Software Requirements Specification

for

SHELVD

Prepared by

Arun Ezekiel S/O Richard
Cheong Yu Qing, Crystal
Kenneth Goh Jing Wei
Muhammad Sufyan Bin Mohd Jais
Nyan Maw Htun
Ryu Hyunsun
Zhu Yuhao

Mawsters , Nanyang Technological University

2024-02-21

1 Table of Contents

2 Problem Statement

The management and discovery of digital books in the current environment present many difficulties for avid readers and book enthusiasts. Conventional book tracking and discovery approaches are frequently unwieldy, ineffective, and time-consuming. They also miss opportunities to fully utilise contemporary technological advancements. This results in a disorganised reading experience makes it harder to locate books that fit individual interests and makes it more difficult to properly manage one's reading list. A comprehensive shelvd application that leverages cutting-edge technology to deliver efficient management tools, personalised book recommendations, and a user-friendly interface is desperately needed. The way readers interact with their books would be completely transformed by this application, which would make finding and organised reading materials fun, easy, and customised to each user's tastes and reading habits.

3 Overview

3.1 Background

The difficulties that readers and book lovers face in the rapidly changing world of digital reading are becoming complicated. People are having a hard time keeping track of and finding new reading material that fits with their particular interests and preferences due to the growing amount of books and digital resources that are available. Current book management and exploration systems are frequently disjointed, impersonal, and underutilize current technological developments. Overwhelming options, ineffective book management, and a less-than-ideal reading experience result from this circumstance. shelvd must develop a cutting-edge, intuitive application that can provide effective management tools, personalized recommendations, and streamlined book discovery.

By enabling users to easily navigate the vast sea of books and customize their reading experience to suit their unique needs and interests, such a solution would greatly improve the reading journey.

3.2 Overall Description

The shelvd application is intended to provide a comprehensive response to the problems associated with managing and discovering digital books. Fundamentally, the application communicates with a strong back-end system that can manage multiple users and transactions at once. The application makes use of technologies like ViteJS, ClerkJS, and a MySQL database hosted on PlanetScale to guarantee dependable and effective performance. For added functionality, it also uses Python libraries and APIs, such as the Google Books API. With features like personalised recommendations and user-friendly navigation, the system is designed to provide a smooth, intuitive user experience for managing reading lists and finding books. With this all-encompassing strategy, the shelvd application is positioned as a key instrument in revolutionising the digital reading experience.

4 Investigation & Analysis Methodology

4.1 System Investigation

Upon evaluation, there are significant usability and functionality gaps in the current digital book management environment. Disjointed systems that provide little personalisation and ineffective book discovery processes frequently cause users to struggle. Users find it difficult to efficiently manage their reading lists and monitor their reading progress as a result of this, which is exacerbated by a lack of integration between various reading platforms. Additionally, a less customised reading experience results from current solutions' inadequate use of machine learning's potential for personalised book recommendations. These problems emphasise the need for the shelvd application, which attempts to solve these shortcomings by offering a unified, user-friendly platform for book discovery and management.

4.2 Analysis Methodology

4.2.1 Feasibility study and requirements elicitation

The creation of a project team with experience in software development and digital book management is essential to the shelvd application's success. Over the project, this team will support continuous communication and teamwork.

Interviewing important stakeholders such as avid readers, librarians, and publishers is essential. These interviews will provide information about how current book management systems are used, highlighting their advantages and shortcomings. Talking with UI/UX designers and software developers who have experience in similar projects can offer insightful viewpoints on practicality and design issues. The numerous suggested features will be assessed in light of the learned lessons through a thorough Feasibility and Risk Assessment study.

To ensure the creation of a successful, user-friendly shelvd application, this study will concentrate on evaluating technical feasibility, user experience optimisation, financial implications, compliance with data protection regulations, and potential risks.

4.2.2 System analysis and requirements specification

4.2.2.1 Perform an analysis of the problem using object-oriented techniques

An external view of the enterprise model of the student registration including student records, department and staff information, course requirements, and class schedules will be developed using Unified Modeling Language (UML). These System Requirement Specifications documents will form part of the documentation for the project. Some desired features of the new system include:

- The ability to search/view books
- Fetch Best-selling/Recommended books
- Create/Edit/Delete book collections
- Adding/removing books to collections
- Fetch books based on recent cache

4.2.2.2 Scope and Limitations

Analysis methodology will involve business analysis, requirement analysis, data analysis, process analysis, (web) and application architecture:

