Digital Library System – Advanced Programming Project

Melwin Chandy

100001905

SRH University

Course number: Advance programming

Instructor: Esteban Pozo

Due Date: December 12, 2025

Contents

Abstract

This project focuses on designing and implementing a small digital library system using Python. The goal was to create a simple but functional application that allows users to store and manage different types of media such as books, movies, and magazines. The system is divided into two parts: a backend developed with Flask and a graphical user interface (GUI) built with Tkinter. Both components communicate through a HTTP communication. The project also includes automated tests for the backend and basic functional tests for the frontend. Throughout development, I applied Python best practices such as modular design, meaningful naming, and clear documentation. This report explains the concept, the implementation decisions, the testing process, and the final evaluation of the system.

Keywords: Flask, Tkinter, HTTP, JSON Storage

Introduction

Designing and implementing a simple software system to manage a digital library is the objective of this project . The application allows users to view media stored in a JSON file, add new media, delete entries, filter by category, search by name, and view details. The backend is responsible for data management and exposes endpoints for communication, while the frontend provides an interface for user interaction. Together, they form a small but complete Python application.

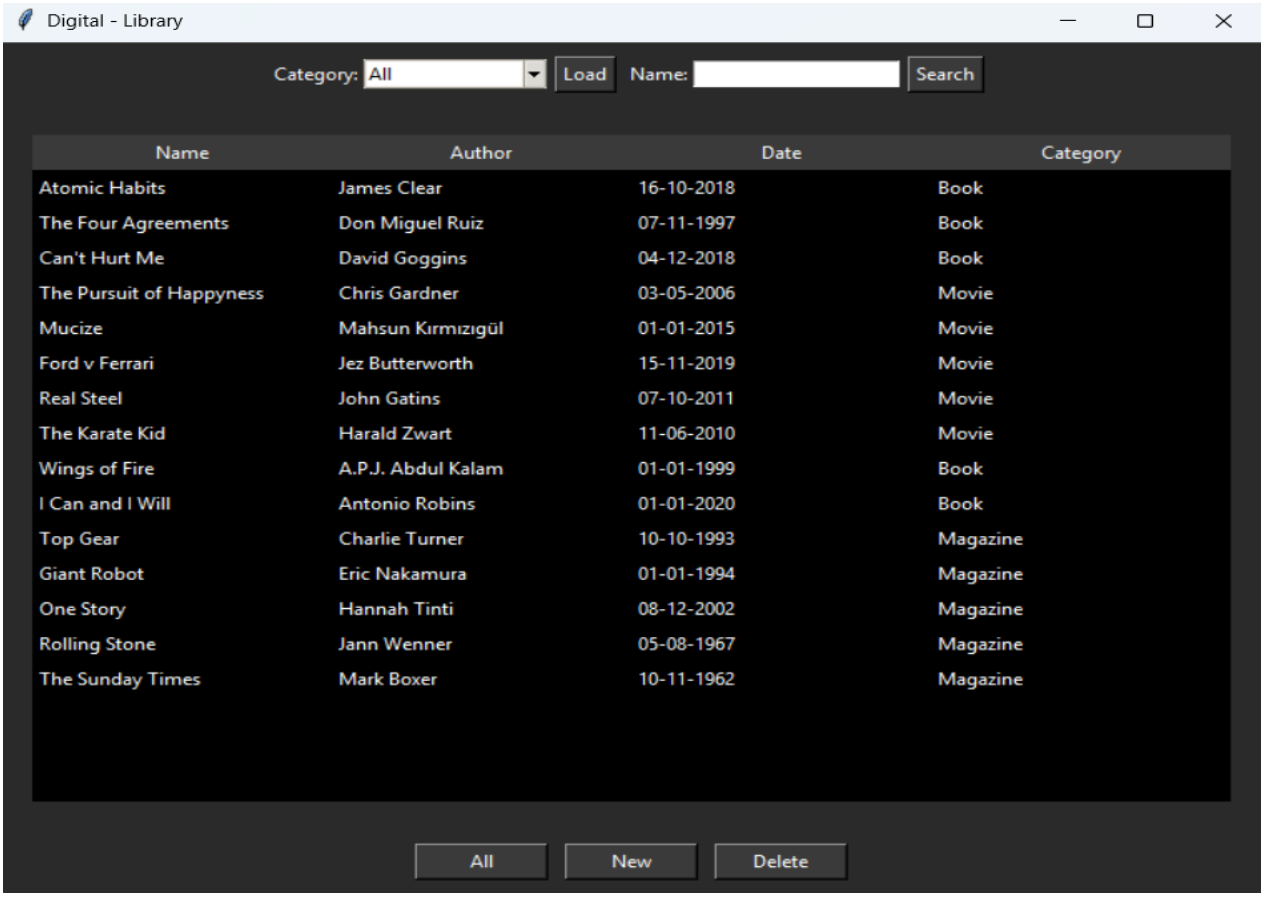

Figure 1: picture of the digital library

Figure 2. JSON file sample

Task 1

Concept and Requirements

Functional Requirements

The system must provide the features like - Display all media items, Filter media by category, Search for media by exact name, View detailed information about a selected item and add new media, Delete existing media, Store data persistently in a file and ensure each media item has a unique ID. These functions represent the core user actions expected from a basic digital library system.

Non-Functional Requirements

The GUI must be simple, clean, and easy for beginners. All operations (loading, searching, adding) should respond quickly. Data should not be lost and the program must handle errors gracefully. Clear structure, modules, classes, and comments are required. The program should run on any machine with Python installed.

Architecture Overview

The system uses a **client–server structure**:

- **Backend**
    - Built with Flask
    - Manages a JSON storage file
    - Implements REST API endpoints
    - Handles validation, data retrieval, addition, and deletion
- **Frontend**
    - Built with Tkinter
    - Sends HTTP requests to the backend
    - Displays data in a table
    - Includes input fields for searching and adding media

Communication (REST API)

The frontend and backend communicate using **HTTP requests**.

Data is always sent and received in **JSON** format.

Example:

- Frontend → GET /media → Backend returns list of all media
- Frontend → POST /media → Backend adds new item

This loose coupling makes the system flexible and easy to extend.

GUI Design Concept

The GUI consists of:

- A **top bar** with:
    - Category dropdown
