# 📄 EduChain Assignment Report

## ✅ Overview

### 🎯 Objective

* Build an MCP server using the EduChain library.
* Generate educational content (MCQs and lesson plans).
* Prepare endpoints for integration with clients such as Claude Desktop.

## GitHub repository link

## <https://github.com/radsc/EduChainMCP>

## ✅ Components

### ⚙️ EduChain

* Python library to generate educational content.
* Provides methods to generate MCQs and lesson plans.

### ⚙️ MCP Server (FastAPI)

* Runs locally on 127.0.0.1:8000.

### Exposes two main API endpoints:

* + /generate-questions for MCQs.
  + /generate-lesson-plan for lesson plans.

#### Example server code (server.py)

from fastapi import FastAPI, HTTPException

from pydantic import BaseModel

from typing import Optional

from educhain import Educhain

from openai import OpenAI  
  
clientkey = OpenAI()

class ContentRequest(BaseModel):

topic: str

num\_questions: Optional[int] = 5

# Initialize FastAPI app

app = FastAPI(title="EduChain MCP Server")

client = Educhain() # Deafault gpt-4o-mini Model

# Create /generate-questions endpoint

@app.post("/generate-questions")

async def generate\_questions(req: ContentRequest):

try:

questions = client.qna\_engine.generate\_questions(topic=req.topic, num\_questions=req.num\_questions)

return {"topic": req.topic, "questions": questions}

except Exception as e:

raise HTTPException(status\_code=500, detail=str(e))

# Create /generate-lesson-plan endpoint

@app.post("/generate-lesson-plan")

async def generate\_lesson\_plan(req: ContentRequest):

try:

lesson\_plan = client.content\_engine.generate\_lesson\_plan(topic=req.topic)

return {"topic": req.topic, "lesson\_plan": lesson\_plan}

except Exception as e:

raise HTTPException(status\_code=500, detail=str(e))

# Create a root route

@app.get("/")

async def root():

return {"message": "EduChain MCP Server is running!"}

## ✅ Testing with curl (Windows CMD)

### MCQ generation

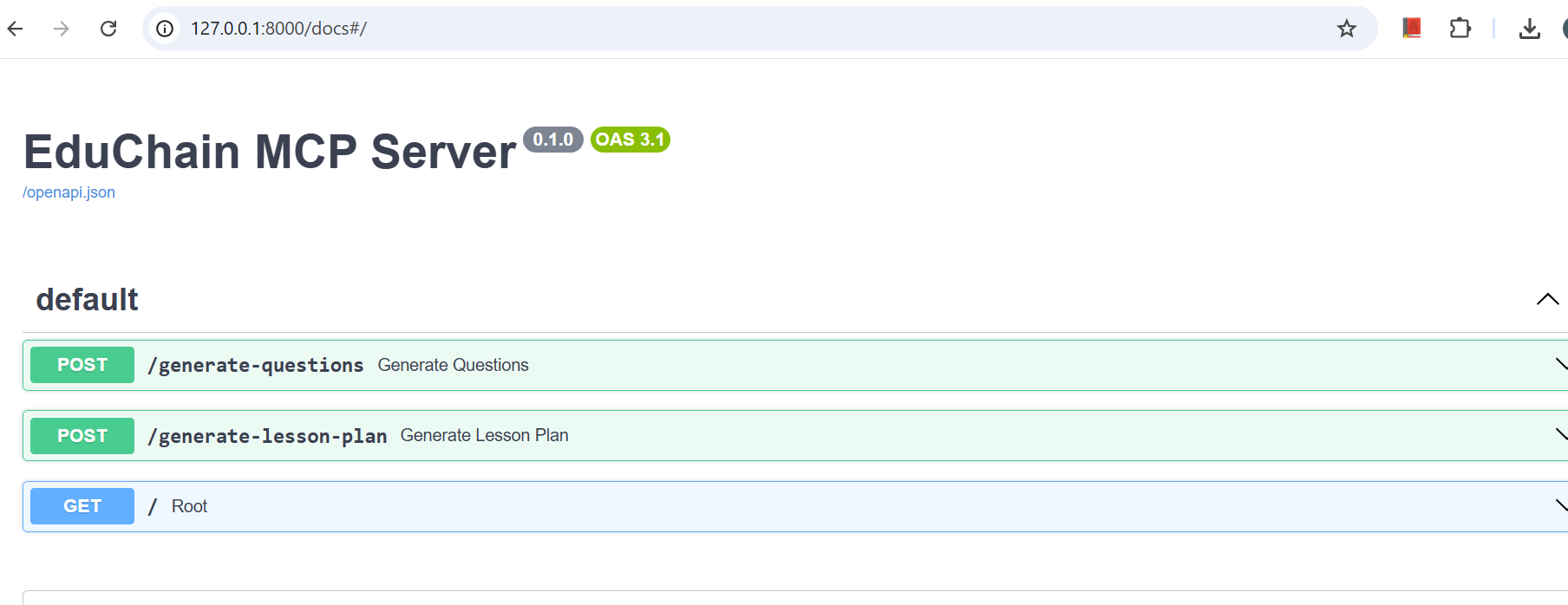
### curl -X POST "http://127.0.0.1:8000/generate-questions" -H "Content-Type: application/json" -d "{\"topic\": \"Python loops\", \"num\_questions\": 5}"

