WASHINGTON STATE UNIVERSITY
VANCOUVER

Software Requirements Specification

for

Retro Game Exchange Database

Version <1.0>

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**

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
Draft Type and Number	Full Name	Information about the revision. This table does not need to be filled in whenever a document is touched, only when the version is being upgraded.	00/00/00
0.1	John McEchron	Draft SRS outline with high-level details for a different product than originally described	10/31/2020
0.2	John McEchron, Logan Tan, Brandon Huang	Diagrams added and requirements finished.	11/5/2020
0.3	Daniel Lee	Added detail Operating Environment and Specific Requirements	11/6/2020
0.4	John McEchron	Updated document to reflect decisions made during early implementation and to append meeting notes	12/11/2020
1.0	John, Logan, Brandon, and David	Updated document to reflect implementation details and decision on user authentication	12/15/2020

1 Introduction

This project describes an online system to support the sharing and exchange of Retro Video Games. This system is intended for users to share their collections, search for games by different criteria, and exchange games, systems, and accessories with other users. Since this system is intended for Retro Video Games it does not cover the current or future generations of console games and also does not include online, digital, or mobile games that lack physical media to share.

1.1 Document Purpose

This document is intended for system developers, testers, and maintainers, and users or instructors interested in the design details. This document is not intended to provide user training on the system itself; necessary training manuals or user guides will be provided online with the application.

This document describes the original design of the system including how users interact with the data, how users create accounts and define their collections, and how users can share games with other users. This document also describes the overall design of the system technology including the online hosting infrastructure, the data definitions and relationships, the core system software classes, the user interface design, and the primary features of the software interface.

While this document version 1.0 will describe the design of the system as it is initially released it may be updated in the future to include any design changes or clarifications that are necessary. The document may also be updated if the system is changed after deployment to describe any new features or enhancements included in each release.

1.2 Product Scope

The purpose of this system is to provide an online portal for individuals to showcase their collections of Retro Video Games and Systems, and to allow others to view, search, and exchange those games. The system will allow users to input details about their collections to identify which games and consoles they own. The system will allow users to search for games based on different criteria like Release Console, Release Year, and Genre to find Retro Video Games that they may be interested in playing. The system will allow users to exchange games that other users make available to share and track which games have been borrowed. The goal of the system is to provide a hub of interaction for Retro Gaming enthusiasts to show off their own collections, learn more about the games, and share them with other enthusiasts to attract new fans of these classics.

The scope of this system will be limited to Video Games that were released in the United States for 20th Century platforms including the Atari 2600, ColecoVision, Commodore 64, Mattel Intellivision, Nintendo NES, Sega Genesis, and the Sony Playstation. This system will exclude newer generations of games for consoles released in the 21st Century like the Nintendo Wii, Xbox, PS3, iOS, and other Digital Platforms. This system will also exclude support for sharing any re-releases of Retro Games for newer platforms, limited-edition versions of games, games released in markets outside the US, and games released in non-English languages.

This system will exclude digital versions of Retro Video Games like MAME files and ROM images that recreate the original game Cartridge or Disc in a digital format. While these digital copies may be beneficial for archiving physical media that has decayed over time, these files may not be legal to host on US servers so our system will not allow users to store or link to those files.

1.3 Intended Audience and Document Overview

This document is organized into 5 major sections, each of which are split into subsections. The major sections in order are the introduction, overall description, specific requirements, other non-functional requirements, and other requirements. The document in general is best read in order. Below are some possible readers and most important sections for the type:

- Developer/Maintainer: For those maintaining or developing the system, start with section 2, with important sections being 2.2, 2.3, and 2.7. Continue on through all the requirement sections after. Section 3.2 is particularly important as well.
- Professor: For those who are using the document to learn about the project, the document should be read in the order presented. Section 3.2 in particular describes in depth all the functionality of the final product. Section 3.3 describes the types of users.
- User: For those who just intend to use the system, the whole document does not need to be read. section 1.4 will help establish acronyms. Section 2 will help give context to what the product does. Other sections are unnecessary.

1.4 Definitions, Acronyms and Abbreviations

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

1.5 Document Conventions

Text in 11 point Arial font comprises the bulk of the text within the document. If italicized, the text is a comment on the section. If highlighted yellow it is subject to change in a later version. Headings are bolded and increased to 14 point Arial font.

