WASHINGTON STATE UNIVERSITY
VANCOUVER

CS 320 Course Project - Software Design Document

for

Retro Game Exchange Database

Prepared by

Group Name: Team 22

Logan Tan	11631408	Logan.tan@wsu.edu
John McEchron	011749059	johnathan.mcechron@wsu.edu
Brandon	011725342	brandon.huang@wsu.edu
Daniel Lee	011546543	daniel.lee4@wsu.edu

1 Introduction

1.1 Project Overview

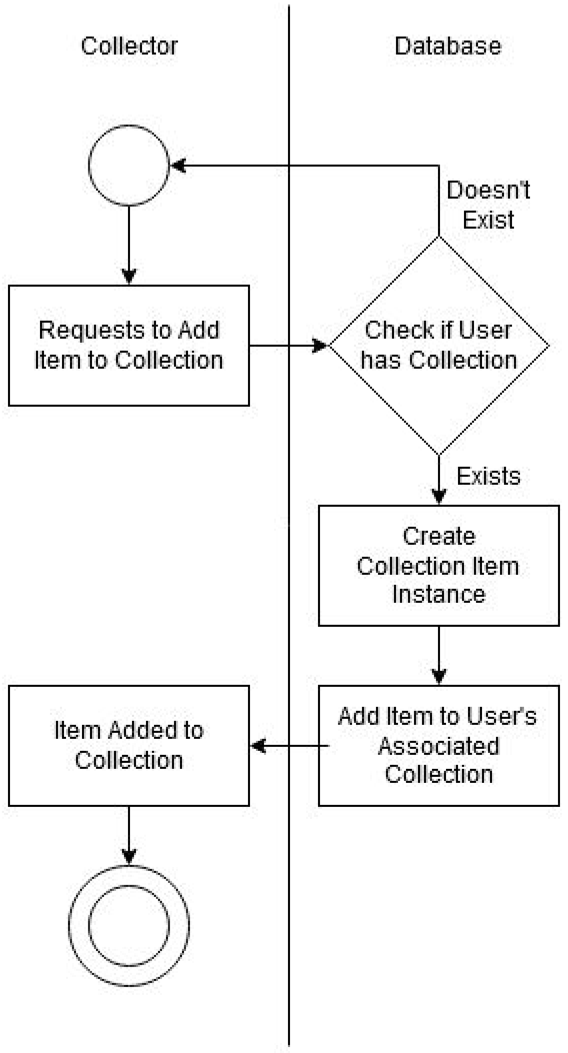
This Project is a Retro Games database. This database will store information, such as console release year, and developer, about each game, along with which users possess the games, and would be willing to exchange them. This will be handled by the database, and it's and helper functions to associate user accounts with games they own, and will help mediate exchanges. In this document We will be showing the class relationships, and which functions call another class within Class diagrams, while we will be showing the step by step actions taken by each function though sequence diagrams. With the broadest scope being displayed thoughActivity diagrams. where the user requests an action to be done, and we show which classes called or are modified.

1.2 Definitions, Acronyms and Abbreviations

Accessory = Video Game Console Peripheral Device including Joysticks, Controller, Light Gun, etc.
AWS = Amazon Web Services
C64 = Commodore 64
Cart = Video Game Cartridge
GCP = Google Cloud Platform
HTTPS = Hyper Text Transport Protocol Secure
NES = Nintendo Entertainment System
PS1 = Sony Playstation

