**Group 5 – Library System**

Software Requirements Specification

Revision History

| **Date** | **Revision** | **Description** | **Author** |
| --- | --- | --- | --- |
| 3/2/2025 | 1.0 | Initial Version | Joshua Charlesworth |
| 3/3/2025 | 1.1 | Updated Overview, Non-Functional Reqs, some Specific Requirements | Wai Ip |
| 3/4/2025 | 1.2 | Use Cases | Joshua Charlesworth/Arsen Dzhuraev |
| 3/4/2025 | 1.3 | Use Case Diagrams | Arseb/Josh/Wai/Jordan |
| 3/4/2025 | 1.4 | Class Diagrams | Jordan Walker |
| 3/4/2025 | 1.5 | Sequence Diagrams | Wai Ip |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

**1.** **Purpose 4**

1.1. Scope 4

1.2. Definitions, Acronyms, Abbreviations 4

1.3. References 4

1.4. Overview 4

**2.** **Overall Description 5**

2.1. Product Perspective 5

2.2. Product Architecture 5

2.3. Product Functionality/Features 5

2.4. Constraints 5

2.5. Assumptions and Dependencies 5

**3.** **Specific Requirements 6**

3.1. Functional Requirements 6

3.2. External Interface Requirements 6

3.3. Internal Interface Requirements 7

**4.** **Non-Functional Requirements 8**

4.1. Security and Privacy Requirements 8

4.2. Environmental Requirements 8

4.3. Performance Requirements 8

# Purpose

This document outlines the requirements for the County Library System (CLS).

**Project found on GitHub at: https://github.com/C0rp0r4l/CS401LibraryProject**

## Scope

This document will catalog the user, system, and hardware requirements for the CLS system. It will not, however, document how these requirements will be implemented.

## Definitions, Acronyms, Abbreviations

ID: identification

GUI: Graphical User Interface

Items: Books, CDs, Laptops, Computers

## References

Use Case Specification Document – Pages 9-20

UML Use Case Diagrams Document – page 21

Class Diagrams – Page 22

Sequence Diagrams – Page 23

## Overview

The Multi-Library System is meant to provide a software package to handle all the necessary functions to run a library. Handling functions such as managing library ID’s, managing site inventory, as well as providing the flexibility to add or remove new library sites to the system.

# Overall Description

## Product Perspective

## Product Architecture

The system will be organized into 4 major modules: the Inventory Management module, the Account Management module, the Item Attainment module, and the Location Management module.

## Product Functionality/Features

The high-level features of the system are as follows (see section 3 of this document for more detailed requirements that address these features):

Librarians should be able to, using a member’s library ID, check out various types of items using the Multi-LIbrary system. Librarians should be able to create or delete member accounts, check the status of an account, and change the status of an account. Librarians should also be able to check the status of a type of item as well as change the status of the item, add to a waitlist, and create/delete items.

## Constraints

The program must be completely written in java.

The program must have a GUI interface

Staff interface must communicate with Data interface

Must network over TCP/IP

No HTML and no other Web components

## Assumptions and Dependencies

It is assumed that the system has stable power sources and that source is never interrupted.

It is assumed that every part of the library system is well maintained and never fails to work due to hardware errors.

It is assumed that the Library system is prepared for high volume of users and will operate consistently under low or high volume of users.

It is assumed that the Staff will not misuse any permissions they have to create excessive library locations or change a member’s status.

It is assumed that the user will not attempt to misuse their ability to reserve items or check out items.

# Specific Requirements

## Functional Requirements

### Common Requirements:

3.1.1.1 Users and Staff should be allowed to log in using their issued id and pin, both of which are alphanumeric strings between 6 and 20 characters in length.

3.1.1.2 Checkout ability must be based on Staff-Set Account status.

3.1.1.3 Users have an account status that is quantified by a certain number of strikes.

3.1.1.4 All actions will be logged with date and time in the system.

3.1.1.5 Users can renew select items such as books and CDs.

### Inventory Management Module Requirements:

3.1.2.1 Users should be able to search the entire library database without having to log in.

3.1.2.2 Staff should be able to transfer items between locations with or without necessity, possibly for the purpose of even distribution of items among libraries.

