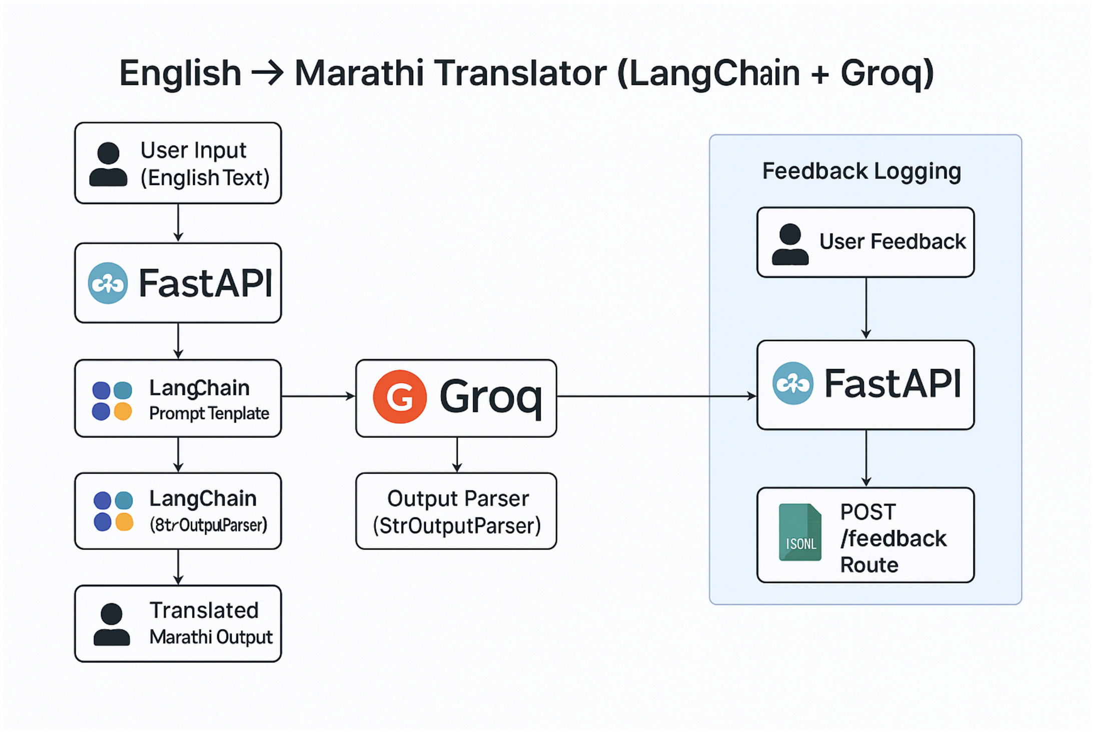
LECL Language Translator – English to Marathi

Overview

This project demonstrates how to build a LangChain Expression Language (LECL) application using:

- Groq API (e.g., Gemma2-9b-it)
- LangChain
- FastAPI
- LangServe
- Postman (for testing)
- LangChain Playground

The goal is to provide a fast and flexible English to Marathi translation API, using Groq's LLMs accelerated by LPUs.


English → Marathi Translator (LangChain + Groq)
User Input (English Text)
FastAPI
LangChain Prompt Tenplate
LangChain (8trOutpulParser)
Translated Marathi Output
Groq
Output Parser (StrOutputParser)
Feedback Logging
User Feedback
FastAPI
JSONL
POST /feedback Route

