Project Report: Bookstore Management System (FastAPI + Streamlit)

Author: Prajyukta Bhanja
Date: 25-10-2025
Project Type: Internship Project

---

1. Project Overview

The Bookstore Management System is a full-stack application that allows users to **manage books** (CRUD operations) using a **FastAPI backend** and a **Streamlit frontend**.
It is designed to be **hackathon-ready**, with options for analytics, AI recommendations, and CSV export/import.

Technologies Used:

- Backend: Python 3.11, FastAPI, SQLAlchemy, SQLite
- Frontend: Python, Streamlit
- API Testing: Swagger UI, Postman
- Package Management: pip, conda
- Optional: AI APIs (OpenAI/HuggingFace) for recommendations

---

2. Project Architecture

```
project-1/
│
├── bookstore_api/
│   ├── main.py        # FastAPI app
│   ├── models.py      # SQLAlchemy models
│   ├── schemas.py     # Pydantic schemas
│   ├── crud.py        # CRUD operations
│   └── database.py    # DB connection & Base
│
├── streamlit_app.py   # Frontend dashboard
├── requirements.txt   # Dependencies
└── books.db           # SQLite database
```

Flow:

1. User interacts with **Streamlit UI**.
2. Streamlit calls **FastAPI endpoints** (GET, POST, PUT, DELETE).
3. FastAPI performs database operations via **SQLAlchemy ORM**.
4. Responses are displayed on the frontend.

3. Backend (FastAPI)

Setup:

```
uvicorn bookstore_api.main:app --reload
```

Endpoints:

Method	Endpoint	Description	Body (JSON) Example
POST	`/books`	Add a new book	`{ "title": "Solo Leveling", "author": "Chugong", "price": 399.0 }`
GET	`/books`	List all books	-
GET	`/search?query=...`	Search books by title/author	-
PUT	`/books/{id}`	Update a book	`{ "title": "Solo Leveling Vol.2", "author": "Chugong", "price": 450 }`
DELETE	`/books/{id}`	Delete a book	-

Database:

- SQLite `books.db`
- SQLAlchemy ORM models and Pydantic schemas

4. Frontend (Streamlit)

Setup:

```
streamlit run streamlit_app.py
```

Features:

- Display all books in a table
- Add, update, delete books via buttons
- Search functionality
- Connected to FastAPI backend via `requests` library
- Optional analytics charts (books per author, price distribution)
- Optional AI recommendations

Sample Streamlit code snippet:

```
import streamlit as st
import requests

API_URL = "http://127.0.0.1:8000"

st.title("Bookstore Dashboard")

# Fetch books
response = requests.get(f"{API_URL}/books")
books = response.json()
st.table(books)
```

---

5. Testing the API

- **Swagger UI:** `http://127.0.0.1:8000/docs`
- **Postman:** Test CRUD operations

Example Workflow:

1. Add book → Check table in Streamlit → Update book → Delete book

---

6. Bonus Ideas (Hackathon Ready)

- Analytics Dashboard: Books per author, price trends
- AI Recommendations: Use OpenAI or HuggingFace to suggest books
- Authentication: JWT for admin users
- CSV Export/Import
- Frontend Enhancements: Search filters, charts, responsive UI

---

7. Challenges Faced & Fixes

- **Python 3.12 incompatibility with Uvicorn ***** `[standard]` ***** dependencies** → Resolved by using Python 3.11 environment
- **Module import errors (**** `database` ***************, ***** `models` *****)** → Resolved by proper `bookstore_api` package structure
- **Connection refused in Streamlit** → Resolved by ensuring FastAPI backend is running before starting Streamlit

---

8. Future Improvements

- Deploy on cloud (AWS/GCP/Heroku)
- Add user roles (admin, reader)
- Integrate real-time notifications
- Implement search with fuzzy matching

End of Report