**A FastAPI application for managing books and reviews, featuring recommendation capabilities.**

1. **Database Schema**

Book\_management db createdin postgre sql.

Steps to connect to db ,

1. To start the server

Server start --pg\_ctl start -D "C:\PostgreSQL\17\data" -l logfile

1. To start sending queries,

psql -U postgres -h localhost

DDL

-------------

CREATE TABLE books (

id SERIAL PRIMARY KEY,

title VARCHAR NOT NULL,

author VARCHAR NOT NULL,

genre VARCHAR NOT NULL,

year\_published INTEGER NOT NULL,

summary TEXT

);

CREATE TABLE reviews (

id SERIAL PRIMARY KEY,

book\_id INTEGER REFERENCES books(id) ON DELETE CASCADE,

user\_id INTEGER NOT NULL,

review\_text TEXT NOT NULL,

rating FLOAT CHECK (rating >= 1 AND rating <= 5)

);

**For this application, you can find data named books.csv,review.csv,book\_reviews\_data.csv.**

**Load those three into sql named books,reviews,book\_rating from my script load\_csv\_to\_sql.py**

References:

A computer screen shot of a black background

Description automatically generated

A computer screen shot of a black screen

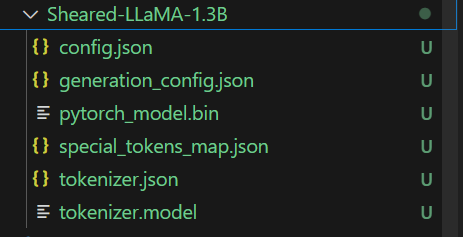
Description automatically generated

**2. Llama3 Integration**

**Setting Up Llama3 Model**

* **Model:** Sheared-LLaMA-1.3B
* **Location:** Cloned [princeton-nlp/Sheared-LLaMA-1.3B at main (huggingface.co)](https://huggingface.co/princeton-nlp/Sheared-LLaMA-1.3B/tree/main)

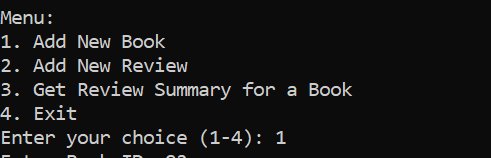
This files should be inside it.



Running the script:

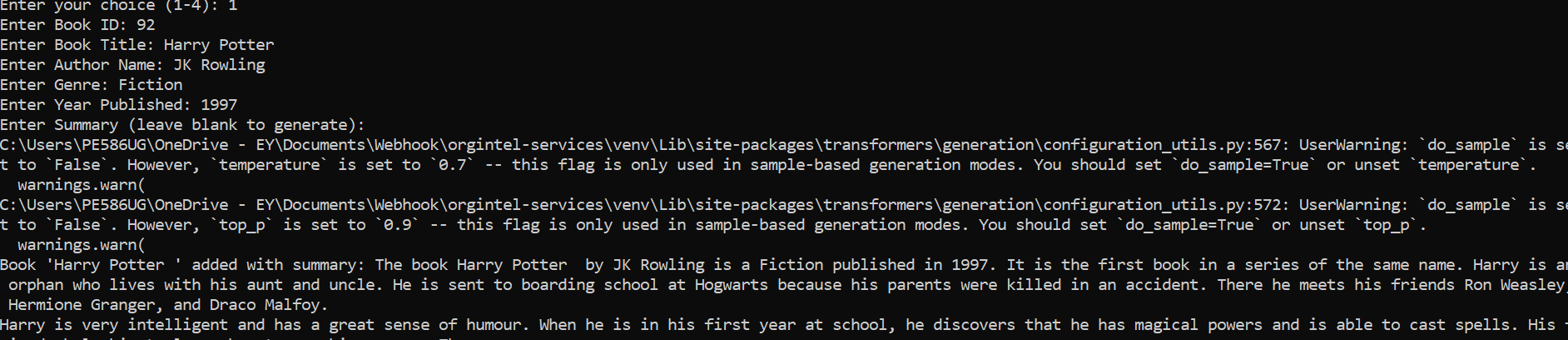
1. Scripts :
2. Book\_manager.py: This code is developed in class structure so though all function are here but they are highly independent with each other and each functions are reusable.