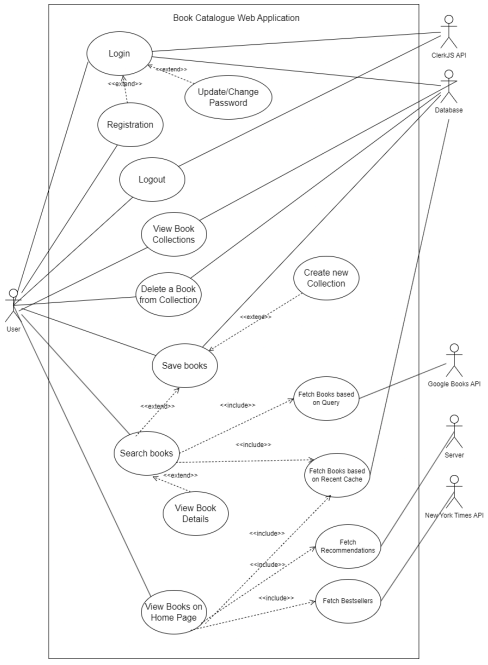
- Business analysis - State the business rules, business system interfaces, business function, business ownership sponsorship and associated project budget requirement
- Requirement analysis - System I/O description, user requirement definition, functional and security requirement
- Data analysis - Involve data collection process, data validation, data storage, manipulation and retrieval
- Process analysis - Data/process flow analysis, process decomposition and system interfaces
- Application architecture - Analyse application information structure, usability, user interface design, interaction and application implementation

4.2.3 Object-oriented design using UML

A comprehensive object-oriented design for the shelvd application will be developed using UML (Unified Modeling Language) for both graphical representation and documentation. This design will focus on the core functionalities of book management and discovery, encompassing the interactions between users and the system. At its core, users will interact with a web-based interface to search, manage, and discover books, facilitated by real-time processing with the back-end system. Additional features will include personalized recommendations, management of reading lists, and feedback on book availability and popularity. Security will be ensured through user authentication mechanisms involving user ID and password.

4.2.3.1 Use Case Diagram

The Use Case Diagram for the shelvd application illustrates the dynamic interactions between users and the system's functionalities.



4.2.4 Prototyping

The development of the shelvd application will employ the Object-Oriented Rapid Prototyping (OORP) methodology to create a functional prototype that represents the core aspects of the application. This prototype will serve as a tangible proof of concept to demonstrate the envisioned functionality and user interaction with the system.

The prototype will focus on the most critical features identified in the requirements, such as the user interface for book search, personalized recommendation mechanisms, and the management of personal reading lists. It will be crafted to interact with the underlying database as well as relevant APIs providing access to sufficient book information.

Primary elements to be included in the prototype are:

- **Web-based Forms:** These will act as the primary method of input for end-users, allowing them to perform book searches, manage their reading lists, and receive recommendations.
- **Database and API Interaction:** The forms will interact in real-time with a back-end database and relevant APIs, demonstrating the application's ability to query and retrieve book data.
- **User Interface Mock-ups:** These will showcase the proposed design and layout of the application's interface, providing stakeholders with a visual understanding of the user experience.

The prototype will be developed with a focus on flexibility to accommodate feedback and iterative improvements. Key objectives during this phase will be to:

- Validate the application's technical feasibility and functional alignment with user needs.
- Gather early user feedback to inform subsequent design and development phases.
- Provide hands-on experience for stakeholders to interact with the concept, ensuring alignment with business objectives and user expectations.

Upon completion, the prototype will be presented to the implementation team, stakeholders, and select user groups for evaluation and feedback. This iterative prototyping process is intended to ensure that the final application is both technically sound and closely aligned with the user requirements and business goals set forth in this SRS document.

5 Constraints

5.1 Scalability

Scalability is key in the development of the shelvd application, as it needs to support an increasing number of users and a growing book database. To enable effective data retrieval and updating, the application's architecture must be built to support smooth integration with a variety of data sources and external APIs, such as the Google Books API.

5.2 Project Schedule

The project's planning, development, implementation, and quality assurance will take place over three months.

5.3 API Limitations

The shelvd application is subject to a technical constraint imposed by the Open Library RESTful API, which restricts the number of requests to 100 per IP address within a 5-minute timeframe. Exceeding this specified rate limit may result in service disruptions or temporary unavailability of data from the Open Library API.

5.4 Data Protection

In alignment with prevailing data protection regulations, including but not limited to the Personal Data Protection Act (PDPA) of Singapore. Compliance with these regulations is imperative, necessitating explicit user consent for data collection, transparent communication about the purpose of data processing, and the implementation of robust security measures to protect personal information. The application must align with the respective data protection laws in all jurisdictions where it operates, ensuring a privacy-centric approach and enabling users to exercise control over their data as mandated by the applicable regulatory frameworks.

6 Operational Requirements

6.1 Help Desk Support

System users have a 24x7 access to telephone assistance for questions that are technical in nature, such as, slow or sluggish system response time, incompatible browser features, application errors, system downtime inquiries, account lock-out assistance, etc. Alternative modes of contact will also be provided and regularly monitored, including a helpdesk email for system users.

6.2 Application Services and Technical support

Programmers and application developers will have access to source code to address bugs or system enhancements as deemed necessary. Network Administrator and DBA support is also required to maintain a 24x7 system uptime.

