Sprints

Capstone Project – BookBazaar

"Library Management and Review System"

#team_8#

☒Objective :

A BookBazaar is a full-stack library management system integrating a relational database (SQLite) for structured data (books, authors, users) and a non-relational database (MongoDB) for reviews. The project includes Flask-based RESTful APIs that facilitate interaction with these databases, all hosted on Apache using mod_wsgi. The project aims to provide an end-to-end solution for a library management system while offering hands-on experience with database management, Python programming, and web deployment.

☒Key Features:

- ❖ Relational database (SQLite) to store book, author, and user data.
- ❖ Non-relational database (MongoDB) to store book reviews.
- ❖ Dual database integration (SQLite and MongoDB).
- ❖ CRUD operations on both databases through Python scripts.
- ❖ RESTful APIs with detailed error handling.
- ❖ Deployment using Apache (html) server and mod_wsgi.

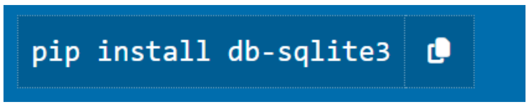
☒Technologies Used:

- ❖ **SQLite**: A relational database used for structured data (books, authors, users).
- ❖ **MongoDB**: A NoSQL database used for storing unstructured data (reviews).
- ❖ **Python**: Used to integrate both databases and create the Flask API.
- ❖ **Flask**: A lightweight web framework for building the RESTful APIs.
- ❖ **Apache**: A web server used to deploy Flask API using mod_wsgi.
- ❖ **Postman**: Tool used for testing API endpoints.

⊠Setup and Installation Instructions:

SQLite Setup :

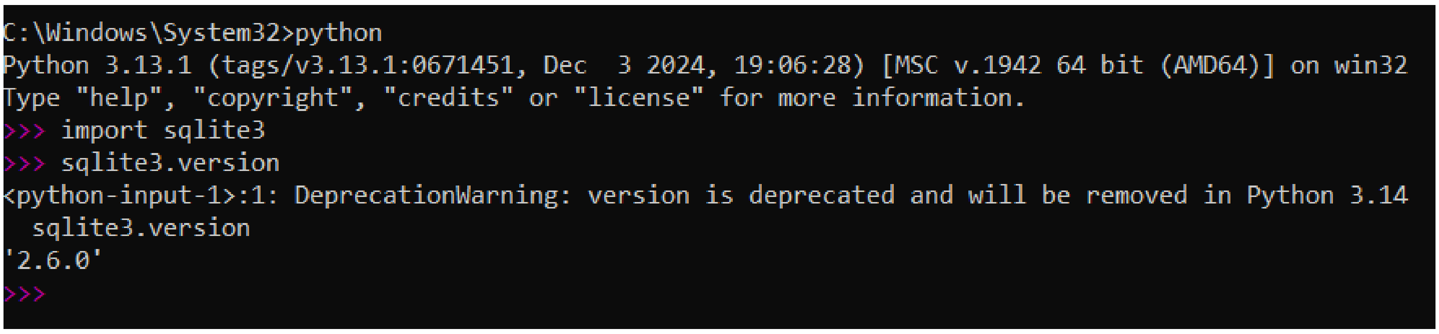
✓ Install SQLite :



```
pip install db-sqlite3
```

✓ Ensure SQLite is installed on your system and accessible :

1- Open Python by typing python in Command Prompt, or start an interactive Python shell.
2- Type the following to import the sqlite3 library:
3- Then, check the version



```
C:\Windows\System32>python
Python 3.13.1 (tags/v3.13.1:0671451, Dec  3 2024, 19:06:28) [MSC v.1942 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import sqlite3
>>> sqlite3.version
<python-input-1>:1: DeprecationWarning: version is deprecated and will be removed in Python 3.14
  sqlite3.version
'2.6.0'
>>>
```

✓ Create SQLite Database:

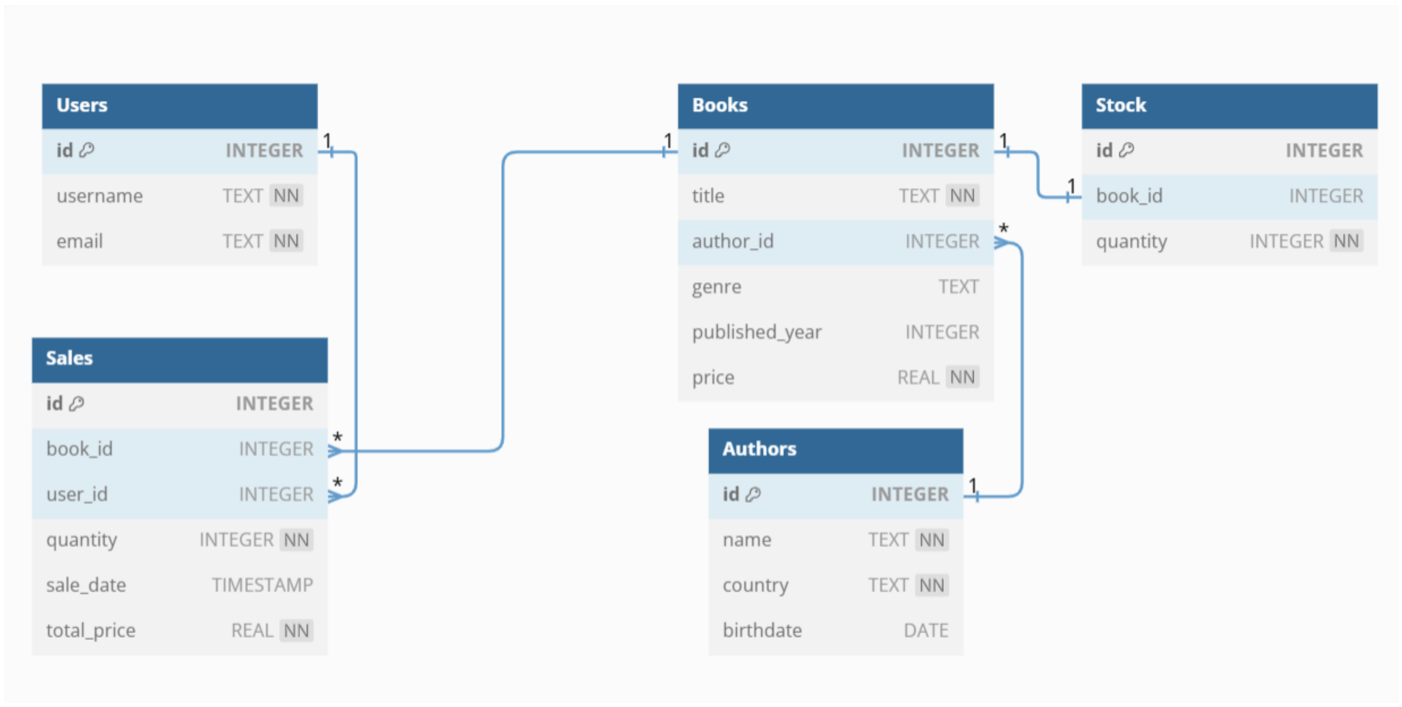
Use the Python script **init_data.py** to create the database (bookbazaar.db) and insert sample data for books, authors, and users, Stock and, Users:

- This script defines the database schema and populates the Books, Authors, Stock, Users, and Users tables with sample data.

✓ SQLite Tables:

- Users: Stores user details (e.g., username, email).
- Authors: Stores author details (e.g., name, country).
- Books: Stores book details (e.g., title, genre, author_id).
- Stock: Stores book stock details (quantity available for each book).
- Sales: Stores information about each sale (user, book, quantity, total price).

