Vintage– Your Guide to Financial Empowerment

Abstract

Imagine navigating the complex world of finance, where terms are intricate, clauses are hidden, and every document feels like it's written in a foreign language. Understanding insurance policies, contracts, and financial documents is overwhelming for most people—even professionals struggle with the heavy jargon and fine print. This confusion leaves people feeling powerless, unable to make fully informed decisions, and often worried they're missing something critical.

Introduction

To solve this problem, we introduce Sage—your financial guide. Sage is a generative AI assistant who will simplify these documents and answer all your questions regarding your Insurance paperwork. Users will no longer feel overwhelmed by complex terms or small-print details, instead, they'll gain clarity, insight, and peace of mind. Sage will assist individuals to make calculated, financially literate decisions by democratizing access to financial understanding and empowering individuals to make confident, informed financial decisions. Let Sage guide you toward clarity and confidence in your financial journey—one simplified document at a time.

Programming Language

FinSage will be implemented using HTML, CSS for the front end, Python for the backend, and Gemini API for the AI-Assistant implementation. We plan to use the Gemini API, which sources data directly from Google. If this API proves insufficient, we may also compile our dataset of insurance documents to enhance the model's understanding of different financial document formats.

Our project leverages a Retrieval-Augmented Generation (RAG) model to help users understand complex financial documents by answering specific questions based on document content. When a user inputs a question into the chatbot, the system processes it by converting the query into an embedding—a numerical representation of

Its semantic meaning—using the Gemini API. The model then searches through pre-indexed document embeddings to find sections most relevant to the question. This

The approach enables users to navigate financial paperwork with ease, finding key information without manually sifting through extensive text.

Datasets

Since we need to be able to parse and understand financial documents, our group is looking to use Google Gemini API as our Large Language Model (LLM) integrated into our backend to process the uploaded user documents. We expect Google Gemini API to be pre-trained, and optimized to handle general financial documents so that we don't need to personally source our datasets.
Google also offers Document AI, a document processing and understanding platform that will allow us to parse through unstructured data and turn them into structured data for easier understanding and analysis.

Existing Code & Extensions

Google Cloud Client Libraries: Python Client Library

https://cloud.google.com/python/docs/reference

Google Cloud Client Libraries offers existing code that can help us access Google Cloud APIs programmatically in Python by using the Python Client Library. This will help our backend implementation as we are planning on using Python as our main backend language. We will tailor this library to use within our project to help with end-to-end integration of document uploading, parsing, and understanding.

Google Cloud AI code samples

https://cloud.google.com/document-ai/docs/samples/documentai-quickstart?hl=en#documentai_quickstart-python
Quickstart is an existing sample code that will help us get started with document parsing. It is included as part of the Google Cloud Document AI documentation and includes code that utilizes the previously mentioned client libraries to parse our intended documents.

Algorithm & Approach

The following steps will detail the approach to our project from initial document parsing to its full analysis and summary for the user.

1. **Document upload and parsing**
   Input: User uploaded financial document (PDF)
   Goal: Parse document to extract relevant data and turn it into structured data for easier analysis
   Tasks:
   - Document upload
   - Document processing
   - Data preprocessing

   Output: Clean and structured data with key financial entities

2. **Data extraction and Recognition**
   Input: Parse document text and structured data for Document AI
   Goal: Identify financial entities and terms necessary for user understanding
   Tasks:
   - Entity Recognition
   - Key Term Extraction

Output: structured representation of the uploaded financial document with extracted key terms identified

3. **NLP and Summarization with Google Gemini**
   Input: Extracted document data
   Goal: Summary of financial document in layman's terms for easy user comprehension
   Tasks:
   - Prompt Construction
   - Model API Call
   - Results Processing

   Output: User-friendly summary of financial document with all key points identified and terms simplified

By implementing these steps in the backend, we will then integrate it with a frontend that enhances user experience by seamlessly coordinating all steps for optimal document comprehension and analysis.

Timeline

Keeping in mind the show timeline for the project, we will implement the Waterfall method for implementation. Dividing the project into 4 runs for 4 weeks. The first week will be focused on the development and design of the website, the second week will be focused on developing the product– Full Stack AI development. During the third week, we will be designing the front end to make the application visually appealing and user-friendly. During the last week, we will focus on testing and iteration, and we may also collect user feedback for improvement in future sprints.

Roles & Responsibilities

We have divided the project based on each of the separate technology stacks to divide the work. Hiba will be working on connecting the front end and back end using the Django framework and working on the frontend. Khushi and Huyen will be working on the backend primarily. This may vary as the project continues if we need more bandwidth in a certain area.