6.3 Administration Features

System security and access levels are provided in the online system. There are varying levels of system access and functional authority. Each user's access is limited to his/her own book catalogues and recommendations. Only authorized system administrator(s) has access to all database information across users.

6.4 System hardware fail over and routine back up

Computer operations center will handle system hardware tasks such as data tape back-up, hardware maintenance, fail over, scheduled system patches and maintenance. A reliable server will be utilised to ensure minimal effort is required for servicing.

6.6 Audit Trail

System audit trails are inherent part of all user registrations. Among others, all transaction records will capture what action was taken, when (time-stamp) the transaction occurred and who made the transaction.

7 Functional Requirements

The shelvd app is a comprehensive system designed to enhance the experience of discovering, managing, and engaging with digital books. It offers a range of self-service features that allow users to personalize and manage their reading experience effectively. All system interfaces adhere to ISO-accepted industry standards for the World Wide Web.

7.1 Functionalities

The app includes the following functionalities:

7.1.1 Book Search

7.1.1.1 The user must be able to search books by entering keywords related to title, author, genre and ISBN in the search bar.

7.1.1.1.1. The search text must be of at least 1 character and less than 128 characters.

7.1.1.2. When the user types into the search bar, the system must fetch the list of books using the search cache or the search endpoint of Google Books API.

7.1.1.2.1 When Google Books API is used to fetch the list of books, the list of books must be cached in a database.

7.1.1.2.2 When the system receives the list of books from Google Books API, the system must display a table for the books.

7.1.1.2.2.1 There must be less than 5 books in each row.

7.1.1.2.2.2 Each cell in the table must contain the title, author and cover image of the book.

7.1.1.2.2.3 If the number of books on a page exceeds 30, the system must provide pagination for the user to navigate to a subsequent page.

7.1.1.3 When the search bar is empty, the system must fetch the top 5 bestsellers using the bestsellers cache or the search endpoint of New York Times API.

7.1.1.3.1 When the New York Times API is used to fetch the list of books, the list of books must be cached in a database.

7.1.1.3.2 When the system receives the list of books from New York Times API, the system must display the top 5 bestsellers inline.

7.1.1.4 When the search bar is empty, the system must fetch the top 5 book recommendations using the recommendation cache or the recommendation endpoint of the server.

7.1.1.4.1 When the server is used to fetch the list of recommended books, the list of books must be cached in a database.

7.1.2 Login

7.1.2.1. The interface must display whether the login was successful or unsuccessful through a pop-up message.

7.1.2.2. The user must be able to choose to register an account if the account does not exist.

7.1.2.3. The user must be able to choose to reset their password if they forget their password

7.1.2.4 Upon successful login, the user will be redirected to the authenticated home page.

7.1.3 Register

7.1.3.1. Email provided by the user must be unique and valid.

7.1.3.2. Password provided by the user must be unique and meet the following conditions:

7.1.3.2.1. Must contain at least one uppercase letter.

7.1.3.2.2. Must contain at least one lowercase letter.

7.1.3.2.3. Must contain at least one digit.

7.1.3.2.4. Must contain at least one special character.

7.1.3.2.5. Length of the password Must not exceed 12 characters.

7.1.3.3 ClerkJS API must send a one time passcode to the email provided. The passcode is valid for a specified time frame.

7.1.3.4 The user must enter a correct one time passcode.

7.1.3.5 When the user registered for an account, 3 default collections Read, Reading and Completed must be created.

7.1.3.6 The user must be redirected to the authenticated home page.

7.1.4 Logout

7.1.4.1 The system must display a dialog box including options to cancel the operation or proceed with the logout to confirm the user's intention to logout.

7.1.4.2 When the user logs out, the user must be redirected to the unauthenticated home page.

7.1.4.2.1 All local storage data must be cleared after logout.

7.1.4.2.2 Authentication token must be invalidated in the database using Clerk's SDK.

7.1.5 Reset / Change Password

7.1.5.1 When the user changes password, the user must be redirected to the change password page.

7.1.5.1.1 The user must be prompted to enter their email.

7.1.5.1.2 ClerkJS API must send a password reset code to the provided email, valid for a specified time frame.

7.1.5.1.3 The user must be prompted to enter the password reset code.

7.1.5.1.4 The user must be prompted to enter their new password.

7.1.5.1.5 New password must follow the same requirements as registration.

7.1.5.2 Upon successful reset, the user must be redirected to the home page.

7.1.5.3 The user must be notified in the user interface upon successful reset through a pop-up message.

7.1.6 Bookmarking (Save Books, Create Collection, Add to Collection)

7.1.6.1 The user must be able to save the books into one or more collections.

7.1.6.2 The user must be able to create personal book collections to store saved books.

7.1.6.2.1 The name of the collection must be at least 1 character and less than 10 characters.

7.1.6.3 When a book is added into a collection, the user must be notified in the user interface.

7.1.6.4 When a new collection is created, the user must be notified in the user interface through a pop-up message confirming the successful creation of a collection.

