# **CSTU Virtual Assistant Documentation**

## **Overview**

The CSTU Virtual Assistant is an AI-powered application designed to assist students at CSTU (California Science and Technology University) with various queries related to courses, international student FAQs, campus locations, and on-campus jobs. It leverages advanced natural language processing (NLP) capabilities provided by OpenAI and utilizes Pinecone for efficient storage and retrieval of structured data.

## **Components and Features**

### **Functionality**

* **User Interaction**: The assistant interacts with users through a text-based interface, processing user queries and providing informative responses.
* **External API Integration**: Integrates with external APIs (OpenAI and Pinecone) to fetch and present relevant information in real time.

### **Integration with OpenAI**

* **GPT Model**: Utilizes OpenAI's GPT model (gpt-4o or gpt-4) to generate responses for user queries that do not directly match structured data in the Pinecone index.
* **Dynamic Conversations**: Implements the chat\_complete\_messages function to handle dynamic conversations and provide contextually appropriate responses.

### **Integration with Pinecone**

* **Data Storage and Retrieval**: Utilizes Pinecone for storing and querying structured data such as course details, FAQs, campus locations, and job listings.
* **Embedding Generation**: Implements the get\_embeddings function to generate embeddings using OpenAI, which are indexed in Pinecone.
* **Data Management**: Implements the upsert\_data function to insert or update data in the Pinecone index (cstu-bot) based on predefined structured data.

### **User Interaction**

* **Query Processing**: Upon receiving a user query, identifies the type of information requested (courses, FAQs, locations, jobs).
* **Data Retrieval**: Uses Pinecone to retrieve the most relevant data based on the user query and presents it to the user.
* **Fallback Mechanism**: If structured data is not found or the query is ambiguous, resorts to using OpenAI to generate a response.

### **Gradio Interface**

* **User Interface**: Provides a user-friendly interface using Gradio, allowing users to interact with the assistant through a textbox.
* **Real-time Interaction**: Displays the user's query and the assistant's response in real-time.
* **Context Management**: Maintains a chat history (chatContext) to keep track of the conversation flow and context.

### **Error Handling**

* **Exception Management**: Includes error handling mechanisms to manage exceptions when generating embeddings or completing chat messages using OpenAI.
* **Robustness**: Ensures smooth operation even in case of API failures or data retrieval issues.

## **Implementation Details**

### **Environment Setup**

* **Dependencies**: Ensures necessary Python packages (python-dotenv, openai, pinecone-client, gradio) are installed.
* **Configuration**: Loads environment variables (like API keys) from .env files for secure API access.

### **Data Population**

* **Structured Data**: The json\_data variable contains structured information such as course details, FAQs for international students, campus locations, and on-campus jobs.
* **Index Management**: Structures data into dictionaries and lists, then uses the upsert\_data function to insert or update entries in the Pinecone index.

### **Chatbot Logic**

* **Main Functionality**: The chatbot function processes user queries and directs them to either Pinecone for structured data retrieval or OpenAI for dynamic responses.
* **Data Querying**: Utilizes the get\_relevant\_info function to query Pinecone based on user input and retrieve relevant information.

## 

## 

## 

## 

## **Chatbot with LLM and Pinecone: A Breakdown**

**1. Product Introduction**

**a) Purpose, Target Audience, and Functionalities**

This chatbot aims to be a virtual assistant for prospective and current students at a university (CSTU). It caters to anyone seeking information about courses, international student resources, campus locations, and on-campus jobs. The chatbot can answer user questions, provide relevant details, and guide them towards their goals.

**b) Problem and Benefits**

Many universities have scattered information across websites and resources. This chatbot tackles this problem by offering a centralized and conversational way to access information. Users benefit from:

* **Accessibility:** Information is readily available through a user-friendly interface.
* **Efficiency:** Quick responses save users time searching through various resources.
* **Personalization:** The chatbot can tailor responses based on user queries.

**2. Chatbot Solution Architecture**

**a) Technology Stack**

* **LLM (Large Language Model):** OpenAI's API is used for generating text, creating summaries, and understanding user intent.
* **Pinecone:** This vector search engine stores and retrieves relevant information based on user queries.
* **python-dotenv:** Manages environment variables securely.
* **gradio:** Creates a web interface for user interaction with the chatbot.

**b) System Architecture and Components**

1. **Data Preprocessing**: Course descriptions, FAQs, location details, and job descriptions are loaded and preprocessed.
2. **Embedding Generation**: OpenAI generates vector embeddings (numerical representations) for the preprocessed text data.
3. **Data Storage (Pinecone)**: The processed data, along with its embeddings and metadata (type and original text), is uploaded to Pinecone for fast retrieval.
4. **User Interface (Gradio)**: This web interface allows users to type their questions and receive chatbot responses.
5. **Chatbot Logic**: This processes user input:
   * If the query matches a specific information type (course, FAQ, etc.), it retrieves relevant data from Pinecone using embeddings.
   * If no direct match is found in Pinecone, it leverages OpenAI to generate a response using the user query and potentially retrieved info from Pinecone for context.

**3. Product Demonstration**

**a) Core Capabilities in Action**

A user asks: "What is the fee structure for masters student”?

Great! Let's break down the fee structure for a first-semester student. Here's what you can expect:

1. \*\*Technology Fee:\*\* $50

2. \*\*Students Association Fee:\*\* $40

3. \*\*Registration Fee:\*\* $250

4. \*\*Graduation Fee (one-time payment):\*\* $250

5. \*\*Per Credit Fee:\*\* $700

Each course carries 1.5 credits, and the duration of each term is 2 months.

Let's assume you're taking a standard full-time load of 4 courses in your first semester:

- \*\*Total Credits:\*\* 4 courses \* 1.5 credits per course = 6 credits

- \*\*Per Credit Fee Total:\*\* 6 credits \* $700 per credit = $4,200

Now, let's add up all the fees for the first semester:

- \*\*Technology Fee:\*\* $50

- \*\*Students Association Fee:\*\* $40

- \*\*Registration Fee:\*\* $250

- \*\*Graduation Fee:\*\* $250

- \*\*Per Credit Fee Total:\*\* $4,200

\*\*Total Fees for the First Semester:\*\* $50 + $40 + $250 + $250 + $4,200 = \*\*$4,790\*\*

Would you like to know more about payment methods or any other details?

Another user asks: "University Location"?