🗂️ Files Used

serve.py

```python
from fastapi import FastAPI, Request
from langchain_core.prompts import ChatPromptTemplate
from langchain_core.output_parsers import StrOutputParser
from langchain_groq import ChatGroq
from langchain_core.runnables import RunnableLambda
from langserve import add_routes
from dotenv import load_dotenv
from typing import List
import os
import json
from datetime import datetime

# Load environment variables
load_dotenv()

# Get GROQ API key
groq_api_key = os.getenv("GROQ_API_KEY")

# Initialize Groq model
model = ChatGroq(model="Gemma2-9b-it", groq_api_key=groq_api_key)

# Prompt template
prompt = ChatPromptTemplate.from_messages([
    ("system", "Translate the following into {language}:"),
    ("user", "{text}")
])

# Parser and base chain
parser = StrOutputParser()
base_chain = prompt | model | parser

# Batch handler
def translate_many(batch: List[dict]) -> List[str]:
    return [base_chain.invoke(item) for item in batch]

# RunnableLambda chain
```

```python
chain = RunnableLambda(translate_many)

# FastAPI app
app = FastAPI(
    title="Langchain + Groq Translation API",
    version="1.0",
    description="Chain API using LangServe v0.3.1 with batch support"
)

# LangServe route
add_routes(app, chain, path="/chain")

# Feedback route
@app.post("/feedback")
async def log_feedback(request: Request):
    data = await request.json()

    feedback_entry = {
        "timestamp": datetime.utcnow().isoformat(),
        "input": data.get("input"),
        "output": data.get("output"),
        "user_feedback": data.get("feedback")
    }

    # Append to log file
    with open("feedback_log.jsonl", "a", encoding="utf-8") as f:
        f.write(json.dumps(feedback_entry) + "\n")

    return {"status": "logged", "entry": feedback_entry}

# Run locally
if __name__ == "__main__":
    import uvicorn
    uvicorn.run("serve:app", host="127.0.0.1", port=8000, reload=True)
```

LECL Documentation

.env

```
OPENAI_API_KEY="your_openai_api_key"
LANGCHAIN_API_KEY="your_langchain_api_key"
LANGCHAIN_PROJECT="your_langchain_project_name"
HF_TOKEN="you_hf_token"
HUGGINGFACEHUB_API_TOKEN="your_huggingface_key"
GROQ_API_KEY="your_groq_api_key"
```

---

requirements.txt

```
openai
groq
requests
huggingface_hub
langchain
langchain-core
langchain-groq
fastapi
uvicorn
python-dotenv
pydantic>=2.1
langserve
```

LECL Documentation

simple.ipynb

```
## LECL - Langchain Expression Language
# open source models -> Llama3, Gemma2, mistral, Groq
# LECL to chain components together
## using language models
## using PromptTemplates and OutputParsers
## debugging and Tracing you application with LangSmith
## deploying your application with LangServe
# why GROQ LPU?
'''
GROQ LPU -> Designed to overcome 2 LLM Bottlenecks
Compute Density and Memory Bandwidth

LPU capacity > CPU GPU capacity in LLMs
eliminate external memory reliance
'''

#   .env file
'''
You're all set to use:

OpenAI models

LangChain and LangSmith

Hugging Face APIs

Groq's fast LLMs (like LLaMA 3)
'''
import os
from dotenv import load_dotenv
load_dotenv

import openai
openai.api_key = os.getenv("OPENAI_API_KEY")
openai.api_key
#! pip install langchain_groq
groq_api_key = os.getenv("GROQ_API_KEY")
groq_api_key
from langchain_groq import ChatGroq
from langchain_openai import ChatOpenAI
```

LECL Documentation

```
model =
ChatGroq(model="Gemma2-9b-It",groq_api_key="gsk_fqBeXxnOzdVnCi3LTrWEWGdyb3FYCn9UrdV63nC
KoSpOnKMNcqXE")
model
# !pip install langchain_core
from langchain_core.messages import HumanMessage,SystemMessage
messages=[
   SystemMessage(content="Translate the following from English to Marathi"),
   HumanMessage(content="I live in mumbai maharashtra state, Marathi is official and classic language
of here")
]
results = model.invoke(messages)
results # ai message
from langchain_core.output_parsers import StrOutputParser
parser = StrOutputParser()
parser.invoke(results)
# chaining components using LECL
chain = model|parser
chain.invoke(messages)
# prompt templates
from langchain_core.prompts import ChatPromptTemplate
generic_template = "Translate the following into {language}:"
prompt = ChatPromptTemplate([("system",generic_template),("user","{text}")])
result1 = prompt.invoke({"language":"Marathi","text":"Hello"})
result1.to_messages()
chain = prompt|model|parser # combined 3 components using LECL
chain.invoke({"language":"Marathi","text":"chhatrapati shivaji maharaj born in junnar"})
# deploy chains using REST API
```