3.1.2.3 Staff should be able to add or remove items without necessity.

3.1.2.4 Users should be able to check out or reserve items if their account is in good standing.

3.1.2.5 Users should not be able to check out or reserve items if their account has 3 strikes.

3.1.2.6 Staff should be able to modify the waitlist of an item.

### Account Management Module Requirements:

3.1.3.1 Only staff is able to create or delete accounts and change account statuses.

3.1.3.2 Each account has a relevant status that determines the ability of an account to check out or reserve items.

3.1.3.3 Each account status is set by a Staff member.

3.1.3.4 Staff members may not change their own account status.

3.1.3.5 Staff should be able to modify a user's reservations or waitlist.

### Location Management Module Requirements:

3.1.4.1 New locations are able to be added by Staff members.

3.1.4.2 Only Staff can add Staff accounts to a location.

3.1.4.3 Any user should be able to search the inventory of a specific Location vs the entire network.

3.1.4.4 Each location should have its own attributes displayed (name, address).

### Item Attainment Module Requirements:

3.1.5.1 Staff must be logged in in order to check out or reserve books.

3.1.5.2 Staff is able to check out books themselves.

3.1.5.3 Users are only able to check out with an active staff member.

## External Interface Requirements

3.2.1 The system must provide an interface to the Library database in order for items to be reliably looked up based on location, genre, type of item, year, title, author, and id.

3.2.2 The interface must be a Java based GUI allowing the users to borrow, return books from the library.

3.2.3 The system should communicate between the user and the server over TCP/IP.

3.2.4 The system should only allow staff to check in/out items, create/delete accounts, add/delete locations, and add staff.

3.2.5 The external interface must not give users even the option to do the same actions as staff. Members should not have a button/option to do privileged actions.

## Internal Interface Requirements

3.3.1 The system will store items by layers of distinction. Layer 1 is Location, Layer 2 is type of item (ex: CD, book, etc.), Layer 3 is genre or year.

3.3.2 The system must process new accounts and items by storing them in correct distinction layers and ensuring duplicates only when valid.

3.3.3 Each account, item, and location will have a unique alphanumeric identifier of 10 characters that can be used to search for a specific id number as well as to track transactions.

# Non-Functional Requirements

## Security and Privacy Requirements

4.1.1 Users and Staff members have to login

4.1.2 Only staff members are able to alter member status, create locations, check-in/out media.

## Environmental Requirements

4.2.1 The system must not require any additional software or hardware other than the existing one to operate properly.

4.2.2 Hardware components should function as intended.

## Performance Requirements

4.3.1 System must render all UI pages in no more than 5 seconds.

4.3.2 The system must be optimized and provide the users or the staff with real time data to manage the borrow/return/overdue functions within 10 seconds.

4.3.3 The GUI must provide the user with a reliable and responsive experience, each user input should have a response within 1 second.

4.3.4 System should process each operation within 4 seconds.

# Use Cases

| **Use Case ID:** | **001** | |
| --- | --- | --- |
| **Use Case Name:** | **Create a Library ID (Library Card)** | |
| **Relevant Requirements** | | **Admin/Librarian Login Access** |
| **Primary Actor:** | | **Librarian and User** |
| **Preconditions:** | | **Librarians must have a functional Administrator account with valid login credentials.** |
| **Postconditions:** | | **After this process there should be a new valid Library ID that will be added to the database of all Library ID’s made.**  **This makes a User into a CardHolder from this point forward.** |
| **Basic Workflow:** | | **User requests a Librarian for a Library Card, the librarian/administrator will open an interface for creating a new Library ID. This will create a random UserID number for the card and the librarian will get information like name, phone number, and address from the user and manually add it to the new ID. Once it is saved the ID is ready to be used for rentals.** |
| **Alternative Workflow:** | |  |
| **Exceptions:** | | **If the librarian cannot connect to the server they cannot make a request for a new Library ID so they cannot begin the process without a newID being created in the system first.** |
| **Related Use Cases:** | |  |