1.6 References and Acknowledgments

Information about the Video Game Titles, Genres, and Release Dates will be collected from the following sources:

[1] Wikipedia, *List of Intellivision Games,* 18 August 2020. Accessed on 05 November, 2020. [Online] Available: *https://en.wikipedia.org/wiki/List_of_Intellivision_games*

[2] Wikipedia, *List of Atari 2600 Games,* 29 October 2020. Accessed on 05 November, 2020. [Online] Available: *https://en.wikipedia.org/wiki/List_of_Atari_2600_games*

[3] Wikipedia, *List of ColecoVision Games,* 31 August 2020. Accessed on 05 November, 2020. [Online] Available: *https://en.wikipedia.org/wiki/List_of_ColecoVision_games*

[4] Wikipedia, *List of Commodore 64 Games,* 04 August 2020. Accessed on 05 November, 2020. [Online] Available: *https://en.wikipedia.org/wiki/List_of_Commodore_64_games*

[5] Wikipedia, *List of Nintendo Entertainment System Games,* 02 November 2020. Accessed on 05 November, 2020. [Online]
Available: *https://en.wikipedia.org/wiki/List_of_Nintendo_Entertainment_System_games*

[6] Wikipedia, *List of Playstation Games (A - L),* 01 November 2020. Accessed on 05 November, 2020. [Online] Available: *https://en.wikipedia.org/wiki/List_of_PlayStation_games_(A%E2%80%93L)*

[7] Wikipedia, *List of Playstation Games (M - Z),* 01 November 2020 Accessed on 05 November, 2020. [Online]
Available: *https://en.wikipedia.org/wiki/List_of_PlayStation_games_(M%E2%80%93Z)*

[8] Wikipedia, *List of Sega Genesis Games,* 05 November 2020. Accessed on 05 November, 2020. [Online] Available: *https://en.wikipedia.org/wiki/List_of_Sega_Genesis_games*

[9] Wikipedia, *List of Sega Master System Games,* 05 November 2020. Accessed on 05 November, 2020. [Online] Available: *https://en.wikipedia.org/wiki/List_of_Master_System_games*

2 Overall Description

2.1 Product Perspective

The Retro Video Game Exchange system described in this document is a new stand-alone system that is not intended to interface with any existing external system. The system will be hosted on a single environment that provides public-facing functionality to users through a web browser. This system is not based on any existing system, although it is possible that similar platforms exist given the popularity of Retro Gaming across many different countries. This system will be free to use and will not require payment to create an account.

2.2 Product Functionality

Primary product functions will include:

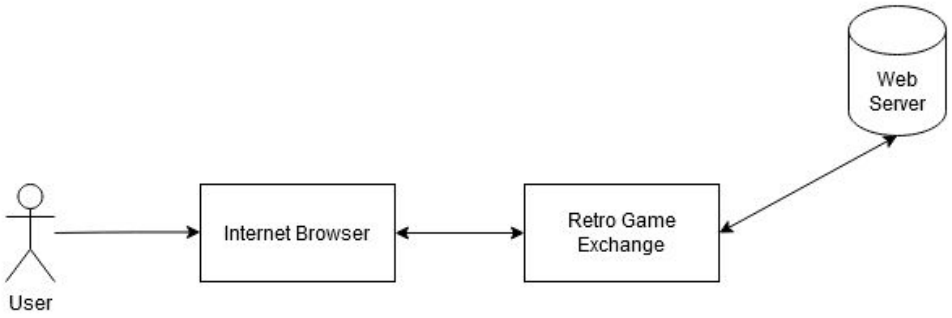
- User Account Creation
- User Account Management
- Add Collection to User Account
- Add Game to User Collection
- Remove Game from User Collection
- Search for Game
- Request to Borrow Game
- Decline Borrow Request
- Approve Borrow Request
- Return Borrowed Item

2.3 Users and Characteristics

1. Game Collector – This type of user will create an account and describe their game collection. They may also request to borrow games or accessories from other users. We expect this type of user to be our primary audience and constitute the majority of user accounts.
2. Game Sampler – This type of user will create an account and request to borrow games without maintaining a game collection of their own.
3. Game Searcher – This type of user will search the Retro Video Game catalog without creating an account of their own. Without an account this type of user would not be able to borrow a game or describe their own collection.