🧪 Postman Test

URL:

POST http://127.0.0.1:8000/chain/invoke

Headers:

Content-Type: application/json

Body:

```
{
  "input": [{
    "language": "Marathi",
    "text": "How are you?"
  }],
  "config": {},
  "kwargs": {}
}
```

Expected Response:

```
{
  "output": [
    "कसे आहात? (Kese aahat?) \n\nThis is the most common way to say \"How are you?\" in Marathi."
  ],
  "metadata": {
    "run_id": "...",
    "feedback_tokens": []
  }
}
```

📮 Feedback Endpoint

URL:

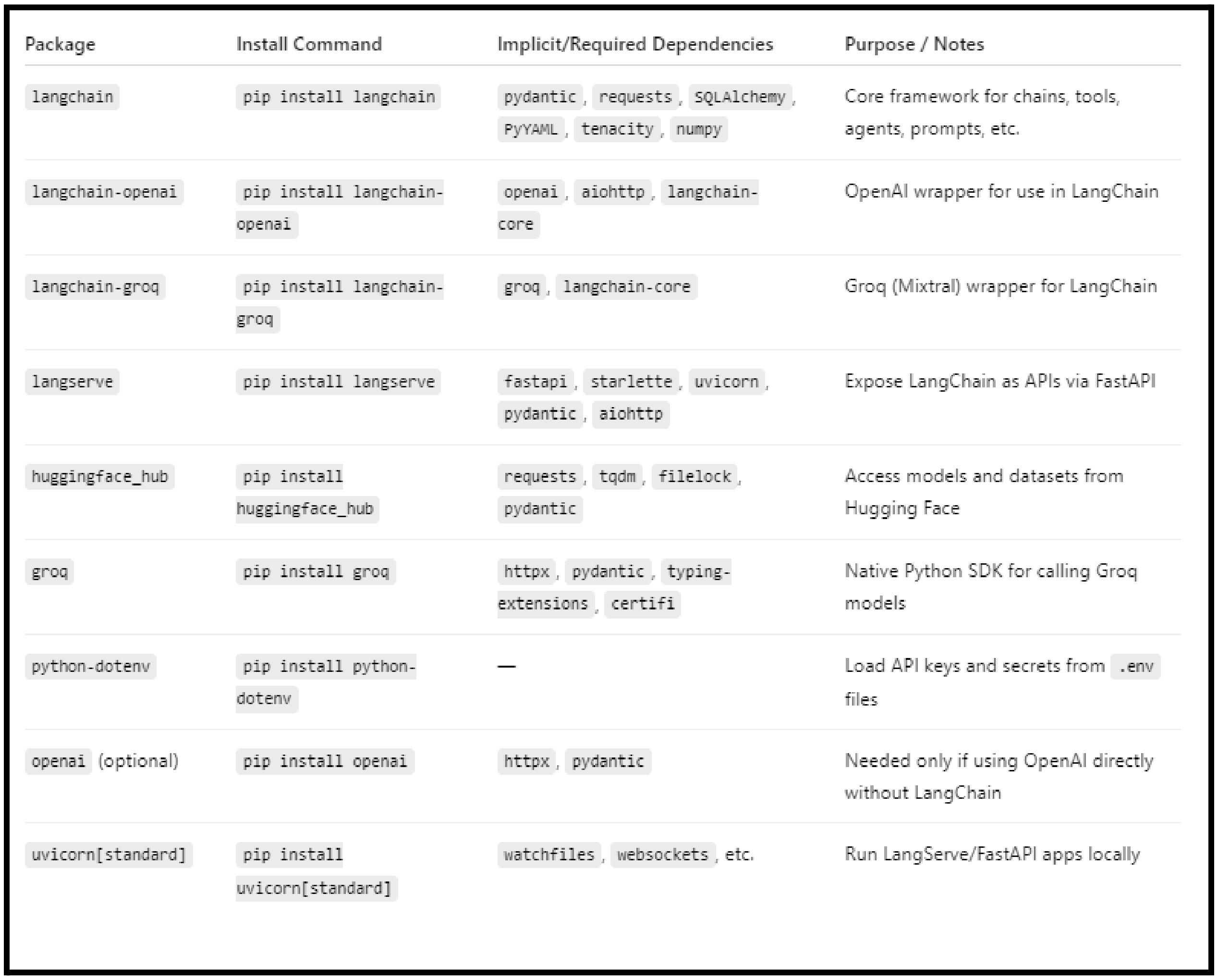
POST http://127.0.0.1:8000/feedback

Body:

```
{
  "input": [
    {
      "language": "Marathi",
      "text": "How are you?"
    }
  ],
  "output": [
    "कसे आहात?"
  ],
  "feedback": [
    "Accurate translation. Sounds natural."
  ]
}
```