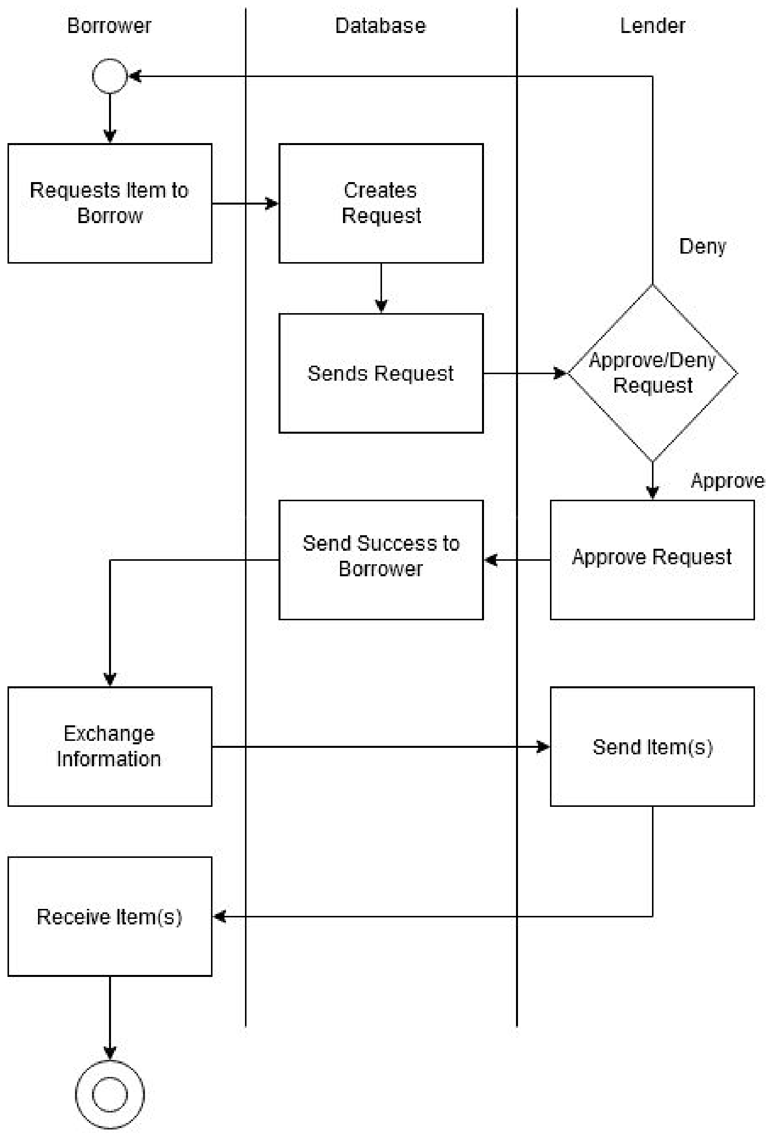
2 Activity Diagram(s)

2.1 Add Item to Collection


Collector
Database
Doesn't Exist
Requests to Add Item to Collection
Check if User has Collection
Exists
Create Collection Item Instance
Item Added to Collection
Add Item to User's Associated Collection