2.4 Operating Environment

The Operating Environment will include a Javascript Web Client and a web API hosted on a Cloud Compute Platform such as AWS, GCP, or Microsoft Azure. Users will interact with the web service using a Modern Web Browser, including Google Chrome version 86 or higher, and Mozilla Firefox version 81 or higher. No other proprietary or paid software will be required to use this system. Compatibility will be tested with Windows 10 and Android OS 10 platforms – compatibility with other operating systems and browsers is not guaranteed.



2.5 Design and Implementation Constraints

The implementation of the login system will be constrained by security concerns. Since the development team does not have experience in secure logins or data encryption, we will limit the amount of sensitive information the system stores.

2.6 User Documentation

A User Guide will be developed and made available through the platform website, if requested.

Context-sensitive help screens will be available on the User Interface to describe any potentially confusing actions, activities, or descriptions for users.

2.7 Assumptions and Dependencies

Assumptions:

- The user will have access to either Firefox or Google chrome.
- Given the age of the console systems, we will assume that no new games will be released over the life of our product and no interface is necessary to add new Retro Video Games published after November 5, 2020.
- This product is only used in the North America region and will not catalog games and console systems released for international markets or non-English languages.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

There will be one primary User Interface provided online and accessible through current Internet Browser Software. Through the User Interface, system users will have a number of options available, and some will be duplicated on multiple pages. The first interface screen will be called the Home Page and will have the following functions available:

- Create an Account
- Sign into an existing Account
- View User Collection Page
- Browse Catalog
- Search Catalog

When a user logs in and opens their User Collection Page, they will have the following functions available:

- View Consoles
- View Games
- Add Game to Collection*
- Remove Game from Collection*

* Each function with this icon will have common attributes:

    - The action has a Cancel button available to return to the previous page without making any changes.
    - When a user clicks Submit they will see a dialog box to confirm their submission. Clicking Cancel on this dialog will return to the previous page without making any changes.

When a user opens the Browse Catalog page, they will be presented with a screen that shows options to view the Catalog by Console with the name and image of each supported console shown.

When a user opens the Search Catalog page, they will be presented with a text field that allows wild-card search characters. When they perform a search, the results can be sorted alphabetically or chronologically by release date.

All User Interface screens will share a common Cascading Style Sheet that provides consistent text formatting throughout the system.

All User Interface screens will provide Links in the left portion of the screen that will indicate what page they are currently viewing and allow the user to return to a previous page in the interface. A link to return to the Home page will be provided on all screens.

3.1.2 Hardware Interfaces

No custom Hardware Interface will be provided by this system. The system will depend on the Internet Browser Software to interface with the user's Keyboard, Mouse, and Screen to collect input and display content.

3.1.3 Software Interfaces

The service will be running in a docker container which creates an emulated linux server environment. Our application code will be ‘baked’ into an image that will run on our Cloud Provider’s servers. Data will likely be stored in a database hosted by our Cloud Provider. Users will interact with our services through their web browser.

3.1.4 Communications Interfaces

The web browser will communicate to the server through https and ssl encryption. Our data will be securely stored on our Cloud Providers servers using their own hardware encryptions and secure account services.

3.2 Functional Requirements

3.2.1 User Account Creation
The system will provide a way for users to create an account with persistent information about their Retro Video Game Collection and any active Lended or Borrowed items. User Email Address and Password will be collected upon account creation and stored in the system.

3.2.2 Catalog
The system will maintain a Catalog of Retro Video Games stored in a JSON data structure. This information will be hierarchical so that each Game and Accessory is linked to a parent Console. The information stored about the Catalog Items will include:

- Title
- Release Date (only Year is required)
- Console
- Game Genre (not required
- URL to Wikipedia article (not required)
- Developer (not required)
- Publisher (not required)
- Unique Index ID (not displayed)

3.2.3 Catalog Search
The Retro Video Game Catalog will allow users to search for specific strings within each Catalog attribute and return the results to the User Interface. Search results will be sorted alphabetically by Game Title.