I found some relevant information about locations:

- University Location California Science and Technology University (CSTU), 100 Innovation Way, Santa Clara, CA 95054 (from locations)

……

**b) Development Challenges**

* **Data Preparation:** Ensuring data quality and appropriate structure for embedding generation.
* **Pinecone Indexing:** Fine-tuning search parameters for optimal retrieval based on user queries.
* **LLM Prompt Engineering:** Crafting effective prompts for OpenAI to generate informative and relevant responses in various scenarios.

**4. Lessons Learned**

**a) Key Learnings**

* Combining LLMs and vector search engines like Pinecone can offer a powerful framework for building intelligent chatbots.
* Data preparation and thoughtful prompt design are crucial for accurate and informative chatbot responses.
* Utilizing a web interface (Gradio) simplifies user interaction and streamlines testing and development.

**b) Successes and Obstacles**

* Successfully integrating LLM and Pinecone to create a dynamic chatbot with knowledge retrieval and response generation capabilities.
* Overcoming challenges in data organization and prompt design to ensure accurate and helpful responses.

**c) Project Scope Adjustments**

The initial scope might have focused solely on LLM-based responses. However, integrating Pinecone for knowledge storage and retrieval likely enhanced the chatbot's efficiency and accuracy.

**5. Areas for Improvement**

**a) Performance Enhancement**

* Implementing user feedback mechanisms to identify areas for improvement in response quality and relevance.
* Fine-tuning Pinecone search parameters for even faster and more accurate information retrieval.

**b) User Feedback and Insights**

* Analyzing user interactions to identify frequently asked questions (FAQs) and incorporate them directly into the knowledge base stored in Pinecone.
* Exploring sentiment analysis to understand user satisfaction and tailor responses accordingly.

**c) Potential Enhancements**

* Expanding the chatbot's capabilities to include functionalities like appointment scheduling or task reminders.
* Integrating with university systems (e.g., student portal) to provide real-time updates and personalized information.

This breakdown highlights how the provided code leverages LLMs and Pinecone to create a valuable chatbot for university information access. By continuously improving

## 

## 

## 

## 

## 

## 

## 

## 

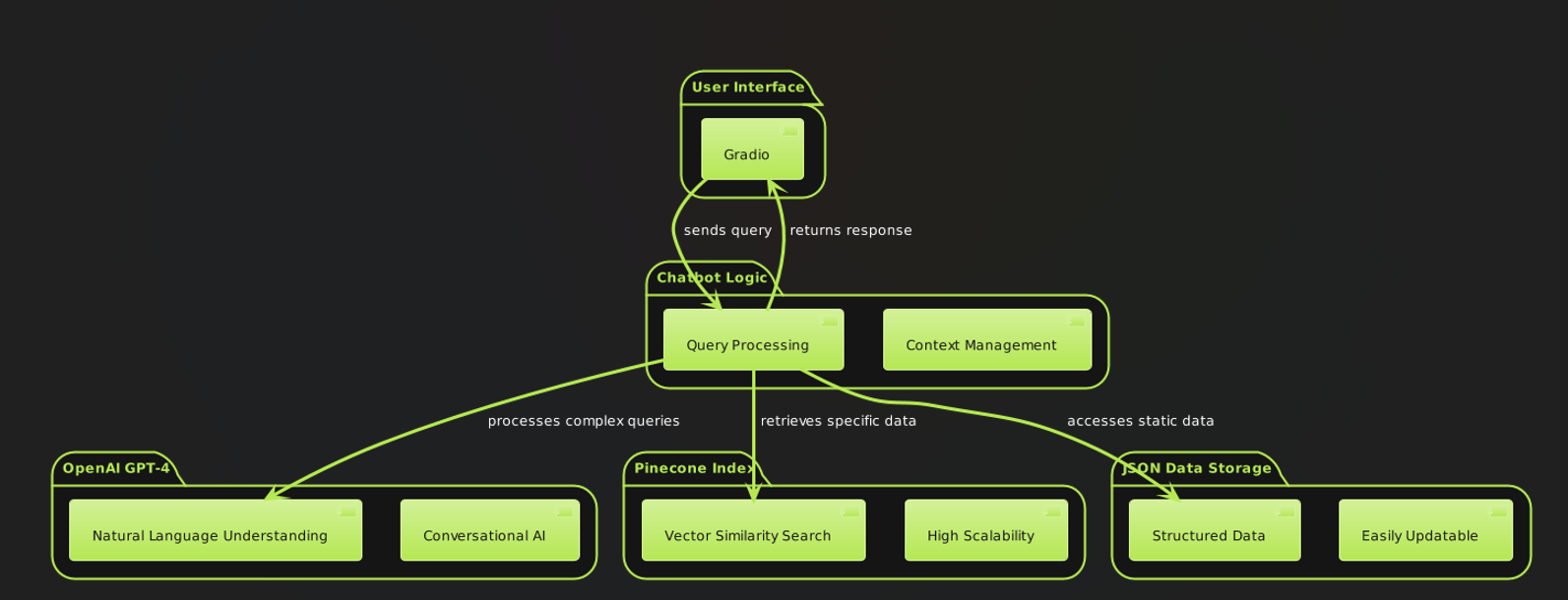
## **System Architecture and Components**

### **Overview**

The CSTU Virtual Assistant system architecture is designed to efficiently handle user queries, retrieve relevant information, and provide accurate and timely responses. The architecture integrates several advanced technologies, including large language models, vector databases, and interactive user interfaces. This section outlines the key components of the system architecture, highlighting innovative and unique features.

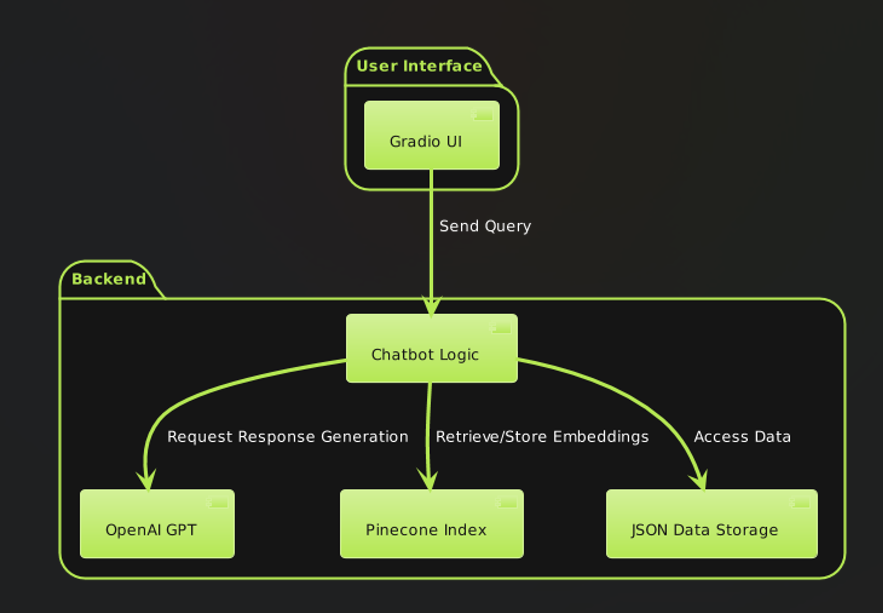
### **System Architecture Diagram**

To visualize the system architecture, here's a simplified representation of the key components and their interactions:



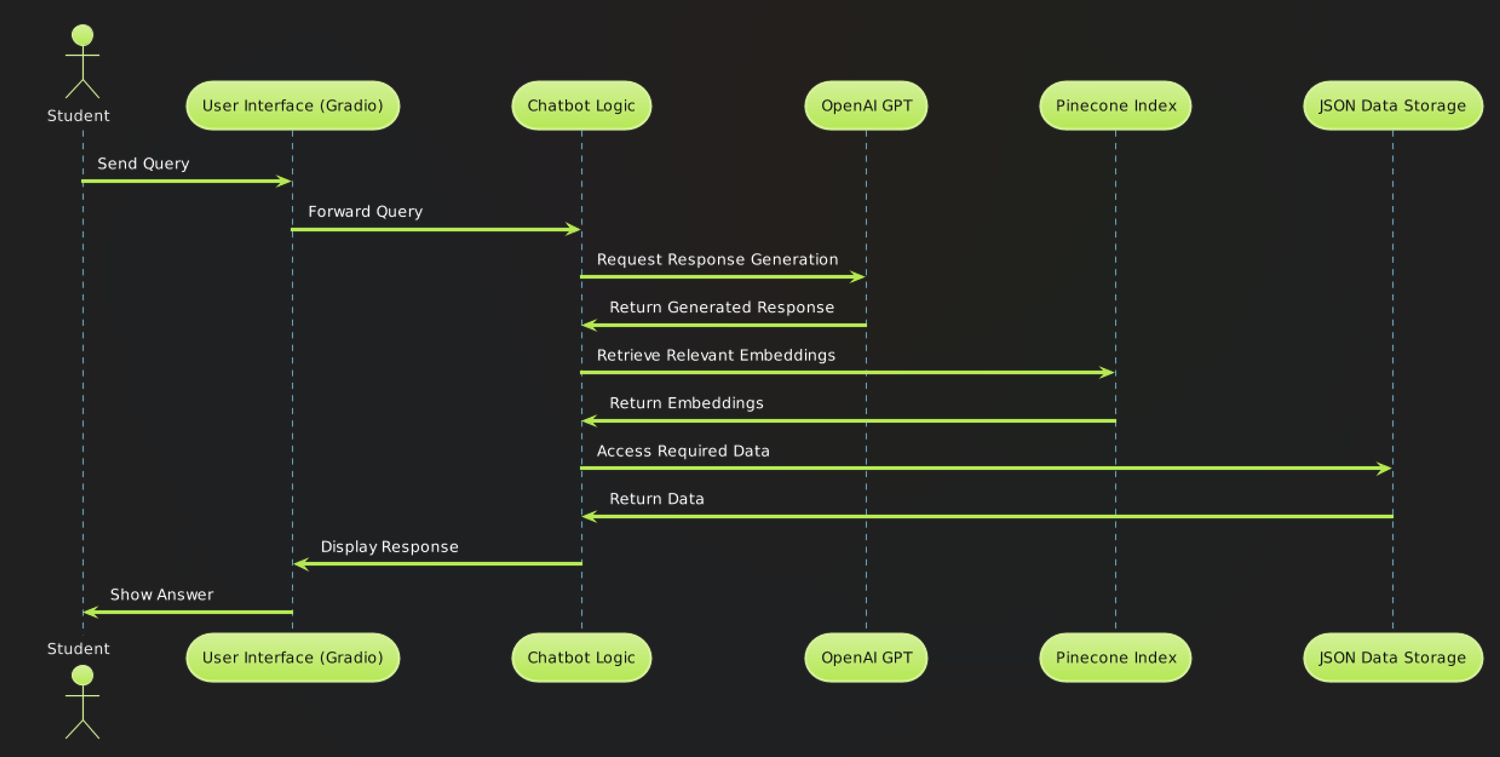
**Component Diagram**:

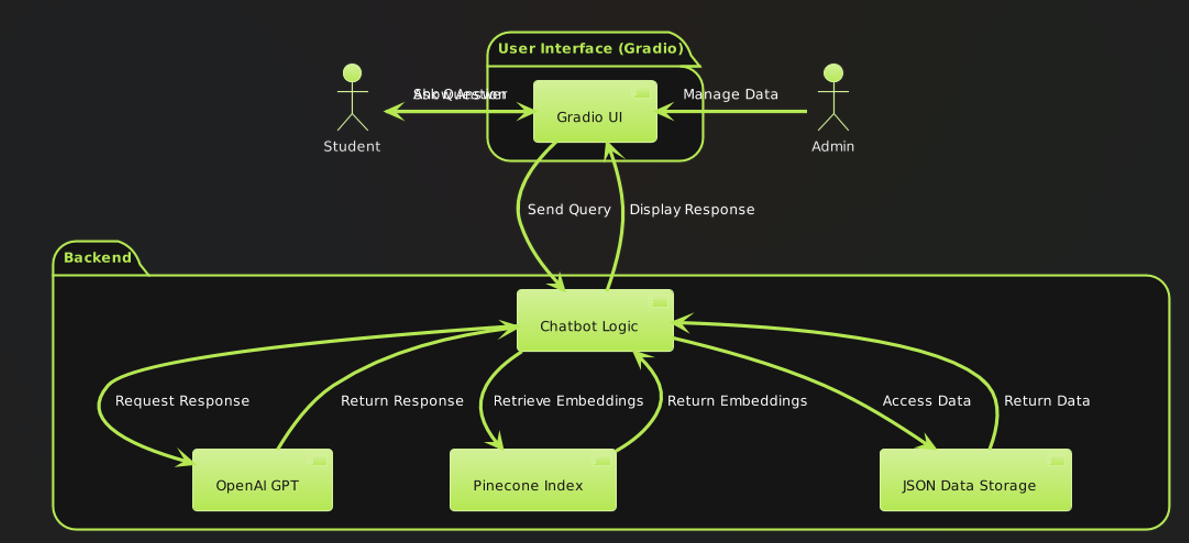
* To show the detailed architecture of each component within the system



