Project Name: RAG SaaS Application
Expected Date of completion: 06/12/24

Project Overview:

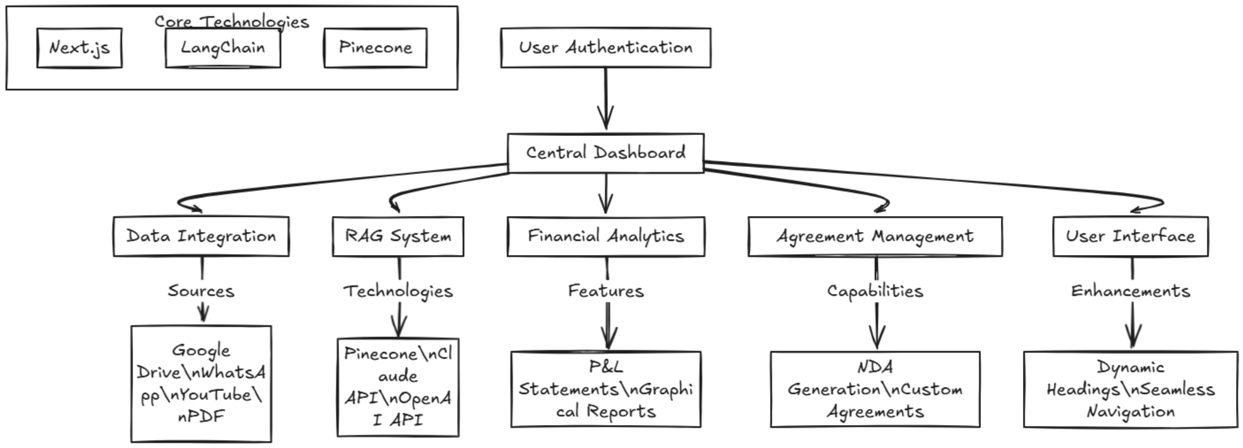
The proposed platform is a SaaS-based solution designed to streamline data processing, retrieval, and analysis through an intuitive and user-friendly interface. It enables seamless user authentication, robust data integration, and automated content summarization. With advanced visualization tools, dynamic customization options, and secure retrieval mechanisms, the platform ensures efficient handling of diverse data sources.
Built with scalability and modularity in mind, it supports document processing, agreement management, and actionable insights through advanced computational techniques. The platform prioritizes accuracy, privacy, and adaptability, making it a comprehensive solution for businesses seeking to enhance operational efficiency and data-driven decision-making.

Key features

1. **Seamless Authentication:** Login via Gmail and LinkedIn for secure access.
2. **Data Integration:** Link and process data from Google Drive and APIs.
3. **Summarization:** Extract insights from WhatsApp chats, YouTube, and PDFs.
4. **Downloadable Outputs:** Export data as CSV or XML files, including financial summaries.
5. **Visualizations:** Generate detailed graphs and perform advanced calculations.
6. **Dynamic Customization:** Modify headings and interface elements easily.
7. **Agreement Management:** Automate NDAs and other document workflows.
8. **Secure RAG Space:** Enable private, retrieval-augmented data access.
9. **Enhanced Search:** Leverage hybrid search for improved accuracy.

Tech Stack

1. **RAG Framework**: LangChain, OpenAI API, Claude API.
2. **Embedding Models**: OpenAI, Hugging Face Sentence Transformers.
3. **Vector Store**: Pinecone
4. **Parsing**: PyPDF2, PDFplumber.
5. **Language**: Python
6. **Frontend:** Next.js, Tailwind CSS
7. **Backend:** Node.js with Express, FastAPI
8. **Database:** MongoDB Atlas
9. **Authentication:** NextAuth.js
10. **Deployment:** Vercel (Frontend), Render (Backend)



Milestones and Timelines:

No. of Days	Milestone	Deliverables
1	Basic Infrastructure Setup	- Install required libraries and frameworks (LangChain, OpenAI API, Claude API, vector database, etc.). - Test APIs for summarization and embeddings.