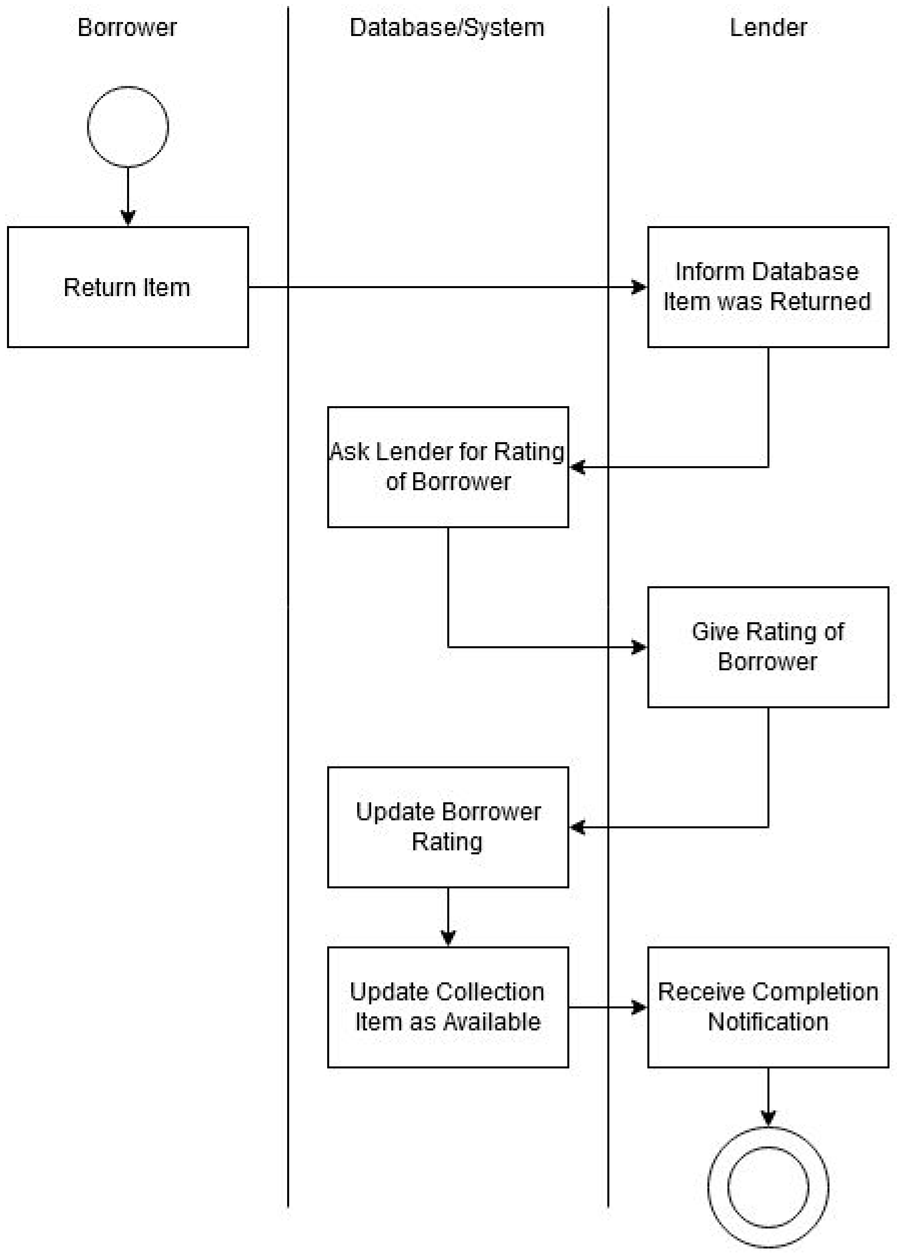
Title: Add Item to Collection
Associated Use Case: Add item to collection (3.2.6)
Start State: A user wants to add an item to their collection.
End Case: The item is added.
Description: A user wants to add an item to their collection and sends a request with relevant information. The system checks if the user has created a collection. If they have, create a new item in the associated users collection and inform them of the success.

2.2 Borrow Request



Title: Borrow Request
Associated Use Case(s): Request Borrowed Item (3.2.9), Respond to Request (3.2.10)
Start State: A borrower requests to borrow an item.
End State: A request is approved or denied.
Description: A borrower, through the database, sends a request to the borrower for an item from a lender. The lender may then approve or deny the request. If the request is approved, the system will inform the borrower, allowing information exchange and then sending of an item.

