Step 1: Introduction to FastAPI Basics

Objective

Create your first web API using FastAPI and understand the fundamental concepts of APIs.

Context

APIs (Application Programming Interfaces) allow different software applications to communicate with each other. FastAPI is a modern, fast web framework for building APIs with Python. In this step, you'll learn the basics of FastAPI and create a simple "Hello World" API.

Why it is required

Understanding APIs is essential for modern software development. APIs enable:

- Communication between different applications
- Sharing of data and functionality
- Building of scalable and modular software systems

Learning FastAPI provides a beginner-friendly introduction to API development with automatic documentation, type checking, and high performance.

How to achieve this

1. Set up your development environment

- Install Python 3.7+ if not already installed
- Create a new directory for your project
- Create a virtual environment and activate it
- Install the required packages

```
# Create a project directory
mkdir ml-api-beginner
cd ml-api-beginner

# Create a virtual environment
python -m venv venv

# Activate the virtual environment
# On Windows:
venv\Scripts\activate
# On macOS/Linux:
source venv/bin/activate

# Create a requirements.txt file with the necessary packages
# (You can copy the requirements from the advanced track)
# Install the required packages
pip install fastapi uvicorn
```

2. Create your first FastAPI application

Create a new file named `main.py` with the following content:

```python
from fastapi import FastAPI

# Create a FastAPI instance
app = FastAPI(
    title="My First API",
    description="A simple API built with FastAPI",
    version="0.1.0"
)

# Create a root endpoint
@app.get("/")
async def root():
    return {"message": "Hello World"}

# Create an endpoint that returns your name
@app.get("/name")
async def get_name():
    return {"name": "Your Name"}

# Create an endpoint that takes a name parameter and returns a greeting
@app.get("/greet/{name}")
async def greet(name: str):
    return {"message": f"Hello, {name}!"}

# Create an endpoint that takes query parameters
@app.get("/query")
async def query_params(name: str = "World", age: int = None):
    response = {"message": f"Hello, {name}!"}
    if age:
        response["age"] = age
    return response
```

3. Run your FastAPI application

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

4. Explore your API

- Open a web browser and go to `http://localhost:8000/` to see the "Hello World" message
- Try the other endpoints:

    - `http://localhost:8000/name`
    - `http://localhost:8000/greet/John`
    - `http://localhost:8000/query?name=Jane&age=25`
- Explore the automatic interactive documentation at `http://localhost:8000/docs`

Examples of usage

Making requests to your API

Using a web browser

Simply navigate to:

- `http://localhost:8000/`
- `http://localhost:8000/name`
- `http://localhost:8000/greet/Alice`
- `http://localhost:8000/query?name=Bob&age=30`

Using curl (command line)

```bash
# Root endpoint
curl http://localhost:8000/

# Name endpoint
curl http://localhost:8000/name

# Greet endpoint with path parameter
curl http://localhost:8000/greet/Charlie

# Query endpoint with query parameters
curl "http://localhost:8000/query?name=Dave&age=35"
```

Using Python requests library

```python
import requests

# Root endpoint
response = requests.get("http://localhost:8000/")
print(response.json())

# Name endpoint
response = requests.get("http://localhost:8000/name")
print(response.json())

# Greet endpoint with path parameter
response = requests.get("http://localhost:8000/greet/Eve")
print(response.json())
```

```
# Query endpoint with query parameters
response = requests.get("http://localhost:8000/query", params={"name": "Frank",
"age": 40})
print(response.json())
```

Tasks for students

1. Set up the development environment as described above
2. Create the `main.py` file with the provided code
3. Run the FastAPI application using Uvicorn
4. Test all the endpoints using a web browser
5. Explore the automatic documentation at `/docs`
6. Add a new endpoint `/add` that takes two query parameters `a` and `b` and returns their sum
7. Add a new endpoint `/info` that returns information about yourself (name, age, favorite programming language)