AL Local Library Management System

Andie Leonhard

System Requirements Specification

Updated 12/20/2024

Table of Contents

Customer Problem Statements & System Requirements:

Problem statement: Even for a smaller locale, managing a public library involves the management of a large number of media and users reliably. This means keeping track of all inventory of various types (DVDs, CDs, books)[1] including adding or removing items from inventory, managing users checking out or returning items, managing holds and waitlists for more popular items, and finding locations of items. Without proper management of inventory and space, problems will arise such as lost or misplaced items and cause major issues in the library's ability to continue to provide service to the community.[2]

Objectives: The system sets out to make resource tracking easy for library staff members by keeping real-time records of the location and status of resources and to allow visitors a way to search available resources and manage their checkouts, reservations, and fees from home. The system should be scalable as the library's inventory and visitor counts may vary.

System requirements:

- Staff should be able to:
    - Add new user/library card to system
    - Log in/out
    - Search inventory items
    - Check out items from inventory on a specified visitor's library card/account
    - Return items to inventory
    - Look up visitor information by library card number (Checkouts, holds, reservations, fees)
- Visitors should be able to:
    - Log in/out/create account from library card
    - Search catalogue
    - View checkout history and holds
    - View fees fees
    - Update user info

Typical customers: The software would be purchased by library management, so likely city officials, but users would be librarians and library visitors, so these will be considered most when developing system.

Functional Requirements Specification:

Functionality: The system will consist of a database to manage library resources, a web app for visitors to view available inventory and spaces, as well as to manage their own checkouts, holds, reservations, and fees, and a traditional application to allow for staff management of resources onsite, including processing checkouts and returns. Both the staff application and the web app will have log ins allowing users to access information from the database in real time (with varying permissions), allowing for visitors and staff to view relevant information in real time.

Requirements: The traditional application must be able to support basic library functions performed by staff members, meaning it must be able to access the database. The web application needs to perform simple functions for library users remotely, so must be able to have limited access to the database.

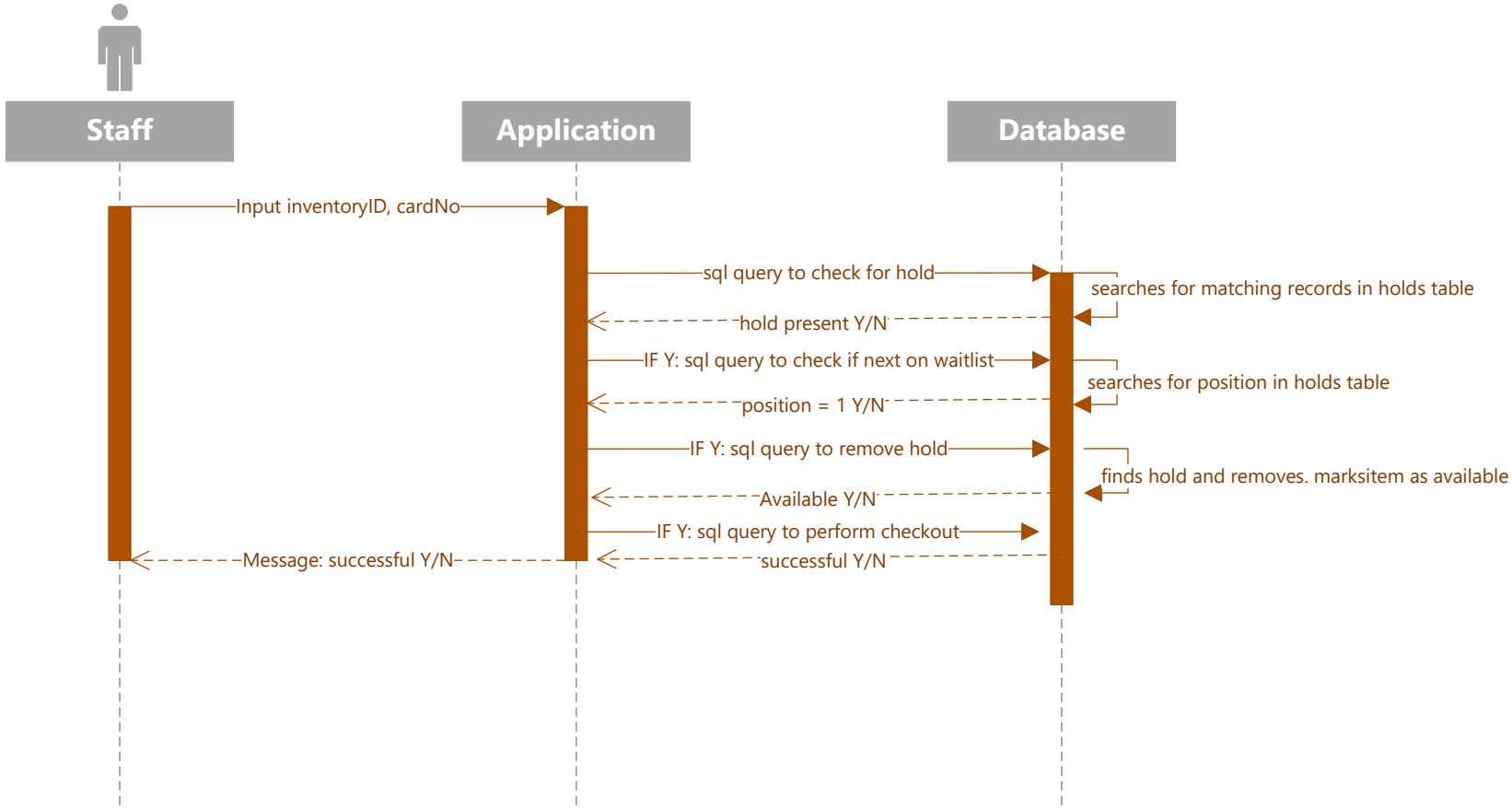
Required methods/use cases:

- Traditional Application:
    - Search inventory items
    - Check out items
    - Return items to inventory
    - Look up visitor information
- Web application:
    - Log in/out/create account
    - **Search catalogue**
    - View item details
    - Place hold
    - Remove hold
    - **View checkout history and holds**
    - View fees/pay fees

Database Requirements: The database must store comprehensive information about inventory as possible, within reasonable time and resource limits. This includes information about the type of media, title, author/artist/director or studio depending on the item type, genre, location within the library, and any other basic identifying information that users might find helpful. In addition, the database must track check out, holds and waitlists, and returns, as well as user information and existing fine balances. This requires a well-built, normalized relational database, which must be compatible with the traditional and web applications that need to access this data.