✓ SQL Schema:



✓ relations:

---> Users and Sales: A user can make multiple purchases, relation ---> (1-to-Many)

---> Authors and Books: An author can write multiple books, relation ---> (1-to-Many)

---> Books and Stock: Each book has a stock quantity, relation ---> (1-to-1)

---> Books and Sales: Each sale references a specific book, relation ---> (1-to-Many)

MongoDB Setup:

✓ Install MongoDB:

- " Follow the MongoDB installation instructions for your operating system. "
- Install pymongo for python:

```
pip install pymongo
```

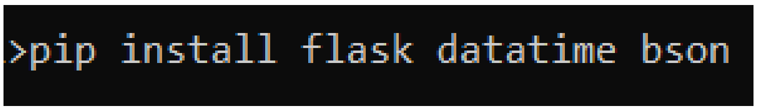
✓ Ensure pymongo is installed on your system and accessible :

- "Just like what we did with SQLite "

- ✓ Create MongoDB Database:
  - Use the script **init_data.py** to create the bookbazaar_reviews database and its reviews collection.

Install Dependencies:

- ✓ To set up the environment and install all required dependencies, run the following command:



```
>pip install flask datatime bson
```

  - Flask: Web framework to build RESTful APIs.
  - datetime: Python module used for handling date and time, particularly useful for managing timestamps in the database (e.g., sale dates).
  - bson: Library for working with BSON (Binary JSON) data format used by MongoDB, essential for manipulating MongoDB documents and handling object IDs.

⌧Deployment Configuration (Windows) :

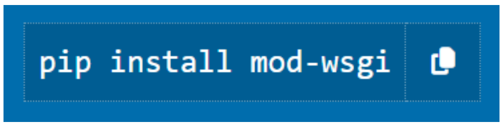
Install Apache Web Server:

- Download Apache for Windows from Apache Lounge.
- Follow the installation guide to set up Apache.

Install mod_wsgi:

" Mod_wsgi is required to host Python web applications on an Apache server, allowing them to communicate seamlessly."

- Install it using pip



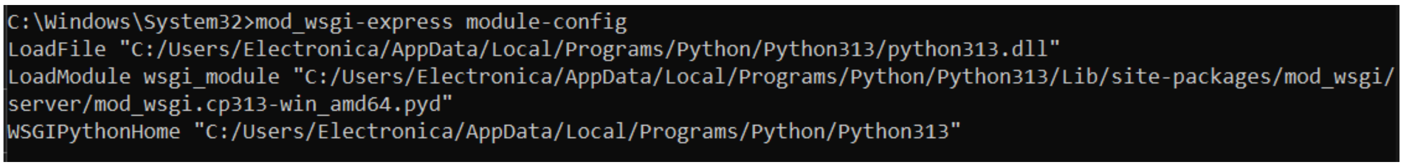
```
pip install mod-wsgi
```

Configure Apache to Serve Flask Application:

✓ **Update httpd.conf**

(After Install mod_wsgi using pip):

- using "mod_wsgi-express module-config" command to get the config which needed to load wsgi_module in apache



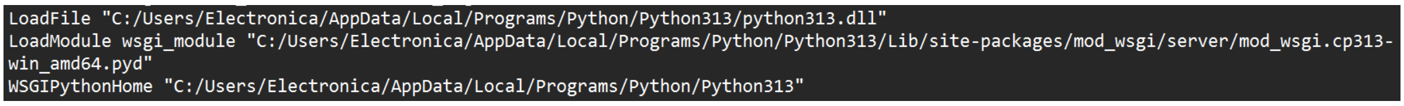
```
C:\Windows\System32>mod_wsgi-express module-config
LoadFile "C:/Users/Electronica/AppData/Local/Programs/Python/Python313/python313.dll"
LoadModule wsgi_module "C:/Users/Electronica/AppData/Local/Programs/Python/Python313/Lib/site-packages/mod_wsgi/
server/mod_wsgi.cp313-win_amd64.pyd"
WSGIPythonHome "C:/Users/Electronica/AppData/Local/Programs/Python/Python313"
```

