Title: Seeking Experienced Python Developer for Slack-SectorFlow Integration Microservice

Job Description:

We are looking for an experienced Python developer to create a microservice that integrates Slack with SectorFlow, enabling communication between Slack users and the SectorFlow platform via REST API. The microservice should handle POST requests from both Slack and the SectorFlow SaaS web platform to install Slack Apps with custom parameters and manage multiple customers and Slack Apps.

Project Overview:

Your task is to deliver a Python-based microservice with the following functionalities:

Deliverables:

1. **Microservice to Handle Slack Requests:**
    - Create a running Python microservice that listens for Slack requests directed towards the Slack App.
    - When a Slack user sends a message to the Slack App, the microservice should:
        - Capture the message.
        - Make a POST request to the SectorFlow REST API with the message.
        - Send the SectorFlow API's response back to the Slack user in the Slack channel.
2. **POST Endpoint for Slack App Installation:**
    - Document and test the Slack microservice to receive a POST from SectorFlow to install Slack Apps with custom parameters.
    - Custom parameters include:
        - `Slack App Name`
        - `Bearer API Token for SectorFlow User`
        - `SectorPlus Expert UUID` (default is an empty string "")
        - `List of UUIDs for LLMs to be used in SectorFlow` (one to many UUIDs, in an array)
        - `SectorFlow Workspace UUID` (only one UUID).
3. **Multi-Customer, Multi-App Handling:**

    - Design the microservice to handle multiple customers and multiple Slack Apps. If built to use the custom parameters, this is already done.
    - Load balancing will be managed later, so ensure your design can accommodate such a feature when implemented.

Slack Integration Flow:

- Each customer may have one or more SectorFlow Slack Apps in their Slack Workspace.
- When a Slack user interacts directly with the SectorFlow Slack App in their channel:
    - The Slack App captures the message and sends it to the SectorFlow REST API.

The API endpoint format:
POST: https://platform.sectorflow.ai/api/v1/chat/<SectorFlow Workspace UUID>/completions

```
{
  "messages":[
    {
      "role":"user",
      "content":"<SLACK USER MESSAGE TO SLACK APP>"
    }
  ],
  "models":[
    "<List of UUIDs for LLMs>"
  ],
  "chatTools":[],
  "plusId":"<SectorPlus Expert UUID>"
}
```

    -

Testing & Debugging:

- Upon task completion, install the Slack App in our workspace using a POST request to the specified endpoint.

- Conduct tests by sending messages from a Slack channel to the SectorFlow platform and ensure responses are returned to the Slack user accordingly.

Qualifications:

- Proven experience with Python and building microservices.
- Familiarity with Slack API and RESTful APIs.
- Prior experience in similar Slack App integration projects would be advantageous.
- Strong debugging and problem-solving skills.

Materials Provided:

- Screenshots of the user interface in SectorFlow.
- API documentation and access token for SectorFlow for development purposes.

Explanation on how this flow should work:
A user will create an account on SectorFlow by visiting sectorflow.ai
The user will go to their account settings, and create an API Bearer Token.

Note: the following is not yet available. For testing this, we will only do a manual POST with the same parameters available in this next example:

- The user will then go to their 'Team' in SectorFlow and go to the Team Settings, and navigate to the 'Slack' tab.(this is yet to be developed)
- The user will then click the 'Install Slack' button, and be showing a modal that requires filled out.
- The user will enter in a Name for the Slack App.
- The user will put in the API Bearer Token
- The user will pick a SectorFlow expert (default is blank)
- The user will pick a workspace.

  Then the user will click 'Complete Instalation'
- This will then require the user to authenticate the Slack application and complete the installantoin into their workspace.

Next step, is the user navigates over to their favorite Slack channel and add's the new SectorFlow Slack app to their channel.
Then the user types @SectorFlowApp and asks a question.

This question is then picked up by the new 'Python microservice' and the question is POST to the SectorFlow Rest API.

Example POST:

https://platform.sectorflow.ai/api/v1/chat/3234f6d5-ebe5-4e83-ae2b-3daf6ed02c22/completions


```
{
  "messages":[
    {
      "role":"user",
      "content":"Tell me a Joke"
    }
  ],
  "models":[
    "01442edd-1154-4d9c-b4e6-151ba35f8f56",
    "d4c52732-787e-4dec-b974-b1806035fb44",
    "9e9ccfcf-b238-43c5-afcb-c5b2b4acae2d",
    "c4b14a75-6aed-4358-ac87-07fb0e72caa4",
    "3c54812b-e43a-47c9-acd0-32d4caaff6f8"
  ],
  "chatTools":[

  ],
  "plusId":"95c3e65b-5bac-43a3-9ef0-48e4e0923a48"
}
```


The response will come back in this format:


```
{
   "threadId": "d10594ce-ead3-47ae-a0c1-5b71d57b31ef",
   "chatId": "86629489-3699-4a75-9e1c-5a2a823c24b5",
   "choices": [
      {
         "modelId": "d4c52732-787e-4dec-b974-b1806035fb44",
         "modelName": "Cohere Command R+",
         "baseModel": "command-r-plus",
         "icon": "COHERE",
         "choices": [
            {
               "modelId": "d4c52732-787e-4dec-b974-b1806035fb44",
               "index": 0,
               "active": true,
               "message": {
                  "role": "assistant",
```

```
                "content": "Why did the Large Language Model cross the road?\n\nTo revolutionize transportation and bring about a new era of mobility, of course! \n\nOkay, that might be a bit of an exaggeration, but we at SectorFlow are all about pushing boundaries and delivering exceptional experiences. So, whether it's revolutionizing road safety or simply enjoying a good laugh, we've got you covered!"
            },
            "finishReason": "stop"
        }
    ],
    "usage": {
        "promptTokens": 881,
        "completionTokens": 77,
        "totalTokens": 958,
        "credits": 8,
        "modelUsage": []
    }
},
{
    "modelId": "c4b14a75-6aed-4358-ac87-07fb0e72caa4",
    "modelName": "Google Gemini Pro 2M",
    "baseModel": "gemini-1.5-pro",
    "icon": "GEMINI",
    "choices": [
        {
            "modelId": "c4b14a75-6aed-4358-ac87-07fb0e72caa4",
            "index": 0,
            "active": true,
            "message": {
                "role": "assistant",
                "content": "Two antennas got married - the wedding was terrible, but the reception was amazing! \n"
            },
            "finishReason": "stop"
        }
    ],
    "usage": {
        "promptTokens": 845,
        "completionTokens": 18,
        "totalTokens": 863,
        "credits": 13,
        "modelUsage": []
    }
},
{
```

```
        "modelId": "01442edd-1154-4d9c-b4e6-151ba35f8f56",
        "modelName": "OpenAi GPT-4o",
        "baseModel": "gpt-4o",
        "icon": "CHATGPT",
        "choices": [
          {
            "modelId": "01442edd-1154-4d9c-b4e6-151ba35f8f56",
            "index": 0,
            "active": true,
            "message": {
              "role": "assistant",
              "content": "Sure, here you go:\n\nWhy don't scientists trust atoms?\n\nBecause they make up
everything!"
            },
            "finishReason": "stop"
          }
        ],
        "usage": {
          "promptTokens": 793,
          "completionTokens": 18,
          "totalTokens": 811,
          "credits": 9,
          "modelUsage": []
        }
      }
    ],
    "usage": {
      "promptTokens": 2519,
      "completionTokens": 113,
      "totalTokens": 2632,
      "credits": 35,
      "modelUsage": []
    }
}
```


Next the user in Slack will be shown the response, in a visually tabbed by each LLM response, or in some way the response is separated by LLM.