ADVANCE PROGRAMMING FINAL REPORT

Name – Saksham Jain

Student ID- 100004921

SRH University

Course Name – Applied Mechatronics Systems

Instructor: Esteban Pozo

Due Date: December 12, 2025

Abstract

This project presents the design and implementation of an online book library system developed using Python. The system follows a client–server architecture consisting of a Flask-based backend and a graphical user interface frontend. The application enables users to browse, search, create, and delete book entries stored persistently on the local system. Each book is described using metadata including title, author, publication year, and category.

The backend exposes RESTful HTTP endpoints to manage the library data, while the frontend provides an intuitive interface for interacting with these endpoints. Persistent storage is implemented using a JSON file, allowing the library data to be saved and reloaded across application restarts. The solution emphasizes modular design, input validation, exception handling, and separation of concerns.

In addition to implementation, the project includes conceptual design, testing, and critical discussion of strengths, weaknesses, and future improvements. Unit tests were implemented to verify both backend and frontend functionality. The result is a functional and extensible book library application that fulfills the requirements of the Advanced Programming assignment.

Contents

Introduction

This project was developed as part of the Advanced Programming module and focuses on creating an online library system for managing books. The objective is to design and implement a software solution that allows users to interact with a digital book collection through a graphical user interface connected to a backend server.

The system stores metadata for each book, including its title, author, publication year, and category. Users can retrieve all available books, filter them by category, search for a specific book by name, view detailed metadata, and perform creation or deletion operations. The backend is implemented using Flask and communicates with the frontend via HTTP requests. The frontend is implemented as a desktop GUI application.

The project follows best practices in software development, including modular architecture, persistent storage, exception handling, and automated testing. The design and implementation are based on the assignment requirements and guided by a reference implementation structure .

Task 1

Concept and System Design

Requirements

Functional Requirements

- The system must store metadata for books:
- Title
- Author

- Publication year
- Category
- Supported categories include:
- Novel
- Philosophy
- Poetry
- The backend must provide HTTP endpoints to:
- List all books
- Filter books by category
- Search for a book by exact title
- Retrieve metadata of a single book
- Create a new book entry
- Delete an existing book
- The GUI must allow users to:
- View all books
- Filter books by category
- Search by title
- View metadata of selected books
- Add new books
- Delete books

Non-Functional Requirements

- Persistence: All book data must be saved in a JSON file.
- Reliability: Errors must be handled gracefully with meaningful messages.
- Maintainability: Code must be modular and well structured.
- Portability: The application must run on any system with Python installed.

Architecture

The system follows a client–server architecture:

Backend (Flask)

- Handles HTTP requests and responses
- Validates user input
- Manages persistent storage
- Returns JSON-formatted data

Storage Layer

- Loads and saves book data to a JSON file
- Ensures unique identifiers for books
- Handles file-related exceptions

Model Layer

- Represents book data using structured Python objects

Frontend (GUI)

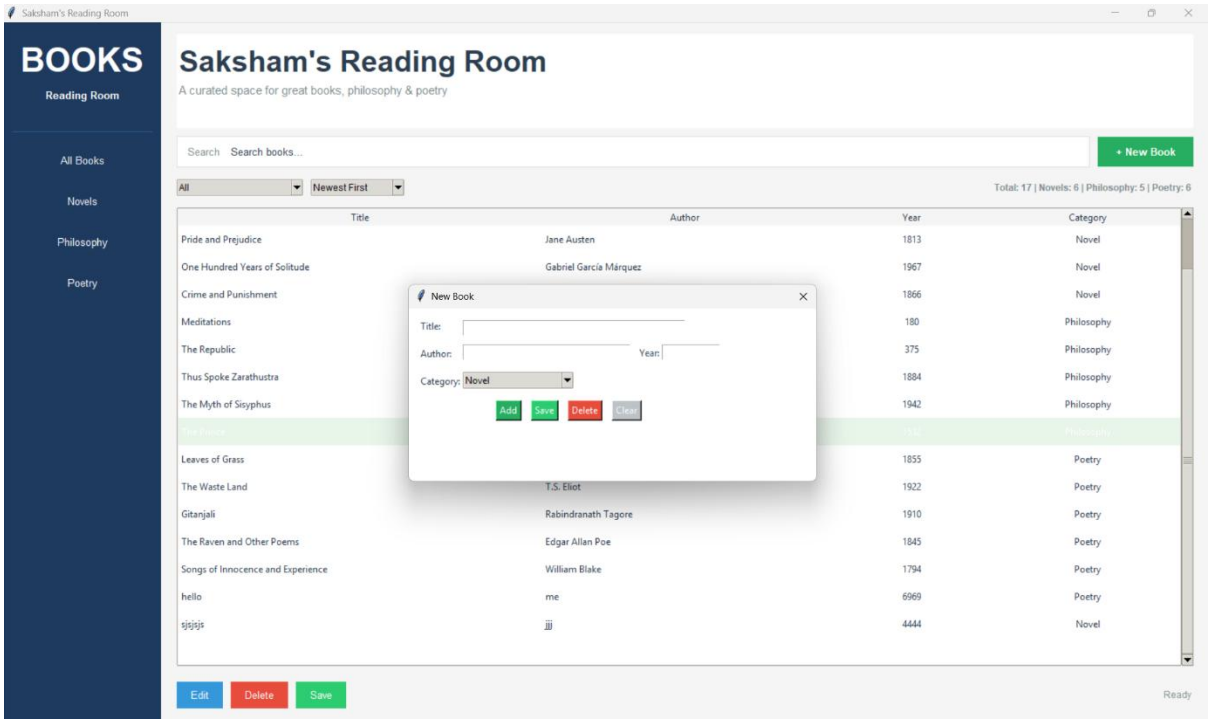
- Displays book lists and metadata
- Sends HTTP requests to backend
- Handles user interactions