**Sequence Diagram**:

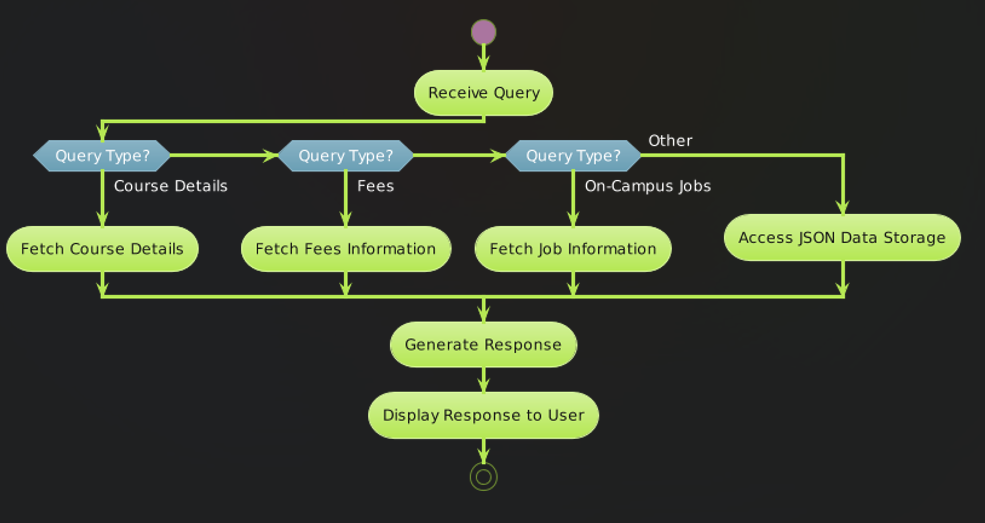
* To illustrate the step-by-step interaction between the user, the chatbot, and various backend components.





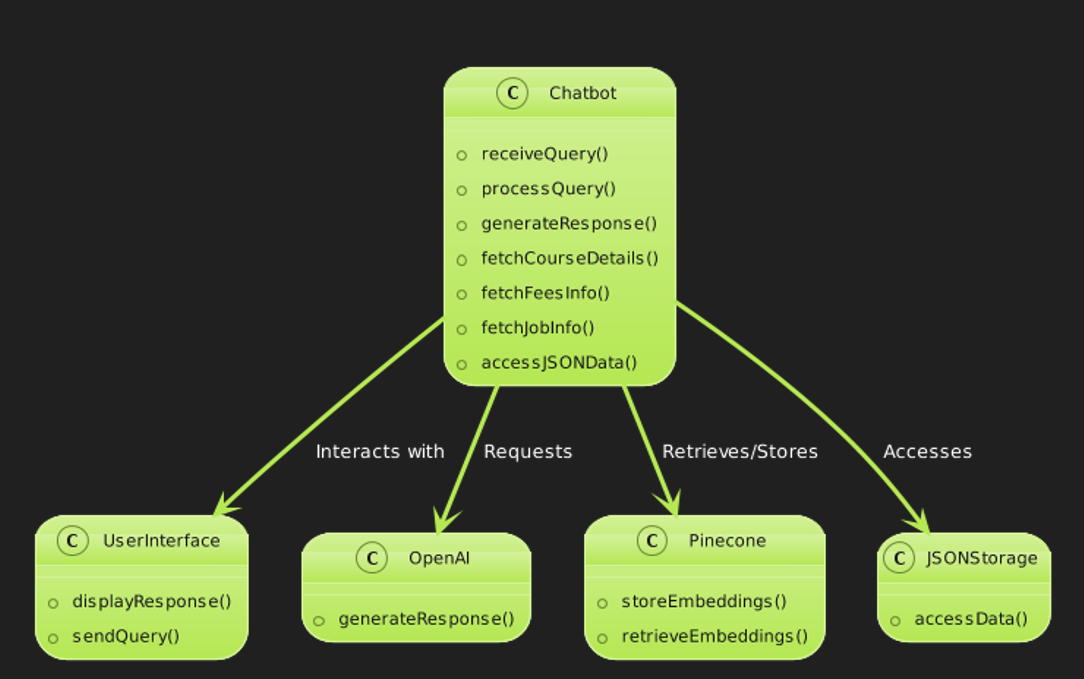
**Detailed Flowchart**:

* To visualize the flow of data and decisions made by the chatbot system for different types of queries.



**Class Diagram**:

* To represent the structure of the system in terms of classes and their relationships.



## **Future Enhancements**

* **Personalization**: Enhance the assistant to remember user preferences or past interactions for a more personalized user experience.
* **Multi-turn Conversations**: Extend functionality to handle multi-turn conversations seamlessly, maintaining context across interactions.
* **Natural Language Understanding**: Improve NLP capabilities to better interpret user queries, including handling variations and nuances in user input.

## **Conclusion**

The CSTU Virtual Assistant is a robust application that harnesses AI technologies to deliver timely and relevant information to students. It seamlessly combines natural language processing with efficient data retrieval techniques to provide a user-friendly interface for interacting with university-related information. With planned enhancements, it has the potential to significantly enhance student support services at CSTU.

**Code Demonstration**  
  
pip install python**-**dotenv

Note: you may need to restart the kernel to use updated packages.Requirement already satisfied: python-dotenv in c:\users\dell\anaconda3\lib\site-packages (1.0.1)

In [2]:

pip install openai

Requirement already satisfied: openai in c:\users\dell\anaconda3\lib\site-packages (0.28.0)Note: you may need to restart the kernel to use updated packages.

