**PDFlex:** ab pdf chatBot

**Introduction:** PDFlex is an innovative chatbot platform designed to revolutionize the way users interact with PDF documents. Powered by the cutting-edge OpenAI GPT-3 model and Langchain, PDFlex offers a seamless multi-user experience, enabling efficient document management and intelligent query resolution. **Key Features:**

* **Multi-User Platform with Kinde Authentication:** The system employs Kinde authentication to ensure a secure and seamless multi-user experience. Users can create accounts, log in securely, and access the platform's features with ease.
* **Dashboard for File Management**: A user-friendly dashboard is provided, allowing users to upload new PDF files and manage or delete previously uploaded documents. These files are stored in a cloud database, ensuring accessibility and reliability.
* **Chat Box Integration:** The core functionality of the platform lies in its chat box, where users can engage in conversational interactions with the chat bot. Users can pose questions related to the uploaded PDF documents and receive relevant responses in real-time.
* **Chat History Storage:** All conversations between users and the chat bot are logged, enabling users to scroll back through their chat history and review previous interactions. This feature enhances user experience and facilitates continuity in the interview preparation process.
* **User-Friendly UI:** The platform boasts a user-friendly interface designed to enhance navigation and improve overall user experience. With intuitive design elements and smooth transitions, users can seamlessly navigate between different sections of the platform.
* **Feedback Page:** To foster continuous improvement, a feedback page is incorporated into the platform, allowing users to share their thoughts **and suggestions with the development team. This feedback loop ensures that the platform evolves to meet users' needs effectively.**
* **Next.js and TypeScript Implementation with TRPC:** The project utilizes Next.js and TypeScript for fast client and server-side rendering, ensuring optimal performance and scalability. TRPC is employed to enforce type safety and enhance the reliability of the application  
  **Methodology:** The methodology behind the project, termed "genAi," combines the capabilities of LangChain's Large Language Models (LLM), OpenAI's GPT-3 model, and PineCone cloud vector database to deliver a sophisticated and efficient interview preparation platform. Here's a breakdown of the methodology:

**1. LangChain LLM and OpenAI Embeddings** - Each page of the PDF document is converted into vector representations using LangChain's Large Language Models (LLM) and OpenAI embeddings.

- These vector representations capture the semantic meaning and context of each page, enabling efficient querying and matching.

**2. Storage in PineCone Cloud Vector Database:** - The vector representations of each page are stored in the PineCone cloud vector database.

- This database facilitates fast and scalable retrieval of vector representations, ensuring optimal performance during query processing.

**3. Query Processing:** - When a user submits a query, the query undergoes the same process of conversion into a vector representation using LangChain LLM and OpenAI embeddings.

- The resulting query vector is then used to find the best matching vector pages from the PDF document stored in the PineCone database.

**4. Matching with PineCone:** - PineCone is utilized to efficiently search and retrieve the six most closely matching vector pages from the PDF document based on the query vector.

- This process ensures that relevant pages are identified swiftly, enhancing the accuracy and effectiveness of the response.

**5. Query History for Improved Accuracy:** - The last five queries and their corresponding results are stored and utilized to improve the accuracy of future responses.

- Analyzing past queries helps in understanding user preferences and refining the matching process for better outcomes.

**6. Formatting Prompt for GPT-3:** - The six most matched vector pages from the PDF document, along with the query vector and the vector representations of the last ten messages, are formatted into a prompt.

- This prompt is structured according to the specifications required by OpenAI's GPT-3 LLM for generating responses.

**7. Utilizing GPT-3 for Response Generation:** - The formatted prompt is passed as an argument to OpenAI's GPT-3 LLM through an API call.

- GPT-3 generates a response based on the provided prompt, leveraging its advanced language generation capabilities.

**8. Real-time Streaming to Client:** - The response generated by GPT-3 is stored in the database and streamed to the client in real-time.

- This ensures seamless interaction and immediate feedback for the user, enhancing the overall user experience.

In summary, the genAi methodology harnesses the power of LangChain LLM, OpenAI's GPT-3 model, and PineCone cloud vector database to create a robust and efficient interview preparation platform. By leveraging advanced techniques in natural language processing and vector representation, the platform delivers personalized and insightful responses to user queries in real-time.