3.2.4 Catalog Detail Display
When a user has searched the Retro Video Game Catalog and clicks on one of the results, they will be directed to a page with more details about the selected item. This page will display the Title, Release Year, and Genre of the item they have selected.

3.2.5 User Collection Creation
When a user has accessed the system with an established account they will have the option to create a Collection. This will establish a new Collection for that user which can be managed through the following functions.

3.2.6 Add Catalog Item to Collection
When a User selects the Add to Collection option, the system will validate that they have a Collection and display an exception message if they do not. If the user has a collection, they will have the ability to Search the Catalog. This Search method will differ from 3.2.3 by providing an additional option to Add Item to Collection for each Console, Game, and Accessory listed in the Search Results. If the user clicks on the Add button and confirms their choice then that Catalog Item will be added to the user's Collection.

3.2.7 View Collection
An authorized User will have the ability to view a list of User Collections sorted by User, and selecting a Collection will navigate them to the View Collection screen. This screen will display the Consoles, Games, and Accessories that exist within the selected User's Collection. This screen will also indicate whether an item in the User's collection is on loan to another user.

3.2.8 Remove Catalog Item from Collection
If an authorized User opens the View Collection screen for their own collection they will see an additional Remove option for each item. If the user clicks on the Remove button and confirms, then that item will be removed from their Collection and no longer appear on the View Collection screen for that User. Users shall only be able to remove items from their own Collection - Users cannot remove items from a different user's Collection.

3.2.9 Request to Borrow Item
When an Authorized User views a Collection they will see an additional option to Request to Borrow each available Game or Accessory in the user's collection. If the Authorized User clicks the Request to Borrow button they will be asked to confirm their request. Users that do not have an Account will not see the option to Request to Borrow an item. Users will not see the Request to Borrow button if they are viewing their own Collection. If an item in a user's Collection has already been loaned to another user then the Request to Borrow button will not be available for that item. Users that submit a Request to Borrow will be asked to provide a Shipping Address

3.2.10 Respond to Request

When a User with a Collection has had another user Request to Borrow an item in their collection, the system will present a new button that allows the user to Respond to the Request. When the user clicks the Respond to Request button they will be taken to a new page that displays each Request to Borrow on a table row that includes the following information:

- Game Title and Console
- User that submitted the request
- Date that the request was submitted
- Button to Approve Request
- Button to Decline Request

When the user clicks either the Approve Request or Decline Request button the system will process the request according to that choice and remove the request from the Respond to Request table.

3.2.11 Approve Request

When a Lender user Approves a Request, that Game or Accessory will be marked as On Loan in the Lender's Collection. The date that the request was approved will be recorded and the requested Game or Accessory will be added to the requesting User's list of Borrowed Items. When a request is approved the Lender will be asked to confirm their Return Shipping Address which will be shared with the user.