7.1.6.5 When creating a new collection, the user must be prompted to enter the name.

7.1.6.6 When there is another collection with the same name, the user interface must display an error message through a pop-up message and the user must be prompted to enter a different name for the new collection.

7.1.6.7 When a book already exists in a collection, the system must display an error message through a pop-up message, providing information about the duplication.

7.1.6.8 The number of collections created by the user should not exceed 500.

7.1.6.9 The number of books in a collection should not exceed 500.

7.1.7 View Book Details

7.1.7.1 The user must be able to view pre-requisites, book metadata, and preview links provided by Google Books API.

7.1.8 Manage Book Collections

7.1.8.1 The user must be able to view a list of book collections fetched from the database.

7.1.8.2 The user must be able to click on a collection to view a list of books belonging to that collection.

7.1.8.3 The user must be able to delete a book from a collection.

7.1.8.4 The system will display a dialog box including options to cancel the operation or proceed with the deletion to confirm the user's intention to delete the book from the collection.

7.1.8.5 When a book is deleted from a collection, the user must be notified in the user interface through a pop-up message confirming the successful deletion.

The use case descriptions are included as follows:

7.2 Use Case Descriptions

7.2.1 Login (for Users)

Use Case ID:	1		
Use Case Name:	Login		
Created By:	Ryu Hyunsun	Last Updated By:	Ryu Hyunsun
Date Created:	04/02/2024	Date Last Updated:	04/02/2024

Actors:	Database, User, ClerkJS API
Description:	The user must be able to login to use the app.
Trigger:	The user clicks on the Login button on the website.
Preconditions:	1. The internet connection is stable. 2. The website is launched.
Postconditions:	1. User has logged in to the website. 2. Authenticated home page is loaded.
Normal Flow:	1. The user clicks on the Login button on the home page. 2. Login screen will be launched. 3. The user enters their username in the ‘Username’ field. 4. The user enters their password in the ‘Password’ field. 5. The user clicks on the ‘Login’ button. 6. ClerkJS API will then verify the username and password with the database. 7. Upon successful login, the authenticated home page will be loaded.
Alternative Flows:	1.4: If the user inputs an incorrect password. 1. If the user inputs an incorrect password, an error message “Username / Password entered is invalid, Please try again.” will be shown. 2. The password field will be cleared. 3. The use case returns to step 4. 1.3: If the user inputs an incorrect username. 1. If the user inputs an incorrect username, an error message “Username / Password entered is invalid, Please try again.” will be shown. 2. The password field will be cleared. 3. The use case returns to step 3. 1.2: If the user clicks on the “Reset Password” button 1. The user clicks on the “Reset Password” button. 2. The use case uses the extended Use Case 3 ‘Reset Password’.

	1.2: If the user clicks on the “Register” button 1. The user clicks on the “Register” button. 2. The use case uses the extended Use Case 2 ‘Register’.
Exceptions:	1.0.E.5: If the password has been found in an online data breach 1. If the password has been found in an online breach, an error message “Password has been found in an online data breach. Please reset your password” will be displayed. 2. The use case uses the extended Use Case 3 ‘Reset Password’.
Includes:	N/A
Priority:	High
Frequency of Use:	Once in three months - When the user initially launches the website or when the login session is terminated.
Business Rules:	
Special Requirements:	SR-1: The system should verify the username and password within 3 seconds.
Assumptions:	-
Notes and Issues:	-

7.2.2 Register (for Users)

Use Case ID:	2		
Use Case Name:	Register		
Created By:	Ryu Hyunsun	Last Updated By:	Ryu Hyunsun
Date Created:	04/02/2024	Date Last Updated:	04/02/2024

Actors:	Database, User, ClerkJS API
Description:	The user must be able create a new account.
Trigger:	The user clicks on the “Register” button.
Preconditions:	1. Internet connection is stable. 2. The user’s account must not already exist in the database.
Postconditions:	1. The user successfully registers an account. 2. The user will be directed to the Login page for login upon successful account creation. 3. 3 default book collections (Read, Reading, To be completed) must be created.
Normal Flow:	1. The user clicks on the ‘Register’ button on the home page./The user clicks on the ‘Register’ button on the Login page. 2. Registration screen will be loaded.

	3. The user must input a valid email and password. 4. The user must confirm the password. 5. The user clicks on the 'Register' button. 6. The system shall verify that the password and the confirm password match. 7. The system shall check with the database to ensure that the registrant's username does not exist in the database. 8. The system must ensure that the password fulfils the criteria. 9. ClerkJS API will verify the user's email address by sending a one-time code with a redirect link to the email address provided. 10. The user will click on the link to enter the verification code. 11. The user clicks on the "Verify Email" button. 12. A dialog box with a message "Account created successfully" will be displayed upon successful account creation. 13. The user will be directed to the Login page. 14. The system returns to the Use Case 1 "Login".
Alternative Flows:	-
Exceptions:	2.0.E.5: If the user attempts to use a password that has been found in an online data breach 1. An error message "Password has been found in an online data breach. Please use another password" will be displayed. 2. The use case returns to step 3. 2.0.E.5: Email already exists in the database 1. System queries and finds a record with the same email address. 2. An error message "The email address is already in use. Please enter a new email address." will be displayed. 3. The use case returns to step 3 2.0.E.8: Password does not fulfil the criteria 1. An error message "The password should contain at least one upper case letter, one lower case letter, one digit, and one special character." will be displayed. 2. The use case returns to step 3. 2.0.E.8: Password exceeds 12 characters 1. An error message "The length of the password must not exceed 12 characters." will be displayed. 2. The use case returns to step 3. 2.0.E.6: The password and confirm password does not match