| **Use Case ID:** | **002** | |
| --- | --- | --- |
| **Use Case Name:** | **Checking Out Item** | |
| **Relevant Requirements** | | **Admin/Librarian Login Access**  **Valid CardHolder** |
| **Primary Actor:** | | **Librarian and CardHolder** |
| **Preconditions:** | | **We assume there is a librarian on hand to finalize this process and the user has a valid Library ID.**  **Must have Item available to rent in site inventory.** |
| **Postconditions:** | | **CardHolder will now officially have a rental linked to their ID, has to return by due date or face a fine.** |
| **Basic Workflow:** | | **The CardHolder will request to rent a piece of Item from the Librarian, the librarian will then go to the Rental interface of their program. They must request the user present a valid Library ID (library card). Once they have a valid Library ID entered in the librarian can link the ItemID of whatever the CardHolder wishes to rent. This will update the inventory of the library location and report with the server to make sure all bookkeeping is updated/adjusted. The rental period is 2 weeks from checkout, a user may ask for a single extension of 2 weeks if the book has a waitlist, otherwise they may request as many extensions as needed.** |
| **Alternative Workflow:** | |  |
| **Exceptions:** | | **CardHolder may not rent any Item if their account has a rental hold placed on it from failing to return a currently rented Item. Until the Item is returned and a fine is paid, the status cannot be changed and no further rental actions can be taken.** |
| **Related Use Cases:** | | **Checking in a piece of Item, Account Rental Holds, Paying a late fee and changing account status.** |

| **Use Case ID:** | **003** | |
| --- | --- | --- |
| **Use Case Name:** | **Adding new item to the library** | |
| **Relevant Requirements** | |  |
| **Primary Actor:** | | **Admin and the Librarian** |
| **Preconditions:** | | **The admin or the librarian must be logged in with the right credentials to process CardHolder requests.** |
| **Postconditions:** | | **The new item is successfully added to the library catalog and is available for search and borrowing.** |
| **Basic Workflow:** | | **The admin or librarian selects the “Add New Item” option from the system menu, enters item details such as title, author, category, and availability status, and submits the form. The system validates the data, ensuring no duplicates exist. If the validation is successful, the system saves the new item to the library catalog, making it available for search and loan requests** |
| **Alternative Workflow:** | |  |
| **Exceptions:** | |  |
| **Related Use Cases:** | |  |

| **Use Case ID:** | **004** | |
| --- | --- | --- |
| **Use Case Name:** | **Checking in Item** | |
| **Relevant Requirements** | | **Admin/Librarian Login Access**  **Valid CardHolder** |
| **Primary Actor:** | | **Librarian and CardHolder** |
| **Preconditions:** | | **Librarian logged in to process CardHolder request,** |
| **Postconditions:** | | **After an item is checked back in, the relevant inventory management protocols should adjust the catalog to reflect that an item is available for search and borrowing at its respective location.** |
| **Basic Workflow:** | | **The CardHolder will request to return an item to a Librarian, who will process the request. The librarian will use the Check-In feature of the item rental interface.** |
| **Alternative Workflow:** | | **In the case the user requests to Check-In an item that is late, the item may be checked back in but the CardHolder must pay a Late Fee to remove the rental hold on their account. Please see the use case for Late Fee and Account status handling.** |
| **Exceptions:** | | **If the server cannot update or confirm the check-in of a Item correctly this will hamper the process of a smooth return** |
| **Related Use Cases:** | | **Checking Out Item, Late Fee Handling, Account Status Handling** |

| **Use Case ID:** | **005** | |
| --- | --- | --- |
| **Use Case Name:** | **Removing item from library** | |
| **Relevant Requirements** | |  |
| **Primary Actor:** | | **Admin and the Librarian** |
| **Preconditions:** | | **The admin or the librarian must be logged in with the right credentials to process CardHolder requests.** |
| **Postconditions:** | | **The new item is successfully removed from the library catalog.** |
| **Basic Workflow:** | | **The admin or librarian selects the “Delete Item” option from the system menu, enters item details such as title, author, category, and availability status, and submits the form. The system validates the data, ensuring the item exists. If the validation is successful, the system deletes the item from the library catalog.** |
| **Alternative Workflow:** | | **When the admin or librarian searches for an item using the title, author, or category and the item does not match any existing items. The system then returns an error message saying the item is not found.** |
| **Exceptions:** | |  |
| **Related Use Cases:** | |  |

| **Use Case ID:** | **006** | |
| --- | --- | --- |
| **Use Case Name:** | **User searching for an item** | |
| **Relevant Requirements** | |  |
| **Primary Actor:** | | **Visitor or Cardholder or Librarian** |
| **Preconditions:** | |  |
| **Postconditions:** | | **The member has successfully found the item they are looking for** |
| **Basic Workflow:** | | **The user or member enters attributes of the item they are looking for such as title, author, category, and availability. The system searches for an item with the matching attributes and returns a list of item(s). The list is then displayed to the user with additional information (in addition to the previously mentioned attributes) such as location, and if a waitlist exists or not.** |
| **Alternative Workflow:** | | **When the user searches for an item using the title, author, or category and the item does not match any existing items. The system then returns an empty list.** |
| **Exceptions:** | |  |
| **Related Use Cases:** | |  |

