100x____ - Discord Bot

Name Ideas

1. 100xSage (convey the idea of wisdom, knowledge, and guidance, rooted in the timeless concept of a "sage" being someone wise and learned)
2. 100xLumos (A Harry Potter-inspired name meaning light, symbolizing clarity)
3. 100xEureka (For that "aha!" moment when doubts get resolved)
4. 100xResolve
5. 100xNova (A star which suddenly turns bright, implying a problem solver which resolves doubts or points to the correct direction like a star would)
6. 100xCipher (Cipher implying d*ecoding complexities with ease, sleek and mysterious)*

▼ Concept + Pitch Document: Discord-Based Agentic Bot for Efficient Doubt Resolution in Web Dev + DevOps Course

Introduction

The current system for resolving student queries in the course involves TAs addressing a wide range of questions, from basic to advanced. This often results in repetitive answering and an inefficient use of TA and instructor time. We propose an intelligent Discord bot that leverages a knowledge base and Retrieval-Augmented Generation (RAG) to streamline the resolution of student doubts, reduce TA workload, and improve response times.

Proposed Solution

The proposed bot will act as an agent that integrates:

1. **Knowledge Base Access**: Query existing Q&A pairs and reference course materials stored in a knowledge base.

2. **Question Summarization & Storage**: Allow TAs to submit answered threads, which will be summarized and stored in the knowledge base in a Q&A format.
3. **Interactive Database**: Facilitate admin and staff management of the database for CRUD operations.
4. **Fallback to Human Assistance**: Seamlessly escalate unresolved queries to TAs.
5. **Future Vision**: Integrate with video lecture transcripts for advanced referencing.

System Flow

1. **User Query Handling**:
    - A student tags the bot on Discord with a query.
    - The bot first searches the vector database for relevant Q&A pairs.
    - Uses RAG to generate a response.
2. **Unresolved Queries**:
    - If no satisfactory answer is found, the bot escalates the query to a TA.
    - The TA answers the query, optionally marking it for addition to the knowledge base.
3. **Knowledge Base Management**:
    - TAs can submit resolved threads to the bot.
    - The bot summarizes the thread into a Q&A format and stores it in the database.
    - Admins (instructors) have full CRUD privileges, while TAs can suggest or challenge database entries.

Core Components

1. Knowledge Base

- **Data Sources**:
  - Existing Q&A pairs answered by TAs.
  - Notion documents, course materials, and summaries.
- **Storage**:
  - Vector database for semantic search and retrieval.
  - Metadata for each Q&A pair, such as topic, difficulty, and source.

2. Discord Bot

- **Features**:
  - Receive queries tagged by students.
  - Perform vector searches in the database.
  - Generate responses using RAG.
  - Escalate unresolved queries to TAs.
  - Provide an interface for TAs to submit threads for database updates.

3. RAG System

- Uses the vector database to retrieve relevant Q&A pairs.
- Leverages a language model to generate or refine answers based on retrieved data.

4. Admin & Staff Portal

- **Admin (Instructor)**:
  - CRUD operations on the database.
  - Approve or reject TA-submitted Q&A pairs.
- **Staff (TAs)**:
  - Suggest edits or challenge database entries.
  - View unresolved queries for manual handling.

5. Future Integration with Video Sessions

- Convert weekly video sessions into transcripts.
- Index transcripts for semantic search.
- Allow students to reference specific video content in their queries.

Technical Stack

- **Backend**: Python with FastAPI for API development.
- **Database**:
  - Vector database (e.g., Pinecone or Weaviate) for semantic search.
  - Relational database (e.g., PostgreSQL) for metadata and management.
- **Discord Integration**: Discord API for bot functionalities.
- **RAG Framework**: OpenAI or similar LLMs for Retrieval-Augmented Generation.
- **Frontend**: React for the admin and staff portal.

Benefits

- **Efficiency**: Saves TA and instructor time by reducing repetitive queries.
- **Scalability**: Handles an increasing volume of questions with ease.
- **Accessibility**: Provides students with immediate access to course knowledge.
- **Collaboration**: Facilitates a transparent and interactive knowledge management system.
- **Future-Ready**: Lays the groundwork for integrating advanced features like video referencing.

Implementation Plan

Phase 1: MVP Development

1. Set up the vector database and integrate it with a basic knowledge base.

2. Develop the Discord bot to:
    - Accept queries.
    - Query the database and return answers.
    - Escalate unresolved queries to TAs.
3. Build the admin and staff portal with basic CRUD functionalities.

Phase 2: Enhanced Features

1. Integrate summarization of TA-resolved threads.
2. Automate Q&A storage into the database.
3. Implement TA permissions for suggesting or challenging database entries.

Phase 3: Advanced Integration

1. Add video lecture transcript indexing and referencing.
2. Enhance the bot's capabilities with contextual understanding of complex queries.

Conclusion

This agentic bot will assist how student queries are handled in the course. By leveraging advanced NLP and a collaborative knowledge management system, we aim to enhance learning experiences while optimizing TA and instructor resources. The bot's modular design ensures scalability and adaptability for future enhancements.