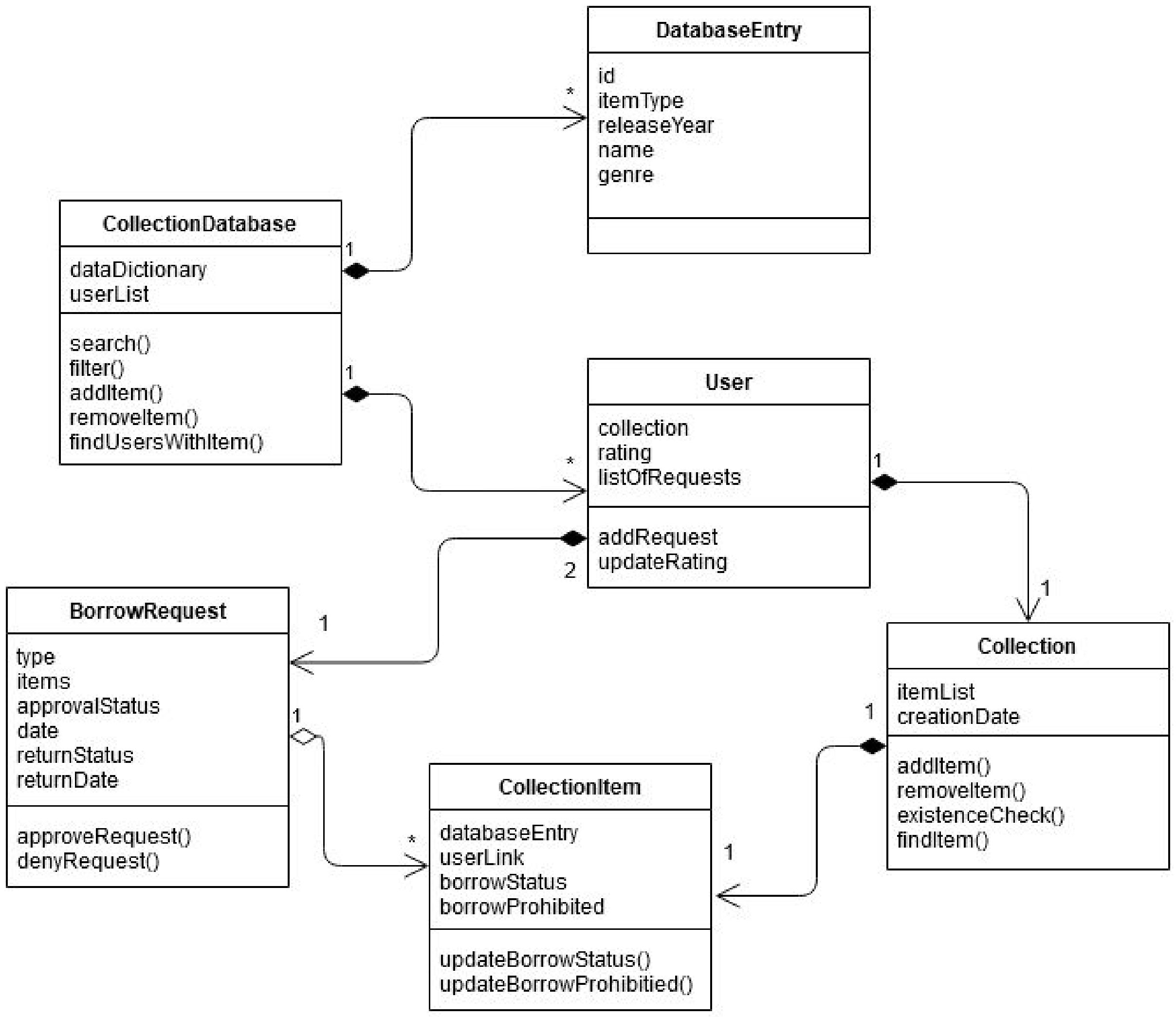
2.3 Return Item


Borrower
Database/System
Lender
Return Item
Inform Database Item was Returned
Ask Lender for Rating of Borrower
Give Rating of Borrower
Update Borrower Rating
Update Collection Item as Available
Receive Completion Notification

Title: Return Item
Associated Use Case(s): Return Item (3.2.13)
Start State: A borrower returns a lent item.
End State: The system is updated with the availability and the borrower is rated.
Description: An item is returned to the lender. The lender lets the system know and gives the borrower a rating based on the condition returned, punctuality and other external factors. The system marks the item as available again and sends a notification to both parties.