Communication

- Frontend and backend communicate via HTTP using JSON data format

GUI Layout Sketch

[FIGURE 1]



Description of layout:

- Left panel: navigation and category filters
- Center panel: list of books
- Right panel: metadata display and action buttons
- Top bar: search input and add-new-book button

Exception Handling

Potential errors and solutions include:

- Missing input fields → backend returns validation error

- Invalid category selection → blocked by backend logic
- Corrupted JSON file → system reinitializes storage safely
- Backend unavailable → GUI displays error dialog
- Deleting a non-existing book → backend returns 404 error

Task 2

Implementation

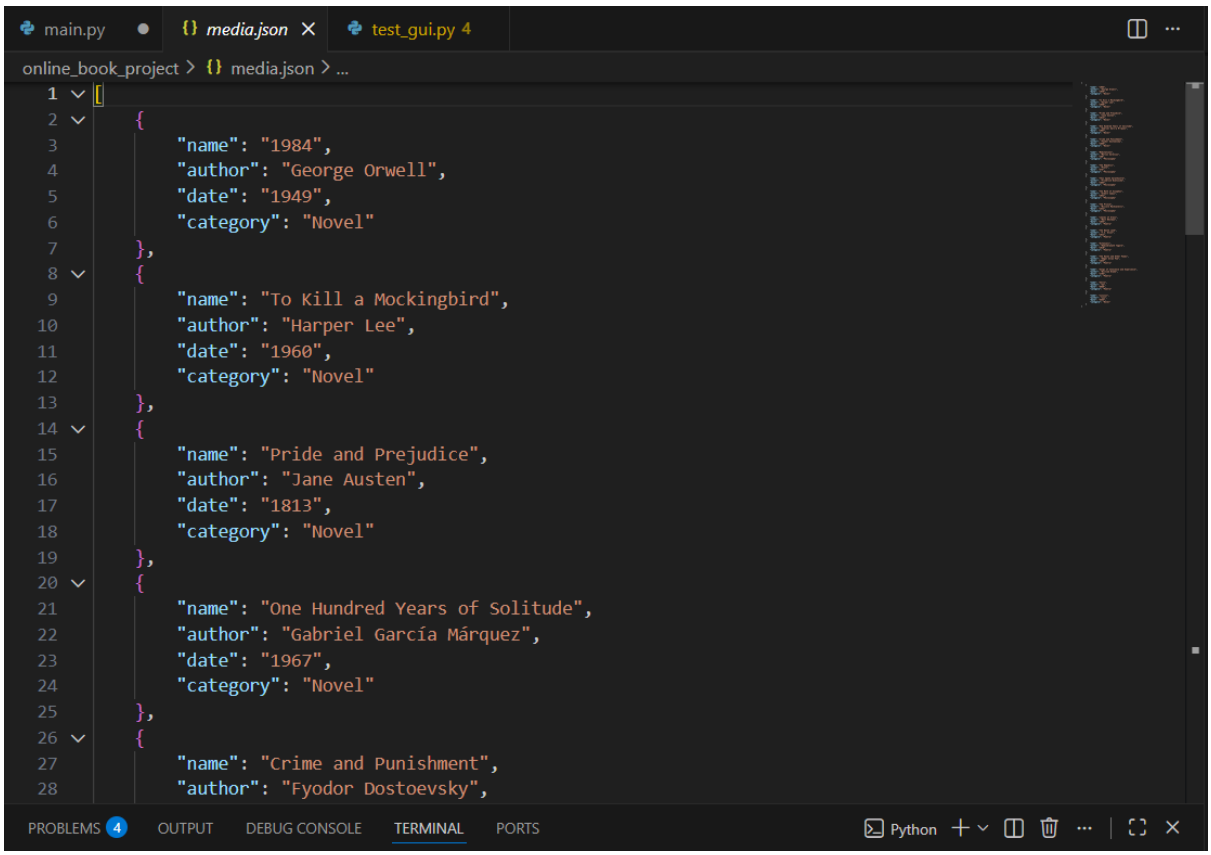
Backend Implementation

The backend is implemented using Flask and provides RESTful endpoints for managing book data. Each endpoint performs input validation and returns appropriate HTTP status codes. The backend is responsible for creating, retrieving, filtering, searching, and deleting book records.