LECL Documentation

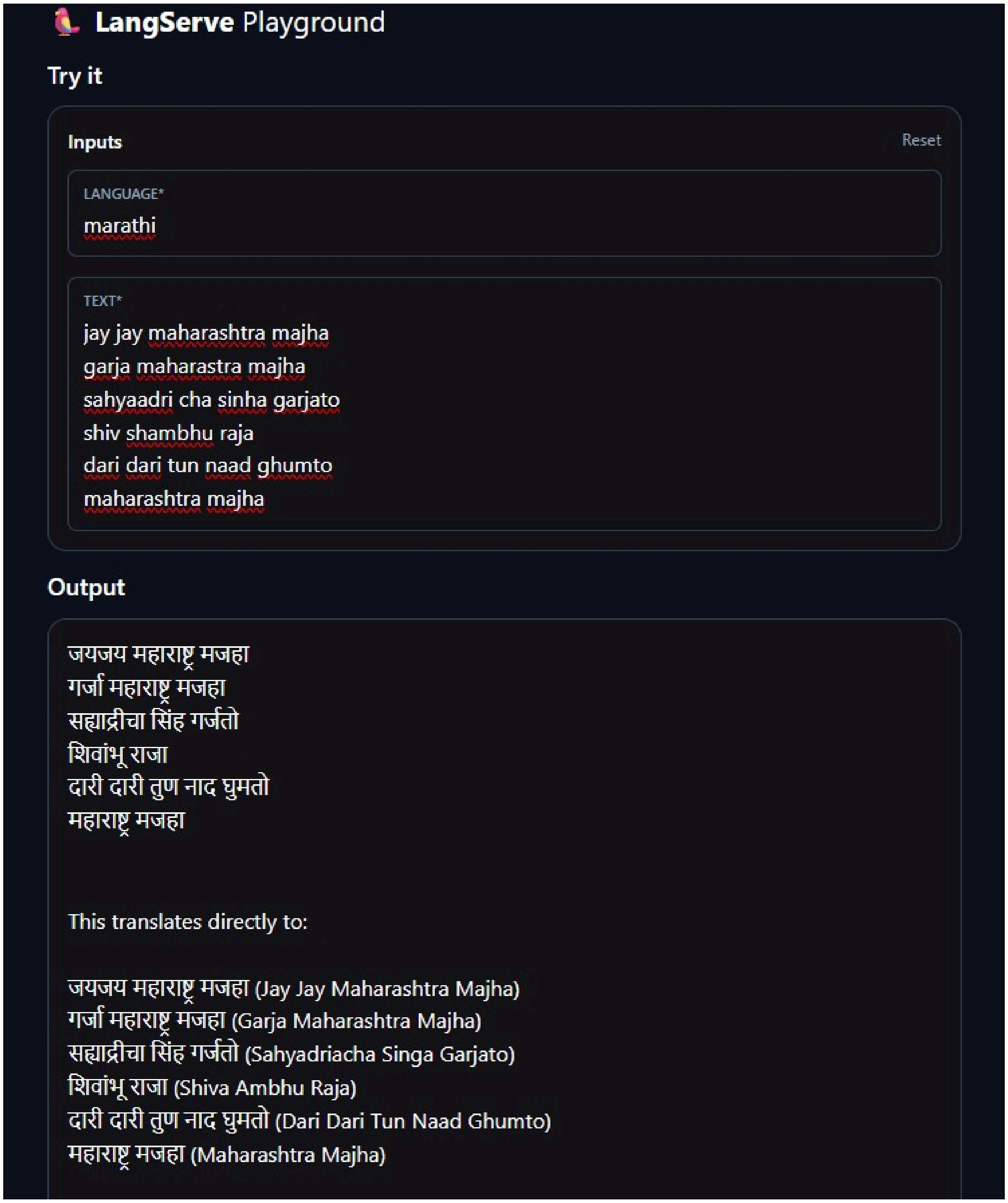
Langchain Ecosystem Dependencies Overview



