TDI PROJECT FOR DATA ENGINEERING WEEK 5

SQLAlchemy Models, PostgreSQL, and Workflow Orchestration

Introduction

Welcome to Week 5! This week, you will build on the database you created in Week 4 by adding a RESTful API layer using **FastAPI**. This assignment will introduce you to the basic concepts of building APIs and how to use them to interact with your **PostgreSQL** database. You will implement the four primary HTTP methods: `GET`, `POST`, `PUT`, and `DELETE` to manipulate data in your database.

PART A: Video Links

Here are some helpful links that will introduce or reinforce the concepts you'll be applying this week:

Requests Library

FASTAPI

RESTFUL API

Functions in Python

HTTP Methods

argparse Library

Branching in Git

SQLAlchemy

Prefect

Airflow

PostgresSQL

PART B: Assignment Tasks and Directions

In this week's assignment, you will use **FastAPI** to build a RESTful API that will interact with the **PostgreSQL** database you created in Week 4. You will implement various endpoints to **GET**, **POST**, **PUT**, and **DELETE** data from the database.

File Structure for This Week's Assignment:

```
week5_assignment/
│
├── .env
├── models/                 # Folder to store SQLAlchemy models
│   └── models.py           # SQLAlchemy models for the PostgreSQL tables
│
├── api/                    # Folder for FastAPI application
│   ├── main.py             # Main FastAPI application file
│   └── crud.py             # CRUD functions to interact with the database
```

```
├── requirements.txt          # Python dependencies (FastAPI, SQLAlchemy, psycopg2, etc.)
```

Task 1: Set Up FastAPI

Objective:
You will set up a FastAPI application and connect it to your PostgreSQL database using **SQLAlchemy**.

1. **Install FastAPI and Uvicorn:**
   Add **FastAPI** and **Uvicorn** to your `requirements.txt` file, and install them in your environment.
   ```
   pip install fastapi uvicorn
   ```

2. **Create a `main.py` file** in the `api/` folder to serve as the entry point for your FastAPI application.

3. **Connect FastAPI to PostgreSQL:**
   Use SQLAlchemy to establish a connection to your PostgreSQL database. Import the models you created in Week 4 from `models/models.py`.

4. **Run the FastAPI Application:**
   Use Uvicorn to run your FastAPI app.
   ```
   uvicorn api.main:app –reload
   ```

Task 2: Create CRUD Functions (`crud.py`)

Objective:
You will write functions to handle **Create**, **Read**, **Update**, and **Delete** (CRUD) operations on your PostgreSQL data.

1. **Define a function to get data:**
   Write a function to retrieve data from the database. Use SQLAlchemy to query the `data warehouse` table.

2. **Define a function to add data:**
   Write a function to insert new records into the database. Use SQLAlchemy ORM to create new rows in the `datawarehouse` table.

3. **Define a function to update data:**
   Write a function to update existing records in the database. Use SQLAlchemy to modify records based on their `id`.

4. **Define a function to delete data:**
   Write a function to delete records from the database. Use SQLAlchemy to remove records based on their `id`.

Task 3: Define API Endpoints in `main.py`

Objective:
You will define API endpoints for each CRUD operation. Each endpoint will correspond to one of the HTTP methods: `GET`, `POST`, `PUT`, and `DELETE`.

1. **GET Endpoint: Fetch Data**
   Define a `GET` endpoint that retrieves all records from the database, or a single record by `id`.

2. **POST Endpoint: Add Data**
   Define a `POST` endpoint that allows you to add new records to the database. Use FastAPI's `Request Body` features to receive data in JSON format.

3. **PUT Endpoint: Update Data**
   Define a `PUT` endpoint that updates existing data in the database. Specify the record by `id` and use the request body to update fields.

4. **DELETE Endpoint: Remove Data**
   Define a `DELETE` endpoint that removes data from the database based on `id`.

Task 4: Test Your API

Objective:
You will use **Postman** or **cURL** to test each of the CRUD operations on your FastAPI application.

1. **Test GET Endpoint:**
   - `GET /data/` - Retrieve all records.
   - `GET /data/{id}` - Retrieve a specific record by ID.
2. **Test POST Endpoint:**
   - `POST /data/` - Add a new record by sending JSON data in the request body.
3. **Test PUT Endpoint:**
   - `PUT /data/{id}` - Update a record by ID, using JSON data in the request body.
4. **Test DELETE Endpoint:**
   - `DELETE /data/{id}` - Delete a record by ID.

Note: Document your testing steps in the README file, explaining how to use each endpoint.

Task 5: Git Branching and Collaboration

Objective:
You will continue to work with **Git** to manage your code and collaborate on **GitHub**.

1. **Create a `feature/api-endpoints` branch:**
   Write all API-related code in this branch.

2. **Merge with `dev` and `prod` branches:**
   Once you've completed your tasks, merge `feature/api-endpoints` into `dev` for testing. After testing, merge `dev` into `prod`.

3. **Collaborate with a Team Member:**
   Invite a teammate to review your code on GitHub. They should create a `review` branch, make suggestions, and submit a pull request back to your `dev` branch.

PART C - Submission Instructions

You are expected to submit your assignment by the end of the week. Submission will be done via Google Forms. You are encouraged to put your work online to help build your portfolio and show your learning

For Twitter Submission:
- Tag the official TDI page: @TDataImmersed
- Tag Annie @DabereNnamani
- Tag the project coordinator @The_Jonathaan
- Tag @python
- Tag @JOloganj
- Use the hashtag TDI

For LinkedIn Submission:
- Tag the TDI page: @TheDataImmersed
- Tag Annie @AnneNnamani
- Tag @josephologunja

For this assignment, you will be required to submit your work via GitHub.

Push your code to GitHub:
- Ensure all code is pushed to GitHub, with the appropriate branching (`dev`, `staging`, `prod`).
- Submit your GitHub repository link and ensure that your `README.md` file explains how to set up the project, run the workflows, and how collaborators can contribute.

PART D - Correction Class

Correction classes will be held every Saturday from 4 pm to 6 pm Nigerian Time on the TDI official Discord or a Google Meet link will be shared before the class.

Good luck with your assignment! We hope you find it both challenging and rewarding. If you have any questions, feel free to reach out to the mentors or the community on the TDI platform.