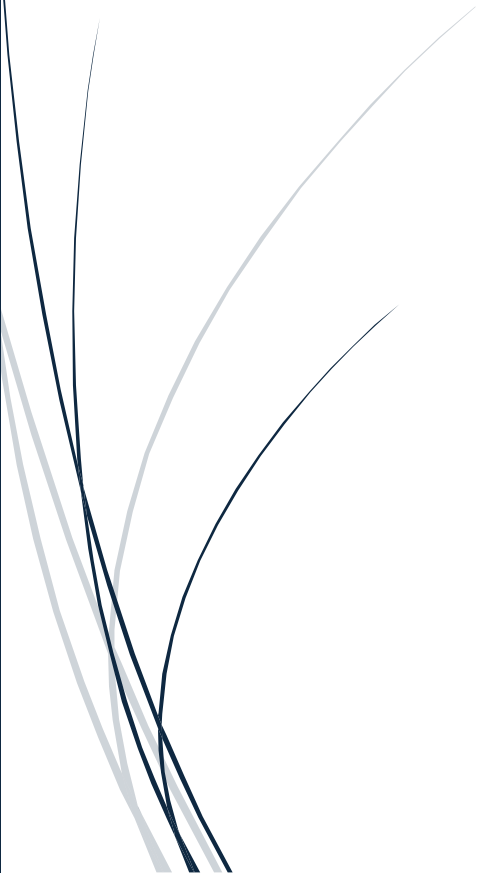


2/26/2024

Bookstore Management System



Ed Seo B00701238
Shu Yang B00839607
Dalhousie University
Prepared for Dr. Issam Hammad

Executive Summary

The Bookstore Management System (BMS) represents a transformative step in addressing traditional evolving challenges for bookstores in the digital age. With the increasing consumer shift towards digital books and audiobooks, physical bookstores need to adopt innovative strategies to enhance operational efficiencies and customer engagement. The BMS, developed in Python and hosted on a public GitHub repository, introduces a solution designed to streamline daily bookstore operations and improve the customer journey. Initially launched as a basic application with potential expansion to a web platform, BMS offers functionalities ranging from book and user management to sophisticated search capabilities. Through design and testing, including leveraging Kaggle's enriched dataset for realistic inventory simulation, the BMS is poised to redefine bookstore management. This report outlines the BMS' design process and critical functionalities with future enhancements to elevate its impact further.

Table of Contents

EXECUTIVE SUMMARY	I
INTRODUCTION.....	1
SYSTEM OVERVIEW	1
SYSTEM STRUCTURE AND DESIGN	1
BOOK MANAGEMENT.....	2
USER MANAGEMENT	3
SYSTEM STRUCTURE.....	4
TESTING AND IMPLEMENTATION.....	5
CONCLUSION	6
REFERENCE	7
APPENDIX A: STATE OF CONTRIBUTION	8
APPENDIX B – REPOSITORY.....	9

LIST OF TABLE

Table 1: Fields for Book Objects	2
--	---

Introduction

The literary landscape has witnessed a marked shift, with readers increasingly gravitating towards digital books and audiobooks, leading to a decline in the time spent with physical books. This trend poses significant challenges for traditional bookstores, indicating the urgent need for innovative strategies to attract and retain customers. This project focuses on developing a basic bookstore management system (BMS) to satisfy certain daily operations and elevate customer journeys. This system will focus on basic applications with the potential to expand into a web application that supports searching and daily managing. This report will go through the design process of the bookstore management system, introduce the functionalities, and discuss potential future improvements that could be made to improve the system.

System Overview

The foundation of BMS lies in its system requirements, encompassing both the functional and non-functional aspects essential for the system's efficacy and user satisfaction. With these principles, a systemic overview of the BMS will be provided in this section.

The development of the BMS utilizes the Python programming language, with the source code posted on a public GitHub repository. Core programming constructs encompass the definition of functions, object-oriented programming (OOP), error handling mechanisms, and sophisticated data manipulation techniques underpinning the system's backend architecture.