### Lesson plan generation

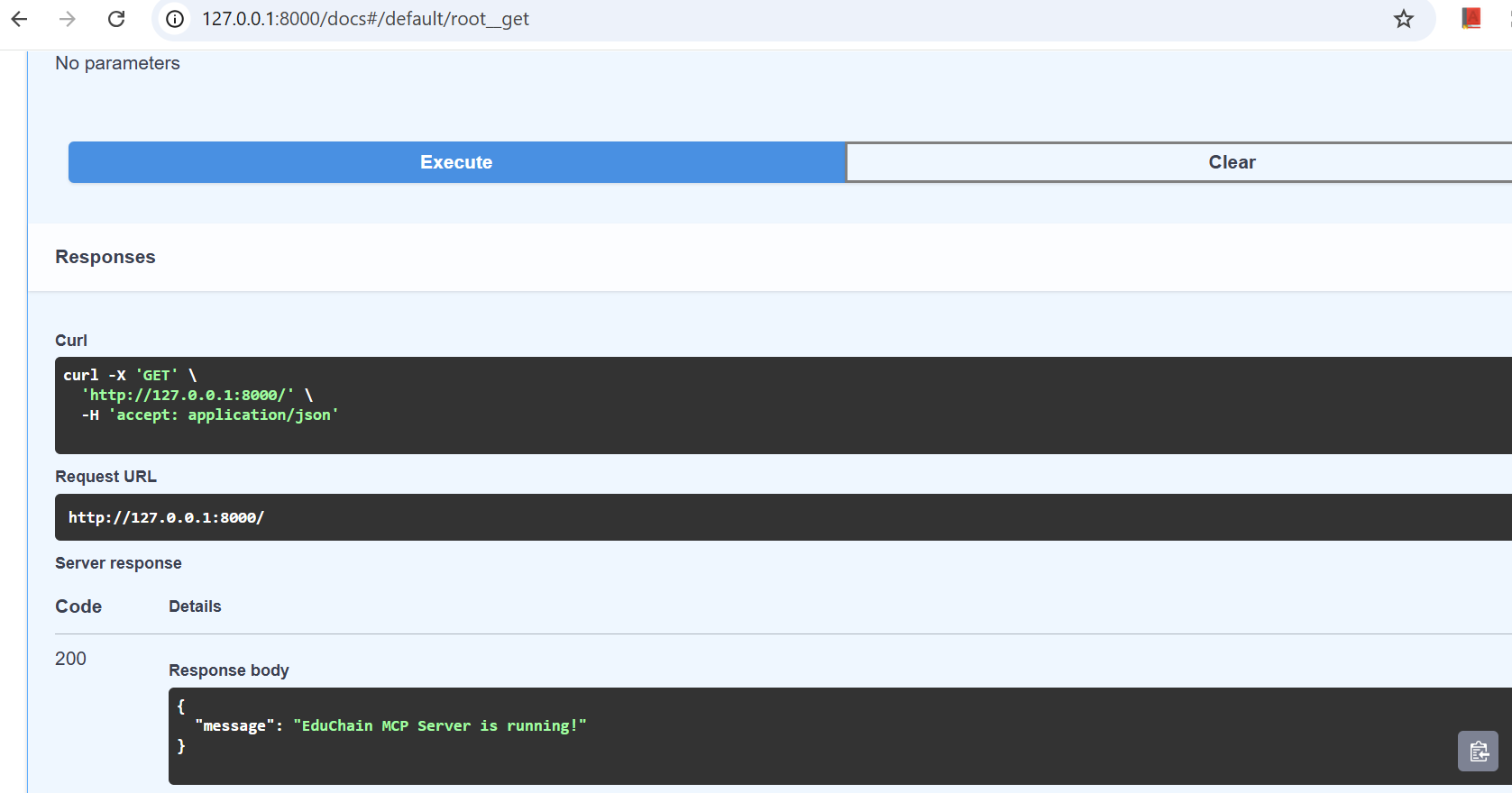
curl -X POST "http://127.0.0.1:8000/generate-lesson-plan " -H "Content-Type: application/json" -d "{\"topic\": \" Algebra \"}"

## ✅ Swagger UI Landing Screen

Run MCP server and visit the browser using the link <http://127.0.0.1:8000/docs> to open the Swagger UI landing page.