3 Class Diagram(s)

3.1 Retro Games Database

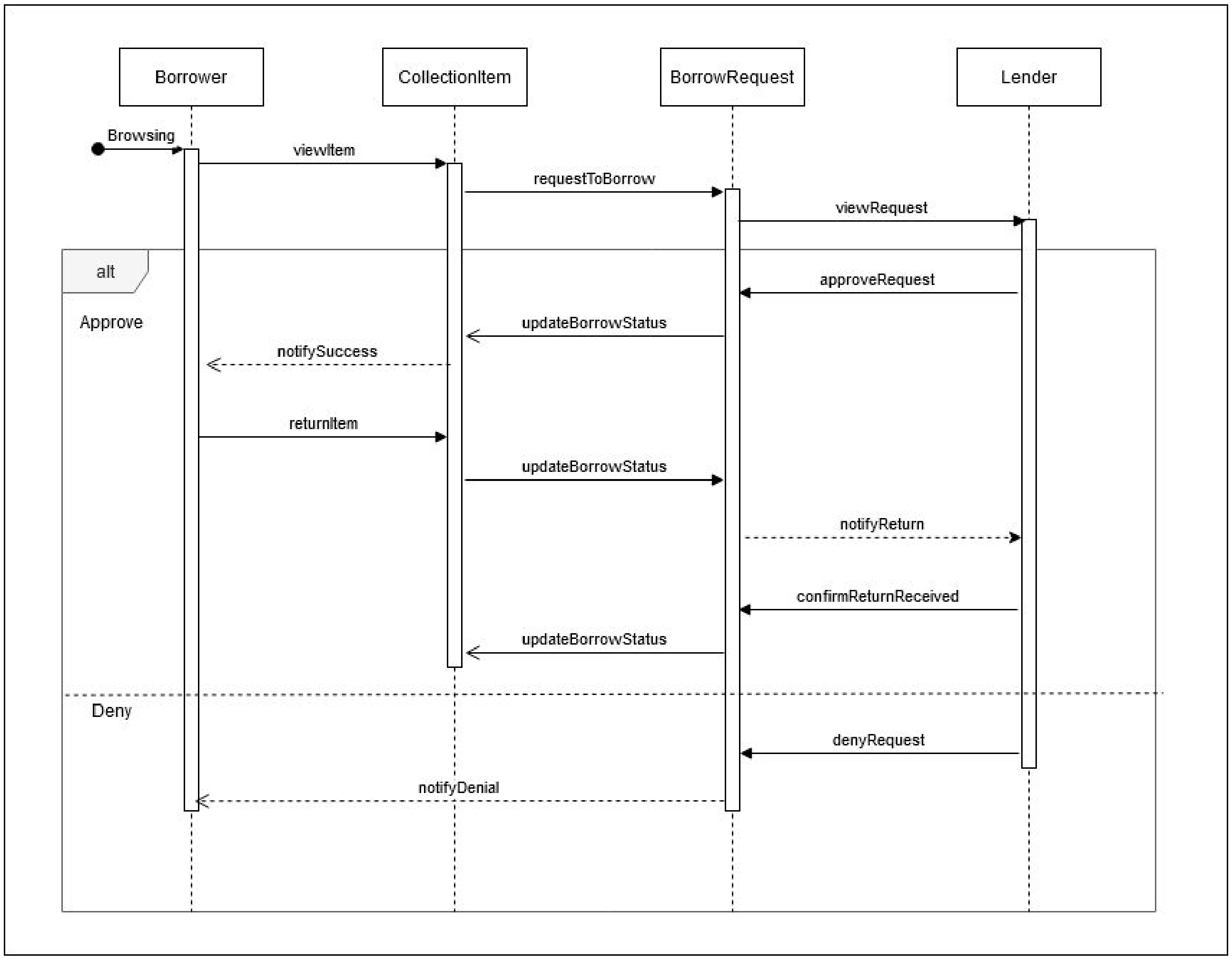


Class Name	Description
CollectionDatabase	Contains a database full of DatabaseEntries for all of the catalogued games/accessories. Can search the database and modify it. Also contains a list of all the users currently registered.
DatabaseEntry	Represents a catalogued item. Has fields for information including type, name, genre, release year, and a unique id to the specific entry.

User	Represents the user. Contains their associated collection, a user rating, and requests for which they are part of.
BorrowRequest	Represents an unresolved borrow request. Contains relevant information like items involved, approval status, date, return status, return date, and whether the associated user is a borrower or a lender. 2 BorrowRequests are made for one actual request, one associated with the Lender, and one with the borrower. Resolved when the Lender informs the system of the return of the items.
Collection	Represents the items in a users collection. Contains a list of CollectionItems and can add or remove from the list as well as search it.
CollectionItem	Represents a specific item in the collection. Can be marked as unavailable through the borrowStatus or borrowProhibited flags. Holds a link to the associated user and the unique ID of the database entry that corresponds to the same item.