BMS invokes security protocols to safeguard data integrity and enforce access controls upon initiation. Authentication mechanisms are integrated to verify user credentials and delineate user permissions. The system is engineered to accommodate two distinct user roles: store administrators and customers, each with distinct functionalities. Administrators can onboard additional staff, modify book placement, and compile comprehensive inventory reports. Conversely, customers are granted the functionality to query the database for book searches exclusively.

System Structure and Design

BMS is founded on a streamlined two-tier architectural framework for book management and a three-tier architectural framework for user management, clearly segregating the application

logic and data storage layers to enhance system clarity and operational efficiency. This section explores the system's core objects and functionalities, essential for the operation.

Book Management

In the BMS environment, books are represented as objects, embodying a collection of attributes detailed in Tab 1. These attributes are initialized via the `__init__` method, Python's essential mechanism for object instantiation.

Table 1: Fields for Book Objects

Fields	Title	Authors	Original Language	First Published (year)	Sales (millions)	Genre	Stored Location (Shelf)	Inventory Level	Price
--------	-------	---------	-------------------	------------------------	------------------	-------	-------------------------	-----------------	-------

The data type is not specified in these fields because Python can dynamically select the data type. Specific error-handling processes are in place to ensure that Python selects the suitable data types. After initialization, a suite of methods associated with the book object facilitates the retrieval and manipulation of book data, enhancing the system's capability to access and modify book-related information efficiently.

Bookshelves, similar to books, are modelled as objects within the BMS. These objects incorporate five critical methods that significantly extend the system's capabilities. The primary method, `'load_from_csv'`, requires a file name argument to import book data from Comma-Separated Values (CSV) files, employing pandas (imported as `pd`) for data reading and extraction. This function efficiently transforms CSV data into a structured data frame, ensuring seamless integration and manipulation within the system.

The `'find_book'` method addresses the need for effective book retrieval, accommodating searches for partial titles and disregarding case sensitivity. The only input for this method is the partial book title, which will return all books with titles containing the input information. This functionality enhances customer service by allowing for flexible search queries and supports bookstore employees by enabling rapid, system-wide book searches.

The `'update_shelf'` method empowers administrators to adjust book locations, mirroring physical rearrangements in the bookstore's layout. Book titles and new shelves should be passed to the function to change the location stored in the system, and the new stored location and shelf letter will be provided once books are switched to their new home. This function ensures that the

data remains in sync with the bookstore's physical setup, providing customers with accurate book location information.

Lastly, a critical function of the BMS involves writing updated book data back to the CSV file, ensuring that subsequent data retrievals reflect the most current information. This capability is essential for maintaining the integrity and relevance of the bookstore's inventory data.

User Management

The system architecture includes an additional layer for user management, encompassing two primary roles: customers and administrators, each identified by a username and password. Customers possess limited privileges, primarily enabling them to search for books by complete or partial titles. In contrast, administrators are granted extended permissions beyond those available to customers.

The user hierarchy is implemented through inheritance, where the `'Customer'` class is derived from the base `'User'` class, and the `'Admin'` class extends the `'Customer'` class, encapsulating additional administrative functionalities. The uniform application of username and password initialization across users enhances system consistency. The `Customer` class introduces a `'search_book'` method, facilitating book searches by invoking the `'find_book'` method of the `'Bookshelf'` class. This process directly displays search results, including book locations and inventory levels, to the user. If a search does not match the database, the system prompts the user, allowing for exploring alternative functionalities.

The `'Admin'` class, an extension of the `'Customer'` class, introduces three specialized methods enhancing administrative capabilities within the BMS.

The first method enables an administrator to register another employee by supplying a new username and password. The system then validates this input against existing records to prevent duplicate registrations, ensuring each username within the BMS is unique. This functionality is crucial for expanding access to the system for new staff.

The second method, `'update_book_shelf'`, facilitates updating a book's physical location within the bookstore. It prompts the administrator for the book's title and new shelf location. Utilizing a process similar to the `'search_book'` function, it executes the

`'bookshelf.update_shelf'` method from the `'Bookshelf'` class to manage the relocation. Upon completion, the system notifies the administrator of the update's success, ensuring transparency in book management operations.