Package	Install Command	Implicit/Required Dependencies	Purpose / Notes
`langchain`	`pip install langchain`	`pydantic`, `requests`, `SQLAlchemy`, `PyYAML`, `tenacity`, `numpy`	Core framework for chains, tools, agents, prompts, etc.
`langchain-openai`	`pip install langchain-openai`	`openai`, `aiohttp`, `langchain-core`	OpenAI wrapper for use in LangChain
`langchain-groq`	`pip install langchain-groq`	`groq`, `langchain-core`	Groq (Mixtral) wrapper for LangChain
`langserve`	`pip install langserve`	`fastapi`, `starlette`, `uvicorn`, `pydantic`, `aiohttp`	Expose LangChain as APIs via FastAPI
`huggingface_hub`	`pip install huggingface_hub`	`requests`, `tqdm`, `filelock`, `pydantic`	Access models and datasets from Hugging Face
`groq`	`pip install groq`	`httpx`, `pydantic`, `typing-extensions`, `certifi`	Native Python SDK for calling Groq models
`python-dotenv`	`pip install python-dotenv`	—	Load API keys and secrets from `.env` files
`openai` (optional)	`pip install openai`	`httpx`, `pydantic`	Needed only if using OpenAI directly without LangChain
`uvicorn[standard]`	`pip install uvicorn[standard]`	`watchfiles`, `websockets`, etc.	Run LangServe/FastAPI apps locally

Output

http://localhost:8000/chain/playground/



LANGUAGE*
marathi

TEXT*
jay jay maharashtra majha
garja maharastra majha
sahyaadri cha sinha garjato
shiv shambhu raja
dari dari tun naad ghumto
maharashtra majha

Output

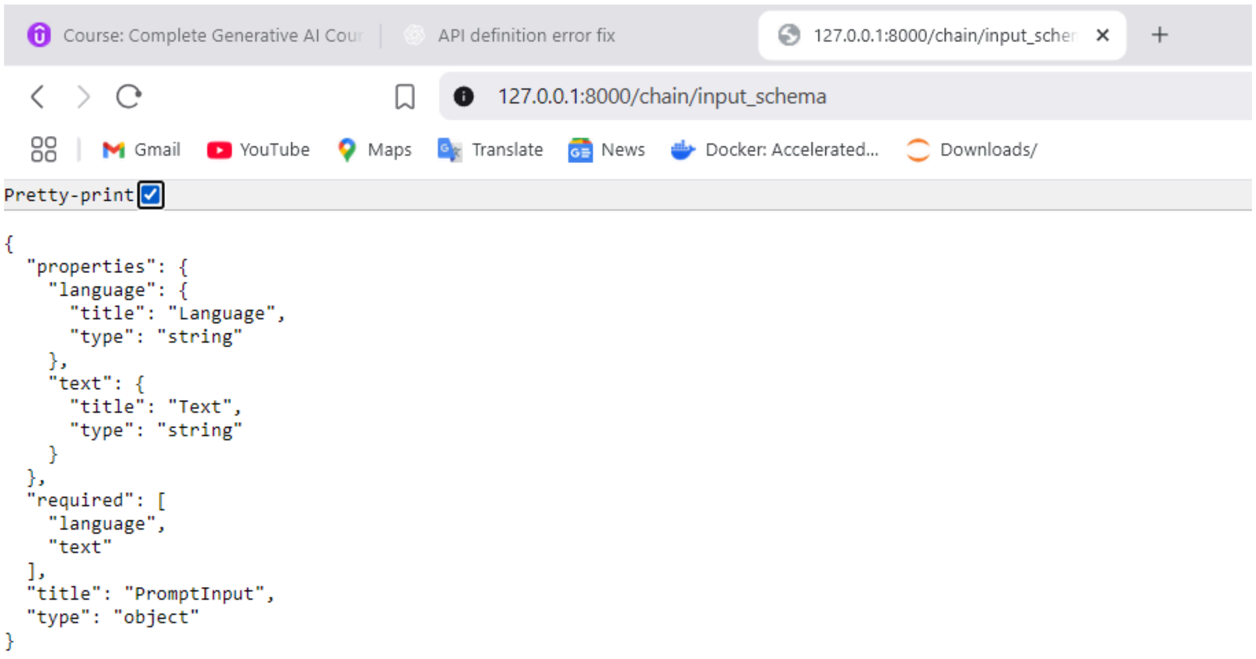
जयजय महाराष्ट्र मजहा
गर्जा महाराष्ट्र मजहा
सह्याद्रीचा सिंह गर्जतो
शिवांभू राजा
दारी दारी तुण नाद घुमतो
महाराष्ट्र मजहा