4 Behavioral Diagram(s)

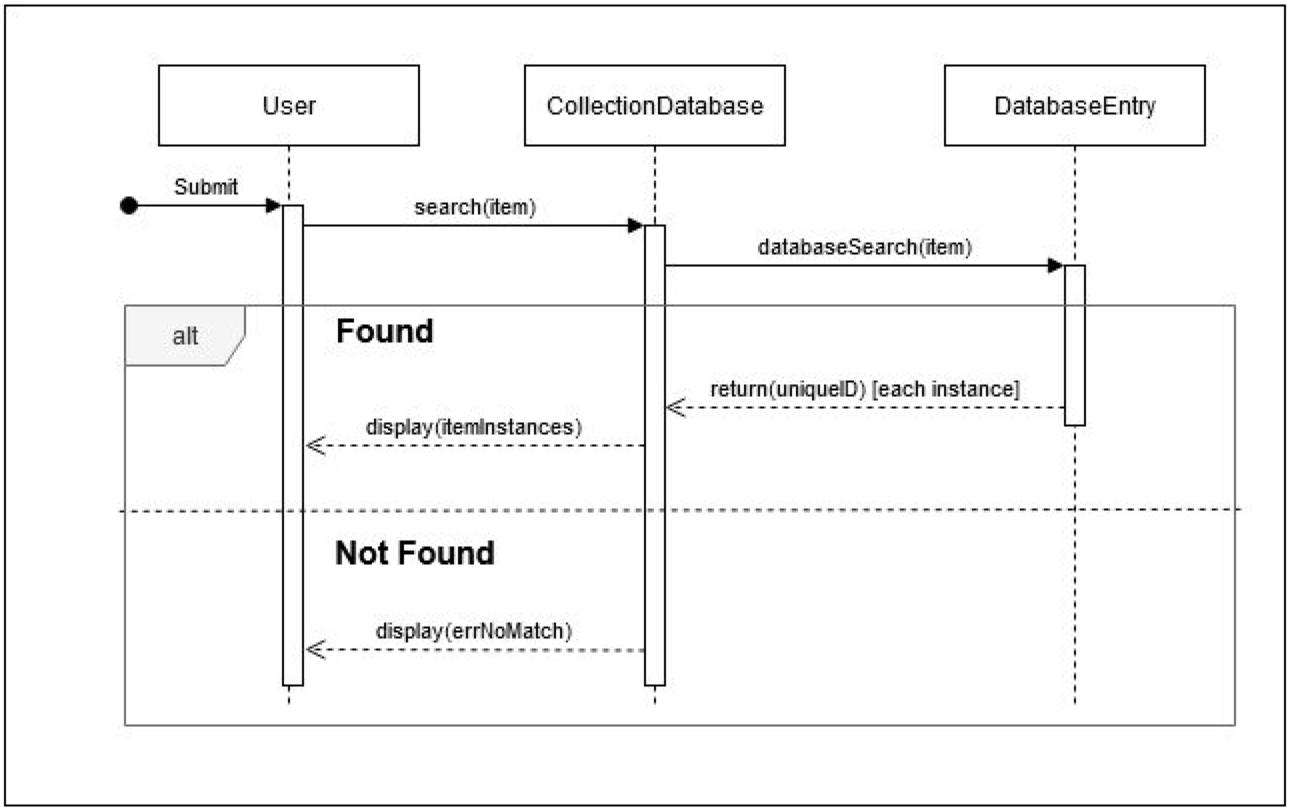
4.1 Lending Sequence Diagram



This diagram describes the sequence of a user requesting to borrow an Item in the Lender's Collection. The Lender may Approve or Deny the request to borrow the item, so there are two alternate paths described. If the Lender denies the request, then the requesting user is notified and no further action is required.