**Set up**

* **Fork and Download Project:**

Navigate to your GitHub repository and click on the "Fork" button to create a copy of the repository under your account.

Once forked, navigate to your forked repository and click on the "Code" button, then select "Download ZIP" to download the project files to your local machine.

Extract the downloaded ZIP file to a location of your choice.

* **Install Dependencies:** Open a terminal window and navigate to the location where you extracted the project files.

Run the following command to install all the project dependencies using npm: npm install

* **Add API Keys to .env File:**

Inside the extracted project folder, locate the .env file.

Open the .env file in a text editor.

Add the required API keys for the various cloud services used in the project. For each service, add the corresponding API keys in the .env file as follows:



**Configure Services:**

Ensure that you have registered accounts for the cloud services mentioned (Kinde authentication, Neon cloud PostgreSQL database, uploadThing, Pinecone cloud vector database, and OpenAI API).

Retrieve the API keys for each service and replace the placeholder values in the .env file with your actual API keys.

**Run the Project:**

After adding the API keys to the .env file, save the changes.

In the terminal, navigate to the project directory if you're not already there.

Run the command to start the project locally:

**Frontend Specifications:**

1. **Tailwind CSS for Responsive Design:** -

* Utilized Tailwind CSS to implement modern and responsive design principles throughout the project.
* -Tailwind CSS facilitates the creation of responsive layouts and components with minimal CSS code, enhancing the project's aesthetics and usability.

2. **ShadCn UI Library:**

* Integrated ShadCn UI Library to streamline the development of basic components such as buttons, dialogue boxes, inputs, and text areas.
* By leveraging ShadCn UI Library, development overhead is reduced, and consistency in UI components is maintained, resulting in a smoother user experience.

**3. react-PdfRender for PDF Document Rendering:**

* - Implemented react-PdfRender to enable the rendering of PDF documents within the project.
* - This feature allows users to upload PDF files and view them seamlessly within the application, enhancing document accessibility and usability.

**4. Chat Box Functionality:**

* - Developed a chat box feature where users can interact with the chat bot to ask queries and receive responses from the API.
* - The chat box provides a user-friendly interface for communication, facilitating intuitive interaction and enhancing the user experience.

**5. Navigation Bar, Dashboard, Landing Page, Pricing Page, Support Page:**

* - Designed and implemented various pages to enhance user experience and facilitate smooth navigation.
* - The navigation bar ensures easy access to different sections of the application, while the dashboard provides a centralized hub for managing files and accessing features.
* - The landing page, pricing page, and support page are tailored to provide relevant information and support to users, improving overall engagement and satisfaction.

**6. Next.js App Router:**

* - Utilized Next.js app router for efficient client-side routing within the application.
* - Next.js app router enhances navigation by providing fast and seamless transitions between pages, resulting in a smoother user experience overall.

By incorporating these frontend specifications, the project ensures a visually appealing, user-friendly, and responsive interface, enhancing usability and engagement for users interacting with the application.

A

**Front End Specifications:**

1. **User Authentication**:

* - Implemented using Kinvey Auth for secure user authentication.

2. **Database Management**:

* - Utilized Neon Cloud PostgreSQL database for efficient data storage.
* - Integrated PRISMA (ORM) for seamless interaction with the database.
* - Maintained separate tables for:
* - User information
* - File information (e.g., userID, fileID, URL, upload date)
* - Message information (related to uploaded files)

**3. File Hosting:**

* - Employed the UploadThing third-party library built on Amazon AWS S3 service for hosting PDF files securely.

**4. Vector Database:**

* - Leveraged PineCone Cloud Vector Database to store vector pages of PDFs and messages.
* - Utilized OpenAI Embeddings and LangChain to generate vector pages for PDFs and messages.

**5. Type Safety:**

* - Utilized TypeScript to ensure type safety throughout the front-end development process.

**6. API Endpoint:**

* - Created a TPRC (Thin Library built over React Query) endpoint for interacting with the backend API, ensuring type-safe API responses.

**Github :** <https://github.com/ayushssshhh/pdflex>**Visit live Link:** <https://pdflex-psi.vercel.app/>