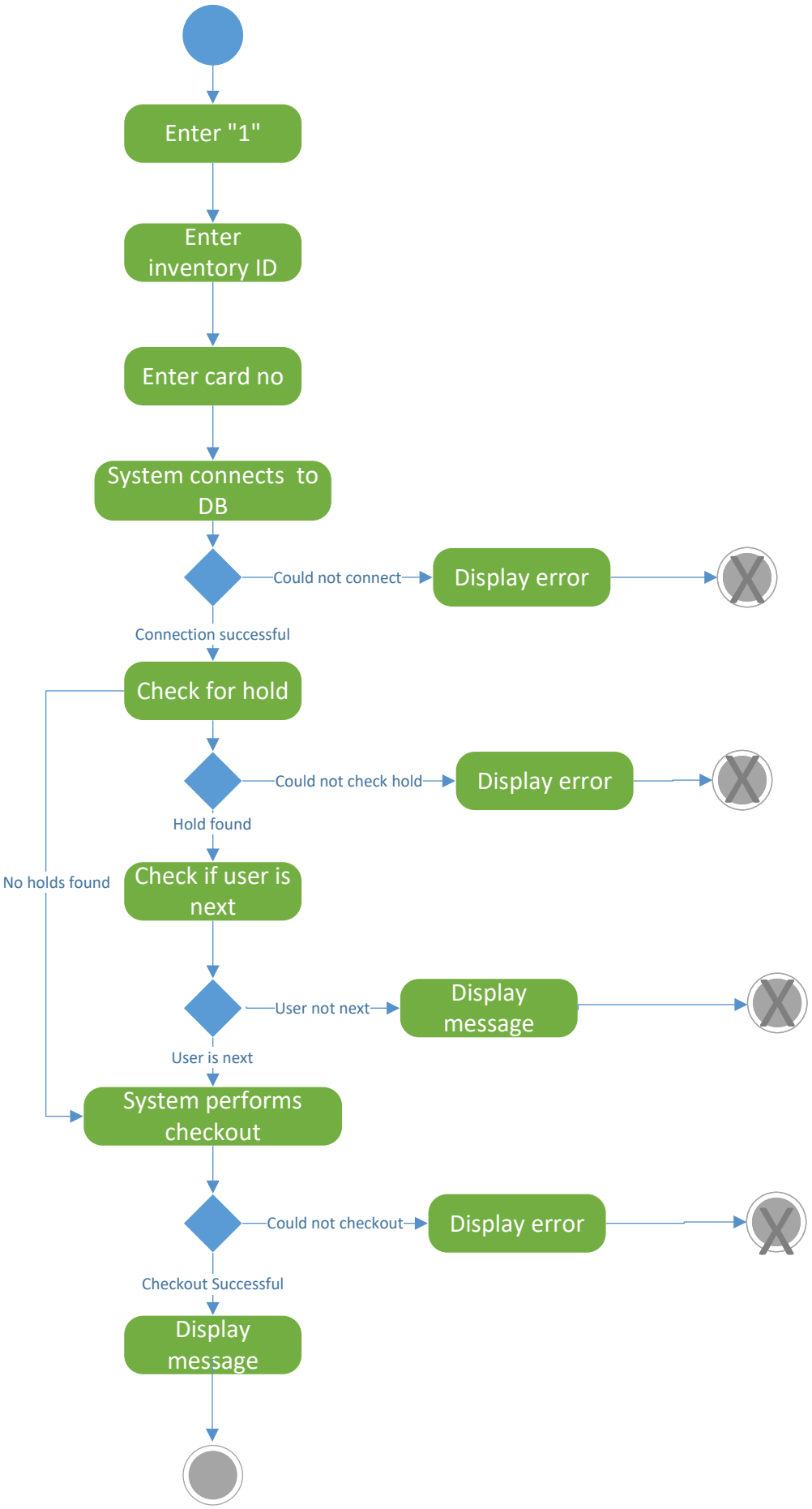
System Sequence Diagram:

Sequence Diagram: Check Out Item

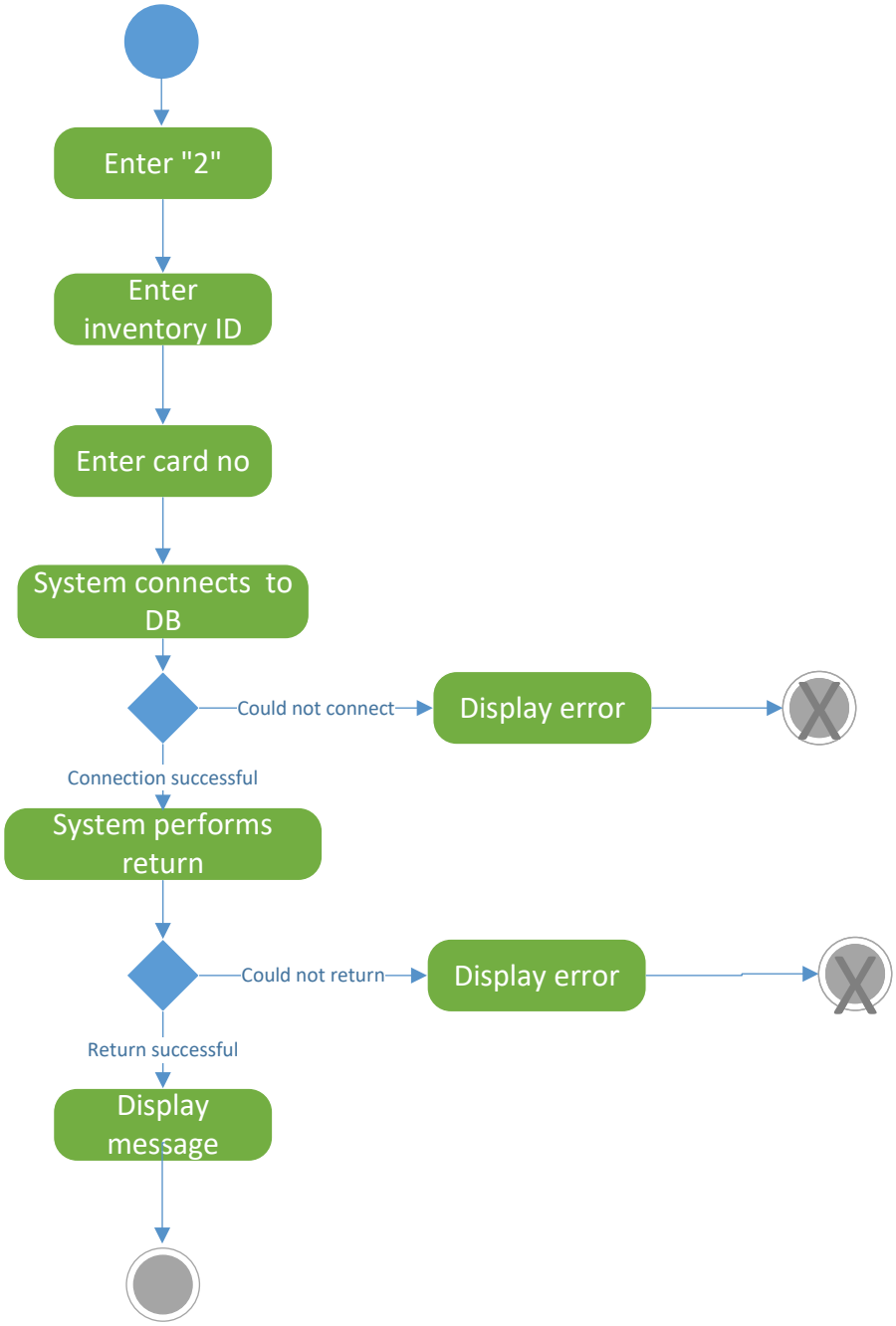


Activity Diagrams:

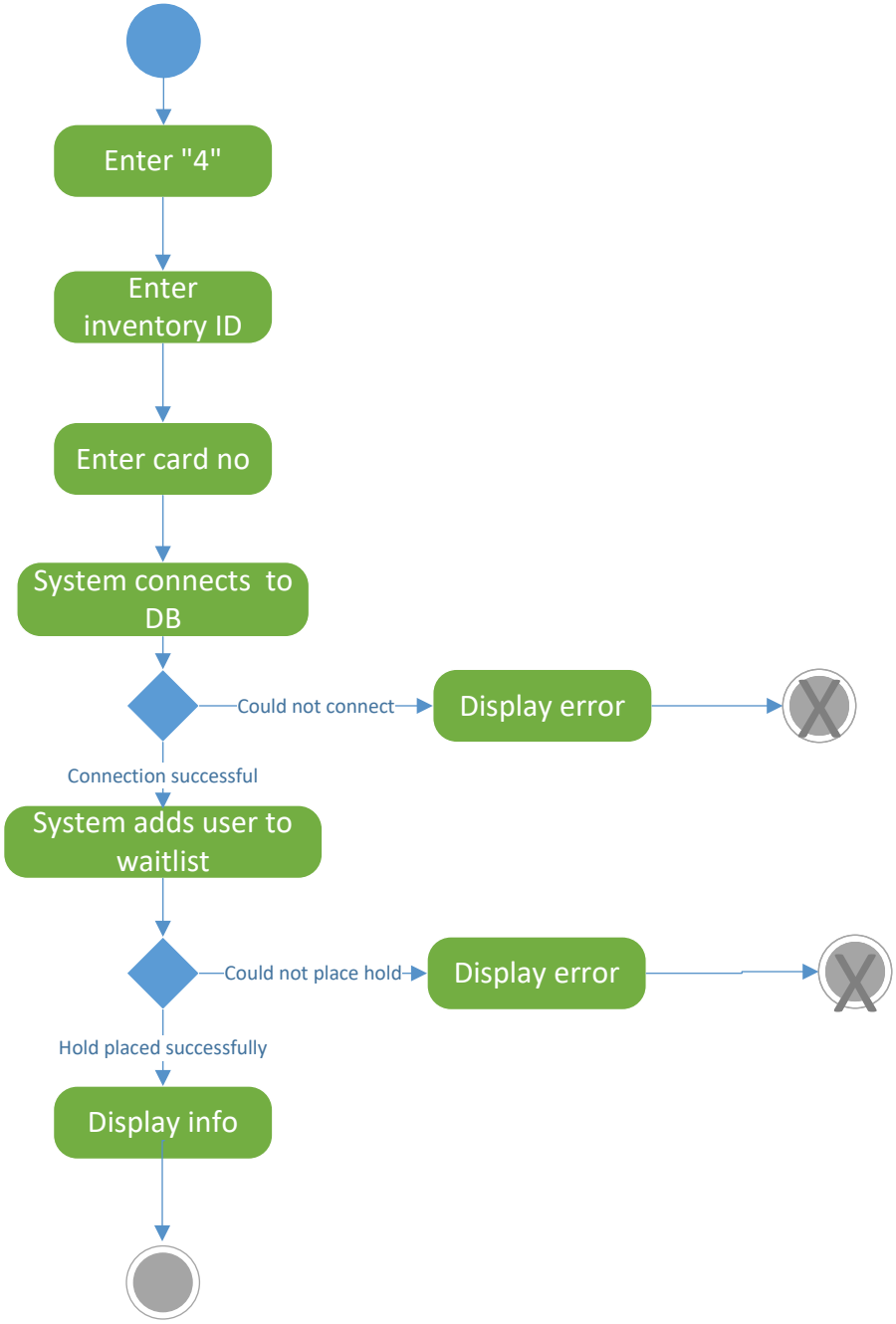
Use Case: Check Out Item



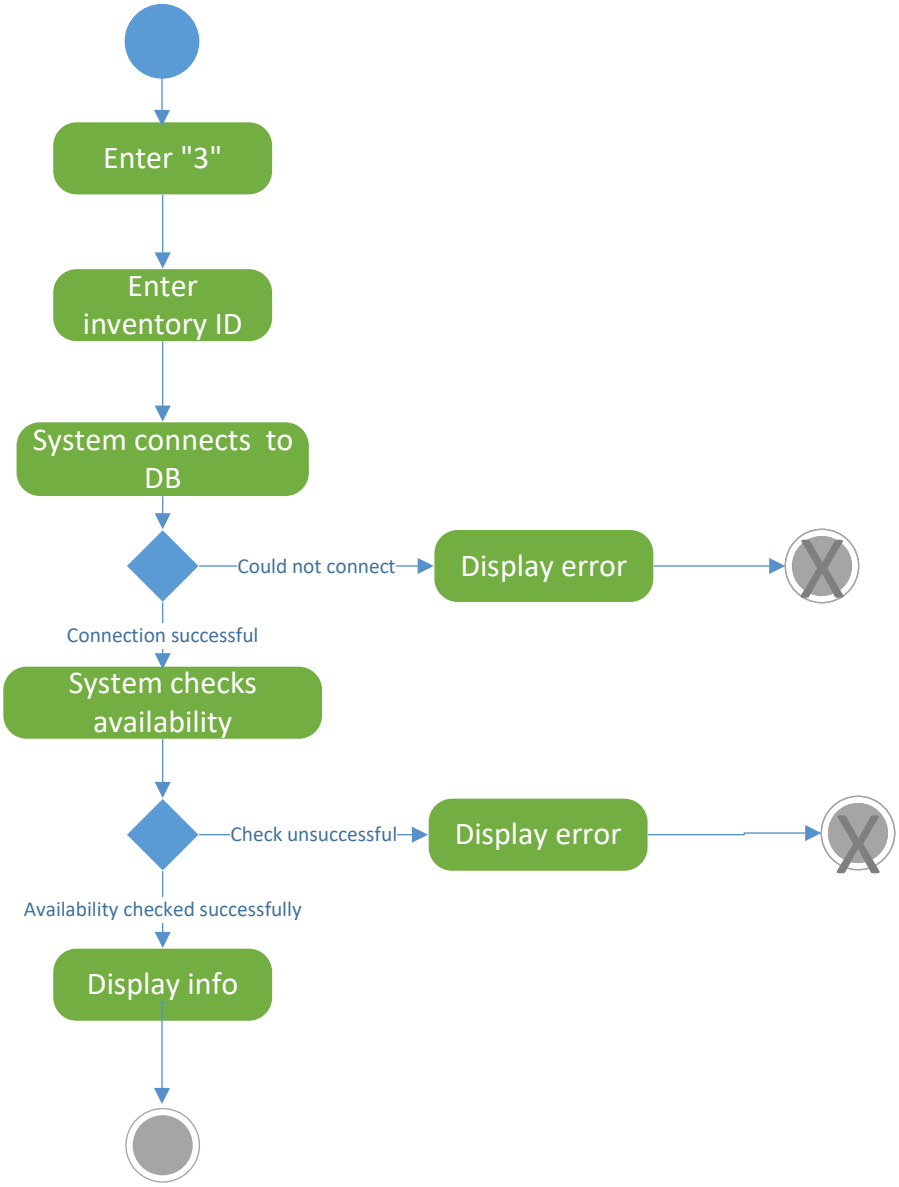
Use Case: Return Item



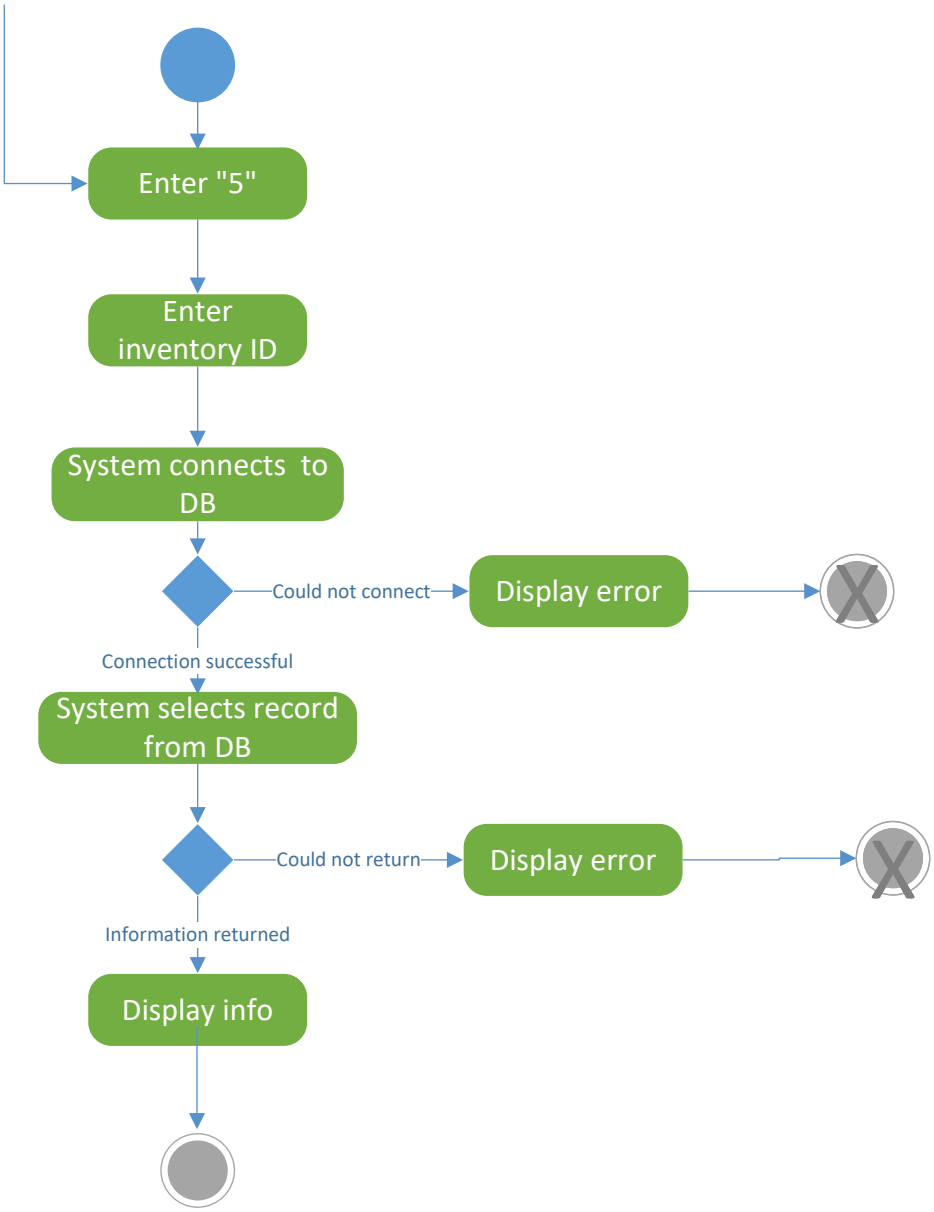
Use Case: Place Hold



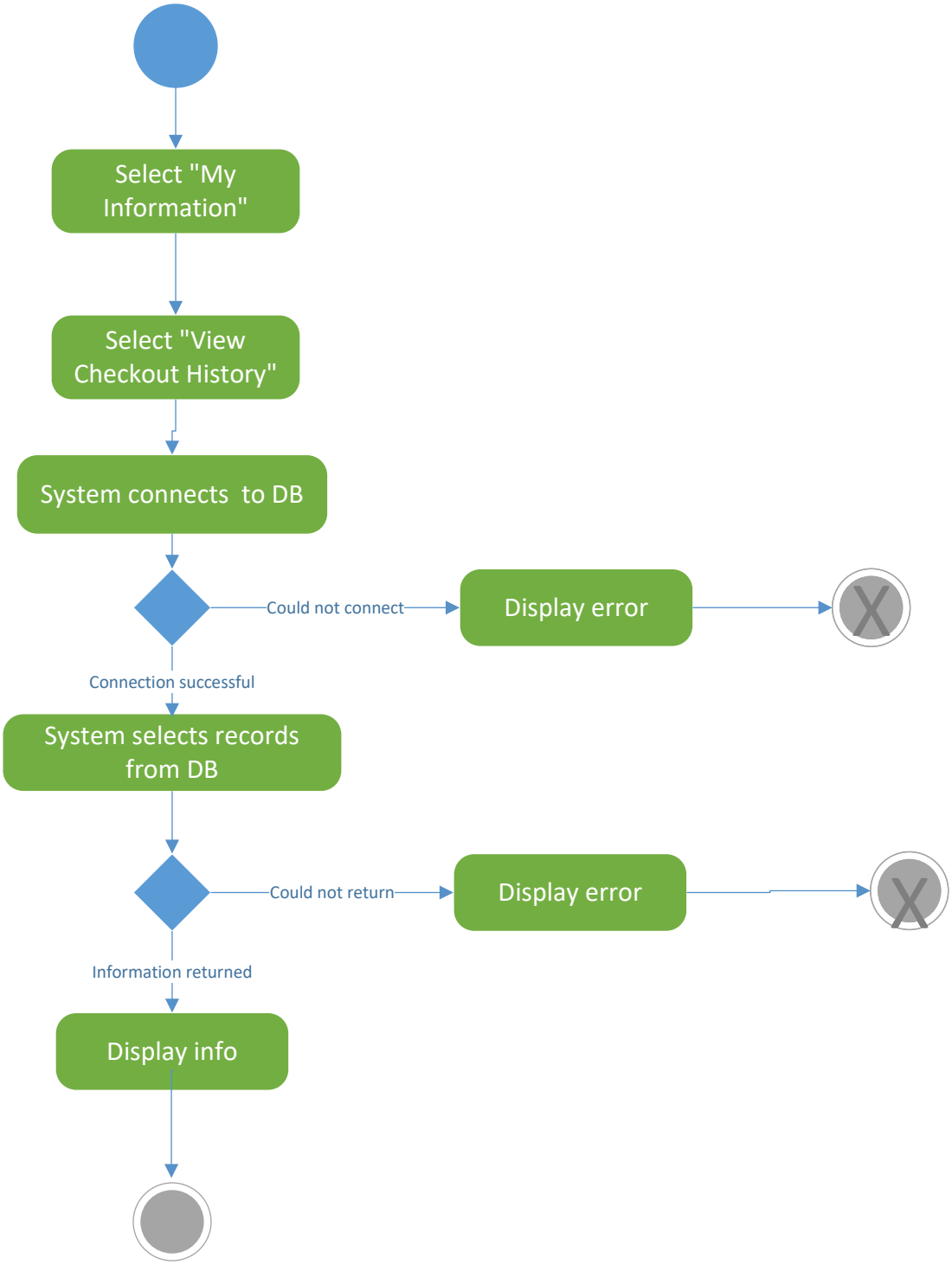
Use Case: Check Availability



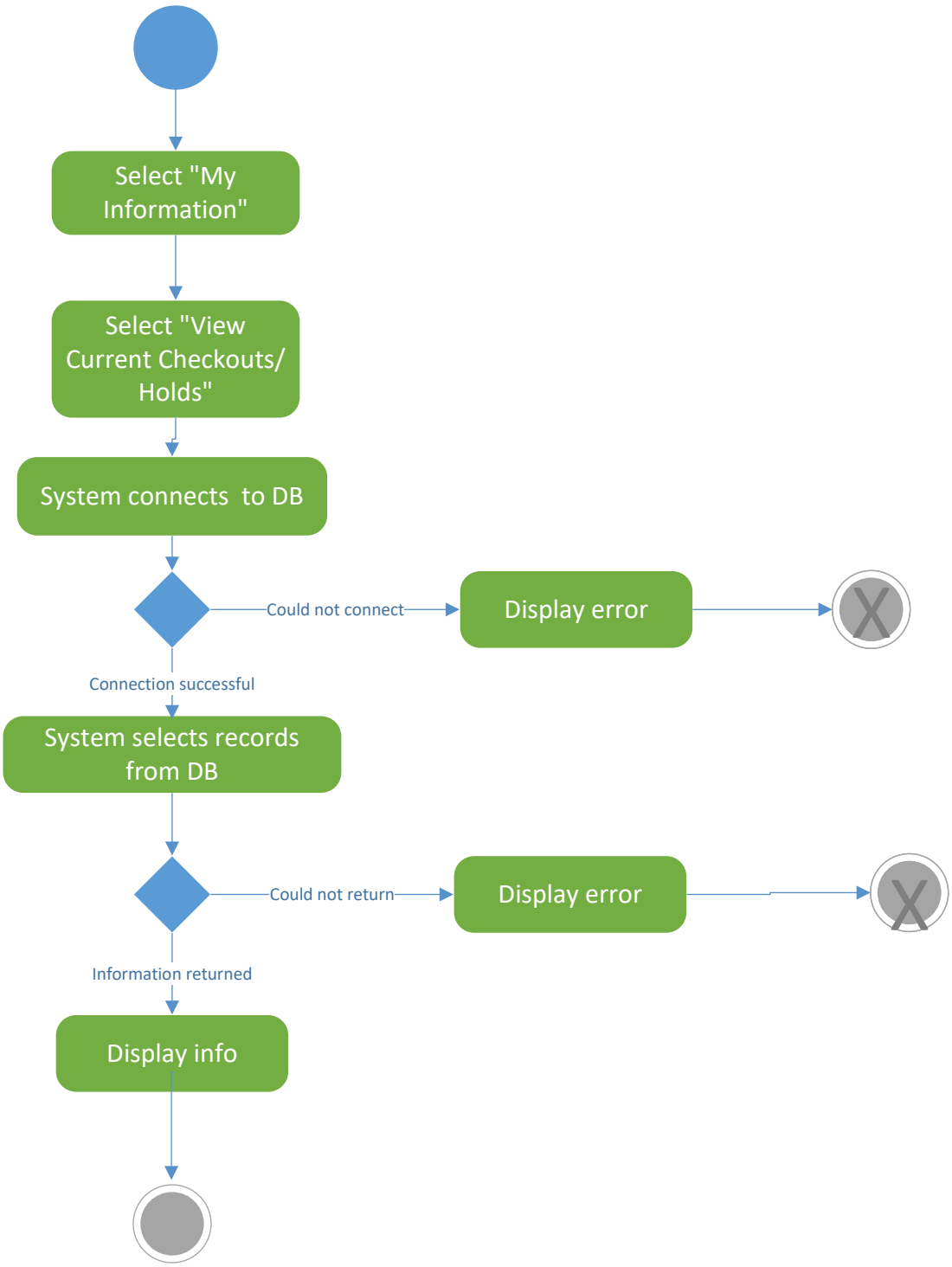
Use Case: Invnetory Search



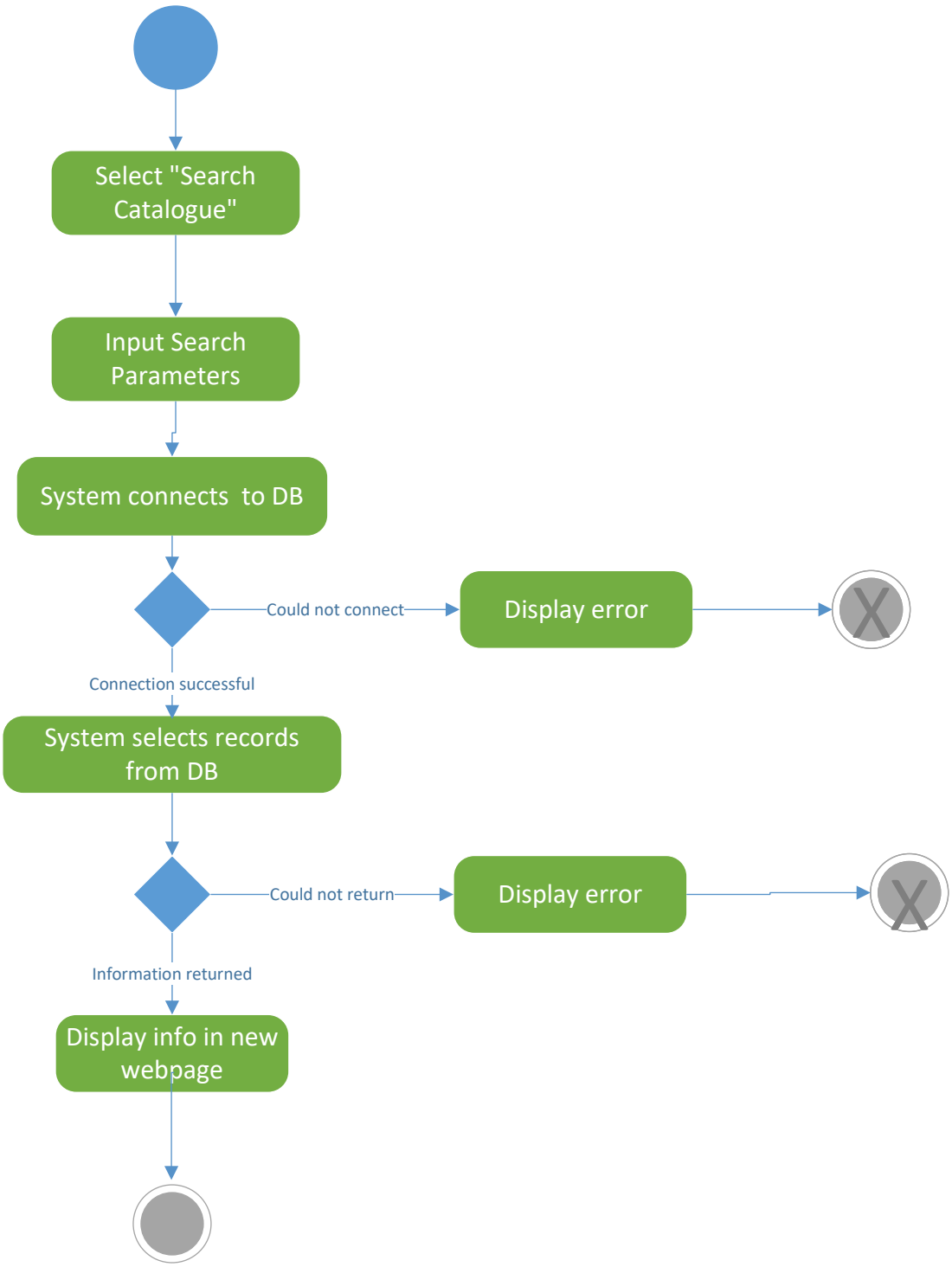
Use Case: View Checkout History



Use Case: View Current Holds and Checkouts



Use Case: Search Inventory



User Interface Specification:

