eBay-like Marketplace - SQLAlchemy & FastAPI Implementation

In this assignment, you will implement a simplified online marketplace inspired by eBay using FastAPI and SQLAlchemy. The goal is to build a database model and expose it through a REST API.

You have access to a GitHub repository containing a template project with FastAPI, SQLAlchemy, and basic configurations.

https://github.com/profSorbus/boilerplatePython

You will modify the code of this codebase to solve this lab.

I strongly advise that you clone my repository on your computer and rename it. You can start from the code of last time if you want. You can even make it a repository on your own github account by using the fork function on github and clone the repository from there.

Part 1: Database Modeling

In this part, you will modify the provided data model to build one that correspond to this problem :

A website like ebay has users. We want to store their name, username, their email address. We also want to be able to know for how long they have been users of the website. Finally, a user will have a rating based on past transactions and the password_hash that allows for authentication.

These users can make transactions with other users. A transaction is between two users, has a price and is also dated. During the transaction, users exchange an item.

The items have a name, a description, a price, belong to one owner. Three status are possible for an item : Available, Sold, Removed

1. **Build the data model on dbdiagram.io or on paper to solve so it contains all classes and attributes necessary.** You will also think of the relations between the table/classes and their multiplicity.

Before moving on to question 2, show me the data model to make sure that the one is close enough to the solution to continue. If you feel at ease with SQL and with the labs so far, you can elaborate and develop the data model to make the application more complex, if not, we will stick with a simple model. You will always be able to update the data model later or expand it so if you are unsure, pick the simple option.

2. Implement that data model in SQLAlchemy by modifying the file : db_model.py in the db folder
3. Modify the db_init.py, and create some sample data to fill the model (This script will help you create the database and fill some beginning data, and reset the database when needed).
4. Create the database in sqlite using the db_init.py

Part 2: API Implementation with FastAPI

1. The company expects you to implement the following methods with the API, so that the client can be built :

    - Create a new user
    - Retrieve a user by ID
    - Update user information
    - Delete a user

For these methods, you will not take into account privileges and admin rights needed. The API will be secured later. We just expect these methods to be available through the API.

2. We also want to build methods that allow to manage items.

    - Creating a new item listing
    - Retrieving all available items
    - Retrieving items by seller
    - Updating an item’s details

3. Finally, we need to build methods that will allow an user to make transactions

    - Implement a route allowing a user to purchase an item
    - Ensure that the same item cannot be purchased multiple times.
    - Update the item's status after a transaction.

Part 3: Advanced Features (Optional)

1. **User Authentication**

    ○ Implement a simple authentication system using password hashing and JWT tokens.
    ○ You can find the tutorial needed to implement those steps in here :

https://fastapi.tiangolo.com/tutorial/security/oauth2-jwt/

    - Restrict actions like purchasing or listing items to authenticated users.

2. **User Ratings**

    - Add a method where only a buyer can rate a seller with which he has done a transaction with.

    - The seller's rating should be an average of all received ratings. It should also be updated a new user rating.

3. **Filtering and Searching**

    - Add query parameters to search for items based on price range, keywords, and seller rating.

What you should do if you have time :

- Update the readme of the github repository so it follows the new functionalities of the repo
- Update the documentation/create documentation for functions you do
- Update the requirements.txt if needed