✓ **Update httpd.conf:**

- Enable(loadModule) mod_wsgi

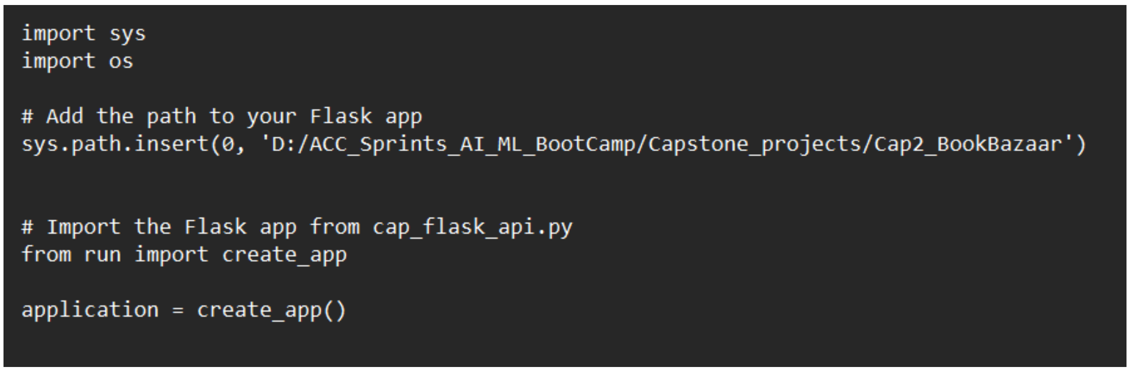
  (Add what we got from "mod_wsgi-express module-config")

- Uncomment this too " #Include conf/extra/httpd-vhosts.conf"



```
LoadFile "C:/Users/Electronica/AppData/Local/Programs/Python/Python313/python313.dll"
LoadModule wsgi_module "C:/Users/Electronica/AppData/Local/Programs/Python/Python313/Lib/site-packages/mod_wsgi/server/mod_wsgi.cp313-
win_amd64.pyd"
WSGIPythonHome "C:/Users/Electronica/AppData/Local/Programs/Python/Python313"
```

- Create cap_app.wsgi:



```
import sys
import os

# Add the path to your Flask app
sys.path.insert(0, 'D:/ACC_Sprints_AI_ML_BootCamp/Capstone_projects/Cap2_BookBazaar')


# Import the Flask app from cap_flask_api.py
from run import create_app

application = create_app()
```

✓ Create Virtual Host Configuration:

- Add the following to httpd-vhosts.conf

```
<VirtualHost *:80>
    # Admin email and domain name for the Flask app
    ServerAdmin www.bookbazaar.test

    # The ServerName directive specifies the domain name of your Flask application
    # This is the URL that users will use to access the Flask app
    ServerName bookbazaar.test

    # Document root is the directory containing your Flask app
    DocumentRoot "D:/ACC_Sprints_AI_ML_BootCamp/Capstone_projects/Cap2_BookBazaar"

    # Default log file locations (for error and access logs)
    ErrorLog "C:/Apache24/logs/error.log"
    CustomLog "C:/Apache24/logs/access.log" combined


    #WSGIScriptAlias is used to define the location of the WSGI application
    WSGIScriptAlias / "D:/ACC_Sprints_AI_ML_BootCamp/Capstone_projects/Cap2_BookBazaar/wsgi.py"


    #  allowing access to the directory where the Flask app resides
    # 'Require all granted' means that all requests are allowed to access this directory
    <Directory "D:/ACC_Sprints_AI_ML_BootCamp/Capstone_projects/Cap2_BookBazaar">

        Require all granted
    </Directory>
</VirtualHost>
```