Requirement already satisfied: aiohttp in c:\users\dell\anaconda3\lib\site-packages (from openai) (3.9.5)

Requirement already satisfied: tqdm in c:\users\dell\anaconda3\lib\site-packages (from openai) (4.64.1)

Requirement already satisfied: requests>=2.20 in c:\users\dell\anaconda3\lib\site-packages (from openai) (2.32.3)

Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\dell\anaconda3\lib\site-packages (from requests>=2.20->openai) (2.0.4)

Requirement already satisfied: certifi>=2017.4.17 in c:\users\dell\anaconda3\lib\site-packages (from requests>=2.20->openai) (2023.7.22)

Requirement already satisfied: idna<4,>=2.5 in c:\users\dell\anaconda3\lib\site-packages (from requests>=2.20->openai) (3.4)

Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\dell\anaconda3\lib\site-packages (from requests>=2.20->openai) (2.2.1)

Requirement already satisfied: multidict<7.0,>=4.5 in c:\users\dell\anaconda3\lib\site-packages (from aiohttp->openai) (6.0.5)

Requirement already satisfied: aiosignal>=1.1.2 in c:\users\dell\anaconda3\lib\site-packages (from aiohttp->openai) (1.3.1)

Requirement already satisfied: attrs>=17.3.0 in c:\users\dell\anaconda3\lib\site-packages (from aiohttp->openai) (22.1.0)

Requirement already satisfied: frozenlist>=1.1.1 in c:\users\dell\anaconda3\lib\site-packages (from aiohttp->openai) (1.4.1)

Requirement already satisfied: yarl<2.0,>=1.0 in c:\users\dell\anaconda3\lib\site-packages (from aiohttp->openai) (1.9.4)

Requirement already satisfied: async-timeout<5.0,>=4.0 in c:\users\dell\anaconda3\lib\site-packages (from aiohttp->openai) (4.0.3)

Requirement already satisfied: colorama in c:\users\dell\anaconda3\lib\site-packages (from tqdm->openai) (0.4.6)

In [3]:

pip install pinecone**-**client

Requirement already satisfied: pinecone-client in c:\users\dell\anaconda3\lib\site-packages (4.1.1)

Requirement already satisfied: certifi>=2019.11.17 in c:\users\dell\anaconda3\lib\site-packages (from pinecone-client) (2023.7.22)

Requirement already satisfied: typing-extensions>=3.7.4 in c:\users\dell\anaconda3\lib\site-packages (from pinecone-client) (4.11.0)

Requirement already satisfied: tqdm>=4.64.1 in c:\users\dell\anaconda3\lib\site-packages (from pinecone-client) (4.64.1)

Requirement already satisfied: urllib3>=1.26.0 in c:\users\dell\anaconda3\lib\site-packages (from pinecone-client) (2.2.1)

Requirement already satisfied: pinecone-plugin-interface<0.0.8,>=0.0.7 in c:\users\dell\anaconda3\lib\site-packages (from pinecone-client) (0.0.7)

Requirement already satisfied: colorama in c:\users\dell\anaconda3\lib\site-packages (from tqdm>=4.64.1->pinecone-client) (0.4.6)

Note: you may need to restart the kernel to use updated packages.

In [4]:

**import** os

**import** json

**from** dotenv **import** load\_dotenv

**import** openai

**import** pinecone

**import** gradio **as** gr

In [5]:

load\_dotenv()

Out[5]:

True

In [6]:

**from** dotenv **import** load\_dotenv *# Optional, for future environment variable needs*

**from** pinecone **import** Pinecone, ServerlessSpec

*# Pinecone API Key (replace with your actual key)*

YOUR\_API\_KEY **=** 'fbbf0a67-1250-4c20-ab2b-c97e5959c05e'

*# Create the Pinecone index*

index **=** pinecone**.**Index(name**=**"cstu-bot", api\_key**=**"fbbf0a67-1250-4c20-ab2b-c97e5959c05e", host**=**"https://cstu-bot-91teuj3.svc.aped-4627-b74a.pinecone.io")

*# Initialize OpenAI API key*

openai**.**api\_key **=** os**.**getenv("OPENAI\_API\_KEY")

*# Create an instance of Pinecone*

pc **=** Pinecone(api\_key**=**"fbbf0a67-1250-4c20-ab2b-c97e5959c05e")

*# Define the index name*

index\_name **=** os**.**getenv("CSTU\_BOT") *# Use environment variable if set*

*# Check if the index already exists*

**if** index\_name **not** **in** pc**.**list\_indexes()**.**names():

*# Define the serverless specification*

spec **=** ServerlessSpec(cloud**=**'aws', region**=**'us-east-1') *# Adjust cloud and region as needed*

*# Name of the index*

CSTU\_BOT **=** "cstu-bot" *# Provide a suitable name for your index*

*# Create the index with the specified specification*

*# pc.create\_index(*

*# name=CSTU\_BOT,*

*# dimension=1536, # Adjust dimension if needed*

*# metric='cosine', # Specify the metric for similarity search*

*# spec=spec # Pass the ServerlessSpec object*

*#)*

In [7]:

*# JSON data directly defined within the code*

json\_data **=** {

"Course\_Details": [

{

"code": "MB/CSE 652",

"title": "Prompt Engineering",

"details": "Explore advanced techniques for understanding and generating human language."

},

{

"code": "MB/CSE 632",

"title": "Introduction to Cloud Computing",

"details": "Dive into the world of on-demand computing resources with this introduction to Cloud Computing! Learn how to leverage the cloud for storage, processing, and applications."

},

{

"code": "MB/CSE 648",

"title": "GPT application",

"details": "Unleash the power of AI: Learn to leverage GPT for automatic topic extraction, improve content organization, and unlock deeper insights from your text."

},

{

"code": "MB/CSE 600",

"title": "Python",

"details": "Python empowers you to automate tasks, analyze data, and build applications. Learn Python from scratch or enhance your skills in this versatile programming language."

},

{

"code": "MB/CSE 590",

"title": "Network Security and AI",

"details": "This course explores how Artificial Intelligence empowers network security strategies, defending against evolving cyber threats."

},

{

"code": "MB/CSE 638",

"title": "Deep learning with TensorFlow",

"details": "Master deep learning concepts and build neural networks for real-world tasks using TensorFlow, a popular open-source framework."

}

],

"International\_Students\_faqs": [

{

"question": "What are the different types of fees for international students?",

"answer": "There are several fees associated with studying at CSTU University for international students. These include:\n\* Technology Fee: $50 (one-time fee)\n\* Students Association Fee: $40 (per semester)\n\* Registration Fee: $250 (one-time fee)\n\* Graduation Fee: $250 (one-time fee)\n\* Per Credit Fee: $700\n\*\*Note:\*\* Each Course has 1.5 Credits and lasts one Term (2 months)."