	1. An error message “The password and confirm password does not match.” will be displayed. 2. The use case returns to step 3.
Includes:	-
Priority:	High
Frequency of Use:	Once per lifetime
Business Rules:	Each email address should be associated with only one user.
Special Requirements:	Registering an account should take less than 1 minute.
Assumptions:	-
Notes and Issues:	-

7.2.3 Reset Password

Use Case ID:	3		
Use Case Name:	Reset Password		
Created By:	Ryu Hyunsun	Last Updated By:	Ryu Hyunsun
Date Created:	04/02/2024	Date Last Updated:	04/02/2024

Actors:	Database, User, ClerkJS API
Description:	The user must be able to reset their password.
Trigger:	The user clicks on the “Reset Password” button.
Preconditions:	1. The internet connection is stable. 2. The user has an existing account.
Postconditions:	1. “Password reset successful” message will be shown and the user will be redirected to the home page.
Normal Flow:	1. The user clicks on the “Reset Password” button on the Login page./The user clicks on the “Reset Password” button after clicking on the profile picture on the home page. 2. “Reset Password” page will be loaded. 3. The user will enter their email address. 4. The user will click on the “Get Verification Code” button. 5. ClerkJS API sends a password reset code to the provided email. 6. The user enters the new password. 7. The user enters the password reset code. 8. The user clicks on the “Reset Password” button. 9. The system will check to ensure that the new password meets the criteria. 10. ClerkJS API will verify the new password and password reset code. 11. Upon successful reset, the user will be redirected to the home page.

Alternative Flows:	3.8. If the user clicks on the “Cancel” button 1. The user interface will display a dialog box to confirm the user’s intention to cancel the operation. 2. The user will be directed to the home page.
Exceptions:	3.0.E.10. If the user attempts to use a password that has been found in an online data breach 1. An error message “Password has been found in an online data breach. Please use another password” will be displayed. 2. The use case returns to step 4. 3.0.E.9.. The new password does not meet the criteria. 1. An error message “The password should contain at least one upper case letter, one lower case letter, one digit, and one special character.” will be displayed. 2. The use case returns to step 6. 3.0.E.10. The password reset code entered is invalid. 1. An error message “The password reset code is incorrect. Please try again.” will be displayed. 2. The use case returns to step 4. 3.0.E.5. The email does not exist. 1. An error message “The email address does not exist.” will be displayed. 2. The use case returns to step 3.
Includes:	-
Priority:	Low
Frequency of Use:	Once per year
Business Rules:	-
Special Requirements:	The email containing the password reset code must be sent within 10 seconds.
Assumptions:	-
Notes and Issues:	-

7.2.4 Logout

Use Case ID:	4		
Use Case Name:	Logout		
Created By:	Ryu Hyunsun	Last Updated By:	Ryu Hyunsun
Date Created:	04/02/2024	Date Last Updated:	04/02/2024

Actors:	Database, User, ClerkJS API
Description:	The user must be able to logout from the system.
Trigger:	The user clicks on the “Logout” button on the home page.
Preconditions:	1. The user is already logged in.
Postconditions:	1. The user will be logged out.

	2. The user will be directed to the unauthenticated home page.
Normal Flow:	1. The user clicks on the “Logout” button on the home page. 2. A message “Are you sure you want to log out?” will be displayed. 3. User clicks on the “Log me out” button. 4. ClerkJS API will deactivate the current session. 5. Upon successful log out, the user will be directed to the unauthenticated home page.
Alternative Flows:	4.3. The user clicks on the “Cancel” button. 1. The user will be directed to the home page.
Exceptions:	-
Includes:	-
Priority:	High
Frequency of Use:	Once per year
Business Rules:	-
Special Requirements:	The user must be logged out within 5 seconds.
Assumptions:	-
Notes and Issues:	-

7.2.5 Save Books

Use Case ID:	5		
Use Case Name:	Save Books		
Created By:	Ryu Hyunsun	Last Updated By:	Ryu Hyunsun
Date Created:	04/02/2024	Date Last Updated:	04/02/2024

Actors:	Database, User
Description:	The user must be able to save a book into a collection.
Trigger:	The user clicks on the save button on the View Book Details page.
Preconditions:	1. The user must have an account. 2. The user must be already logged in.
Postconditions:	1. The book will be saved into the selected collection. 2. The user will be notified in the user interface through a pop-up message upon successful addition.
Normal Flow:	1. The user clicks on the save button on the View Book Details page. 2. The user will have the option to create a new book collection, add to an existing collection, or cancel on a dialog box. 3. The user clicks on the “Add to Collection” button. 4. A list of book collections created by the user will be displayed.

