



# Waypoint: Find your way at IA

Rebanto Nath & Sreyas Sabbani

Sponsor  
Dr. Isabell

## Driving Question

How can we leverage **AI** at Innovation Academy to **help students and staff** easily access **accurate information** about school events, schedules, and resources?

## Proposed Solution

Our project uses an AI chatbot powered by **Retrieval-Augmented Generation** (RAG) to answer questions about school schedules, events, and resources. With a **custom-built interface** and **semantic search** over school documents, it delivers fast, relevant, and private assistance—right from a school computer.

## User Experience

We used the ReactJS frontend library and TailwindCSS for styling.

### Chatbot

The chatbot features a clean and **uncluttered layout** that creates a strong first impression of simplicity and ease of use. Clear input buttons and a **focused message** area help users quickly understand how to interact, reducing confusion and **encouraging engagement**.

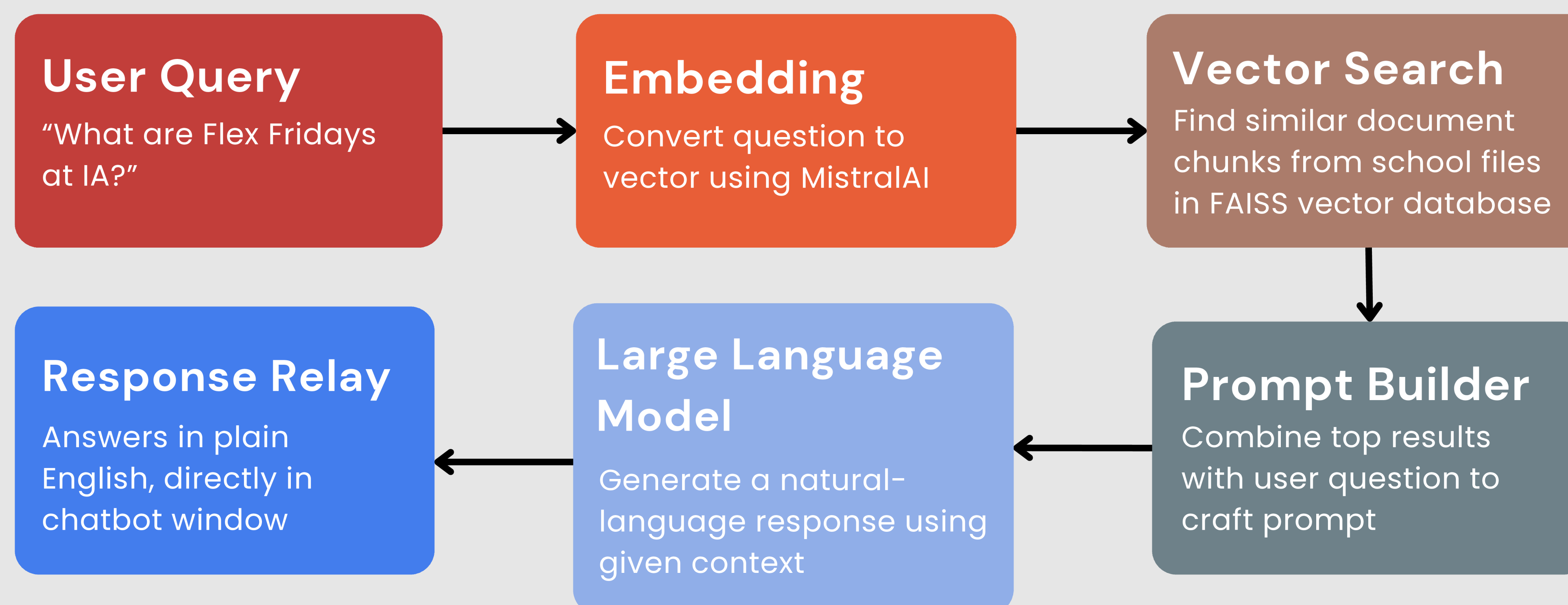
### Informative Views

The club and event pages use a clean, card-based layout that emphasizes **organization** and **clarity**. By highlighting key details like **dates** and **descriptions**, the design makes it easy for users to find **relevant information quickly**, enabling **accessibility** and **efficiency**.

## Tools



## Resource Augmented Generation



Our application uses **Retrieval-Augmented Generation** (RAG) to ensure **accurate, document-based answers**. When a user asks a question, the query is **converted** into a **vector** and matched against **pre-embedded school documents**. The most **relevant** results are passed to a **large language model**, which uses them to generate a **context-aware response**.

## Backend

### Web Router (Express.js)

Once the client's **HTTP POST** request carrying the user prompt is received, the router starts a **Python child process** that initiates the RAG pipeline. This **modularity** ensures clean separation between the **Node.js server logic** and the **Python-based AI processing**. This setup allows easier debugging, flexible updates to the AI model, and smooth communication between components.

### RAG Model (Langchain)

The RAG pipeline begins by using **Mistral AI's embedding model** to convert the user query into a vector, which is then matched against preprocessed school documents stored in a **FAISS vector database**. The top-matching chunks are combined with the original question and sent to **Gemini Flash 2.0**, a high-speed NLP LLM, which generates a context-aware and accurate response based on the retrieved content.

## Next Steps

### Accounts & Auth

We have already implemented a user accounts system and plan to leverage it to tailor chatbot behavior based on user roles—such as students, teachers, and administrators—enabling role-specific access control, response customization, and analytics.

### Realtime Webscrapping

We want to enable the system to dynamically ingest and embed new content every day. This will allow the model to respond with up-to-date information beyond the static vector store.

### Containerize Application

We plan to containerize the system using Docker, allowing us to package the frontend, backend, and RAG components into isolated environments. We also aim to use Kubernetes for orchestration, enabling horizontal scaling.

## Contact Us