},

{

"question": "What payment methods are available for international students?",

"answer": "International students can pay their fees through the following methods:\n\* \*\*Credit Card:\*\* Accepted credit cards include Visa, Mastercard, and American Express. A secure online payment portal is available within the university system.\n\* \*\*Bank Transfer:\*\* International students can initiate a bank transfer to the university designated account. Details and instructions will be provided upon request."

},

{

"question": "When are the deadlines for fee payments?",

"answer": "Fee payment deadlines are typically set before the start of each semester. Specific deadlines will be communicated to students via email and are also available on the university website. Timely payments are crucial to avoid late fees and potential registration holds."

},

{

"question": "What happens if I have questions about my student account or need assistance with payments?",

"answer": "The university International Student Office is here to help! You can contact them via email at iso@cstu.edu. They can assist with any questions or concerns regarding your student account and fee payments."

},

{

"question": "What is Curricular Practical Training (CPT)?",

"answer": "Curricular Practical Training (CPT) is an authorization for F-1 international students to gain practical experience directly related to their major field of study. It must be an integral part of your academic program and requires approval from your Designated School Official (DSO). There are limitations on the duration of CPT, so be sure to consult the International Student Office for details and eligibility requirements."

},

{

"question": "What is Optional Practical Training (OPT)?",

"answer": "Optional Practical Training (OPT) allows F-1 students temporary employment authorization to gain practical experience after completing their academic studies. There are two types of OPT: pre-completion (before graduation) and post-completion (after graduation). Students must apply for OPT through USCIS with approval from their DSO. STEM (Science, Technology, Engineering, and Mathematics) program graduates may be eligible for an extended OPT period."

},

{

"question": "What is SEVIS (Student and Exchange Visitor Information System)?",

"answer": "The Student and Exchange Visitor Information System (SEVIS) is a U.S. Department of Homeland Security (DHS) database that tracks F and M nonimmigrant students enrolled in U.S. schools. All educational institutions approved to enroll international students must report student information to SEVIS. Maintaining a valid SEVIS record is crucial for your immigration status as an international student."

},

{

"question": "Where can I find more information about CPT, OPT, and SEVIS?",

"answer": "We recommend consulting the following resources for more detailed information on CPT, OPT, and SEVIS:\n\* CSTU University International Student Office website\n\* U.S. Department of Homeland Security Study in the States website (https://studyinthestates.dhs.gov/assets/SEVP\_InternationalStudentLifecycle\_2022-BW.pdf)."

},

{

"question": "What are the deadlines for the application?",

"answer": "The deadlines for the application varies with each term, Here are the Deadlines for the Undergraduation and Post-Graduation Summer term: Apirl 1st 2024,\n\* Fall term: June 1st 2024,\n\* Spring term: September 1st 2024."

}

],

"On\_Campus\_Jobs": [

{

"job\_title": "Teaching Assistant",

"description": "Supports professors with courses, leads discussions & grades assignments. Eligibility: Enrolled students with strong subject knowledge & excellent communication skills."

},

{

"job\_title": "Research Assistant",

"description": "Aids researchers in experiments, data collection & analysis. Eligibility: Students with research experience, strong analytical skills & attention to detail."

}

],

"locations": [

{

"name": "University Location",

"address": "California Science and Technology University (CSTU), 100 Innovation Way, Santa Clara, CA 95054",

"google\_maps\_link": "https://www.google.com/maps/place/California+Science+and+Technology+University+(CSTU)+100+Innovation+Way,+Santa+Clara,+CA+95054"

},

{

"name": "Medical Services or Hospitals",

"address": "Santa Clara Valley Medical Center, 751 S. Wells Fargo Dr, Santa Clara, CA 95054",

"google\_maps\_link": "https://www.google.com/maps/place/Santa+Clara+Valley+Medical+Center,+751+S+Wells+Fargo+Dr,+Santa+Clara,+CA+95054"

},

{

"name": "Police Services",

"address": "Santa Clara Police Department - Main Station, 150 Santa Clara Ave, Santa Clara, CA 95050",

"google\_maps\_link": "https://www.google.com/maps/place/Santa+Clara+Police+Department+-+Main+Station,+150+Santa+Clara+Ave,+Santa+Clara,+CA+95050"

}

]

}

*# Function to generate embeddings using OpenAI*

**def** get\_embeddings(texts):

response **=** openai**.**Embedding**.**create(

model**=**"text-embedding-ada-002", *# Use the appropriate embedding model*

input**=**texts

)

**return** [embedding["embedding"] **for** embedding **in** response['data']]

*# Function to upsert data into Pinecone index*

**def** upsert\_data(index):

**for** metadata\_type, data **in** json\_data**.**items():

vectors\_to\_upsert **=** []

**if** metadata\_type **==** "Course\_Details":

**for** doc **in** data:

text **=** doc['title'] **+** " " **+** doc['details']

doc\_id **=** doc['code']

vectors\_to\_upsert**.**append((doc\_id, text))

**elif** metadata\_type **==** "International\_Students\_faqs":

**for** doc **in** data:

text **=** doc['answer']

doc\_id **=** doc['question']

vectors\_to\_upsert**.**append((doc\_id, text))

**elif** metadata\_type **==** "On\_Campus\_Jobs":

**for** doc **in** data:

text **=** doc['description']

doc\_id **=** doc['job\_title']

vectors\_to\_upsert**.**append((doc\_id, text))

**elif** metadata\_type **==** "locations":

**for** item **in** data:

text **=** item['name'] **+** " " **+** item['address']

doc\_id **=** item['name']

vectors\_to\_upsert**.**append((doc\_id, text))

**else**:

**continue** *# Skip unrecognized metadata types*

*# Upsert vectors to Pinecone*

**for** doc\_id, text **in** vectors\_to\_upsert:

embedding **=** get\_embeddings([text])[0]

index**.**upsert([(doc\_id, embedding, {"type": metadata\_type, "text": text})])

print(f"Document '{doc\_id}' from {metadata\_type} has been upserted into Pinecone successfully.")

*# Call the function to upsert data*

upsert\_data(index)

Document 'MB/CSE 652' from Course\_Details has been upserted into Pinecone successfully.

Document 'MB/CSE 632' from Course\_Details has been upserted into Pinecone successfully.

Document 'MB/CSE 648' from Course\_Details has been upserted into Pinecone successfully.

Document 'MB/CSE 600' from Course\_Details has been upserted into Pinecone successfully.

Document 'MB/CSE 590' from Course\_Details has been upserted into Pinecone successfully.