Stakeholders: Library management and staff, community library patrons

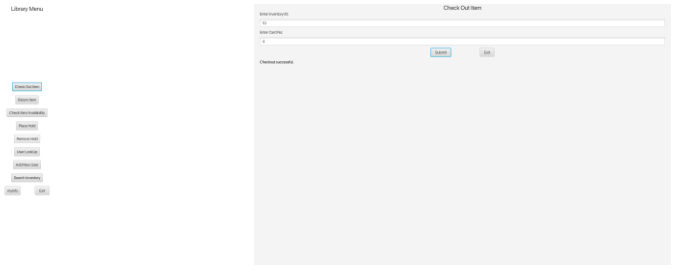
Actors and Goals:

- Library Management: Manage inventory, users, and balances successfully to minimize errors and unnecessary complications and overhead
- Library Staff: To perform basic library functions conveniently and accurately
- Library Users: Access library resources and manage account easily and remotely

Use Cases:

Staff: Checkout item[3]

- Starting at menu, click corresponding button to check out item
- Enter (or, practically, scan in) inventoryID of item to be checked out
- Enter (or, practically, scan in) cardNo of user checking out item



Navigation Path: Menu > Input Inventory ID > Input Card No > Success

Number of clicks: 1 + 3 (to set cursor 2x and then click enter)

Number of Keystrokes: 2+ (depending on values of inventoryID and cardNo)

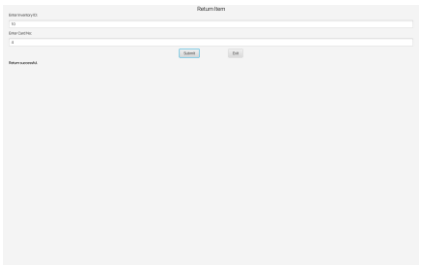
Staff: Return item

- Starting at menu, click corresponding button to return item
- Enter (or, practically, scan in) inventoryID of item to be returned
- Enter (or, practically, scan in) cardNo of user returning item