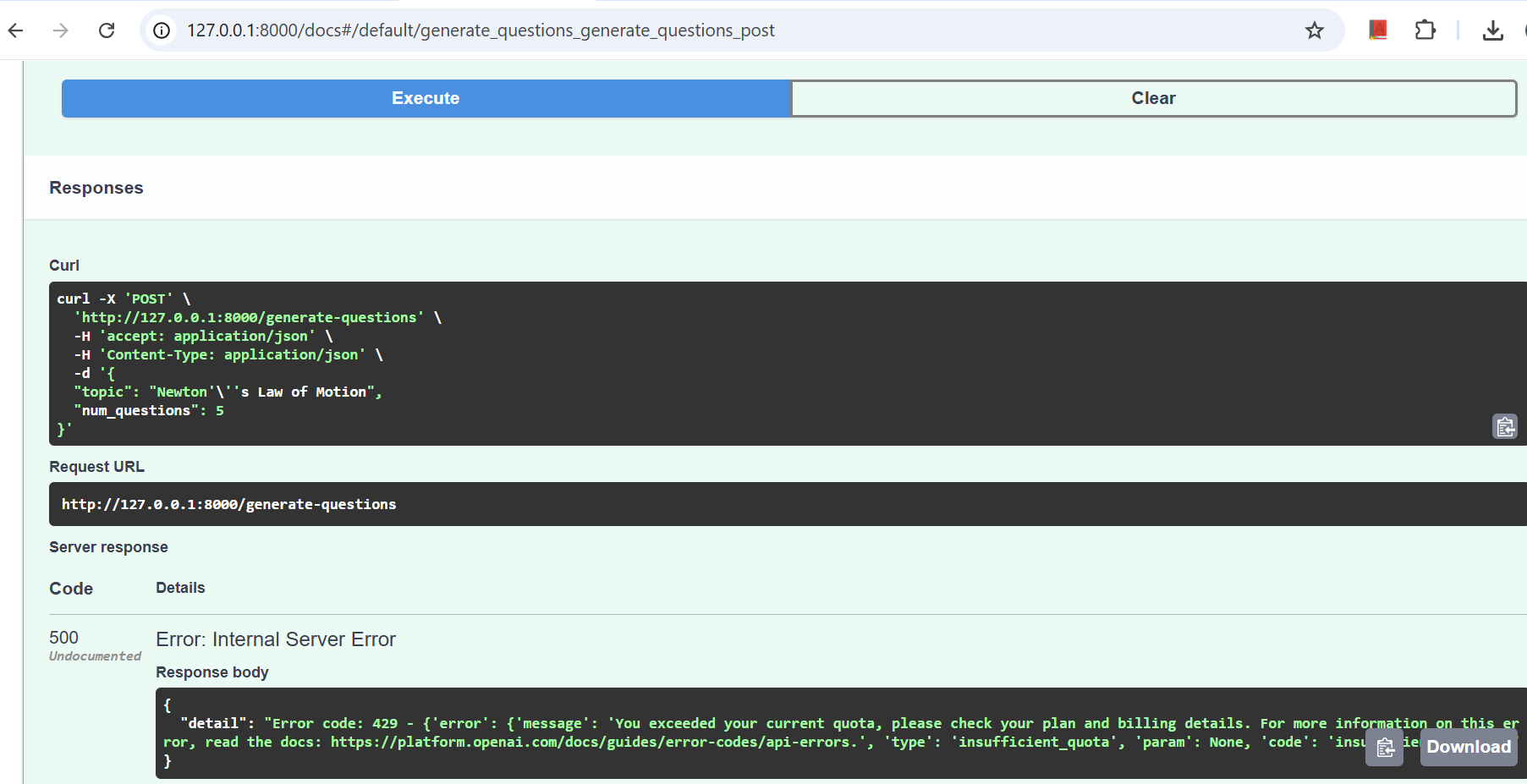


Click on the endpoint “GET : /” and execute it. You will see the response –



Try and execute generate questions endpoint, you will see the response:

Since the Open API request quota is exceeded, you will see below response. Else you will json response with 5 MCQs.



## ✅ Claude Desktop Integration Attempt

### Config used in claude\_desktop\_config.json

{  
 "tools": [  
 {  
 "name": "EduChain MCQ Generator",  
 "description": "Generate MCQs on a given topic",  
 "endpoint": "http://127.0.0.1:8000/generate-questions",  
 "method": "POST",  
 "parameters": {  
 "topic": "string",  
 "num\_questions": "integer"  
 }  
 },  
 {  
 "name": "EduChain Lesson Planner",  
 "description": "Generate a lesson plan for a given topic",  
 "endpoint": "http://127.0.0.1:8000/generate-lesson-plan",  
 "method": "POST",  
 "parameters": {  
 "topic": "string"  
 }  
 }  
 ]  
}

### Status

* Thepublic version 0.11.6 (0aa9ce) of Calude Desktop, doesn’t support local MCP setup via JSON.
* Instead, it can be further explored with Extension 🡪 Install .dxt or Pro version.

## ✅ Recommendations

* ✅ Use curl/Postman for immediate local testing.
* ✅ Optionally Swagger UI can be used to test endpoints.