**Imported fastapi and uvicorn in backend**

**1. FastAPI**

FastAPI is a modern, high-performance web framework for building APIs with Python 3.7+ based on standard Python type hints. It is designed for speed, ease of use, and automatic generation of OpenAPI documentation.

**Key Features of FastAPI:**

* **Fast**: As fast as Node.js and Go.
* **Automatic Documentation**: Generates Swagger UI and ReDoc.
* **Type Hints**: Uses Python type hints to validate data.
* **Dependency Injection**: Simplifies authentication and authorization.
* **Asynchronous Support**: Uses async and await for non-blocking operations.

**Basic Example of FastAPI**

from fastapi import FastAPI

app = FastAPI()

@app.get("/")

def read\_root():

return {"message": "Hello, FastAPI"}

@app.get("/items/{item\_id}")

def read\_item(item\_id: int, q: str = None):

return {"item\_id": item\_id, "query": q}

**2. Uvicorn**

Uvicorn is a lightning-fast ASGI (Asynchronous Server Gateway Interface) server that runs FastAPI applications. It is built on **uvloop** and **httptools**, making it much faster than traditional WSGI-based servers like Gunicorn.

**Why Use Uvicorn?**

* **Asynchronous**: Uses ASGI instead of WSGI for better performance.
* **Lightweight**: Minimal overhead.
* **Supports WebSockets**: Works well with real-time applications.

**Running a FastAPI App with Uvicorn**

Once you create a FastAPI app (app), you can run it using Uvicorn:

**uvicorn main:app --reload**

Here:

* main → The Python file where the FastAPI instance is created.
* app → The FastAPI instance.
* --reload → Enables auto-reloading on code changes (useful in development).

**Commonly Used Codes in FastAPI**

**1. Path and Query Parameters**

@app.get("/users/{user\_id}")

def get\_user(user\_id: int, details: bool = False):

return {"user\_id": user\_id, "details": details}

**2. Request Body (POST Request)**

from pydantic import BaseModel

class Item(BaseModel):

name: str

price: float

is\_offer: bool = None

@app.post("/items/")

def create\_item(item: Item):

return {"item": item}

**3. Handling HTTP Status Codes**

from fastapi import HTTPException

@app.get("/error/")

def trigger\_error():

raise HTTPException(status\_code=404, detail="Item not found")

**4. Middleware Example**

from fastapi.middleware.cors import CORSMiddleware

app.add\_middleware(

CORSMiddleware,

allow\_origins=["\*"],

allow\_credentials=True,

allow\_methods=["\*"],

allow\_headers=["\*"],

)

**How FastAPI and Uvicorn Work Together**

1. **Create a FastAPI application** (main.py).
2. **Run it with Uvicorn**:

uvicorn main:app --host 0.0.0.0 --port 8000 --reload

1. **Access API at** http://127.0.0.1:8000
2. **View API documentation at**:
   * Swagger UI: http://127.0.0.1:8000/docs
   * ReDoc: http://127.0.0.1:8000/redoc