Navigation Path: Menu > Input Inventory ID > Input Card No > Success

Number of clicks: 1 + 3 (to set cursor 2x and then click enter)

Number of Keystrokes: 2+ (depending on values of inventoryID and cardNo)



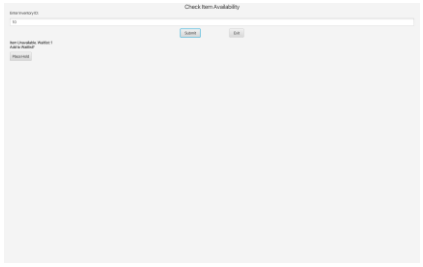
Staff: Check Availability

- Starting at menu, click corresponding button to check availability
- Enter (or, practically, scan in) inventoryID of item to check availability

  Navigation Path: Menu > Input Inventory ID > Success

  Number of clicks: 2 (1 to set cursor, 1 to click submit)

  Number of Keystrokes: 1+ (depending on value of inventoryID)



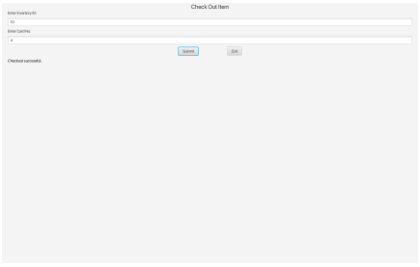
Staff: Place Hold

- Starting at menu, click corresponding button to place hold
- Enter (or, practically, scan in) inventoryID of item to be placed on hold
- Enter (or, practically, scan in) cardNo of user placing the hold

Navigation Path: Menu > Input Inventory ID > Input Card No > Success

Number of clicks: 1 + 3 (to set cursor 2x and then click enter)

Number of Keystrokes: 2+ (depending on values of inventoryID and cardNo)



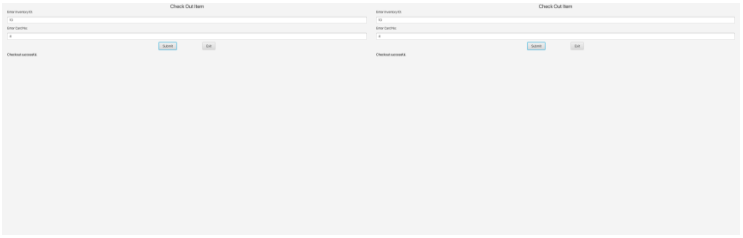
Staff: Inventory Search

- Starting at menu, enter corresponding number to inventory search
- Select item type radio button
- Enter search parameters

Navigation Path: Menu > Select type > Input search parameters > Submit > Success

Number of clicks: 1 + 1 (select type) + 1(or more to set cursor, multiple times if needed) + 1

Number of Keystrokes: 1+ (depends on search parameters



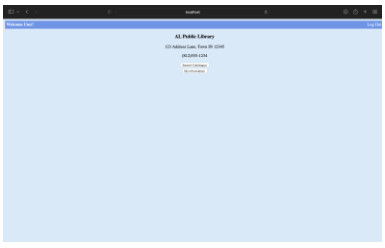
Guest: View Current Checkouts and Holds

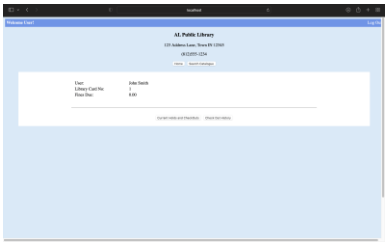
- Start at Home. Select button for “My Information”
- On the new webpage, select the button for viewing current checkouts and holds

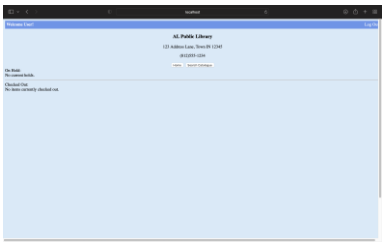
Navigation Path: Home > My Information > Current Checkouts > Success

Number of clicks: 2

Number of Keystrokes: 0







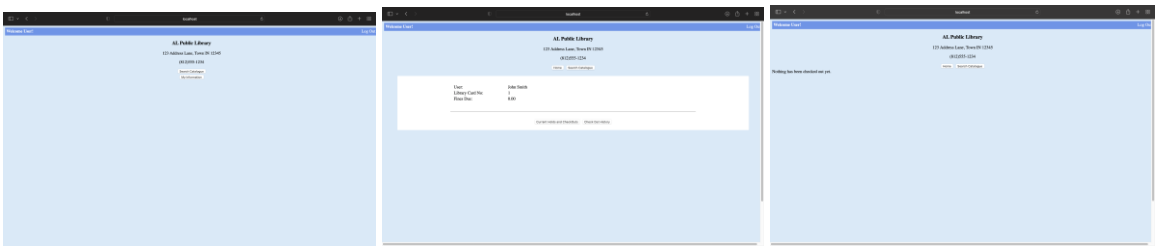
Guest: View Check Out History

- Start at Home. Select button for “My Information”
- On the new webpage, select the button for viewing checkout history

Navigation Path: Home > My Information > Checkout History > Success

Number of clicks: 2

Number of Keystrokes: 0



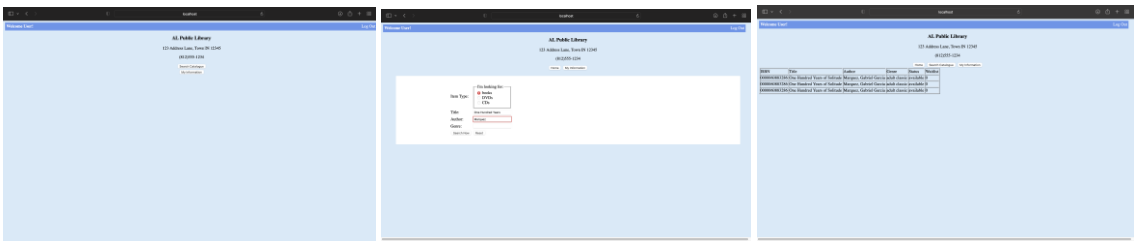
Guest: Inventory Search

- Start at Home. Select button for “Search Catalogue”
- On the new webpage, select the radio button for the desired item type
- Input information in the search fields.
- Select the button for “Search”
- View results on new webpage

Navigation Path: Home > Catalogue Search > Enter Information > Success

Number of clicks: 3+ (may be more to enter cursor into different input fields)

