**Project Documentation: FastAPI Server Execution in Google Colab**

**1. Project Overview**

The purpose of this project is to deploy and interact with a **FastAPI** server inside a Google Colab environment, execute HTTP requests against it, and receive structured JSON responses. The server exposes a /brief POST endpoint that processes a request payload containing topic details and returns a structured brief.

**2. System Design**

**2.1 High-Level Architecture**

pgsql

[Google Colab Notebook]

|

| (start server using uvicorn)

v

[FastAPI Application] <-- uvicorn -->

|

| (POST request to /brief)

v

[Endpoint Logic] --> process request --> return JSON

**2.2 Components**

**a) Server Start & Process Management**

* The script kills any existing uvicorn processes to prevent port conflicts.
* Launches the FastAPI server in a subprocess.

**b) API Application Structure**

* app/api.py contains the main FastAPI app and endpoint definitions.
* api.py can be used instead of the folder structure for simpler setups.

**c) Endpoint: /brief**

* **Method:** POST
* **Request Body:**

json

{

"topic": "string",

"depth": int,

"follow\_up": bool,

"user\_id": "string"

}

* **Response:** JSON object containing the processed data.

**3. File & Folder Structure**

bash

project/

│

├── app/

│ ├── \_\_init\_\_.py

│ └── api.py # FastAPI app definition

│

├── main\_colab.py # Script to manage server start & API calls

└── requirements.txt # Dependencies

**4. Detailed Execution Flow**

**Step 1: Kill Previous Servers**

python

for p in psutil.process\_iter(attrs=["pid","name","cmdline"]):

if p.info["cmdline"] and "uvicorn" in " ".join(p.info["cmdline"]):

os.kill(p.info["pid"], signal.SIGKILL)

* **Purpose:** Avoid multiple uvicorn instances using the same port.

**Step 2: Start the Server**

python

server = subprocess.Popen(

["uvicorn", "app.api:app", "--host", "0.0.0.0", "--port", "8000"],

cwd=str(BASE),

stdout=subprocess.PIPE, stderr=subprocess.STDOUT, text=True

)

time.sleep(3) # Give server time to start

* **Command:** uvicorn app.api:app
* **Meaning:**
  + app.api → Path to the api.py file inside app folder
  + app → FastAPI app instance inside api.py

**Step 3: Send Request to API**

python

payload = {

"topic": "LLM evaluation methods in production",

"depth": 2,

"follow\_up": False,

"user\_id": "u1"

}

r = requests.post("http://127.0.0.1:8000/brief", json=payload, timeout=180)

* **Requests Library** sends an HTTP POST to the local server.
* timeout ensures the request does not hang indefinitely.

**Step 4: API Processing**

Inside app/api.py:

python

from fastapi import FastAPI

from pydantic import BaseModel

app = FastAPI()

class BriefRequest(BaseModel):

topic: str

depth: int

follow\_up: bool

user\_id: str

@app.post("/brief")

async def brief(request: BriefRequest):

return {

"message": "Brief generated successfully!",

"data": request.dict()

}

* **Validation:** pydantic ensures incoming JSON matches the model.
* **Response:** Returns the request data along with a success message.

**5. Troubleshooting & Common Errors**

**Error: ModuleNotFoundError: No module named 'app'**

* Cause: Missing app folder or api.py file.
* Fix: Create the folder and file as described above.

**Error: ConnectionRefusedError**

* Cause: Server failed to start before the request.
* Fix: Increase time.sleep(3) or check server logs.

**6. Execution in Colab (Cell-by-Cell)**

**Cell 1: Install dependencies**

python

!pip install fastapi uvicorn nest\_asyncio psutil requests

**Cell 2: Create API**

python

!mkdir -p app

!touch app/\_\_init\_\_.py

%%writefile app/api.py

from fastapi import FastAPI

from pydantic import BaseModel

app = FastAPI()

class BriefRequest(BaseModel):

topic: str

depth: int

follow\_up: bool

user\_id: str

@app.post("/brief")

async def brief(request: BriefRequest):

return {

"message": "Brief generated successfully!",

"data": request.dict()

}

**Cell 3: Start Server**

python

import subprocess, time, requests, json, os, signal, psutil, sys, nest\_asyncio

nest\_asyncio.apply()

# Kill previous uvicorn processes

for p in psutil.process\_iter(attrs=["pid","name","cmdline"]):

try:

if p.info["cmdline"] and "uvicorn" in " ".join(p.info["cmdline"]):

os.kill(p.info["pid"], signal.SIGKILL)

except Exception:

pass

server = subprocess.Popen(

["uvicorn", "app.api:app", "--host", "0.0.0.0", "--port", "8000"],

stdout=subprocess.PIPE, stderr=subprocess.STDOUT, text=True

)

time.sleep(3)

**Cell 4: Send Request**

python

payload = {"topic": "LLM evaluation methods in production", "depth": 2, "follow\_up": False, "user\_id": "u1"}

r = requests.post("http://127.0.0.1:8000/brief", json=payload, timeout=180)

print("Status:", r.status\_code)

print(json.dumps(r.json(), indent=2))

**7. Possible Enhancements**

* **Add logging** to capture server output for debugging.
* **Switch to Gradio** for a Colab-friendly interface.
* **Add async processing** for long-running tasks.
* **Enable OpenAPI docs** (/docs) for API testing.
* **Deploy externally** via ngrok for public access.