| **Use Case ID:** | **007** | |
| --- | --- | --- |
| **Use Case Name:** | **Late Fee Handling** | |
| **Relevant Requirements** | | **Admin/Librarian Login Access**  **Valid CardHolder** |
| **Primary Actor:** | | **Librarian and CardHolder** |
| **Preconditions:** | | **Librarian logged in to process CardHolder request, CardHolder has an item past due** |
| **Postconditions:** | | **Past Due items will be checked back into the system with relevant inventory bookkeeping and the user has paid a fee to have an account in good standing once again with a strike on their account. Otherwise the account stays in a rental freeze state.** |
| **Basic Workflow:** | | **CardHolder will request to check in an item past its original rent date of 2 weeks without an extension. The process will be similar to checking in any other item, however the Librarian will be notified the account and item in question is past due and requires a late fee be paid. CardHolder pays this fee and the Librarian can manually release the account hold to allow rentals and checkouts once again for the CardHolder.** |
| **Alternative Workflow:** | | **The process is similar to the above, however if the CardHolder does not pay the late fee the account status will not be released back into good standing, it will stay in a rental freeze until the fee is paid in full.** |
| **Exceptions:** | | **If the account status was not marked correctly as past due this could lead to an issue where the user is never penalized for returning an item late** |
| **Related Use Cases:** | | **Checking Out Item, Checking In Item, Account Status Handling, Deleting an Account** |

| **Use Case ID:** | **008** | |
| --- | --- | --- |
| **Use Case Name:** | **Adding New Library Location** | |
| **Relevant Requirements** | | **Admin/Librarian Login Access** |
| **Primary Actor:** | | **Librarian** |
| **Preconditions:** | | **Librarian logged in with valid credentials** |
| **Postconditions:** | | **Librarian has added a new library location/site to the database. System inventory adjusts to reflect this.** |
| **Basic Workflow:** | | **The Librarian should have the option to create a new library location and transfer over Items or use existing Add Items processes to build out a new inventory for this particular location/site and have it appear in the overall database for item searches and other inventory management purposes.** |
| **Alternative Workflow:** | |  |
| **Exceptions:** | | **If there is improper internet connection between server and client. You cannot create a new location locally, it must be verified and saved to the main server database.** |
| **Related Use Cases:** | | **Adding new item to the library, Removing item from library, Transfer Items** |

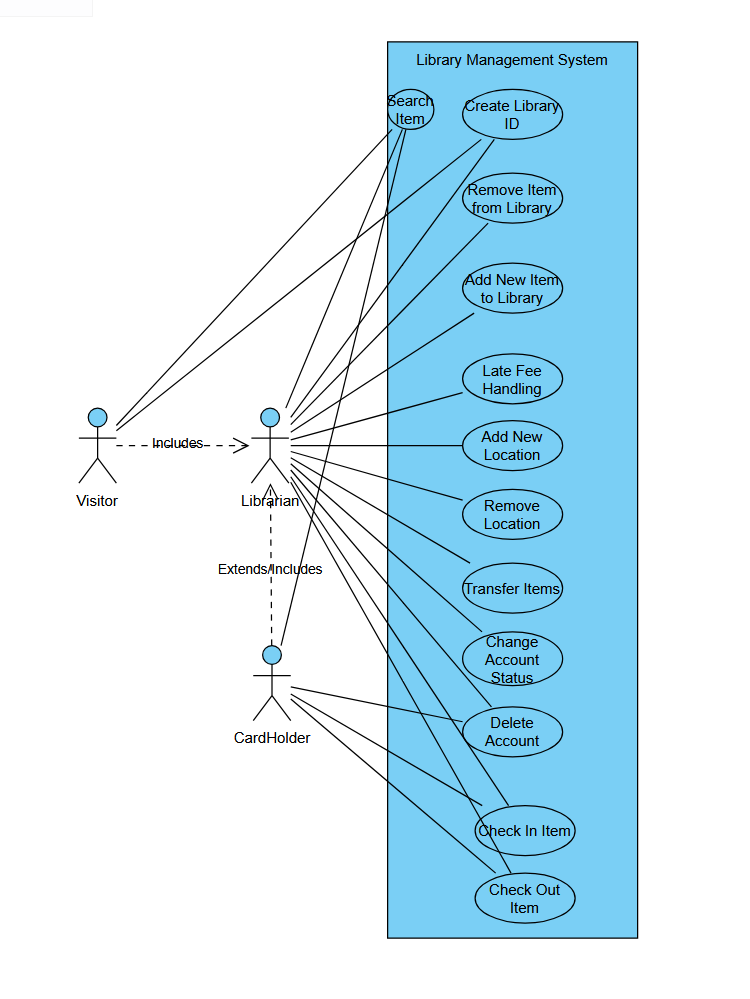
| **Use Case ID:** | **009** | |
| --- | --- | --- |
| **Use Case Name:** | **Removing a Library Location** | |
| **Relevant Requirements** | | **Admin/Librarian Login Access** |
| **Primary Actor:** | | **Librarian** |
| **Preconditions:** | | **Librarian logged in with valid credentials** |
| **Postconditions:** | | **Librarian has removed a library location/site to the database. System inventory adjusted to reflect this.** |
| **Basic Workflow:** | | **The Librarian should have the option to remove a library location and Transfer Over Items or use existing Remove Items processes to remove inventory for this particular location/site and have it reflected in the overall database for item searches and other inventory management purposes.** |
| **Alternative Workflow:** | |  |
| **Exceptions:** | | **If there is improper internet connection between server and client. You cannot remove a location locally, it must be verified and saved to the main server database.** |
| **Related Use Cases:** | | **Adding new item to the library, Removing item from library, Transfer Items** |