Model and Storage Implementation

The book model is defined using structured Python classes to ensure consistency. The storage module handles reading and writing data to a JSON file, ensuring persistence across sessions. Auto-generated identifiers ensure that each book entry is unique.

[Insert Figure 2 here – JSON file structure]



Frontend Implementation

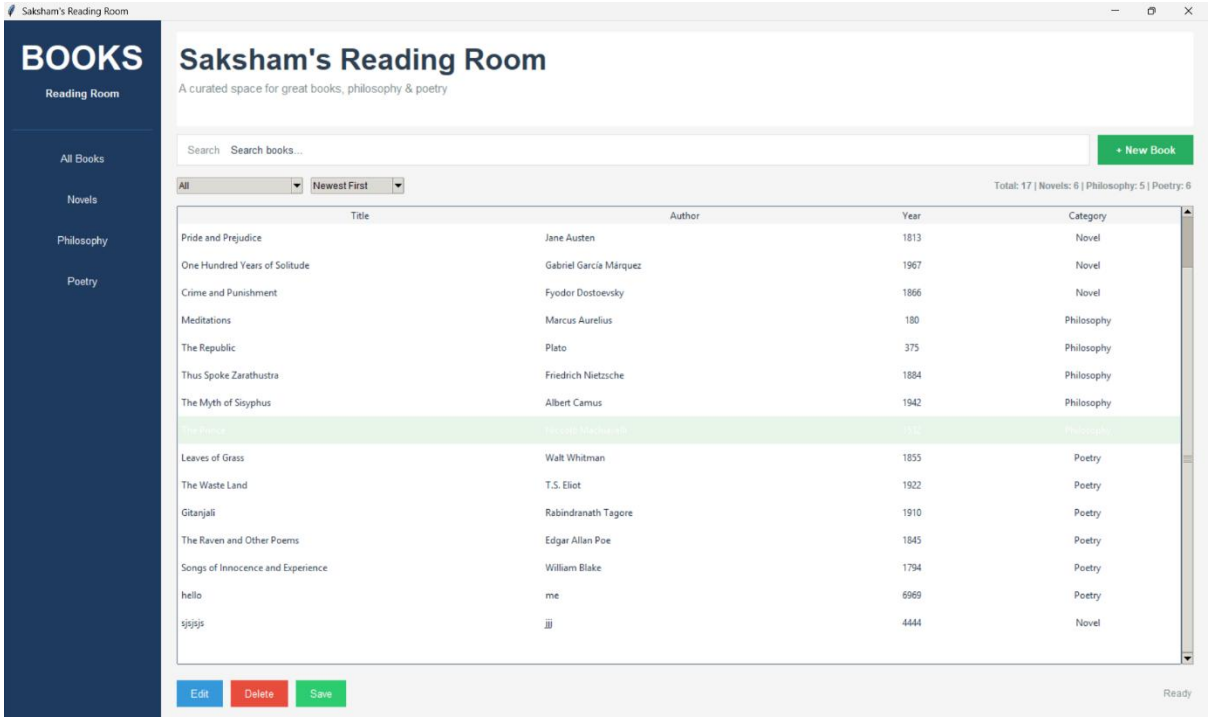
The frontend is implemented as a desktop GUI application. It provides:

- A list view for displaying books

- Dropdown menu for category filtering
- Search field for exact title lookup
- Buttons for creating and deleting books
- A detail panel for viewing metadata

User actions trigger HTTP requests to the backend, and responses are displayed dynamically in the interface.

Figure 3 here – Book Library GUI screenshot]



Task 3

Testing

Unit tests were implemented to ensure correctness and reliability.

Backend Tests

- Test book creation
- Test listing all books
- Test filtering by category
- Test search by exact title

Frontend Tests

- Test formatting of book display strings
- Test response handling from backend

These tests ensure that both backend logic and frontend presentation work as expected.

Task 4

Discussion

Strengths

- Clear separation between backend and frontend
- Persistent data storage
- Modular and maintainable code structure
- User-friendly GUI
- Proper input validation and error handling

Weaknesses

- No edit/update functionality for existing books
- JSON storage is not suitable for large datasets
- Limited GUI styling
- No pagination or sorting

Future Improvements

- Replace JSON storage with a database such as SQLite
- Add edit/update functionality

- Improve GUI design using PyQt or PySide
- Add user authentication
- Enable export of book data to PDF or CSV

References

Python Software Foundation. (2024). Flask Documentation. https://flask.palletsprojects.com/

Tkinter GUI Programming. https://docs.python.org/3/library/tkinter.html

Pytest Documentation. https://docs.pytest.org/en/stable/

Appendix

Use of AI Tools

AI tools were used to assist with architectural planning, clarification of REST API concepts, and documentation structuring. All generated content was reviewed and adapted manually to meet assignment requirements. The final implementation, testing, and verification were performed independently.

Source Tree Image