This translates directly to:

जयजय महाराष्ट्र मजहा (Jay Jay Maharashtra Majha)
गर्जा महाराष्ट्र मजहा (Garja Maharashtra Majha)
सह्याद्रीचा सिंह गर्जतो (Sahyadriacha Singa Garjato)
शिवांभू राजा (Shiva Ambhu Raja)
दारी दारी तुण नाद घुमतो (Dari Dari Tun Naad Ghumto)
महाराष्ट्र मजहा (Maharashtra Majha)

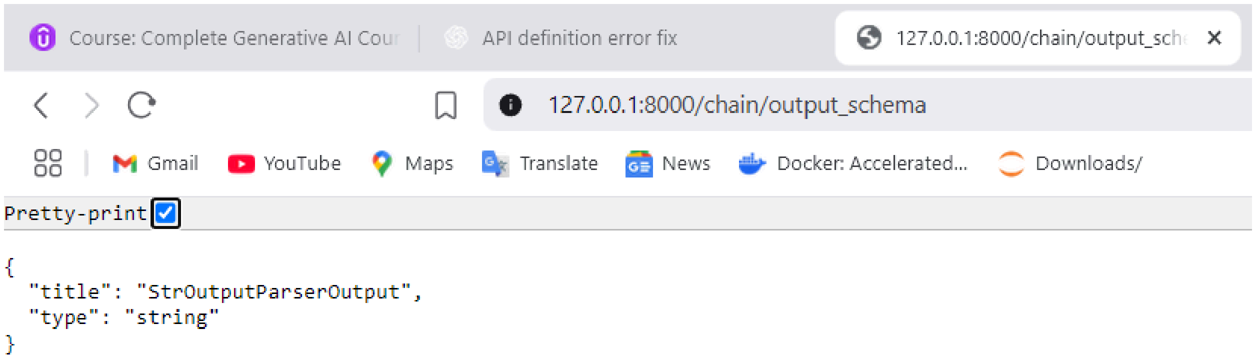
LECL Documentation

Input_schema


Course: Complete Generative AI Cour
API definition error fix
127.0.0.1:8000/chain/input_scher
127.0.0.1:8000/chain/input_schema
Gmail
YouTube
Maps
Translate
News
Docker: Accelerated...
Downloads/
Pretty-print

```
{
  "properties": {
    "language": {
      "title": "Language",
      "type": "string"
    },
    "text": {
      "title": "Text",
      "type": "string"
    }
  },
  "required": [
    "language",
    "text"
  ],
  "title": "PromptInput",
  "type": "object"
}
```

