**Project Structure**

* **Book\_Manage.py**: Book Management Microservice (Provides API for CRUD operations).
* **Book\_cart.py**: Shopping Cart Microservice (Handles user cart operations and inventory synchronization).
* **Database**: PostgreSQL (Stores book and shopping cart data).
* **Book\_Dashboard.py**: Administrator Dashboard (Manages book data).

**Code Structure**

| **File Name** | **Description** |
| --- | --- |
| Book\_Manage.py | Provides book CRUD API, manages inventory |
| Book\_cart.py | Shopping cart API, supports adding, removing, and checkout |
| Book\_Dashboard.py | Admin UI for book management |

**Shopping Cart Microservice (Book\_cart.py)**

**Features:**

* User login (JWT authentication).
* Retrieve cart contents (Query from database & catalog microservice).
* Add items to cart (Checks stock before deduction).
* Update cart item quantity (Prevents overselling).
* Remove items from cart (Restores inventory).
* Clear cart (Deletes all items and restores inventory).

**Key Technologies:**

* Flask-JWT-Extended: User authentication.
* PostgreSQL + SQLAlchemy: Stores shopping cart data.
* Redis Caching: Reduces database queries and improves performance.

**API Examples:**

| **API Route** | **Method** | **Description** |
| --- | --- | --- |
| /login | POST | User login, returns JWT token |
| /cart | GET | Retrieve user shopping cart |
| /cart | POST | Add item to shopping cart |
| /cart/<book\_id> | PUT | Update cart item quantity |
| /cart/<book\_id> | DELETE | Remove item from cart |
| /cart/clear | DELETE | Clear shopping cart |

**Book Management Microservice (Book\_Manage.py)**

**Features:**

* Admin login (JWT authentication).
* Retrieve all books.
* Query book by ID.
* Add a new book.
* Update book details (Price, stock).
* Delete books.

**API Examples:**

| **API Route** | **Method** | **Description** |
| --- | --- | --- |
| /login | POST | Admin login |
| /books | GET | Retrieve all books |
| /books/<book\_id> | GET | Query specific book |
| /books | POST | Add a new book |
| /books/<book\_id> | PUT | Update book details |
| /books/<book\_id> | DELETE | Delete book |

**Book Management Dashboard (Book\_Dashboard.py)**

**Features:**

* Admin login.
* View book list.
* Search books.
* Add books.
* Update book details.
* Delete books.

**Technology Stack:**

* Dash + Dash Bootstrap Components: Builds admin UI.
* Plotly Express: Data visualization (Expandable).
* Flask API Requests: Interacts with Book\_Manage.py.

**Functional Components:**

| **Component** | **Function** |
| --- | --- |
| Login Form | Admin login |
| Book List | Display & search books |
| Add Book | Input title, price, stock & submit |
| Update Book | Select and modify book details |
| Delete Book | Select and remove book |

**Project Setup**

1 **Start Database**

Ensure PostgreSQL is running and create the database:

CREATE DATABASE bookstore;

2 **Run Book Management Microservice**

python Book\_Manage.py

API runs at: http://127.0.0.1:5000

3 **Run Shopping Cart Microservice**

python Book\_cart.py

API runs at: http://127.0.0.1:5001

4 **Run Admin Dashboard**

python Book\_Dashboard.py

Admin dashboard runs at: http://127.0.0.1:8050

**API Testing**

1 **Admin Login**

curl -X POST http://127.0.0.1:5000/login \

-H "Content-Type: application/json" \

-d '{"username": "admin", "password": "admin123"}'

Response:

{"token": "your\_jwt\_token"}

2 **Add a Book**

curl -X POST http://127.0.0.1:5000/books \

-H "Authorization: Bearer your\_jwt\_token" \

-H "Content-Type: application/json" \

-d '{"title": "Python Basics", "author": "John Doe", "price": 29.99, "stock": 10}'

3 **Add Item to Cart**

curl -X POST http://127.0.0.1:5001/cart \

-H "Authorization: Bearer your\_jwt\_token" \

-H "Content-Type: application/json" \

-d '{"book\_id": 1, "quantity": 2}'

4 **Retrieve Cart**

curl -X GET http://127.0.0.1:5001/cart \

-H "Authorization: Bearer your\_jwt\_token"

Response:

[

{

"title": "Python Basics",

"quantity": 2,

"total\_price": 59.98

}

]

**Conclusion**

* **Book Management Microservice**: CRUD operations for books, manages inventory.
* **Shopping Cart Microservice**: Handles cart operations, maintains inventory consistency.
* **Admin Dashboard (Dash)**: Provides a user-friendly interface for book data visualization.
* **JWT Authentication**: Ensures secure access and prevents unauthorized usage.