		- Set up the basic **Next.js** frontend with **Tailwind CSS** for styling. - Create the main layout, and basic UI components and establish routing. - Integrate seamless login with **Gmail** and **LinkedIn** using **NextAuth.js** - Ensure the authentication system is secure and functions smoothly.
1	Document Parsing, Preprocessing and Data flow on platform	- Develop parsers for PDFs (using PyPDF2 or PDFplumber). - Implement emoji-friendly WhatsApp chat file parsing. - Integrate OCR for image-based PDFs. - Set up the backend framework (e.g., **Node.js/Express** or **Next.js API routes**) and establish a basic structure for handling requests. - Ensure the server is ready to handle user login data and any preliminary API interactions. - Implement basic data flow between the frontend and backend, including handling user data securely. - Ensure that authenticated users can interact with the platform (e.g., uploading files, and saving settings).

2-3	Vector Store, Embeddings, RAG API handling, Google Drive API integration, and summary input-output interface	- Set up ChromaDB or Pinecone. - Generate embeddings for parsed documents using Hugging Face/OpenAI models. - Implement hybrid search mechanisms. - Implement functionality for users to link their **Google Drive** account and upload files. - Use **Google Drive API** to fetch attachments and allow seamless access to user files. - Implement endpoints for fetching summaries based on data from WhatsApp chats, YouTube, and PDFs. - Build the frontend UI to allow users to upload files, view summaries, and manage their documents (e.g., view, edit, delete). - Include basic options for downloading the summarized outputs (CSV/XML).
1	Chunking, Retrieval and working on Graph UI	- Develop and test chunking strategies for documents. - Implement retrieval logic using embeddings. - Add UI components that allow users to interact with and modify graphs or calculations.

		- Enable features such as zooming, data point hovering, or exporting graphs as images (e.g., PNG/PDF). - Allow users to download graph data in formats like CSV for offline analysis.
1	Summarization and Q&A Upload PDF, text files and connect the app to RAG API	- Integrate OpenAI GPT-4 and Claude APIs. - Test summarization and Q&A functionality for various document types. - Implement file upload functionality with support for **PDFs** and text files (WhatsApp chats). - Integrate backend logic for file parsing and processing using libraries like pdf-lib. - Connect the app to the summarization API (RAG system or external APIs like OpenAI/Claude). - Display summarized data on the frontend
2-3	Session Management, Memory, and automated agreement for NDAs	- Add session memory to track multi-step queries. - Ensure memory works seamlessly across multiple documents. - Integrate automated management for **NDAs** and other agreements, allowing users to upload, sign, and store agreements securely. - Implement a feature to automatically generate new agreements based on

		templates and user inputs - Allow users to modify headings and labels dynamically on the frontend. - Perform end-to-end testing of all functionalities. - Optimize API calls and vector database queries
1-2	Final Testing and Deployment	- Conduct thorough testing across all modules. - Prepare comprehensive documentation for the RAG implementation, including architecture and workflows. - Deploy the **Next.js frontend** on **Vercel**. - Deploy the backend on **Render** or another suitable service.

Deliverables:

- Fully functional frontend with Next.js and Tailwind CSS, integrated with secure login via Gmail and LinkedIn using NextAuth.js.
- Implemented PDF parsers, emoji-friendly WhatsApp chat file parsing, and OCR for image-based PDFs. Backend structured to handle user data and API interactions securely.
- Integrated ChromaDB or Pinecone, generated embeddings using Hugging Face/OpenAI models, and implemented hybrid search mechanisms.
- Enabled users to connect Google Drive, fetch attachments, and upload files seamlessly.
- Built APIs for summarizing WhatsApp chats, PDFs, and YouTube content, with outputs available for download in CSV/XML formats.
- Developed interfaces for file uploads, managing summaries, and displaying data dynamically with options for editing and deletion.

- Added graph UI with zooming, data point interaction, and export capabilities, alongside downloading graph data as CSV.
- Integrated GPT-4 and Claude APIs for summarization and Q&A functionality across diverse document types.
- Implemented session memory for multi-step queries and automated NDA management with templates.
- Successfully tested all modules, documented workflows, and deployed the frontend on Vercel and the backend on Render.