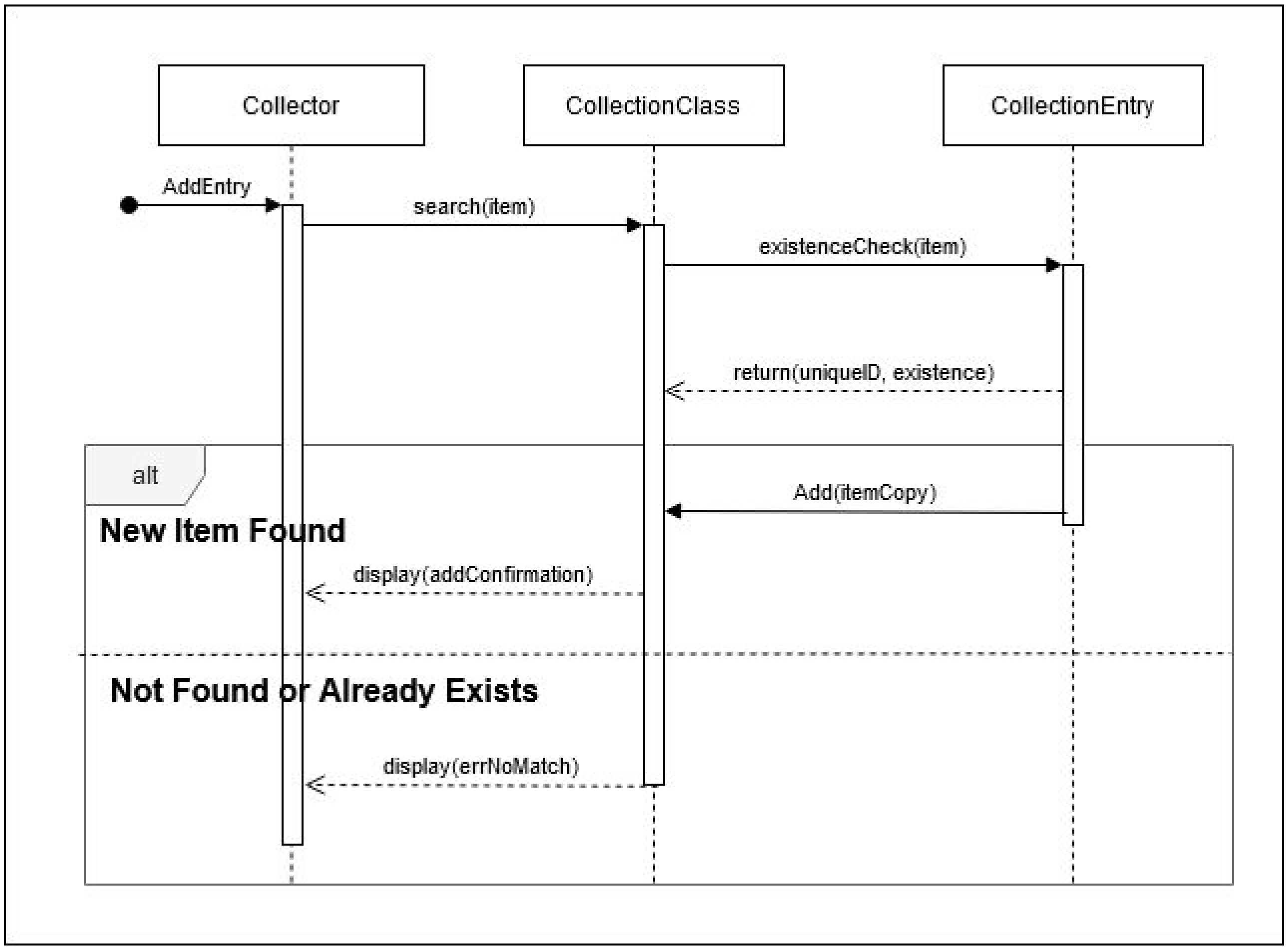
If the Lender approves the request, then the Item's borrowing status is updated to show the item is no longer available for other users to borrow and the Requesting User is notified of the approval. Once the Requesting User has returned the Item, then the Lending User is notified. Once the Lending User has confirmed receipt of the returned item then the Item's Borrowing Status is updated to indicate it is available to be borrowed again.

4.2 Search Sequence Diagram



This diagram describes the sequence of a user searching for an item in the catalog. There are two alternate paths that will be followed if the search has results to return to the user or if no results are found. The user is notified of the search completion whether there are results found or not.

4.3 Add Item to Collection Sequence Diagram



This diagram describes the sequence of a User with a Collection (Collector) adding an item to his or her collection. The system will verify that the item exists and is not already contained in the user's collection before adding an instance of the item to the collection. If the item does not exist or is already in the user's collection, then an error message will be displayed and no changes will be made to the user's collection.