Document 'MB/CSE 638' from Course\_Details has been upserted into Pinecone successfully.

Document 'What are the different types of fees for international students?' from International\_Students\_faqs has been upserted into Pinecone successfully.

Document 'What payment methods are available for international students?' from International\_Students\_faqs has been upserted into Pinecone successfully.

Document 'When are the deadlines for fee payments?' from International\_Students\_faqs has been upserted into Pinecone successfully.

Document 'What happens if I have questions about my student account or need assistance with payments?' from International\_Students\_faqs has been upserted into Pinecone successfully.

Document 'What is Curricular Practical Training (CPT)?' from International\_Students\_faqs has been upserted into Pinecone successfully.

Document 'What is Optional Practical Training (OPT)?' from International\_Students\_faqs has been upserted into Pinecone successfully.

Document 'What is SEVIS (Student and Exchange Visitor Information System)?' from International\_Students\_faqs has been upserted into Pinecone successfully.

Document 'Where can I find more information about CPT, OPT, and SEVIS?' from International\_Students\_faqs has been upserted into Pinecone successfully.

Document 'What are the deadlines for the application?' from International\_Students\_faqs has been upserted into Pinecone successfully.

Document 'Teaching Assistant' from On\_Campus\_Jobs has been upserted into Pinecone successfully.

Document 'Research Assistant' from On\_Campus\_Jobs has been upserted into Pinecone successfully.

Document 'University Location' from locations has been upserted into Pinecone successfully.

Document 'Medical Services or Hospitals' from locations has been upserted into Pinecone successfully.

Document 'Police Services' from locations has been upserted into Pinecone successfully.

In [9]:

initial\_chat\_context **=** [

{'role':'system', 'content': f"""

Objective: You are a smart, friendly virtual CSTU Chatbot Assistent, assisting students with their quires and assiting with knowing information on courses, International Students faqs, Locations, and Jobs.

Procedure:

1. Greet the student and inquire about their reason for contacting you.

2. If the student asks about fees, proceed with step 3.

3. Identify the student's program and semester to retrieve their specific fee structure.

4. Explain the fee structure clearly, including details like tuition, technology fees, and any additional program-specific fees.

5. Offer options for viewing a detailed breakdown of fees (e.g., link to university portal).

6. If the student wants to pay fees, guide them through the available payment methods (e.g., credit card, bank transfer).

7. For online payments, offer a secure and user-friendly interface within the LLM system (if possible) or direct the student to a secure university payment portal.

8. For other payment methods, provide clear instructions and relevant links (e.g., bank account details for transfer).

9. After successful payment, provide confirmation and offer a receipt for download or email delivery.

Task Steps:

1. When a student inquires about fees, ask for their program and semester to personalize the response.

2. Based on the provided information, retrieve the relevant fee structure from the university database.

3. Present the fee structure in a clear and concise format, highlighting key elements like total fees, due dates, and payment methods.

4. If the student chooses to pay online, guide them through the secure payment interface within the LLM (if available) or redirect them to the university's secure payment portal.

5. For offline payments, provide detailed instructions and relevant information (e.g., bank account details).

6. Once payment is processed, confirm the success with the student and offer a downloadable or emailed receipt.

The final step:

1. For online payments within the LLM system, ensure a secure and successful transaction with proper confirmation and receipt generation.

2. Regardless of the payment method, provide a clear overview of the student's remaining balance, if any.

3. Offer additional resources like FAQs or links to the university's financial aid office for further assistance.

[Fee structure:]

The fee structure includes:

- Technology fee: $50

- Students Association fee: $40

- Registration fee: $250

- Graduation fee(one time payment): $250

- Per Credit fee: $700

- note: Each Course has 1.5 Credit, Duration: one Term (2 months)

Procedure to search data in the Pinecone index (cstu-bot):

1. To retrieve course details, international student FAQs, locations, or jobs information, specify the metadata type (e.g., 'Course\_Details').

2. Use the Pinecone index 'cstu-bot' to query the relevant information based on the student's request.

3. Retrieve and present the requested data to the student.

"""},

]

In [10]:

*# Function to generate embeddings using OpenAI*

**def** get\_embeddings(texts):

**try**:

response **=** openai**.**Embedding**.**create(

model**=**"text-embedding-ada-002",

input**=**texts

)

**return** [embedding["embedding"] **for** embedding **in** response['data']]

**except** Exception **as** e:

print(f"Error generating embeddings: {e}")

**return** []

*# Function to handle the chatbot conversation*

**def** chat\_complete\_messages(messages, temperature**=**0.7):

**try**:

completion **=** openai**.**ChatCompletion**.**create(

model**=**"gpt-4o", *# Or use "gpt-4" if available for your use case*

messages**=**messages,

temperature**=**temperature

)

**return** completion**.**choices[0]**.**message['content']

**except** Exception **as** e:

print(f"Error completing chat message: {e}")

**return** "I'm sorry, I couldn't process your request at the moment."

*# Function to query Pinecone for relevant info*

**def** get\_relevant\_info(query, metadata\_type, top\_k**=**4):

query\_embedding **=** get\_embeddings([query])[0]

query\_result **=** index**.**query(

vector**=**query\_embedding,

top\_k**=**top\_k, *# Number of top results to return*

include\_metadata**=True**

)

*# Filter results to ensure they match the metadata\_type*

filtered\_results **=** [

match **for** match **in** query\_result['matches']

**if** match['metadata']['type'] **==** metadata\_type

]

**return** filtered\_results

*# Main chatbot loop (for interactive use)*

print("Welcome! I'm here to assist you with information about courses, International Students FAQs, locations, and jobs.")

*# Function to handle Gradio inputs and outputs*

**def** chatbot(user\_input, chatContext):

print(f"User Input: {user\_input}")

chatContext**.**append({"role": "user", "content": user\_input})

*# Determine the type of information the user is requesting*

metadata\_type **=** **None**

**if** any(keyword **in** user\_input**.**lower() **for** keyword **in** ["course", "class", "module", "study"]):

metadata\_type **=** "Course\_Details"

**elif** any(keyword **in** user\_input**.**lower() **for** keyword **in** ["faq", "fees", "payment", "international student"]):

metadata\_type **=** "International\_Students\_faqs"

**elif** any(keyword **in** user\_input**.**lower() **for** keyword **in** ["location", "place", "map", "address"]):

metadata\_type **=** "locations"

**elif** any(keyword **in** user\_input**.**lower() **for** keyword **in** ["job", "position", "work", "assistant"]):

metadata\_type **=** "On\_Campus\_Jobs"

**if** metadata\_type:

*# Query Pinecone for nearest neighbors based on user input*

query\_results **=** get\_relevant\_info(user\_input, metadata\_type)

**if** query\_results:

response\_message\_content **=** f"I found some relevant information about {metadata\_type}:\n"

**for** match **in** query\_results:

response\_message\_content **+=** f"- {match['metadata']['text']} (from {match['metadata']['type']})\n"

**else**:

response\_message\_content **=** f"I'm sorry, I couldn't find any relevant information about {metadata\_type}."

