DACA Chatbot API - Documentation

Project Overview

This project is a basic implementation of a chatbot backend using FastAPI and Pydantic, built for the DACA (Data-Agentic Conversational AI) tutorial series. It includes user message handling, metadata management (timestamp and session ID), and generates structured API responses using BaseModel.

Features

- GET and POST endpoints
- Auto-generated timestamps and UUIDs
- Query and path parameter support
- Validation using Pydantic
- Interactive API docs (Swagger & ReDoc)

Project Structure

- main.py: FastAPI application
- README.md: Documentation file
- pyproject.toml: Project metadata & dependencies

Installation Guide

1. Install uv (Universal Virtual Environment)

```
powershell -ExecutionPolicy ByPass -c "irm https://astral.sh/uv/install.ps1 | iex"
```

2. Create Project Directory

```
uv init daca-chatbot
cd daca-chatbot
```

3. Create & Activate Virtual Environment

```
uv venv
.venv\Scripts\activate
```

4. Add Dependencies

```
uv add fastapi
uv add "pydantic>=2.0"
uv add --dev pytest pytest-asyncio
```

DACA Chatbot API - Documentation

Sample pyproject.toml

```
[project]
name = "daca-chatbot"
version = "1.0.0"
description = "DACA Chatbot using FastAPI and Pydantic"
readme = "README.md"
requires-python = ">=3.10"
dependencies = ["fastapi>=0.115", "pydantic>=2.7"]
[dependency-groups]
dev = ["pytest", "pytest-asyncio"]
```

How to Run

```
uv run uvicorn main:app --reload
```

OR

```
fastapi dev main.py
```

API Endpoints

1. GET / : Returns a welcome message.
2. GET /users/{user_id}?role=admin : Returns user ID and role.
3. POST /chat/ : Accepts user message and returns chatbot reply.

Code Explanation

MetaData Class:

- timestamp: Automatically sets current time with timezone
- session_id: Generates a unique UUID

Message Class:

- Accepts user_id, text, metadata, tags

Response Class:

- Returns user_id, reply, metadata

chat() Function:

- Validates message
- Constructs dynamic reply
- Returns a structured Response

Conclusion

This project helps you understand:

- FastAPI basics (routing, async handlers)
- Path/query parameters
- Request/response models with Pydantic
- Auto-generated documentation