Lecture 21 – FastAPI

FastAPI is a modern Python web framework designed specifically for building APIs. It has quickly become one of the most popular frameworks because of its speed, simplicity, and ability to automatically generate interactive API documentation. Unlike traditional frameworks, FastAPI fully supports asynchronous programming, which allows it to handle multiple requests simultaneously with high efficiency. These features make it an excellent choice for building scalable web services, including AI-powered applications.

What is FastAPI?

FastAPI is a web framework built on top of standard Python type hints. It is designed to simplify the process of creating APIs by reducing boilerplate code and automating many routine tasks. The framework is not only fast to run but also fast to develop with, thanks to its clean syntax and automatic documentation features.

Key characteristics include:

- **Asynchronous request handling**: Multiple clients can be served at once without blocking.
- **Automatic API documentation**: FastAPI generates Swagger and ReDoc documentation for all endpoints.
- **High performance**: Its speed is comparable to Node.js and Go.
- **Ease of learning**: The framework is beginner-friendly yet powerful enough for production-grade projects.

Why Use FastAPI?

There are several reasons for choosing FastAPI over other web frameworks such as Flask or Django:

1. **Ease of Use** – The framework requires minimal setup and reduces repetitive coding.
2. **Integration with AI and Generative Models** – FastAPI is often used in AI projects for exposing models as APIs.
3. **Scalability and Security** – Suitable for both small applications and large-scale enterprise systems.
4. **Modern Features** – Uses Python type hints for validation and supports asynchronous code execution.

Because of these features, FastAPI is well-suited for applications ranging from simple APIs to complex services in data science and machine learning.

Setting Up FastAPI

The first step is to install FastAPI along with **Uvicorn**, which is an ASGI server required to run FastAPI applications.

```
pip install fastapi uvicorn
```

After installation, create a file (for example, `app.py`) and write a minimal FastAPI program:

```
from fastapi import FastAPI

app = FastAPI(
    title="My First API",
    description="A Simple API Using FastAPI",
    version="1.0.0"
)

@app.get("/")
def read_root():
    return {"message": "Hello, World! Welcome to FastAPI"}
```

To run the application, use the command:

```
uvicorn app:app --reload
```

The `--reload` flag automatically restarts the server when changes are made to the code. After starting the server, you can open your browser at `http://127.0.0.1:8000` to view the output.

Creating Custom API Endpoints

FastAPI allows developers to define their own endpoints quickly. For example, a personalized greeting endpoint can be defined as:

```
@app.get("/hello/{name}")
def greet_user(name: str):
    return {"message": f"Hello, {name}!"}
```

In this example, `{name}` is a path parameter. When a user accesses `http://127.0.0.1:8000/hello/Ahmad`, the API returns:

```
{"message": "Hello, Ahmad!"}
```

This approach demonstrates how FastAPI automatically extracts parameters from URLs and passes them into Python functions.

Automatic API Documentation

One of the most powerful features of FastAPI is its automatic generation of interactive API documentation. Without writing any extra code, you can access:

- **Swagger UI** at `http://127.0.0.1:8000/docs`
- **ReDoc documentation** at `http://127.0.0.1:8000/redoc`

These interfaces display available endpoints, input parameters, and expected outputs. They also allow developers to test APIs directly from the browser, which makes debugging and validation much easier.

Running and Stopping FastAPI Applications

- **Running**: Use `uvicorn app:app --reload` in the terminal.
- **Stopping**: Press `CTRL + C` to terminate the server.

This simple workflow allows for quick development cycles and interactive testing.

Conclusion

FastAPI is a modern Python framework that combines high performance with ease of use. In this lecture, we explored its basic setup, created simple and custom endpoints, and examined its automatic documentation features. The framework is particularly effective for AI and machine learning projects, where rapid deployment and scalability are critical.

By mastering FastAPI, developers can build APIs that are not only fast and reliable but also maintainable and easy to extend. Future lectures will cover more advanced features such as request and response models, data validation, and deploying FastAPI applications to production environments.