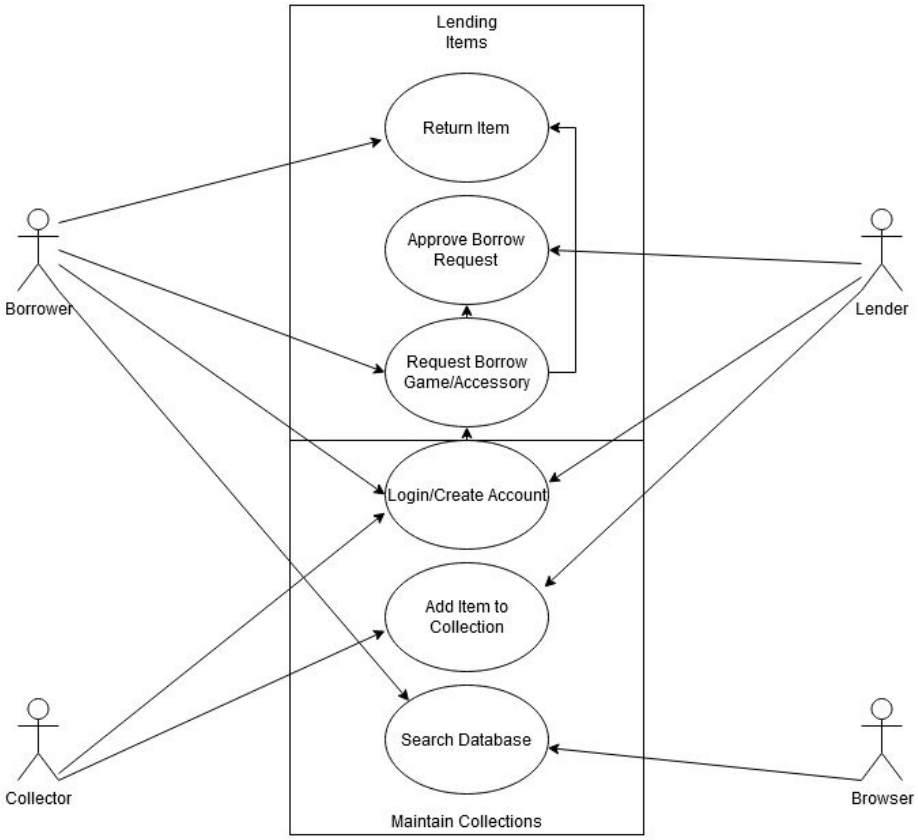
3.2.12 Borrowed Items

When a User's Request to Borrow has been approved both the Borrowing User and Lending User will see a new option to view the Borrowed Items page. The Borrowed Items page will list the Item Title, Borrower, Lender, and the Shipping Information for both Users. This is the only page where the User Shipping Information will be presented to minimize any security concerns. Borrowers will see an additional option on this page to Return an Item. When a Borrower has Returned an Item the Lender will see an option to Confirm Return for that item.

3.2.13 Return Item

When a Borrower has clicked on the Return an Item button they will be asked to confirm that they have shipped the Borrowed Item back to the Lender. When the Borrower has marked the item as Returned, the Borrowed Items page will reflect that status change and present the Lender with a button to Confirm Return of the item. Once the Lender has clicked the Confirm Return button then the Collection Item will be removed from the Borrowed Items page and again be eligible for a Request to Borrow from the Collection page.

3.3 Behavior Requirements



3.3.1 Use Case View

Actors:

- Borrower: A user which intends to request items from the database to be borrowed from collectors/lenders
- Lender: A user who intends to lend out his collection to borrowers.
- Collector: A user who intends to display their collection on the database
- Browser: A user who has no intention to create an account or borrow/lend, and only wishes to peruse the collections.

Use Cases:

- Return Item: A user wishes to return a borrowed item from a collection.
- Request Borrow Game/Accessory: A user can request an item from a lender.
- Approve Borrow Request: A user can approve borrow requests from their collection.
- Login/Create Account: A user can create an account on the application and log into existing accounts.
- Add Item to Collection: A user can add a retro game or accessory to their collection and add it to the database.
- Search Database: A user can search all collections, without logging in or creating an account.

4 Other Non-functional Requirements

4.1 Performance Requirements

The system will need to in real type, flag and display whether or not any given game is available to borrow. This means that there should be no more than 2 seconds of delay between different users flagging the game as unavailable, or as borrowed to avoid collisions with multiple inputs.

The searching and filtering functionality should be able to sort out the games catalog efficiently. That means within 5 seconds there should be a full list returned, assuming the database is storing a reasonable amount of entries.

4.2 Safety and Security Requirements

This product should have very minimal safety requirements, as we are only logging data that is voluntarily given. We should have some security to not allow users to access others accounts, including any admin accounts. The amount of personal Data stored on the Web server will be minimal. It would be best to use Outsourced Security measures to verify users, or allow for personal logins, using emails as the user ID. This UID will be public information, as this is a games exchange site, it is necessary to verify that there is another person on the other end before an account is generated, and for other users to have the ability to contact other users.

4.3 Software Quality Attributes

The Database Should be expandable, and cover different User Libraries, as well as accept new games into the collection, though approval, of either the moderators, or will be done though community polling. The search and filter functionality of the database will be maintained by regular indexing, to allow for faster sorting methods, as this is a “retro” games database, the release year and consoles are the most important traits to index and sort by. knowing this the internal list of games should be sorted by ascending release year. For faster searching for consoles, we will implement an indexed tree to sort by console alphabetically. Indexing will be set to recur every night around midnight, to be less of a burden.