Lastly, the `'add_book'` method empowers administrators to augment the bookstore's inventory by introducing new titles. This function prompts essential book details from the administrator, streamlining the process of inventory expansion in response to bookstore needs or customer demand.

System Structure

This section will dive into the auxiliary functions of the system and its principal operation. The authentication process is implemented as a standalone function, not encapsulated within any class. It prompts users for their username and password. Incorrect credentials trigger a login failure notification, allowing users to attempt to log in again. Successful authentication greets the user with a personalized welcome message, incorporating the username.

The core (main) program operates within a persistent `'while'` loop, designed to continue until reaching a predefined termination condition. Upon initiating the loop, users are guided through the authentication process. Once verified, the system differentiates user roles to tailor the subsequent interface and options available. Customers are directed to the `'search_book'` function, a feature of the `'Customer'` class, allowing them to query the bookstore's inventory or choose to exit, breaking the loop and concluding the session. In contrast, administrators are granted access to a broader range of functionalities. The loop for administrators persists until they log off, maintaining the session's continuity for executing multiple administrative actions.

By designing these objects and methods, the BMS secures a high level of operational functionality and user engagement, adeptly catering to the comprehensive demands of bookstore management and operations. This structured approach to object and function definition underpins the system's architecture, facilitating a robust and efficient mechanism for managing bookstore inventory.

Testing and Implementation

The BMS implementation phase leveraged a dataset from Kaggle's `best-selling-books-notebook.csv`, enriched with additional columns to encompass storage shelves, prices, and inventory levels. These enhancements were autogenerated to simulate a realistic bookstore inventory environment, providing a comprehensive basis for system functionality testing.

BMS strategically employs Python dictionaries to manage user data, encapsulating critical information such as usernames, user roles (Customer or Admin), and passwords. This choice of data structure is instrumental in optimizing the user authentication process, offering several key advantages that enhance the system's security, efficiency, and overall user management capabilities.

Both intended functions and error-handling mechanisms were tested throughout development to ascertain their performance and reliability. All system functionalities were tested to ensure they operated as expected and adhered to our design specifications. Special attention was given to error-handling capabilities to guarantee that the system gracefully manages exceptions and provides meaningful feedback to users under various scenarios. Comprehensive testing efforts will be demonstrated through detailed examination videos in product presentations.

Conclusion

The development of the BMS marks a significant milestone in leveraging technology to enhance the traditional bookstore model. By integrating advanced functionalities such as efficient book management, user role differentiation, and dynamic search capabilities, the BMS addresses critical operational challenges while offering an enriched customer experience.

Throughout the implementation and testing phases, the BMS demonstrated its potential to streamline bookstore operations and customer service significantly. The use of Python dictionaries for user data management, alongside the strategic application of OOP principles and data manipulation techniques, has established a solid foundation for the system's functionality.

The BMS promises further enhancements, including integrating web application capabilities and exploring new functionalities to adapt to evolving market needs. The continued development and refinement of the BMS will undoubtedly contribute to the sustainability and growth of traditional bookstores in the digital era.

In conclusion, the BMS emerges as a crucial tool for bookstores navigating the challenges of digital transformation. Its development showcases the potential of software solutions in revitalizing traditional retail models and paves the way for future innovations in bookstore management.

Reference

OpenAI. (n.d.). ChatGPT-4. Retrieved from <https://openai.com>.

Rahulsingh, D. (2023, October). Best Selling Books Notebook. Retrieved February 16, 2024 from <https://www.kaggle.com/code/drahulsingh/best-selling-books-notebook/input>.

Appendix A: State of Contribution

The authors state that they distributed the work evenly between two members. Both members have contributed to the code development in the shared space. Design ideas are exchanged during the regular meeting. During the design phase, Ed looked at the dataset and implemented the code to handle book information input and output. Specifically, he worked on the `Book` and `Bookshelf` classes. Shu worked on integrating different user roles into the system, with classes handling the users and authentications. Both of us worked on the error-handling and de-bugging at the end. In the implementation phase, Ed focused on preparing the presentation and presenting the implementation of our application. Shu focused on preparing the design report.

Appendix B – Repository

This section provides the link to our copied shared working repository on GitHub:

<https://github.com/laggyDev/Bookstore-Management-System>