MCP SERVER GENERATOR

Done By:

Sanjay V V – 71762133038 (M.Sc Decision and Computing Sciences)

Source Code: https://github.com/sanjaywick/api-to-mcp

1. Project Overview

The **Model Context Protocol (MCP) Server** is designed to provide a **standardized, LLM-friendly interface** to interact with APIs. It allows LLMs or other clients to call REST APIs dynamically without dealing with varying API conventions, authentication, or path/query parameter handling.

For this implementation, we used **PetStore** - https://petstore.swagger.io/ as the target API to demonstrate MCP functionality.

2. AI Architecture & Solution Design

2.1 Components

1. **MCP Server (mcp_server.py)**
    - FastAPI-based server.
    - Exposes endpoints of APIs as MCP tools.
    - Handles request transformation (path, query, body) and response parsing.
2. **Generator (core/generator.py)**
    - Converts Swagger/OpenAPI specifications into MCP tools.
    - Each tool contains: name, method, path, input_schema, output_schema.
3. **Request Handler (core/request_handler.py)**
    - Handles actual API calls using requests.
    - Supports GET, POST, PUT, DELETE requests.
    - Handles path parameter substitution, query params, and JSON bodies.
    - Includes error handling.
4. **Error Handling (core/error_handler.py)**
    - Centralized module to catch and log API errors.

- Returns user-friendly error messages.

5. **Swagger/OpenAPI Spec**
   - Used as the single source of truth for endpoints.
   - Stored locally in spec/petstore.json

3. Code Standards Followed

1. **Naming Conventions**
   - Snake_case for variables and functions.
   - PascalCase for classes.
2. **Project Structure**

```
mcp-generator/
├─ server/
│   └─ mcp_server.py
├─ core/
│   ├─ generator.py
│   ├─ request_handler.py
│   └─ error_handler.py
├─ spec/
│   └─ petstore.json
├─ venv/
└─ README.md
```

3. **Documentation & Comments**
   - Each function is commented explaining input/output.
   - README.md includes setup instructions.

4. How Generator Works

1. Load Swagger/OpenAPI spec (petstore.json).
2. Parse paths section into a list of endpoints.
3. For each path + method:

4. Return list of MCP tools ready for server invocation.

Assumptions:

- Spec uses OpenAPI 3.x standard.
- Path parameters follow {param} syntax.
- Request/response bodies are JSON.

5. Demo / How to Use

1. Run MCP Server

uvicorn server.mcp_server:app –reload

2. List Tools

http://127.0.0.1:8000/tools

3. Invoke an API Tool

http://127.0.0.1:8000/invoke/addPet

{

"body": {

"id": 123,

"name": "Chubramani",

"tag": "dog" }}

4. Response

{

"status": "success",

"data": {

"id": 123,

"name": "Chubramani",

"photoUrls": [],

"tags": []

}

}

7. Future Improvements

- Add **authentication support** for APIs requiring API keys or OAuth.
- Build **Streamlit UI** to list tools and send requests interactively.
- Add **caching** for frequent GET requests.