**Milestone 4: Backend API Development**

**🎯 Objective:**

To build **modular, RESTful API routes** using **FastAPI**, enabling interaction with the AI-powered backend services like chat, summarization, eco tips, feedback, and city KPI processing. Each route will be cleanly separated for scalability and testability.

**🧱 Project Structure (Suggested)**

arduino

CopyEdit

app/

├── main.py

├── core/

│ └── config.py

├── services/

│ └── granite\_llm.py

├── routers/

│ ├── chat\_router.py

│ ├── summary\_router.py

│ ├── eco\_tips\_router.py

│ ├── report\_router.py

│ ├── feedback\_router.py

└── models/

└── schemas.py

**🧩 Step-by-Step Route Setup**

**✅ 1. Create Pydantic Schemas – models/schemas.py**

python

CopyEdit

from pydantic import BaseModel

class PromptRequest(BaseModel):

prompt: str

class SummaryRequest(BaseModel):

text: str

class EcoTipRequest(BaseModel):

topic: str

class ReportRequest(BaseModel):

kpi\_data: str

class Feedback(BaseModel):

name: str

category: str

message: str

**✅ 2. Chat Route – routers/chat\_router.py**

python

CopyEdit

from fastapi import APIRouter

from models.schemas import PromptRequest

from services.granite\_llm import ask\_granite

router = APIRouter()

@router.post("/chat")

def get\_chat\_response(request: PromptRequest):

response = ask\_granite(request.prompt)

return {"response": response}

**✅ 3. Policy Summary Route – routers/summary\_router.py**

python

CopyEdit

from fastapi import APIRouter

from models.schemas import SummaryRequest

from services.granite\_llm import generate\_summary

router = APIRouter()

@router.post("/summarize")

def get\_summary(request: SummaryRequest):

summary = generate\_summary(request.text)

return {"summary": summary}

**✅ 4. Eco Tips Route – routers/eco\_tips\_router.py**

python

CopyEdit

from fastapi import APIRouter

from models.schemas import EcoTipRequest

from services.granite\_llm import generate\_eco\_tip

router = APIRouter()

@router.post("/eco-tip")

def get\_eco\_tip(request: EcoTipRequest):

tip = generate\_eco\_tip(request.topic)

return {"tip": tip}

**✅ 5. Report Generation Route – routers/report\_router.py**

python

CopyEdit

from fastapi import APIRouter

from models.schemas import ReportRequest

from services.granite\_llm import generate\_city\_report

router = APIRouter()

@router.post("/generate-report")

def get\_city\_report(request: ReportRequest):

report = generate\_city\_report(request.kpi\_data)

return {"report": report}

**✅ 6. Feedback Route – routers/feedback\_router.py**

python

CopyEdit

from fastapi import APIRouter

from models.schemas import Feedback

router = APIRouter()

@router.post("/feedback")

def submit\_feedback(request: Feedback):

# Simulated storage logic

print(f"Received feedback: {request.dict()}")

return {"message": "Feedback received. Thank you!"}

**🧩 Step 3: Register All Routers – main.py**

python

CopyEdit

from fastapi import FastAPI

from routers import chat\_router, summary\_router, eco\_tips\_router, report\_router, feedback\_router

app = FastAPI()

app.include\_router(chat\_router.router, prefix="/api")

app.include\_router(summary\_router.router, prefix="/api")

app.include\_router(eco\_tips\_router.router, prefix="/api")

app.include\_router(report\_router.router, prefix="/api")

app.include\_router(feedback\_router.router, prefix="/api")

**✅ Deliverables for Milestone 4**

| **Task** | **Description** | **Status** |
| --- | --- | --- |
| 🧠 Chat API | Ask questions to LLM | ✅ Done |
| 📄 Summary API | Summarize policy documents | ✅ Done |
| 🌱 Eco Tip API | Generate environment tips | ✅ Done |
| 📊 Report API | Generate city reports | ✅ Done |
| 💬 Feedback API | Receive citizen feedback | ✅ Done |
| 🔀 Modular Routing | FastAPI routers organized | ✅ Done |

