Module 1: Backend Development with FastAPI

🚀 Overview

- Backend development fundamentals
- REST APIs and FastAPI basics
- Setting up your development environment
- Python virtual environments

What Is Backend Development?

- The backend powers:
  - Application logic
  - Databases
  - APIs
- Handles:
  - Data processing
  - Authentication
  - Business logic
  - Communication with the frontend

Backend vs Frontend

Frontend	Backend
Runs in browser	Runs on server
HTML, CSS, JS	Python, Java, Node.js
UI/UX focused	Logic & data focused
Immediate feedback	Handles data & logic

Client-Server Model

Request/Response cycle:

- Browser (client) sends HTTP request
- Backend (server) sends response
- Stateless communication

What is an API?

API = Application Programming Interface , allows systems to communicate

- REST APIs use HTTP methods: GET, POST, PUT, DELETE
- Backend provides **endpoints** that clients can call

REST Overview

REST = Representational State Transfer

- Principles:
    - Stateless
    - Resource-based (URL represents data)
    - Standard HTTP methods
- Examples:
    - `GET /users` → fetch users
    - `POST /users` → create user

Introduction to FastAPI

FastAPI is a modern web framework for building APIs
Built on **Starlette** (ASGI) and **Pydantic**

Key Features:

- Type hints & data validation
- Automatic API docs (Swagger/OpenAPI)
- Async support
- Super fast!

Backend Tech Stack (This Course)

- **Language:** Python 3.11+
- **Framework:** FastAPI
- **Database:** PostgreSQL
- **ORM:** SQLAlchemy
- **Async:** asyncio, httpx
- **Deployment:** Docker + Render

Setting Up Your Dev Environment

- Install Python 3.11+
- Install VS Code or PyCharm
- Git and GitHub setup
- HTTP clients: Postman, Insomnia, Bruno
- Python virtual environments:

```
python -m venv .venv
source .venv/bin/activate
```

Intro to Git and Version Control

- What is `git`?
  - VCS (Version Control System)
  - Tracks code changes
  - Enables collaboration
- Common commands:
  - `git init` - Initialize a repo
  - `git add .` - Stage changes
  - `git commit -m "message"` - Commit changes
  - `git push origin main` - Push to remote repo

Publish a Repo to GitHub

- Create a new repository on GitHub: e.g., `your-repo`
- Git commands to push local repo to GitHub:
  - `git init` - Initialize a repo
  - `git add .` - Stage changes
  - `git commit -m "Initial commit"` - Commit changes
  - `git remote add origin https://github.com/yourusername/your-repo.git` - Link to remote repo
  - `git branch -M main` - Rename branch to main
  - `git push origin main` - Push to remote repo

Uvicorn - ASGI Server

- Fast, lightweight ASGI server used to run Python web apps
- It handles incoming HTTP requests, passes them to your FastAPI app, and returns the responses back to clients.
- It is the engine that runs your FastAPI app.
- **ASGI** = Asynchronous Server Gateway Interface

WSGI vs ASGI

WSGI	ASGI
synchronous	asynchronous
Client --> Web Server --> App	Client --> Web Server --> App
Handles 1 request at a time	Handles multiple requests concurrently
Blocking: each request waits	Non-blocking: requests can run in parallel

Project: Hello FastAPI - Development

```
pip install fastapi uvicorn
```

main.py

```python
from fastapi import FastAPI
app = FastAPI()

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

Project: Hello FastAPI - Running & Testing

- Run the server:

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

- Test in terminal:

```
curl http://localhost:8000/
```

- Test in browser:

```
http://localhost:8000/
```

Homework

- Set up your environment
- Create a GitHub repo
- Build a `/hello` endpoint returning a JSON message

🎯 Remember

- Backend development fundamentals
- Client-server model
- REST APIs and FastAPI basics
- Setting up your development environment
- Python virtual environments