**else**:

*# Use OpenAI to generate responses if the type is not recognized*

response\_message\_content **=** chat\_complete\_messages(chatContext)

chatContext**.**append({"role": "assistant", "content": response\_message\_content})

print("ChatBot: ", response\_message\_content)

*# Ensure the updated chatContext is returned*

**return** response\_message\_content, chatContext

*# Gradio interface*

**with** gr**.**Blocks() **as** demo:

gr**.**Markdown("# CSTU Virtual Assistant")

**with** gr**.**Row():

**with** gr**.**Column():

chatbot\_input **=** gr**.**Textbox(lines**=**1, label**=**"Your question:", placeholder**=**"Ask anything about CSTU...")

chatbot\_output **=** gr**.**Textbox(lines**=**10, label**=**"Chatbot Response:")

chatbot\_history **=** gr**.**State(initial\_chat\_context) *# Initialize chat history with the system message*

*# Define interaction*

**def** respond(user\_input, chatContext):

response, updated\_history **=** chatbot(user\_input, chatContext)

*# Update the UI state with the latest chat history*

chatbot\_history**.**value **=** updated\_history

**return** response, updated\_history

chatbot\_input**.**submit(respond, inputs**=**[chatbot\_input, chatbot\_history], outputs**=**[chatbot\_output, chatbot\_history])

*# Run the Gradio interface*

demo**.**launch(share**=True**, inbrowser**=True**)

Welcome! I'm here to assist you with information about courses, International Students FAQs, locations, and jobs.

Running on local URL: http://127.0.0.1:7866

Could not create share link. Please check your internet connection or our status page: https://status.gradio.app.

Out[10]:

User Input: Hello

ChatBot: Hello! How can I assist you today? Are you looking for information on courses, fees, international student FAQs, locations, jobs, or something else?

User Input: What is the fee strcture

ChatBot: Sure, I can help you with that! Could you please let me know your program and the semester you're in? This will help me provide you with the most accurate fee structure.

User Input: 1 st sem

ChatBot: Great! Let's go over the fee structure for the first semester. Here is a breakdown of the fees you can expect:

1. \*\*Technology Fee:\*\* $50

2. \*\*Students Association Fee:\*\* $40

3. \*\*Registration Fee:\*\* $250

4. \*\*Graduation Fee (one-time payment):\*\* $250

5. \*\*Per Credit Fee:\*\* $700

6. \*\*Each Course has 1.5 Credits. The duration of one term is 2 months.\*\*

To give you an example, if you're enrolled in 4 courses for the first semester:

- \*\*Total Credits:\*\* 4 courses \* 1.5 credits/course = 6 credits

- \*\*Total Per Credit Fee:\*\* 6 credits \* $700/credit = $4200

- \*\*Total Fees:\*\*

- Technology Fee: $50

- Students Association Fee: $40

- Registration Fee: $250

- Graduation Fee: $250

- Per Credit Fee: $4200

\*\*Total:\*\* $4790

If you'd like a detailed breakdown or have any specific questions, please let me know!

Would you like information on how to pay these fees?

User Input: 2nd sem

ChatBot: Sure, let's go over the fee structure for your second semester. Here is a breakdown of the fees:

1. \*\*Technology Fee:\*\* $50

2. \*\*Students Association Fee:\*\* $40

3. \*\*Registration Fee:\*\* $250

4. \*\*Per Credit Fee:\*\* $700

Since you only pay the Graduation Fee once, it won't be included in the second semester.

To give you an example, if you're enrolled in 4 courses for the second semester:

- \*\*Total Credits:\*\* 4 courses \* 1.5 credits/course = 6 credits

- \*\*Total Per Credit Fee:\*\* 6 credits \* $700/credit = $4200

- \*\*Total Fees:\*\*

- Technology Fee: $50

- Students Association Fee: $40

- Registration Fee: $250

- Per Credit Fee: $4200

\*\*Total:\*\* $4540

If you have any specific questions or need a detailed breakdown, feel free to ask!

Would you like information on how to pay these fees?

User Input: University Location

ChatBot: I found some relevant information about locations:

- University Location California Science and Technology University (CSTU), 100 Innovation Way, Santa Clara, CA 95054 (from locations)

User Input: Opt

ChatBot: It looks like you want to opt for a specific service or information. Could you please clarify what you would like to opt for? Are you interested in courses, fees, international student FAQs, locations, jobs, or something else?

User Input: what is opt

ChatBot: OPT stands for Optional Practical Training. It is a program that allows international students on F-1 visas to work in the United States in a job related to their field of study. Here are some key points about OPT:

1. \*\*Duration:\*\*

- \*\*Pre-completion OPT:\*\* You can apply to work part-time (20 hours or less per week) while school is in session and full-time during breaks.

- \*\*Post-completion OPT:\*\* You can work full-time after completing your degree program. The maximum duration for post-completion OPT is 12 months.

2. \*\*STEM Extension:\*\*

- If you have a degree in a STEM field (Science, Technology, Engineering, Mathematics), you may be eligible for a 24-month extension of your post-completion OPT, for a total of 36 months.

3. \*\*Eligibility:\*\*

- You must have been enrolled in a full-time degree program for at least one academic year.

- You must be in F-1 status at the time of application.

- The work must be directly related to your major field of study.

4. \*\*Application Process:\*\*

- You need to apply for an Employment Authorization Document (EAD) from the U.S. Citizenship and Immigration Services (USCIS).

- You should apply 90 days before you complete your degree but no later than 60 days after completing your degree.

- You need a recommendation from your Designated School Official (DSO).

5. \*\*Cap-Gap Extension:\*\*

- If you have a pending or approved H-1B petition, you may be eligible for a cap-gap extension, which allows you to continue working until your H-1B status begins.

Would you like more detailed information on any specific aspect of OPT?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_