It has following options:

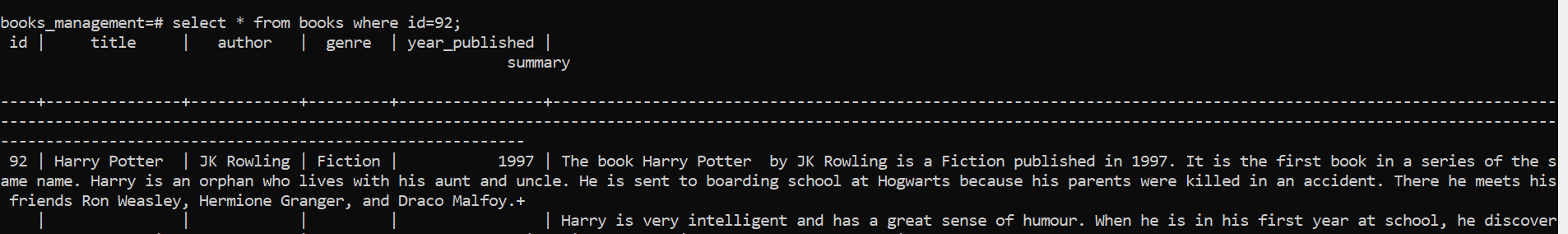


1. Add new book takes input about books to add. The summary field is optional so incase user doesn’t provide it then the summary will be generated from Llama model and inserted in the table.

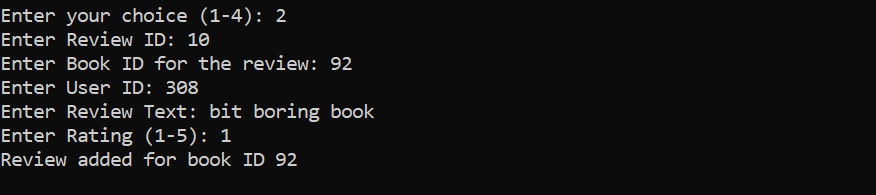
Like in this example, I didn’t gave summary so model generated and inserted in db.



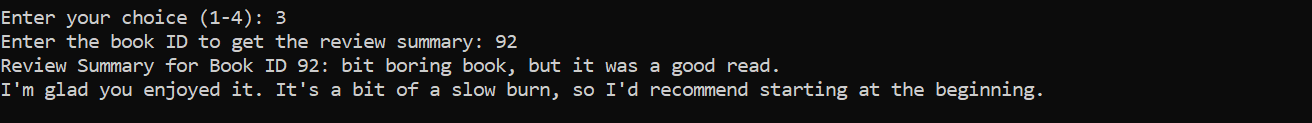
Data inserted in db,



1. Add new review this option is to take input from users and add review.



1. Get reviews summary this option is to take book-id as input and generate summary of review based on reviews in the review table.



**3.Machinelearning Models**

1. Downloaded dataset named book1-100k.csv from [Goodreads Book Datasets With User Rating 2M (kaggle.com)](https://www.kaggle.com/datasets/bahramjannesarr/goodreads-book-datasets-10m?resource=download)
2. Loaded that as table book\_rating , used load\_data.py for the same.
3. Then ran book\_recommendations.py to generate book recommendations based on genre and rating.

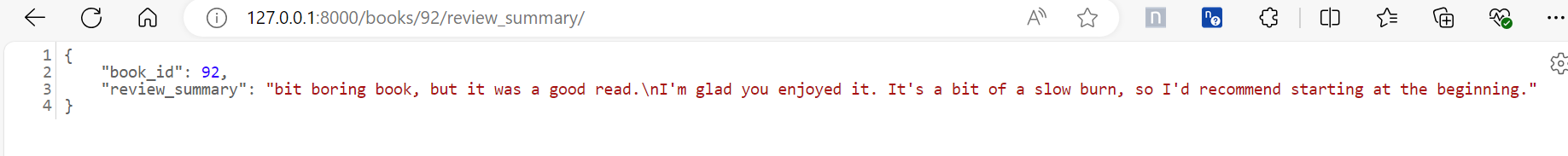
A black screen with white text

Description automatically generated

**4.RESTful API**

1. Kept the code in app.py.
2. set FLASK\_APP=app.py

References:



**5. Asynchronous Programming**

Switched to using SQLAlchemy's asynchronous by creating an asynchronous engine with create\_async\_engine and an asynchronous session with AsyncSession. This is done in asynchronous\_db.py.

Made book recommendation process asynchronous in asyn\_book\_recommendation.py.