Number of Keystrokes: 1+ (depends on input search criteria



Traceability Matrix

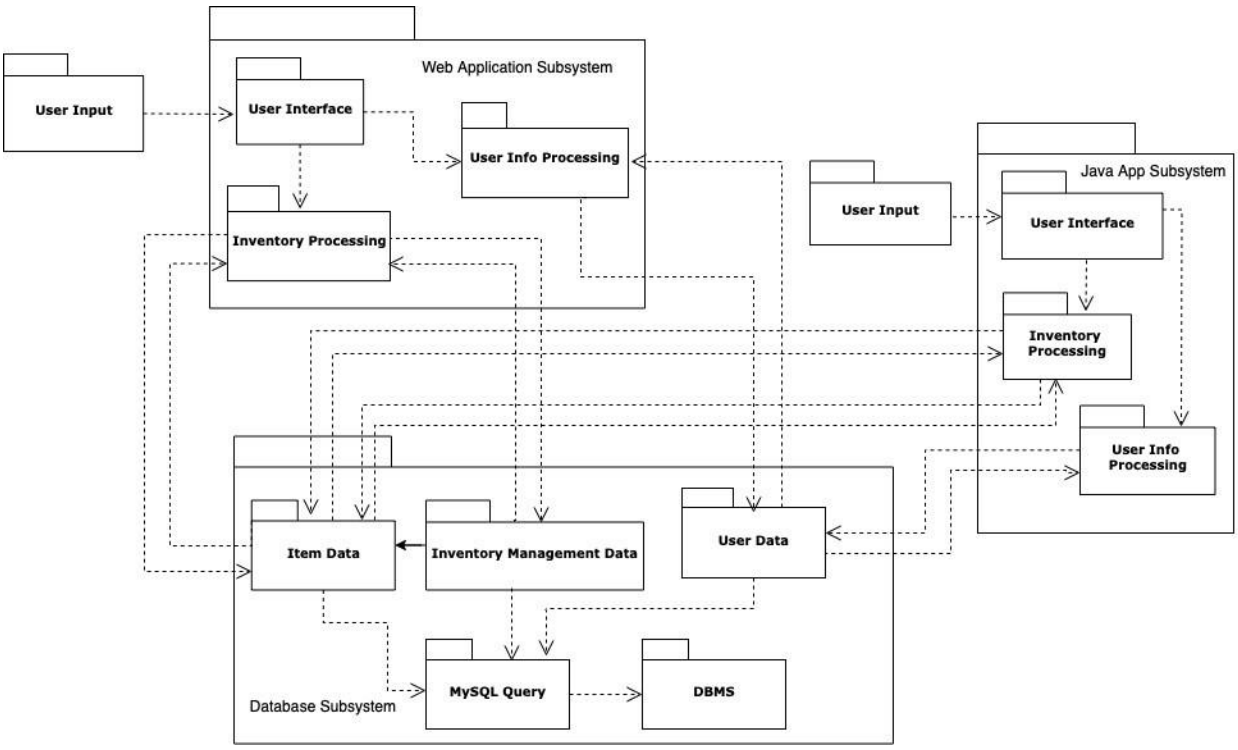
Req.	Description	Test Case	Description	Status
J1	Add new user to system with new library card and either phone or email.	TC1	Verify user can be added with phone number, email, or both, and verify that records in DB are updated.	Complete
J2	Staff log in/out with sufficient security checks	TC2	Verify that a staff member can log in and log out, check input security and that session details are purged after logging out.	Complete
J3	Use application to search inventory: books by title, author or genre, DVDs by title, studio, director, or genre, and CDs by title, artist and genre.	TC3	Verify that items can be searched by each parameter with accurate results in all three type categories. Verify that multiple search parameters can be used at once successfully.	Complete
J4	Check out items from inventory to a specific user's library card.	TC4	Verify that a user can check out an item by library card number and inventoryID. Verify that records are updated in the database, and that checkouts only occur on available items.	Complete
J5	Return items to inventory and mark as available or move up waitlist.	TC5	Verify that a checked out item can be returned to inventory and that the waitlist or availability status is update in the database. Ensure that only checked out items can be returned.	Complete
J6	Check availability of specific items.	TC6	Verify that the availability of an item can be accurately checked using its inventoryID.	Complete
J7	Look up visitor information by library card number, including details about checkouts, holds, fees, name, and contact info.	TC7	Verify that user information can be looked up by entering the library card number. Verify that current and historical checkout data can be viewed and its accuracy.	Complete

W1	User log in/out or create user from library card number.	TC8	Verify that users with a library card number can make an online account with an email and/or phone number, and verify that the user can then log in and out.	In Progress
W2	Search catalogue from web application (same search terms and parameters as for J3 search using application)	TC9	Verify that inventory search is successful for all types of items and with all combinations of search parameters.	Complete
W3	View user checkout history and holds, and current checkouts, and view name, contact info, and balance due.	TC10	Verify that logged in users can view their account information, and that they can access details about current and historical checkouts and holds.	Complete
W4	Change user information or update password.	TC11	Verify that user information can be updated by logged in users and that password changes can occur, with these changes being updated instantaneously in the database.	Complete
D2	Database can be queried for reports.	TC12	Verify that data is up to date, accurate, and able to be queried for specific information regarding inventory and user data.	In Progress

System Architecture and System Design

Architectural Styles: This system is client-server, where for both the Java application and the web application, the client is responsible for user input and interface, but the server-side programming accesses the data and processes the results. The system is two-tiered, involving both the client and server without additional middleware. My system uses a XAMPP server to process data from php programming and using MySQL to manipulate the database, and the results are returned to either the Java application or the web application on the client-side. Because I am using a local XAMPP server, the system is tested and operational on a single machine, but the intent is for the system to be able to run on multiple machines by instead using a non-local server address as this capability would be more practical for business purposes.

Identifying Subsystems: In the package diagram below, the three subsystems are shown, the web application, the Java application, and the relational database.



Mapping Subsystems to Machines: These systems can all be run on the same machine, and are for testing purposes, but generally the web application would be hosted on the Internet for access from basically any device, while the database and Java application would be run from computers (either a single machine or multiple) at the library location.

Persistent Data Storage: The system does require data storage to outlive a single execution of either application. Persistent data storage will be managed by the relational database, the database subsystem in the diagram above. Any edits made in either the web application or the Java application to database records will be reflected in the database instantaneously.

Network Protocol: The web application uses HTTP network protocols, while the Java application uses JDBC protocols to communicate with the server and access the database.

Global Control Flow:

- **Execution orders:** The system is event-driven, as both the web application and the Java application react to user clicks and input to generate options in various different orders. From the web application, the user can click through and perform various tasks and access different webpages by clicking on different navigation buttons, and in the Java application, the user selects options from the menu to begin tasks but can branch off to different tasks from certain windows as well, so the experience is more dependent on user events.
- **Time dependency:** The system processes data in real time. This is important to make sure waitlist positions and availability is accurate when the information is requested, returns are processed immediately so subsequent checkouts can occur immediately, and so inventory items are easy to find without waiting for periodic updates. This is especially important when the system uses multiple machines.
- **Concurrency:** The system does not use multiple threads.

Hardware Requirements: The system depends on screen display and communication with the server, which requires hard disk storage. The system can run either on a single machine or on multiple, in which case the server may not be local, but in my testing of the system, the XAMPP software is stored locally.

The most basic requirements to run this system would be a color display with a minimum 640 x 480 pixels, a desktop for the web application and either a desktop or mobile device with Internet access for the web application, and at least 1 GB of RAM and 1GB of hard drive space, though these requirements may increase if the size of the database increases. The system uses roughly 30 kbps of bandwidth per user so this would be the minimum bandwidth required for the smallest scale of the system (single machine), but this requirement would need to be scaled up as the system expands.

Algorithms and Data structures (If you have those)

Algorithms:

1. **Search Algorithm:** Implements a search function to find inventoyr items by title, author, or genre.
    a. **Algorithm**: **Linear Search**: Because the database tables are unsorted, the linear search is the most straightforward search and was used for this reason. As the database expands to include more items, using SQL to sort the tables and then using a binary search would be more efficient.
2. **Authentication Algorithm:** Handles secure user registration and login.
    a. **Algorithm**: **Session Management:** This system checks user credentials on log in, and stores information in the session details. To increase security on future versions, password hashing could be considered.