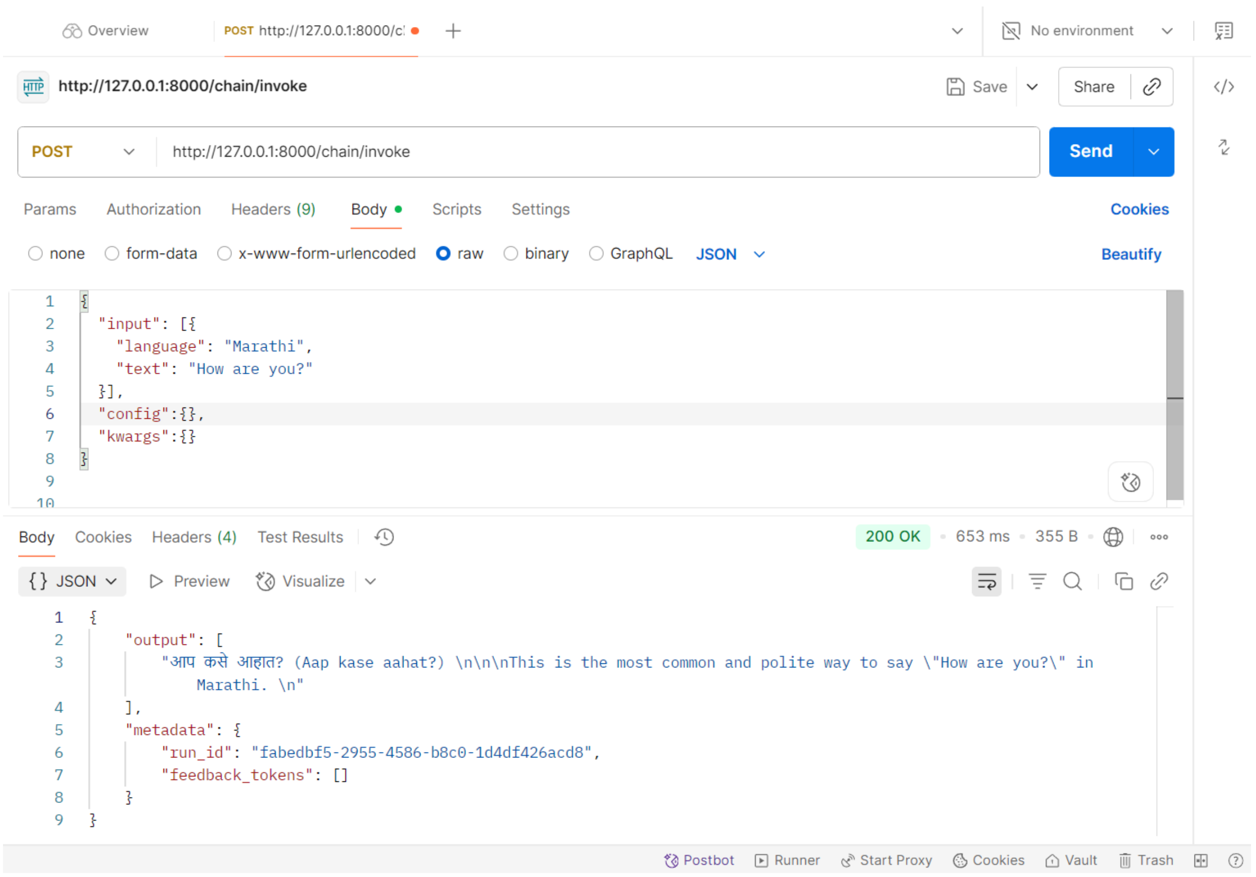
Output_schema


Course: Complete Generative AI Cour
API definition error fix
127.0.0.1:8000/chain/output_sch
127.0.0.1:8000/chain/output_schema
Gmail
YouTube
Maps
Translate
News
Docker: Accelerated...
Downloads/
Pretty-print

```
{
  "title": "StrOutputParserOutput",
  "type": "string"
}
```

Postman

LECL Documentation



POST http://127.0.0.1:8000/chain/invoke

```
{
  "input": [{
    "language": "Marathi",
    "text": "How are you?"
  }],
  "config":{},
  "kwargs":{}
}
```

200 OK • 653 ms • 355 B

```
{
    "output": [
        "आप कसे आहात? (Aap kase aahat?) \n\n\nThis is the most common and polite way to say \"How are you?\" in
          Marathi. \n"
    ],
    "metadata": {
        "run_id": "fabedbf5-2955-4586-b8c0-1d4df426acd8",
        "feedback_tokens": []
    }
}
```

Key	Value
date	Sun, 27 Jul 2025 16:40:45 GMT
server	uvicorn
content-length	229
content-type	application/json

Prepared by:
Yash K. Gadhave
Mail : yashgadhave49@gmail.com
LinkedIn
GitHub