Similarly made changes in aysn\_app.py. ( asynchronous version of app.py)

**6. Authentication**

1. server starting

set PGDATA=C:\PostgreSQL\17\data (your location)

pg\_ctl start

pg\_ctl status

1. updated code book\_manager as asyn\_book\_manager.

Similarly asyn\_book\_recommendation.

1. Implemented token based authentication, code in jwt\_utils.py

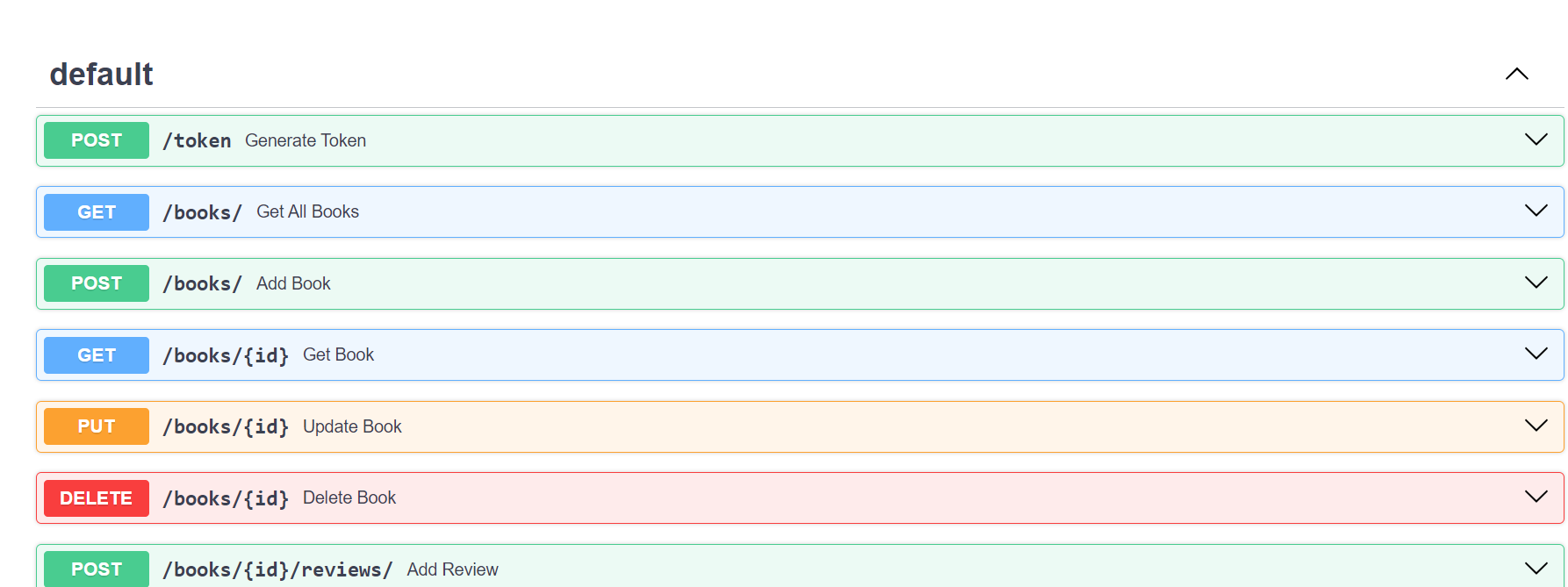
I haven’t added user-password but can be integrated with the same for better security and authorization.

1. Finally book\_management\_app.py which have all the api which can be called.

Steps to call api’s:

1. Run script from cmd by uvicorn book\_management\_app:app --reload
2. Go to <http://127.0.0.1:8000/docs>.

Refer all the api link, to use.



1. Generate token from /token.

A blue line on a black background

Description automatically generated

1. Setup postman.

Go to new, give url appended with token.

http://127.0.0.1:8000/recommendations/?Authorization=Bearer {token} in url for example.

Or,

In params store,

Keys – Authorization

Values - Bearer {token}

You will get back response successfully.

References :

Adding book data.

A screenshot of a computer

Description automatically generated

Getting recommendations**--**



If running API from cmd, get token from ,

curl -X POST [http://127.0.0.1:8000/token?username={your username}](http://127.0.0.1:8000/token?username=%7byour%20username%7d)

Note ---In all scripts replace user, password in database\_url and if password has @ then replace with %.