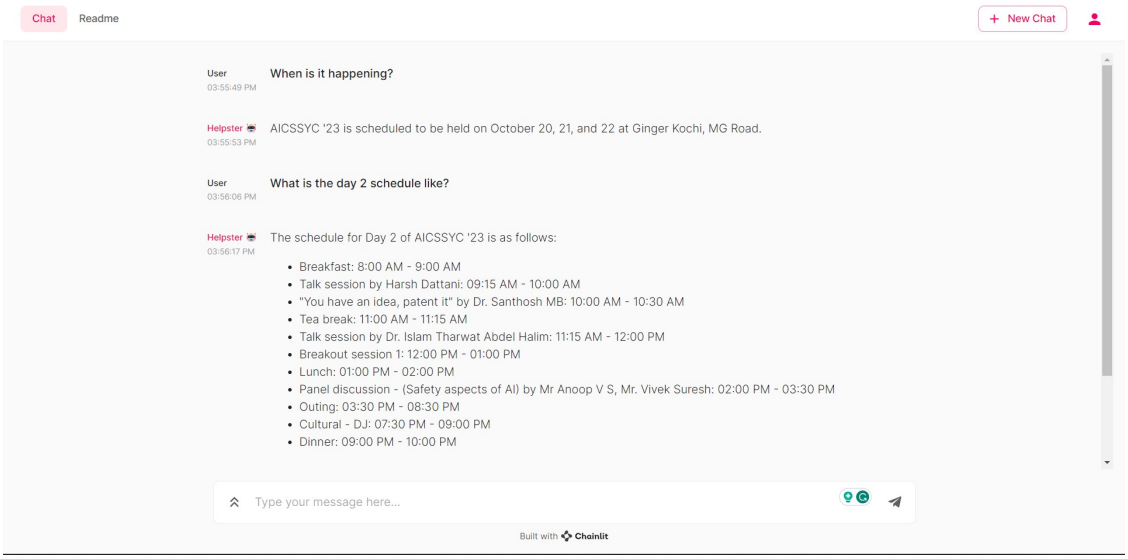
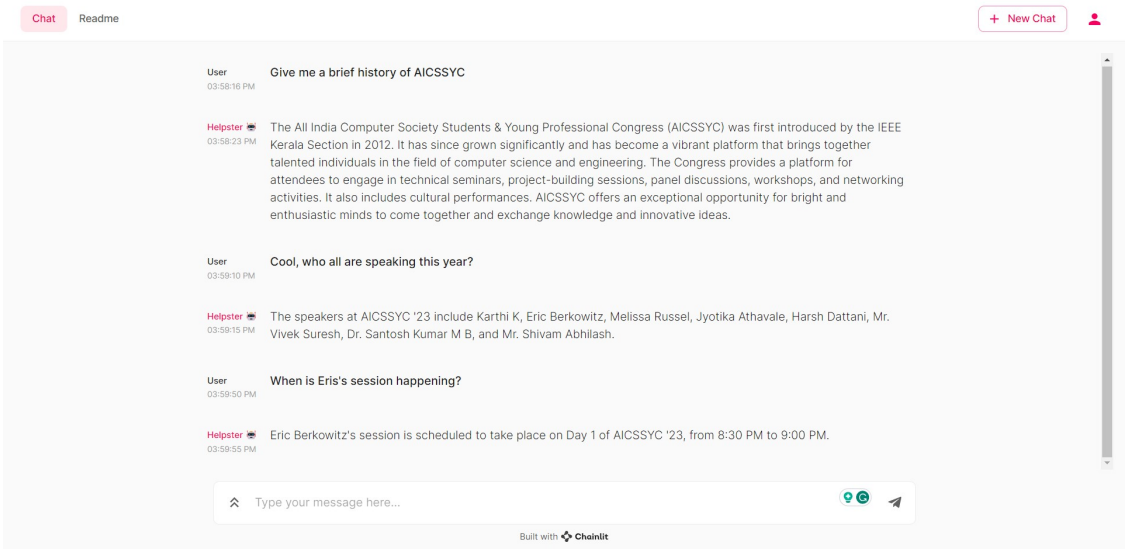
# Helpster - Documentation

Helpster is an LLM based bot made as an entry to **Inquiry Bot Challange** by IEEE CS KS. It can answer questions related to AICSSYC 23, previous AICSSYC's, IEEE CS KS, , IEEE CS, IEEE, and other general questions.

Built using:

- Python
- LangChain
- OpenAI
- ChainLit
- Pinecone




# Demo

Demo Link: https://helpster.onrender.com/

## Points to note

- The demo uses OpenAI's API with credits purchased using my own funds, hence please**keep the demo link confidential** and use it **only for evaluation purposes**.
- The app is hosted on Render, a platform which provides free hosting with limited capacity. The app is hosted on a free plan, hence the app is succeptible to **Cold Starts** (the app may take upto a minute to load initially).

# Development Journey

## Choosing the tools

I have a fair bit of experience in working with LLMs in the past. But, it was mostly the open-source ones like Llama2. Hence, I decided to use OpenAI's API for this project. I have also worked with Pinecone before, so I decided to use it for knowledge retrieval. Then I needed a quick way to come up with a UI for a chat app. Thoght StreamLit would do the trick.

