**FastAPI Learning Summary**

*By Mushahid Hussain*

**Overview**

I have completed a comprehensive 19-hour course on FastAPI, covering a wide range of topics, including CRUD operations, database integration with PostgreSQL, authentication mechanisms, ORM with SQLAlchemy, JWT tokens, deployment, and testing. This document summarizes my hands-on learning and key takeaways, emphasizing how FastAPI can be used to build scalable, performant, and secure APIs.

**Course Details**

* **Instructor:** Sanjeev Thiyagarajan
* **Course Duration:** 19 hours
* **Topics Covered:** FastAPI, PostgreSQL, SQLAlchemy ORM, Authentication, Alembic migrations, Heroku, Ubuntu, Docker, and CI/CD Pipelines

**Skills Acquired**

**1. Setting Up FastAPI**

* **Virtual Environment:**  
  Set up and activated a virtual environment using:

bash

Copy code

py -3 -m venv venv

source venv/Script/Activate.bat

* **Running the App with Uvicorn:**  
  Executed the FastAPI app using Uvicorn:

bash

Copy code

uvicorn main:app --reload

**2. FastAPI Basics**

* Built APIs using HTTP methods such as GET, POST, PUT, DELETE, PATCH
* Utilized FastAPI decorators to create routes and handle requests
* Used Pydantic for schema validation
* Managed HTTP Exceptions and custom status codes
* Introduced proper documentation and API testing using the automatic interactive docs FastAPI provides (Swagger UI)

**3. Database Integration**

* **PostgreSQL Setup & Management**:
  + Set up PostgreSQL with PGAdmin for database management
  + Created and managed tables with constraints and default values
* **Database Connection:**
  + Implemented connection retries for reliable database connections using psycopg2
  + Used SQLAlchemy ORM for building and managing the database models
* **CRUD Operations with ORM:**
  + Defined database models using SQLAlchemy
  + Implemented CRUD functionalities (Create, Read, Update, Delete) with raw SQL and ORM

**4. Authentication & Authorization**

* Implemented OAuth2 authentication with JWT tokens
* Added login routes to authenticate users and generate tokens
* Restricted access to APIs by verifying tokens and protecting routes

**5. Deployment**

* **Heroku Deployment:**
  + Deployed the FastAPI application to Heroku, using PostgreSQL add-ons
  + Managed environment variables and migrations with Alembic in Heroku
* **Ubuntu Deployment:**
  + Configured Ubuntu for production deployment
  + Set up the environment, handled PostgreSQL configurations, and managed app restarts with Gunicorn and Nginx
  + Configured SSL certificates and security settings

**6. Testing & CI/CD**

* Wrote unit and integration tests using pytest
* Set up GitHub Actions for CI/CD to automate testing, building Docker images, and deploying to Heroku/Ubuntu
* Ensured code quality with coverage tracking and consistent testing practices

**7. Advanced Features**

* **JWT Authentication**: Created custom JWT tokens and secured endpoints
* **Foreign Key Relationships**: Established relationships between models (Users & Posts) and added validation rules for ownership
* **Pagination & Filtering**: Built pagination and filtering capabilities into the APIs to handle large datasets
* **Alembic Migrations**: Managed database schema changes through migrations using Alembic
* **Voting & Like System**: Implemented a voting/like system using SQL joins and relationships

**8. Dockerization & CI/CD Pipelines**

* Created a Dockerfile to containerize the FastAPI app
* Used Docker Compose to manage multi-container apps, integrating PostgreSQL and FastAPI
* Set up GitHub Actions for continuous integration and deployment, automating build processes, and deployment to Heroku and Ubuntu

**Technologies & Tools**

* **Languages:** Python
* **Framework:** FastAPI
* **Database:** PostgreSQL
* **ORM:** SQLAlchemy
* **Deployment:** Heroku, Ubuntu, Docker
* **Authentication:** OAuth2, JWT Tokens
* **Testing:** Pytest
* **CI/CD:** GitHub Actions
* **Package Management:** pip, virtualenv
* **Other Tools:** Uvicorn, PGAdmin, Gunicorn, Nginx

**Key Takeaways**

FastAPI, as a modern web framework, provides high performance, intuitive API building, and seamless integration with modern libraries such as Pydantic and SQLAlchemy. The course helped me gain expertise in setting up a complete backend API with authentication, database connections, deployment, and CI/CD automation, making me confident in building and deploying scalable APIs for production environments.

**Source Code**

Github: <https://github.com/mushahid1/fast-api-course>

Youtube: <https://www.youtube.com/watch?v=Yw4LmMQXXFs&list=PL8VzFQ8k4U1L5QpSapVEzoSfob-4CR8zM>

<https://www.youtube.com/watch?v=1N0nhahVdqs>