Data Structures:

1. **Relational Database:** Normalized to 3NS, the relational database is used to dynamically track inventory and user information for the whole library system in a way that minimizes redundancies and
    a. **Tables:** Store a collection of records of various kinds of data in the form of a 2D array or matrix, with dependencies existing between several tables.
        i. Data tables in library database: artist, audioGenre, author, book, bookInventory, cd, cdInventory, checkOutHistory, director, dvd, dvdDirector, dvdInventory, genre, holds, inventory, itemStatus, staff, staffPosition, studio, type, user
        ii. Views:
            1. bookinfo(inventoryID, ISBN, title, author, status)
            2. cdinfo(inventoryID, cdID, cdTitle, artistName, status)
            3. dvdinfo(inventoryID, dvdID, dvdTitle, genre, status)
            4. inventorytitles(inventoryID, title)
            5. staffview(employeeID, employeeFName, employeeLName, username, positionTitle)
    b. **Indexes:** Unique indexes in each table for faster querying
    c. **Referential Integrity:** normalized to 3NS. cascading deletes disabled for data safety (I'm clumsy), so related records will need to be deleted before referenced columns can be deleted, and records are constrained so as to require referenced data to be present in the database prior to being referenced by another table (it is not automatically triggered, but that could be implemented with updates)

2. **Arrays/Array Lists:** Used to store search parameters in order to add them to search queries. These arrays are not stored for use beyond the query execution.
3. **Session variables:** Session data is stored to be able to keep users from having to constantly repeat input while maintaining important details for functionality and security.

User Interface Design and Implementation, Design of Tests

User Interface Design and Implementation:

- **Java Application:**
  - **Users:** library staff
  - **Pages:**
    - Menu:
      - Options for: check out, return, check availability, place hold, remove hold, look up user, add new user, inventory search, my information, and exit
    - Login Page: Staff log in.
    - Check Out Item: check out item for a user by entering inventoryID and library cardNo
    - Return Item: return an item by entering inventoryID and user's library cardNo
    - Check Item Availabilty: Request availability informaiton (status, waitlist) for an item by entering its itemID
    - Place Hold: places a hold for a user by entering the library cardNo and the inventory ID, displays user's position on the waitlist if successful.
    - Remove Hold: removes an existing hold for a user on a specific item by entering the inventoryID and the user's cardNo, and takes the user off the waitlist and moves everyone behind them up in the queue.
    - User LookUp: allows staff to search for specific user by card number. From there, additional pages with information regarding current holds and checkouts or historical checkouts and returns can be accessed.
    - Add New User: allows staff to set up new patron accounts with new library card given name and contact information and entering the card number.

        - Search Inventory: Opens a page with options to search for books, DVDs, or CDs with a few different parameters. Clickign search will open another window with information about all the search results.
        - MyInfo: A staff member can open this page to view their own details, and from here an additional window can be accessed from where they can update their password.
    - **Web Application:**
        - **Users:** library patrons
        - **Pages:**
            - Log In Page: allows users with a library card to sign up for an online account or log in with their existing one.
            - Home Page: This page is just a landing page with library information that allows users to select to view their own information or to search the catalogue.
            - My Information: This page displays user information like name, contact information, and fines due, as well as allowing users to continue to pages where their current holds and checkouts or their checkout history can be viewed.
            - Inventory Search: allows users to select whether they would like to search for books, cds, or dvds, and then allows them to enter search parameters, which are then displayed on a new page.
- **Technologies Used:** HTML, CSS, JavaScript for the frontend on the web application. JavaFX for the application.
- **Implementation Details**
    - Goal was simplicity and consistency. The Java app was kept black and white for accessibility, and the same order was used for input prompts. Input was kept simple and easy with large buttons. The web app used few colors, but tried to maximize contrast in areas where legibility was important for accessibility reasons. The web app and the application line up pretty well in terms of displaying user and inventory information, which could be useful for those using one system at work and one at home.
    - In future updates, more attention could be paid to design and aesthetics, but the general goal of accessibility and usability seems to have been met.

Design of Tests:

- **Unit Testing**: Testing each individual function on both the web app and java application. Completed before integration with database, or in the case of the java application, before development of the GUI.
- **Integration Testing**: Testing interactions between modules, testing as an end user and ensuring the database is updated accordingly.
- **Performance Testing**: Due to resource constraints (I am a single person without access to large amounts of people to test at once), the system has only been tested running on 2 machines at a time, which is not sufficient, but more performance testing would be completed before go-live with clients.

Test Case	Description	Expected Result	Status
TC1	Verify user can be added with phone number, email, or both, and verify that records in DB are updated.	Success message. User can be found in the 'user' table in the database with correct card number. User can checkout items or be looked up in the app.	Passed
TC2	Verify that a staff member can log in and log out, check input security and that session details are purged after logging out.	Menu page loads, and when viewing "myInfo" the information is that of the specified employee.	Passed
TC3	Verify that items can be searched by each parameter with accurate results in all three type categories. Verify that multiple search parameters can be used at once successfully.	New window opens with search results that are accurate and complete for all kinds of search query. Information displayed is up to date.	Passed
TC4	Verify that a user can check out an item by library card number and inventoryID. Verify that records are updated in the database, and that checkouts only occur on available items.	Success message. New record in checkOutHistory can be found in the database for the transaction. The status of the item in the inventory table has been updated to be unavailable.	Passed
TC5	Verify that a checked out item can be returned to inventory and that the waitlist or availability status is update in the database. Ensure that only checked out items can be returned.	Success message. Record for the tansaction in checkOutHistory has a return date, and the status is listed as available in the inventory table or the waitlist has been updated accordingly.	Passed
TC6	Verify that the availability of an item can be accurately checked using its inventoryID.	Message displayed. Check for accuracy in the database.	Passed

TC7	Verify that user information can be looked up by entering the library card number. Verify that current and historical checkout data can be viewed and its accuracy.	User information displays and is accurate. Clicking on historical data button displays another window with accurate data, as does clicking on the current data button.	Passed
TC8	Verify that users with a library card number can make an online account with an email and/or phone number, and verify that the user can then log in and out.	Success message. User can log in with the chosen password right away. User record has been updated in the user table to show password.	In Progress
TC9	Verify that inventory search is successful for all types of items and with all combinations of search parameters.	New page opens with search results that are accurate and complete for all kinds of search query. Information displayed is up to date.	Passed
TC10	Verify that logged in users can view their account information, and that they can access details about current and historical checkouts and holds.	User information displays and is accurate. Clicking on historical data button displays another window with accurate data, as does clicking on the current data button.	Passed
TC11	Verify that user information can be updated by logged in users and that password changes can occur, with these changes being updated instantaneously in the database.	Success message. Information is updated on the page, and is seen updated in the user table in the database.	Passed

Project Plan:

Project planning: For the purpose of this project, the system will consist of a web app, a traditional application, as well as a database, all of which will be basic enough to be hosted on my local machine, so everything should be able to run from one machine, but the app will have to be mac/pc compatible while the web app and database will be run from the web. In the case of a business, a larger system would be more practical, so scalability is important.

Development approach: As of now, the plan is to create a relational database in phpMyAdmin, the web application will be created using html/javascript with php, and the web application will primarily be programmed with java. Users will use the web app and java app interfaces to access the database indirectly to provide more security and access controls as well as a better and more usable graphical interface.

Development plan:[4]

- Complete additional modeling/planning activities for system (ongoing)
  - Progress: Most of the planning is done, but some modeling for additional use cases is still required. In addition, requirements and plans are updated as necessary as the project progresses.
- Develop basic database framework and normalize (~1-2 week)
  - Progress: Database is in working order and normalized, but additional views, triggers, or procedures may be added if required as project progresses.
- Create basic functioning application, integrating access to database (~3-4 weeks, ongoing)
  - Progress: While several basic functions have been implemented in the app, several more are still required. In addition, the GUI needs implemented as well. Integration with the database has been successful so far.
- Create web application, integrating access to database (~3-4 weeks, ongoing)
  - Progress: The web application functions at the most basic level and integration with the database has been successful so far, but additional features will be implemented as well as work on the visual appearance.
- Test functionality and usability (~1 week)
- Adapt/make changes/finalize (~1 week)

---

[4]Project schedule was adapted due to new due date not leaving time for planning stages to occur prior to development beginning. In addition, original schedule was incredibly optimistic about my coding skills and time management, so development will continue concurrently with other steps.