### Building the bot

I hacked together the components to build a simple chat bot first. Then I connected it to Pinecone and added a sample document to the index. Now I had a bot which could answer questions from a single document. By chance, I got to know about ChainLit, which is a purpose built framework for chatbots. I replaced StreamLit with ChainLit. I also wanted to try out LangSmith for a while, I could do it now as I got access to the Beta version. Also, I made a simple script to load data into Pinecone.

### Collecting data

This was arguably the most difficult part of the project. I had to collect data from various sources and clean it. I used `instaloader` to download the instagram posts.I also had to manually collect data from the official website and facebook page. There wasn't much data available on older AICSSYC's, so I had to use the archived websites from Wayback Machine. I also used Wikipedia to collect data about IEEE, IEEE CS, and IEEE CS KS. After this I ran the data through the script to load it into Pinecone.

### Optimizations

Token count matter when using OpenAI's API. So I had to optimize the bot to use as few tokens as possible. Conversational memory buffer was used at first, but the token count was too high. Then I tried out Conversational summarization, which summarizes the chat history periodically, this allowed longer memory at less token count. But here too the token count was too high as summarization itself was adding to token count. As I used the tool, I found out that It is less likely for people to ask follow ups for question beyond 2 or 3 positions into the past. Hence I settled with a simple memory buffer which stores the last 3 questions and answers. This allowed me to reduce the token count to a great extent.

One more thing I noticed while using the bot was that the increased token count is also due to large documents which are retrieved from Pinecone. Hence I decided to use a summarization model to summarize the document and relode it to Pinecone. This reduced the token count even further.

Also, multiple documents were being fetched at a time. This too was increasing the token count. Hence I limited the number of documents to be fetched to 1 per query.

Hallucinations were also a problem. The bot was answering questions which it was not trained to answer. Hence, I explicitly instructed the bot to not answer questions where the context is not present. This made the bot more robust and prevented hallucinations.

## Setup

Clone the repository

```
git clone https://github.com/RohitEdathil/helpster
```

Install dependencies

```
pip install -r requirements.txt
```

Setup environment variables

| Variable | Description |
| --- | --- |
| `OPENAI_API_KEY` | OpenAI API Key |
| `PINECONE_API_KEY` | Pinecone API Key |
| `PINECONE_ENV` | Pinecone Environment Name |

If you want to enable LangChain tracing (Optional)

| Variable | Description |
| --- | --- |
| `LANGCHAIN_TRACING_V2` | Enable version 2 |
| `LANGCHAIN_ENDPOINT` | API endpoint |
| `LANGCHAIN_API_KEY` | API key |
| `LANGCHAIN_PROJECT` | Project name |

Create an index in Pinecone with the name `helpster` and dimension `1536`

## Usage

### Load

```
python3 load.py
```

Running this command will load documents from `data` folder to Pinecone index. All files must be plain text files with `.txt` extension. One file is considered as one document in the index.

**Note:** The index `helpster` must be created in Pinecone before running this command.

### Run

```
chainlit run main.py
```

This will start the server at `http://localhost:8000` where you can ask questions similar to ChatGPT.

## Working

### RAG

This bot uses RAG(Retrieval Augmented Queries). It uses a Pinecone index to retrieve the most similar document to the question. Then it uses the retrieved document as a context to the GPT-3 model to generate the answer.

### ConversationBufferWindowMemory

The bot uses a memory buffer to store the last 3 questions and answers. You can thus ask follow up questions to the bot. It was limited to 3 because of the token consumption limitations.

### Information Constraints

The bot is explicitly instructed to not answer questions which it is not trained to answer. This prevents hallucination and makes the bot more robust. Also the bot will stay close to its purpose.

## Data Sources

Here are the data sources used to train the bot:

- Official website of AICSSYC 23
- Official instagram page of AISSYC 23
- Instagram pages of previous AICSSYC's
- Facebook page of AICSSYC 23
- Archived website of previous AICSSYC's
- IEEE CS KS website
- IEEE CS website
- Wikipedia

## Data Handling

### Training Data (Public)

The data obtained to train the data is publically available. It was obtained mostly manually and a tool called `instaloader` was used to download the instagram posts. The data was then cleaned and formatted to be used for training. Even though the data is publically available, it is still securely stored only in the development system and secure Pinecone servers.

### Chat Logs

All the **chat information is monitored** through `LangSmith`, a sister project of `LangChain`. It is a tool to diagnose and monitor LLM apps. The chat information is stored in a secure database and is only **used for debugging purposes**. Developers may look into the chat information to improve the bot. The chat information is **not shared with any third party**.

### LLM

The LLM used by the app is provided by OpenAI. Hence, data is visible to OpenAI.

Read more about OpenAI's API privacy policy here.

From OpenAI's privacy policy we can see that data from API Platform is **not used to train** the models. Also, the data is not shared with any third party and is SOC 2 Type 2 compliant.

Read more about SOC 2 Type 2 here.

Hence we can conclude there **won't be any data leaks** (user information presented to other users).