# Product Requirements Document: Online Bookstore with Desktop Application

**Version:** 1.0

**Date:** 2024-10-20

# 1. Introduction

This document outlines the product requirements for a desktop application designed for an online business to sell and rent books. The system will consist of a backend RESTful API built with Java or Python Flask and a MySQL database, and a client-facing desktop GUI application built in either Java or Python.

The primary goal is to provide a seamless experience for customers to discover, purchase, and rent books, while also giving managers the necessary tools to administer orders and payments.

# 2. User Roles

There are two primary user roles for this system:

- **Customer:** A registered user who wants to find and acquire books. They can browse the catalog, search for specific titles or authors, and place orders to either buy or rent books.
- **Manager:** An administrative user responsible for business operations. They can log in to a specialized interface to manage books, view orders and update the payment status of those orders.

# 3. Functional Requirements

## 3.1 User Account Management

| ID | Requirement | Details |
|---|---|---|
| FR1.1 | User Registration | A new user must be able to create an account. Required fields: username, password, and a valid email address. Passwords must be securely hashed on the backend. |

| ID | Requirement | Details |
|---|---|---|
| FR1.2 | User Login | A registered user must be able to log in using their username and password. The system must authenticate the user and establish a session. |
| FR1.3 | Session Persistence | Once logged in, the user should remain authenticated until they explicitly log out. All subsequent actions requiring authentication must be tied to their session. |

## 3.2 Book Catalog and Search

| ID | Requirement | Details |
|---|---|---|
| FR2.1 | Book Search | A logged-in user must be able to search for books. The search functionality should accept a keyword. |
| FR2.2 | Search by Title/Author | The search keyword should be matched against both book titles and author names. |
| FR2.3 | Search Results Display | The search results should be displayed in a clear list format, showing at a minimum: Book Title, Author, Price (to buy), and Price (to rent). |

## 3.3 Ordering and Transactions

| ID | Requirement | Details |
|---|---|---|
| FR3.1 | Buy a Book | A logged-in user must be able to select one or more books from the search results to purchase. |
| FR3.2 | Rent a Book | A logged-in user must be able to select one or more books from the search results to rent. |
| FR3.3 | Place Order | The user can finalize their selection of books to buy and/or rent in a single transaction, creating an order. |

## 3.4 Billing and Notifications

| ID | Requirement | Details |
|---|---|---|
| FR4.1 | Bill Generation | Upon placing an order, the system must generate a detailed bill. The bill should include a unique order ID, list of items (bought/rented), individual prices, and the total amount due. |
| FR4.2 | Email Notification | The generated bill must be automatically sent to the user's registered email address. This requires integration with an email sending service. |

## 3.5 Manager Functions

| ID | Requirement | Details |
|---|---|---|
| FR5.1 | Manager Login | A manager must have a separate, secure login mechanism. This role should be distinct from a standard customer role. |
| FR5.2 | View Orders | A logged-in manager must be able to view a list of all buy and rental orders in the system. |
| FR5.3 | Update Payment Status | The manager must be able to select an order and update its payment status (e.g., from "Pending" to "Paid"). |
| FR5.4 | Create and Update Book Information | The manager must be able to add new books or update their information (e.g., a book is returned from rental and available for the next one). |

# 4. Non-Functional Requirements

## 4.1 Technology Stack

| Area | Requirement |
|---|---|
| **Backend** | Java or Python (with the Flask framework). |
| **Database** | MySQL. |
| **API Design** | Must follow RESTful principles. |

| Area | Requirement |
|---|---|
| **Frontend** | A native desktop GUI application developed in either Java (e.g., Swing) or Python (e.g., Tkinter). **Note: This is not a web application.** |

## 4.2 Performance

| ID | Requirement | Details |
|---|---|---|
| NFR1.1 | API Response Time | API endpoints should respond to typical requests in under 500ms. |
| NFR1.2 | GUI Responsiveness | The desktop client should remain responsive and not freeze during network operations (e.g., searching for books). Asynchronous calls to the backend are required. |

## 4.3 Security

| ID | Requirement | Details |
|---|---|---|
| NFR2.1 | Password Storage | All user passwords must be stored in the database using a strong, one-way hashing algorithm (e.g., bcrypt). |
| NFR2.2 | API Authentication | All API endpoints that handle user-specific data or actions (e.g., placing an order, searching) must be protected and require a valid authentication token. |
| NFR2.3 | Authorization | Users must only be able to access their own data. Managers must not be able to perform customer actions and vice-versa. |

# 5. Assumptions and Dependencies

- An external SMTP service or email gateway will be used for sending bill notifications. Credentials for this service will need to be securely managed.
- The initial population of the book database (titles, authors, prices) is outside the scope of this document but will be required for testing.
- The application does not need to handle real-time payment processing (e.g., credit card integration). The manager's function to update payment status is sufficient.

# 6. Future Considerations (Out of Scope for v1.0)

- User profiles with order history.
- Book reviews and ratings.
- Inventory management for physical books.
- More advanced search filters (e.g., by genre, publication year).
- A system for handling book returns (for rentals).