Appendix A - Group Log

Meeting on October 15, 2020
Attendees: 5:00 PM
Start Time: John, Logan, Brandon, Daniel

- We decided to create a shared Google Drive folder to allow editing of the SRS document
- We will create a GitHub repository for version control
- We discussed possible roles for the team and experience levels with JavaScript and web-based applications
- We discussed a need to investigate existing JS libraries that could be helpful with our product
- We discussed meeting time preference and may switch our primary meeting to Saturday instead of Thursday
- We discussed ways to divide the SRS work by section – place your name by the section you want to claim

1.2 Purpose: In this time of frequently changing schedules, we want to provide a time management tool that helps people organize their day. Our application will provide users with a daily schedule that helps them track events in half-hour increments.

1.3 Scope: To help people with time management and scheduling
Daily calendar allows user to input multiple events for a given day
Each event can span from 1 to 48 half-hour time segments
Each event can be marked Complete by the user
TBD: Will this only cover a single day or will it allow 'zooming out' to multiple days/weeks?
TBD: Will events be allowed to repeat in the future – weekly/monthly?
TBD: Organize events by category, such as Work, School, etc.
Then allow users to filter tasks by category

1.3 Intended audience includes the Professor, the Users, and the Development Team

1.4 We will add to this later, if needed

1.5 Font and Formatting TBD

1.6 Any external libraries or standards that we plan to use?

2.1 This is a new product intended for individual users to manage their time. We will provide a diagram of the overall system design, but do not expect to use multiple servers or user entry points.

2.3 Users that access the system will have a single level of access
Users will create an account with a login or cookie – TBD – OAUTH2?
Administrator to manage user accounts and reset passwords

2.4 Application will be given a DNS name and made available through modern web browsers.

10/31 Meeting Minutes
Meeting on October 31, 2020 at 5:00 PM
Attendees: Logan, John
Start Time: 12:00 PM

Topics:

- We discussed the overall project direction and whether to change course since multiple other teams are also building calendar applications that are similar to our idea
- We considered alternative products including a retro video game lending service with search features
- Logan agreed to inquire about the ability to change course with our product idea
- John agreed to email the group with more detail about an alternate application

Meeting on November 5, 2020 at 5:00 PM
Attendees: Logan, John, Brandon
Start Time: 5:00 PM

Topics:

- Finished up all relevant areas of SRS
- Discussed next meeting

Group Meeting November 7, 2020

Attendees: Brandon, John

Topics Discussed:

- Milestone 2
  - Discussed tools to use for creating diagrams
  - Need to investigate Software Design Document structure/contents, if needed for M2
  - Need to create GitHub repository
- Activity/Sequence Diagrams
  - Created a shared Google Doc to track which use cases need to be diagramed
    - Definitely will need one for Borrowing process with 3 actors: Borrower, Lender, and Database
    - Another for Collection Maintenance with 2 actors: Collector and Database
- Class Diagrams
  - May need one for each use case?
  - Created a first draft of class Diagrams including class structure, methods, and fields
  - Need to show how classes interact
- Discussed gaps to be discussed later:

  - Security level – should we encrypt addresses in the system, or would it be easier not to store them long-term so no encryption is needed?
  - How best to organize borrowing requests? Tie them to borrow, lender, or both?
  - Additional features like borrower rating and messaging between borrower and lender

Group Meeting November 14, 2020

Attendees: Brandon, Logan

Topics Discussed:
- Assignment of tasks to do for Milestone 2.
- Logan will handle the Readme, and Github repository. As well as this weeks' meeting minutes.
- Brandon and John will work on the Software Design Documents, and the charts that will be used in it.

Group Meeting November 19, 2020

Attendees: Brandon, Daniel, John, Logan

Topics Discussed:
- Walked through the SDD diagrams and cleaned up the document for submission.
- Discussed approach for Milestone 3 including options for programming language, hosting services, and online interface.
- Discussed any changes that may be needed to the database design before implementation to avoid potential problems with late changes to the data structure.
- Discussed options for how to implement search and filtering.