Unbound Coding Challenge

Welcome to the **Unbound Hackathon Challenge**! In this challenge, you'll build a **lightweight chat UI** integrated with a **secure backend server** that routes user prompts to various **simulated language model providers**.

Your solution should showcase a **modular application** with **clear integration** between the chat interface and dynamic routing logic.

Technical Guidelines

Backend

- Use a **framework of your choice** to implement endpoints that:
  - Handle **chat completions**
  - List **supported models**
  - Apply **custom regex-based routing policies**

Database

- Use a **database of your choice** to store the data.

Frontend

- Use a framework of your choice to develop a **minimal chat UI** and an **admin portal** to manage regex policies and file upload routing.

Milestones

Milestone 1: Models Endpoint

Endpoint: `GET /models`

Behavior:

Fetch and return a list of supported models from the database.

Example Response:

```
[
  "openai/gpt-3.5",
  "anthropic/claude-v1",
  "gemini/gemini-alpha"
]
```

Milestone 2: Chat Completions Endpoint

Endpoint: `POST /v1/chat/completions`

Request Body:

```
{
  "provider": "openai",
  "model": "gpt-3.5",
  "prompt": "Hello world!"
}
```

Behavior:

1. Validate the **provider** and **model** against the list of models in your DB.
2. **Route the request** to the corresponding **stub LLM** based on the specified provider and model. You do not need to call real provider APIs; instead, return predefined static responses for each provider. Ensure modular implementation where each provider has a separate logic component that generates its unique dummy response string.

Example Responses:

✅ **OpenAI Provider:**

{

"provider": "openai",

"model": "gpt-3.5",

"response": "OpenAI: Processed your prompt with advanced language understanding. Response ID: openai_response_001"

}

✅ **Anthropic Provider:**

{

"provider": "anthropic",

"model": "claude-v1",

"response": "Anthropic: Your prompt has been interpreted with ethical AI principles. Response ID: anthropic_response_002"

}

Milestone 3: Prompt Interference & Regex-Based Routing Policy

Requirement:

- Store **regex rules** in **PostgreSQL** along with an **associated routing policy** (original model, regex pattern, and redirect model).
- Before responding to a prompt, check if it **matches any regex patterns** from the database.

Behavior:

If a user's prompt matches a **stored regex pattern** for a specific model, reroute the request to the **configured redirect model**.

Example:

Rule

Regex	Model	Redirect Model
`(?i)(credit card)`	`gpt-4o`	`gemini-alpha`

✅ **User Prompt:**
`"I lost my credit card!"`

If the request specifies provider `"openai"` and model `"gpt-4o"`, the system should **reroute it** to `"gemini-alpha"`.

Milestone 4: Simple Chat UI

Requirement:

Build a **minimal web-based chat UI** that:

- Fetches and lists **available models** via the `/models` endpoint.
- Allows users to select a **provider and model** from dropdown menus.
- Accepts a **prompt input**.
- Sends a request to the `POST /v1/chat/completions` endpoint.
- Displays the **response**.

Milestone 5: Admin Portal for Regex Policies

Requirement:

Develop an **admin interface** (as a separate web page or under a different path) that allows an **admin** to:
✅ **Add** new regex rules.
✅ **Edit or delete** existing regex rules.
✅ **Associate** each regex rule with a **specific redirect model and original model**.
✅ Ensure changes **update the routing policies** in PostgreSQL accordingly.

Milestone 6: File Upload Support in Chat Portal

Requirement:

Enhance the chat UI to allow **file uploads** (e.g., PDFs).

Behavior:

The response for a file upload from the backend server should include an **additional message** indicating **file processed**.

Milestone 7: Special Routing for File Uploads

Requirement:

- Add **configuration options** in the database (bonus points if it is from the **admin portal**) to set a **special routing policy** for file uploads.
- If a user **uploads a file**, the system should **route the request** to a **designated provider/model** for file processing.

Example:

✅ **Admin Configuration:**

File Type	Redirect Provider	Redirect Model
PDF Upload	`anthropic`	`claude-v1`

✅ **Response on File Upload:**

{

"provider": "anthropic",

"model": "claude-v1",

"response": "Anthropic: File processed with secure file analysis. Response ID: anthropic_file_response_004"

}

Evaluation Criteria

✅ **Functionality**:
Does your solution meet the **core objectives** and **milestones**?

✅ **Code Quality**:
Is your code **modular, easy to read and well-documented**?

✅ **Innovation**:
Extra points for **creative solutions** or **additional features** beyond the milestones.

✅ **Documentation**:
Provide a **detailed** `README.md` including:

- **Setup instructions**
- **Design choices and architecture**
- **Usage details**
- **A demo video** showcasing your project in action

Submission Guidelines

✅ **GitHub Repository**

- Create a **private GitHub repository** and invite `vigneshsubbiah16` as a collaborator.

✅ **Commits**

- Use **atomic commits** with **meaningful messages**.

✅ **Questions?**

- Please email **vis@unboundsecurity.ai** if you have any questions.