5 Other Requirements

Appendix A – Data Dictionary

Object	Element	Constraint	Description
User	Email_Address	Primary Key	User Email Address used for authentication and communication
User	has_Collection	Not null	Boolean value to indicate existence of collection
User	Ship_Address		Default Shipping Address to be used when Lending or Borrowing
Catalog	ID	Primary Key	Unique ID number assigned to each item in the catalog
Catalog	Type	Not null	The type of item: [Console, Game, Accessory]
Catalog	Title	Not null	The Title or Name of the Console, Game, or Accessory
Catalog	Rel_Year	Not null	The Year that the Catalog Item was released in North America
Catalog	Genre		The Genre assigned to games, if applicable
Collection	Email_Address	Primary Key	The Email Address of the User that created this Collection
Collection	Item_ID	FK: Catalog.ID	The Unique Catalog ID of the item in the user's collection
Collection	canBorrow	Not null	Boolean value to indicate whether the user is willing to lend the item
Collection	isBorrowed	Not null	Boolean value to indicate whether the item has been lended to another user
Borrowing	Borrow_ID	Primary Key	Unique ID number assigned to each borrowed item
Borrowing	Item_ID	FK: Catalog.ID	The Unique Catalog ID of the item in the user's collection
Borrowing	Active	Not null	Boolean value to indicate whether the item is currently on loan to another user (false when returned)
Borrowing	Lender	Not null	Email Address of the Lender
Borrowing	Borrower	Not null	Email Address of the Borrower
Borrowing	Lender_Address	Not null	Shipping Address of the Lender
Borrowing	Borrow_Address	Not null	Shipping Address of the Borrower
Borrowing	Date_Requested	Not null	Date and Time the Borrowing Request was submitted

Borrowing	Date_Approved		Date and Time the Borrowing Request was Approved by the Lender
Borrowing	Date_Declined		Date and Time the Borrowing Request was Declined by the Lender
Borrowing	Date_Returned		Date and Time the Borrowing Request was marked as Returned by the Borrower
Borrowing	Date_Return_Confirmed		Date and TIme the Returned Item was accepted by the Lender

Appendix B - Group Log

Meeting on October 15, 2020
Attendees: John, Logan, Brandon, Daniel
Topics:

- 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

Meeting on November 14, 2020
Attendees: Brandon, Logan
Topics:

- 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.
- Adjusting meeting schedule to account for Thanksgiving break

Meeting on November 21, 2020
Attendees: Brandon, Logan, John, Daniel
Topics:

- Discussed approach to storing data in JSON files
    - Catalog is static and works well as a large JSON file
    - Collection and User data could be stored differently since they will be more dynamic and need to be modified by the JS classes
- Reviewed feedback for Milestone 2 diagrams and discussed impact on design

Meeting on December 5, 2020
Attendees: Brandon, Logan, John, Daniel
Topics:

- Reviewed UI mockups and discussed possible frameworks that could be used for the front-end UI
    - VUE JS was selected as a good approach for the UI
- Discussed sample User Collection data structure
    - Agreed to generate sample file based on discussed changes
- Given the approaching deadline for Milestone 3 we agreed to hold additional meetings during implementation phase
- Confirmed approach to use JavaScript to implement classes

Meeting on December 9, 2020
Attendees: Brandon, Logan, John, Daniel
Topics:

- Reviewed code, web interface, and data structures that have been created so far
- Discussed options for API implementation to link the data to the web interface
    - Selected Express and Require frameworks to use to help with API implementation
- Daniel lead a demonstration of using Node.js to set up and host APIs on local machines to help with unit testing while we develop the application individually
- Discussed our selected time to provide the final product demo on Wednesday, December 16 at 12:20 PM

Meeting on December 12, 2020
Attendees: Brandon, Logan, John, Daniel
Topics:

- Discussed approach to implementing requests and the need for a new JSON file associated with this data
    - Each request should be given a unique Index ID number in addition to the data collected
- Reviewed the error handling that has been added to the Remove Game from Collection function
- Discussed how data is bing passed to the Front-end and remaining paths to implement
- Discussed an additional meeting on December 15 to finalize testing and prepare for the demo

Meeting on December 15, 2020
Attendees: Brandon, Logan, John, Daniel
Topics:

- Discussed changes left to make to Final Report document
- Discussed approach to the demo
- Completed testing of recent bug fixes