Week 1: Backend Development with FastAPI

Course Introduction

What you'll learn:

- Build high-performance APIs
- Use databases, authentication, and Docker
- Deploy real-world backend systems

What Is Backend Development?

The backend powers:

- Application logic
- Databases
- APIs

Handles:

- Data processing
- Authentication
- Business rules

2

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

Example:

- `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 or Insomnia

- Python virtual environments:

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

Intro to Git and Version Control

- What is `git`?
  - 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

Project: Hello FastAPI

```
pip install fastapi uvicorn
```

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

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

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

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