**Project Database List**

**Overview**

Our database team focuses on information storage for the chatbot system. We have designed an interface that processes user-uploaded files and uses AI to generate responses based on the content of those files. Our system utilizes **Python's Flask** as the backend API, **PostgreSQL** as the database, and **pgvector** to handle embeddings. The system allows users to upload files, which are stored in a local PostgreSQL database, and uses OpenAI’s API-generated embeddings to enable search and retrieval functionality.

**Key Components of the Backend**

**Flask API:**

Flask serves as the primary backend framework, handling HTTP requests and acting as the bridge between the frontend and the database. Flask provides routes for file uploads (/upload) and keyword searches (/search), enabling user interaction through the web interface.

**PostgreSQL:**

We chose **PostgreSQL** as the database because of its scalability and its ability to support large volumes of data. Specifically, it supports the **pgvector** extension, which allows for vector-based similarity searches, making it suitable for the project’s needs to store and search information.

**pgvector:**

**pgvector** is a PostgreSQL extension that allows for efficient storage and search of vector embeddings. The reason we chose **pgvector** is its capability to perform similarity searches using embeddings. This is crucial for allowing the chatbot to retrieve relevant files based on the semantic meaning of user queries. We use the **pgvector** to compute the similarity between the query embedding and stored document embeddings. This operator returns the distance between vectors, enabling us to rank documents by relevance.

**OpenAI:**

**Implementation Steps**

**File Upload and Searching**

Users can upload files through the web interface, which are processed by the */upload* route in Flask.

**Why we choose pgvector and postersql?**