| **Use Case ID:** | **010** | |
| --- | --- | --- |
| **Use Case Name:** | **Transfer Items** | |
| **Relevant Requirements** | | **Admin/Librarian Login Access** |
| **Primary Actor:** | | **Librarian** |
| **Preconditions:** | | **Librarian logged in with valid credentials** |
| **Postconditions:** | | **Items can be transferred from one Library location to another with inventory bookkeeping adjusted.** |
| **Basic Workflow:** | | **Librarian logs in with their credentials into their machine, this allows them access to the Transfer Items interface, they can manage their on site inventory and transfer it to another library location** |
| **Alternative Workflow:** | |  |
| **Exceptions:** | | **If there is improper internet connection between server and client you cannot update inventory.** |
| **Related Use Cases:** | | **Adding new item to the library, Removing item from library** |

| **Use Case ID:** | **011** | |
| --- | --- | --- |
| **Use Case Name:** | **Account Status Handling** | |
| **Relevant Requirements** | |  |
| **Primary Actor:** | | **Librarian** |
| **Preconditions:** | | **Librarian is logged in** |
| **Postconditions:** | | **Member's standing is changed** |
| **Basic Workflow:** | | **Librarians log in to the system, and clicking the search users button, they can type attributes of a member (Name, ID) and the system will return a list of members with matching attributes. Librarian will then click the one of the buttons that adjusts standing of a member.** |
| **Alternative Workflow:** | | **If a librarian fails to login, System will ask them to try again.  If a user does not exist with matching attributes, the system will return an empty list.** |
| **Exceptions:** | | **If there is improper internet connection between server and client you cannot update account status.** |
| **Related Use Cases:** | | **Late Fee Handling** |

| **Use Case ID:** | **012** | |
| --- | --- | --- |
| **Use Case Name:** | **Deleting an Account** | |
| **Relevant Requirements** | |  |
| **Primary Actor:** | | **Librarian** |
| **Preconditions:** | | **Librarian is logged in** |
| **Postconditions:** | | **Member's account is removed from the database** |
| **Basic Workflow:** | | **Librarians log in, and clicking the search users button, they can type attributes of a member (Name, ID) and the system will return a list of members with matching attributes. Librarian clicks the delete account button where they have to confirm again they want to do this action.** |
| **Alternative Workflow:** | | **If a librarian fails to login, System will ask them to try again.  If a user does not exist with matching attributes, the system will return an empty list.** |
| **Exceptions:** | | **If there is improper internet connection between server and client you cannot update the database** |
| **Related Use Cases:** | | **Account Status Handling, Late Fee Handling** |

# UML Use Case Diagrams



# Class Diagrams

## 

# Sequence Diagrams

## 