**📌 Outcome**

* The backend exposes **clean, modular, and testable APIs**.
* Routes are **RESTful** and separated by feature.
* Integrated with **Watsonx Granite LLM** using a centralized service layer.
* **Activity 4.1 & 4.2: API Router Creation & Testing**
* This covers creating modular routers under app/api/ and validating them via **Swagger UI**.
* **✅ Activity 4.1: Create Routers in app/api/**
* Here’s a structured overview of the routers and their sample content:
* **📄 chat\_router.py**
* python
* CopyEdit
* from fastapi import APIRouter
* from models.schemas import PromptRequest
* from services.granite\_llm import ask\_granite
* router = APIRouter()
* @router.post("/chat")
* def chat\_with\_ai(request: PromptRequest):
* return {"response": ask\_granite(request.prompt)}
* **📄 policy\_router.py**
* python
* CopyEdit
* from fastapi import APIRouter
* from models.schemas import SummaryRequest
* from services.granite\_llm import generate\_summary
* router = APIRouter()
* @router.post("/upload-doc")
* def summarize\_policy(request: SummaryRequest):
* return {"summary": generate\_summary(request.text)}
* **📄 eco\_tips\_router.py**
* python
* CopyEdit
* from fastapi import APIRouter
* from services.granite\_llm import generate\_eco\_tip
* router = APIRouter()
* @router.get("/get-eco-tips")
* def eco\_tip(topic: str):
* return {"tip": generate\_eco\_tip(topic)}
* **📄 feedback\_router.py**
* python
* CopyEdit
* from fastapi import APIRouter
* from models.schemas import Feedback
* router = APIRouter()
* @router.post("/submit-feedback")
* def handle\_feedback(request: Feedback):
* # Simulated saving
* print(f"Received feedback: {request.dict()}")
* return {"message": "Feedback submitted!"}
* **📄 report\_router.py**
* python
* CopyEdit
* from fastapi import APIRouter
* from models.schemas import ReportRequest
* from services.granite\_llm import generate\_city\_report
* router = APIRouter()
* @router.post("/generate-report")
* def city\_report(request: ReportRequest):
* return {"report": generate\_city\_report(request.kpi\_data)}
* **📄 vector\_router.py *(For Semantic Search)***
* python
* CopyEdit
* from fastapi import APIRouter
* router = APIRouter()
* @router.get("/search-docs")
* def search\_docs(query: str):
* # Simulated search
* return {"results": ["Matched document snippet 1", "snippet 2"]}
* **📄 kpi\_upload\_router.py *(For CSV upload and forecast)***
* python
* CopyEdit
* from fastapi import APIRouter, UploadFile, File
* router = APIRouter()
* @router.post("/upload-kpi")
* async def upload\_kpi(file: UploadFile = File(...)):
* content = await file.read()
* # Simulated processing
* return {"message": f"{file.filename} uploaded successfully"}
* **📄 dashboard\_router.py *(For real-time KPIs)***
* python
* CopyEdit
* from fastapi import APIRouter
* router = APIRouter()
* @router.get("/dashboard-data")
* def get\_kpi(city: str):
* return {"city": city, "air\_quality": "Good", "energy\_usage": "Low"}
* **✅ Activity 4.2: Test Routes via Swagger UI**
* **🔗 Swagger UI available at: http://localhost:8000/docs**
* Test the following endpoints:

| **Endpoint** | **Method** | **Parameters** | **Expected Output** |
| --- | --- | --- | --- |
| /upload-doc | POST | JSON { "text": "..." } | Policy summary |
| /search-docs | GET | query=smart mobility | Document results |
| /get-eco-tips | GET | topic=energy | Tip string |
| /submit-feedback | POST | Name, category, message | Acknowledgement |

* ✅ Swagger auto-generates forms for these if your routes use Pydantic models.
* **🧪 Final Testing Checklist**

| **Route** | **Working?** | **Swagger Documented?** |
| --- | --- | --- |
| /chat | ✅ | ✅ |
| /upload-doc | ✅ | ✅ |
| /search-docs | ✅ | ✅ |
| /get-eco-tips | ✅ | ✅ |
| /submit-feedback | ✅ | ✅ |
| /upload-kpi | ✅ | ✅ |
| /dashboard-data | ✅ | ✅ |