	5. The user clicks on one of the book collections. 6. When a book is successfully added into a collection, the user will be notified in the user interface through a pop-up message.
Alternative Flows:	5.2. If the user clicks on the “Create New Collection” button 1. The use case will use the extended Use Case 6 “Create Collection.” 2. The use case will return to step 6. 5.2 If a user clicks on the “Cancel” button 1. The dialog box will be closed.
Exceptions:	5.0.E.5 If the book already exists in the selected collection 1. The user will be notified through a pop-up message stating: “Could not be added! Already exists!” 2. The operation will be cancelled. 5.0.E.5 If the number of books in the collection already exceeds 500 1. The user will be notified through a pop-up message stating: “Maximum of 500 books reached!” 2. The operation will be cancelled.
Includes:	-
Priority:	Medium
Frequency of Use:	Once per week
Business Rules:	• The user must be able to add a book into one or more collections. • A book cannot be added into the same collection more than once. • The number of books in a collection must not exceed 500.
Special Requirements:	The user must be able to save a book into a collection within 15 seconds.
Assumptions:	-
Notes and Issues:	-

7.2.6 Create Collection

Use Case ID:	6		
Use Case Name:	Create Book Collection		
Created By:	Ryu Hyunsun	Last Updated By:	Ryu Hyunsun
Date Created:	04/02/2024	Date Last Updated:	04/02/2024

Actors:	Database, User
Description:	The user must be able to create a new collection.
Trigger:	The user clicks on the “Create New Collection” button after clicking on the “Save Books” button on the View Book

	Details page / The user clicks on the “Create New Collection” on the Book Collections page.
Preconditions:	1. The user must be already logged in.
Postconditions:	1. The user will be notified in the user interface through a pop-up message confirming the successful creation of a collection.
Normal Flow:	1. The user interface will display a dialog box with a textbox for the user to input a name for the new collection, along with “Add” button and “Cancel” buttons. 2. The user will enter the name. 3. The user will click on the “Add” button. 4. The user will be notified in the user interface through a pop-up message confirming the successful creation of a collection.
Alternative Flows:	6.3.. When a user clicks on the cancel button 1. The dialog box will be closed.
Exceptions:	6.0.E.3. When a collection with the same name already exists 1. The user interface should present an error message via a pop-up. 2. The user will be prompted to enter a different name for the new collection. 6.0.E.3. When the number of collections created by the user exceeds 500 1. The user interface should present an error message via a pop-up. 2. The operation will be cancelled.
Includes:	-
Priority:	Medium
Frequency of Use:	Once per month
Business Rules:	The user will be able to create a maximum of 500 collections.
Special Requirements:	The user must be able to create a collection within 15 seconds.
Assumptions:	-
Notes and Issues:	-

7.2.7 View Books on Home Page

Use Case ID:	7		
Use Case Name:	View Books on Home page		
Created By:	Zhu Yu Hao	Last Updated By:	Zhu Yu Hao
Date Created:	07/02/2024	Date Last Updated:	07/02/2024

Actors:	New York Times API, User, Server
Description:	The user must be able to view the list of books including top bestsellers on the home page.
Trigger:	1. The user clicks on the home page 2. The user first loads the website
Preconditions:	1. The user's computer must be connected to the internet.
Postconditions:	The Home page will be loaded.
Normal Flow:	1. The system attempts to retrieve the top 5 bestsellers from the bestsellers cache. 2. The system will display the top 5 bestsellers under the "bestsellers" section. 3. The system attempts to retrieve the top 5 recommendations from the recommendations cache. 4. The system will display the top 5 recommendations under the "Recommendations" section.
Alternative Flows:	-
Exceptions:	7.0.E.1 The system fails to retrieve from the bestsellers cache 1. The system fetches the top 5 bestsellers from the New York Times API. 2. The top 5 best sellers are cached into the bestsellers cache. 3. Return to step 2. 7.0.E.2 The system fails to retrieve from the recommendations cache 1. The system fetches the top 5 recommendations from the server. 2. The top 5 recommendations are cached into the recommendations cache. 3. Return to step 4.
Includes:	-
Priority:	High
Frequency of Use:	Once per day
Business Rules:	-
Special Requirements:	The books must be displayed within 5 seconds.
Assumptions:	-
Notes and Issues:	-

7.2.8 Search Books

Use Case ID:	8		
Use Case Name:	Search Books		
Created By:	Zhu Yu Hao	Last Updated By:	Zhu Yu Hao
Date Created:	07/02/2024	Date Last Updated:	07/02/2024