    - Search bar
    - Load/Search buttons
- A **central table** showing the media list
- Bottom buttons:
    - Show All
    - Add New
    - Delete

The design focuses on clarity and minimalism.

Anticipated Exceptions & Handling

1. **Empty search field** → Show popup message
2. **Search result not found** → Show "No media exist"
3. **Adding incomplete media** → Error popup
4. **Backend not running** → Request failure (handled by try/except)
5. **Deleting nothing** → "Select an item first" warning

Task 2

Implementation

I structured my code using meaningful variable and function names, splitting the project into modules, adding comments, and organizing the logic into classes and clear functions. My implementation consists of two main components: a Flask-based backend and a Tkinter-based frontend. I used Flask for backend because it is lightweight, easy to understand, and ideal for creating simple REST APIs. I created a dedicated Storage class responsible for loading and saving data. This class encapsulates all data-related operations and keeps the backend clean and modular. It also includes functions for adding new items, deleting items, searching by name, filtering by category, and retrieving all stored media. Each item is given a unique ID, which prevents conflicts and ensures reliable data management.

The frontend was implemented using Tkinter. I designed the interface to be organized and user-friendly. The window includes a category dropdown, a search bar, a table to display all media items, and buttons for adding, deleting, and reloading the media list.
Communication between the frontend and backend is handled using the HTTP requests.

Task 3

Testing

1.Backend Tests

I created tests files that check listing of all media, Adding media, Searching by name, Retrieving an item by ID, Deleting a media item, Handling invalid input. These tests confirm that every endpoint behaves correctly.

2 .Frontend Tests

- I performed some basic functional GUI tests like verifying GUI loads without errors, Checking that the table populates with data, Testing search functionality, Testing error messages for invalid user input.

  All primary backend endpoints work as expected. Frontend functions run without crashing, and error messages appear properly. Through these tests I confirm that the system is stable for typical use.

Task 4

Discussion

Developing this project helped me understand how different parts of a software system work together. The challenge that I faced was connecting the GUI with the backend. Because, The Tkinter normally works locally and the Flask uses a server-style structure. Moreover, learning how to use HTTP requests inside a desktop app was an important new concept for me.

If I need to add something in my library, I would add a feature like edit the details of the media, a review window for each media items, and the status of the media item. Overall, the project helped me learn full-stack development in Python in a practical way.

References

References Pozo, E. (2025). Advanced Programming Project Description. SRH Fern Hochschule – The Mobile University.

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

Python Software Foundation. (2024). Tkinter Documentation. https://docs.python.org/3/library/tkinter.html