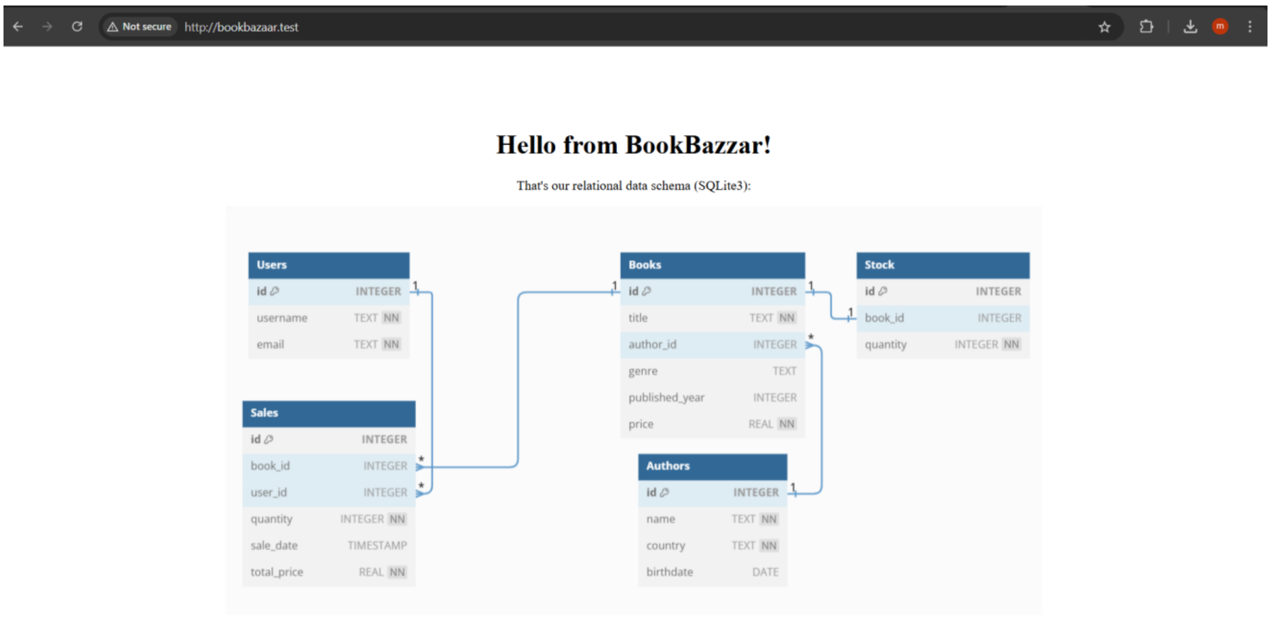
- Update hosts in this directory "C:\Windows\System32\drivers\etc":

```
127.0.0.1 localhost
127.0.0.1 usersapi.local
127.0.0.1 flaskapp.com
127.0.0.1 Sec_virtualdomain.com
127.0.0.1 BookBazzar.com
127.0.0.1 bookbazaar.test
```

----> Note: our BookBazzar.test (as out local domain) is what we care about here

Then restart the apache using " httpd -k restart" then test it :-

✓ Our Home page using apache:



Hello from BookBazzar!

That's our relational data schema (SQLite3):

☒API Documentation:

- ✓ **Base URL:**
  - **using standalone Flask:**
    - http://localhost:5000/
- ✓ **using Apache http:**
    - http://bookbazzar.com/
- ✓ **Endpoints:**
  - Books Management:
    - GET /books: Retrieve all books.
    - POST /books: Add a new book.
    - PUT /books/<id>: Update book details.
    - DELETE /books/<id>: Delete a book.
  - Reviews Management:
    - GET /books/<id>/reviews: Retrieve reviews for a specific book.
    - POST /books/<id>/reviews: Add a review to a book.
    - PUT /reviews/<review_id>: Update a review.
    - DELETE /reviews/<review_id>: Delete a review.

☒Troubleshooting Tips

Database Errors:

- Ensure SQLite and MongoDB services are running.
- Verify database connections in Python scripts.

Deployment Issues:

- Check Apache httpd.conf for correct paths.
- Ensure mod_wsgi is installed and configured properly.

API Errors:

- Use Postman to verify JSON formatting in requests.
- Debug Flask errors with logs.