

# lessoncode5

This Python script creates a **web-based chatbot** that allows users to select from multiple LLM models using a dropdown menu. The app uses **Gradio** for the web interface and **Ollama CLI** to generate AI responses based on the chosen model. It gives you flexibility by supporting various language models within one interface.

## Code Explanation

### 1 Importing Required Modules

```
import gradio as gr
import subprocess
```

- **gradio** : Used to build the interactive web UI.
- **subprocess** : Allows running system commands (to call the Ollama CLI).

### 2 Function to Run the Selected LLM

```
def run_ollama_prompt(prompt, model):
    """Runs a prompt using Ollama CLI and returns the response."""
    command = ["ollama", "run", model, prompt]

    try:
        result = subprocess.run(command, capture_output=True, text=True, c
heck=True, encoding="utf-8")
        return result.stdout.strip()
    except subprocess.CalledProcessError as e:
        return f"Error running Ollama: {e}"
```

#### Key Points:

- **Dynamic model selection:** The function accepts a **model** parameter, allowing different LLMs to be used.
- **Execution:** It runs the **ollama** command with the provided model and prompt.

- **Error handling:** If something goes wrong, it returns an error message.

### 3 Linking the Function to the Web App

```
def generate_response(prompt, model):  
    return run_ollama_prompt(prompt, model)
```

- **Purpose:** Connects the user's input (prompt and selected model) with the function that interacts with Ollama.

### 4 Creating the Gradio Interface

```
with gr.Blocks() as app:  
    gr.Markdown("# Multi-LLM Chat App")  
  
    with gr.Row():  
        prompt_input = gr.Textbox(label="Enter your prompt", placeholder="Type something...")  
        model_selector = gr.Dropdown(label="Select LLM Model", choices=["deepseek-r1:1.5b", "model2", "model3"], value="deepseek-r1:1.5b")  
  
    with gr.Row():  
        submit_button = gr.Button("Generate Response")  
  
    response_output = gr.Textbox(label="Response", interactive=False)  
  
    submit_button.click(fn=generate_response, inputs=[prompt_input, model_selector], outputs=response_output)
```

#### What This Does:

- **Title and Layout:** Uses `gr.Markdown` to display a title and organizes the UI into rows.
- **Input Elements:**
  - A **textbox** for the prompt.
  - A **dropdown menu** (`model_selector`) for selecting the LLM model from a list.
- **Output:** Displays the AI-generated response in a non-editable textbox.

- **Button Interaction:** Clicking the "**Generate Response**" button triggers the `generate_response` function with both the prompt and the selected model.
- 

## **Launching the Web App**

```
app.launch()
```

- **Starts the Gradio app** on a local server, allowing you to interact with it via a web browser.
- 

## **How to Run the App**

1. **Install dependencies** (if needed):

```
pip install gradio
```

2. **Run the script:**

```
python multiLLMapp.py
```

3. **Open the provided URL** in your web browser and start chatting by entering a prompt and selecting a model.
-