Actors:	User, Google Books API
Description:	The user must be able to search for books using a search bar.
Trigger:	The user clicks on the search bar.
Preconditions:	The user must be on the home page.
Postconditions:	The Search Books page will be loaded based on the search query.
Normal Flow:	1. The user types the title/genre/author/ISBN of the book in the search bar. 2. The system will send requests to the cache database to retrieve recent books that match the search. 3. The books are displayed in a pagination view. 4. The search bar will show a greyed out text beside the actual query for autocompletion. 5. The user can press Tab to autocomplete the query to the exact book. 6. The user can select a book to view book details using extended Use Case 10 “View Book Details”.
Alternative Flows:	-
Exceptions:	8.0.E.2 The system fails to retrieve the books from the cache. 1. The system will send a request to Google Books API to retrieve the list of books 2. Return to step 3. 8.0.E.3 Google Books API replies with an error response. 1. The system will display the error message “Unable to retrieve books” in the centre of the page.
Includes:	-
Priority:	High
Frequency of Use:	Once per day
Business Rules:	1. The search text must be of at least 1 character and less than 128 characters. 2. There must be less than 5 books in each row displayed in the pagination. 3. Each cell in the table must contain the title, author and cover image of the book in the pagination. 4. If the number of books on a page exceeds 30, the system must provide pagination for the user to navigate to a subsequent page.
Assumptions:	-
Notes and Issues:	-

7.2.9 View Collections

Use Case ID:	9		
Use Case Name:	View Collections		
Created By:	Zhu Yu Hao	Last Updated By:	Zhu Yu Hao
Date Created:	07/02/2024	Date Last Updated:	07/02/2024

Actors:	User, Database
Description:	The user can display the list of collections created by the user.
Trigger:	The user selects the View Collections navigation tab.
Preconditions:	1. User is logged in
Postconditions:	1. The Collections Page will be loaded with a list of collections created by the user.
Normal Flow:	1. The user selects the View Collections navigation tab. 2. The system queries the database for the collections owned by this user. 3. The system displays the collections owned by this user. 4. The user can select one of the collections.
Alternative Flows:	-
Exceptions:	9.0.E.1 The database does not contain any collections owned by this user 1. The system displays “no collections owned by this user” in the centre of the page.
Includes:	-
Priority:	Medium
Frequency of Use:	High
Business Rules:	-
Special Requirements:	The collections created by the user including the default collections should be loaded within 10 seconds.
Assumptions:	-
Notes and Issues:	-

7.2.10 View Book Details

Use Case ID:	10		
Use Case Name:	View Book Details		
Created By:	Zhu Yu Hao	Last Updated By:	Zhu Yu Hao
Date Created:	07/02/2024	Date Last Updated:	07/02/2024

Actors:	User
Description:	The user can see a list of details of the book.

Trigger:	The user selects a book in the list of books displayed in the pagination view.
Preconditions:	The user selects one of the categories in the home page, or selects one of the books from the search query.
Postconditions:	
Normal Flow:	1. The user clicks on a link of a book cell in the pagination. 2. The system will retrieve the specific book information stored temporarily during the search query. 3. The system will display the book description, book pre-requisites and book preview link. 4. The user can click on the book preview link to read the preview.
Alternative Flows:	
Exceptions:	
Includes:	
Priority:	High
Frequency of Use:	High
Business Rules:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

7.2.11 Delete a book from Collection

Use Case ID:	11		
Use Case Name:	Delete a book from Collection		
Created By:	Zhu Yu Hao	Last Updated By:	Zhu Yu Hao
Date Created:	07/02/2024	Date Last Updated:	07/02/2024

Actors:	User, Database
Description:	The user can delete a book from the selected collection.
Trigger:	The user clicks on the “Delete” button on the Collections page.
Preconditions:	The user is in the specified collection page.
Postconditions:	The selected book will be deleted from the collection .
Normal Flow:	1. The user selects the delete icon beside the book cell in the collection. 2. The system sends a confirmation pop-up message on deletion of the book. 3. The system requests a delete operation to the database for the user collection for that book. 4. The system displays a “successfully deleted book” pop-up message to the user.

Alternative Flows:	11.2 Cancel confirmation of pop-up 1. Returns to the specified collection page. 11.3 Delete operation failed 1. System displays a “Failed to delete book” message to the user.
Exceptions:	
Includes:	
Priority:	Medium
Frequency of Use:	Medium
Business Rules:	
Special Requirements:	
Assumptions:	
Notes and Issues:	

8 Input Requirements

8.1 User identifier key and user access

Each user is assigned a unique authentication key upon account creation in the shelvd application. This key maps to all their personal reading lists, preferences, and history within the app. User accounts may be disabled upon user request or due to inactivity over an extended period. This key creation is facilitated with ClerkJS.

8.2 Book Identification

Unique identifiers, such as ISBNs, are used to catalog books within the system, and via APIs. Users can utilize these identifiers to search for specific books. Book details, along with these identifiers, will be accessible to the users to assist in their exploration and management of books.

8.3 Action Codes

All user actions within the app, such as 'add to reading list' or 'mark as read', will be logged with unique transaction codes. These codes will be used for reference and to facilitate user activities within the app.

9 Process Requirements

The following are among the inherent requirements that the shelvd system must be able to handle.

9.1 Backend and API Transactions

The system must be capable of sending, receiving, and processing transactions with the back-end database and relevant APIs reliably.

9.2 Data Integrity

The application will commit fully completed transactions and rollback any unfinished or timed-out transactions to maintain data integrity.

9.3 Data Validation

The system will include data validation and error-handling routines to manage data errors gracefully, whether they occur on the user end or within the back-end processing.

9.4 Performance

The application will be optimized to handle high concurrency and provide 24/7 availability, resolving any potential locking issues and enhancing the overall user experience.

9.5 Data Repository

The shelvd application will maintain a robust database, likely MySQL hosted on PlanetScale, as the main repository for all user and book data.

10 Output Requirements

10.1 Transaction summary and confirmation

Users will receive a summary and confirmation of their actions during each session, particularly after managing their reading lists or preferences.

10.2 Exception reports

The system will generate exception reports to document any anomalies or unusual activities, such as multiple failed login attempts or data fetching errors.

10.3 Usage Reports and Summaries

Administrators will have the ability to extract and view summarized user activity data and system

performance metrics for analysis and improvement purposes.

11 Hardware Requirements

11.1 Network

A reliable internet connection is essential for accessing the shelvd application.

11.2 Client Computers

The application will be accessible from various devices, including smartphones, tablets, laptops, and desktop computers across different operating systems, provided there is access to internet browsers.

11.3 Servers

Robust server infrastructure to host the application, database, and related services.

11.4 Production support systems

Necessary hardware for hosting the web application, including server computers, backup systems, and uninterrupted power supplies (UPS).

12 Software Requirements

12.1 Client Operating Systems

The application will be accessible on various operating systems, including UNIX, macOS, and Windows.

12.2 Web Application

The shelvd web application will be compatible with all major browsers:

- Google Chrome
- Microsoft Edge
- Mozilla Firefox
- Safari
- Opera

12.3 Network system

The application will use standard network protocols to ensure secure and reliable data transmission:

- TCP/IP
- HTTP/HTTPS for web traffic

- FTP for file transfers as needed

12.4 Backend system

The application uses a resilient backend system, utilizing technologies like ViteJS for frontend development and ClerkJS for authentication, with MySQL serving as the database solution.

12.5 Licenses

All third-party software used in the development, testing, and production of the application will be properly licensed, ensuring compliance with legal and industry standards.

13 Deployment Requirements

The deployment of the shelvd application is a critical phase that ensures the application is available to users with minimal downtime and high reliability. The following requirements outline the necessary steps and considerations for a successful deployment:

13.1 Environment Setup

Production Server Specifications: The production server should run a stable Linux distribution, have at least 16 GB of RAM, a quad-core processor, and SSD storage of at least 256 GB for optimal performance.

Staging Environment: A staging environment identical to the production setup will be established for pre-deployment testing to ensure reliability.

13.2 Deployment Process

Automation: Use tools like Jenkins or GitHub Actions for automated deployment.

Version Control: All code will be managed in a Git repository, with a clear branching model and release management strategy.

13.3 Dependency Management

Software Dependencies: Document and lock all software dependencies using package managers like npm or pip to ensure consistent builds.

Service Dependencies: Configuration for external services will be managed via environment variables and secure secret management systems.

13.4 Data Migration and Backup

Data Migration Strategy: A detailed data migration plan will be developed to transfer existing data to the new system with minimal downtime.

Backup Procedures: Automated daily backups of the database will be configured using tools like mysqldump or PlanetScale's backup services.

13.5 Security Measures

Encryption: TLS encryption will be enforced for all data in transit, and at-rest encryption will be enabled for sensitive data in the database.

Access Controls: Access to production servers will be restricted to authorized personnel using SSH keys and VPNs.

Security Testing: Regular security audits and penetration testing will be conducted before deployment.

13.6 Compliance and Regulations

Legal Compliance: The deployment process will adhere to the relevant software compliance standards, such as PCI DSS for payment processing and GDPR for user data protection.

Data Protection: Data protection policies will be implemented to comply with privacy laws, including proper user data handling and right to erasure procedures.

13.7 Rollback Strategy

Contingency Planning: In the event of a failed deployment, a rollback mechanism will be in place to revert to the last known good state of the application.

13.8 Documentation and Training

Deployment Documentation: Detailed documentation of the deployment process will be maintained and regularly updated.

Staff Training: Technical staff will be trained on the deployment process and emergency procedures.

13.9 Continuous Deployment

CI/CD Pipeline: A robust CI/CD pipeline will be implemented to enable frequent, safe deployments of the application as new updates and features are ready.