PlayVault Curator (PVC)

Team Rocket #12

Jack Evans, Paul Sihavong, Anthony Douglas, Robert Martinez

Software Requirements Specification Document

Version: (1) **Date: (02/27/2025)**

Table of Contents

1 Purpose

The purpose of this project is to create a plug-in for a digital game library management system that assists users in optimizing the use of their console/computer's data storage, tracking gaming history, and exploring new titles to the user.

Gamers have trouble keeping track of their game libraries and finding space for new titles they want to download. This plug-in will streamline user experience with their game libraries, detecting storage waste and alerting the user for best storage management practices based on their individual storage consumption, game history and interests. While it does not automatically clean up waste, it does make it easier for gamers to offload or clear up storage for incoming games.

This application aims to enhance a user's gaming experience by maintaining games & add-ons (additional purchases for said game), in the user's library through time tracking and in-game purchases. This application will share reports of information it gathers to solely the account holder, maintaining the privacy and security of the user's information. This application will provide alerts on necessary updates to download games, along with details of how much storage the update will take up. If storage defined by **"free up space"** parameters (###.## bytes used out of ###.## bytes) exceeds a user defined threshold, it will alert user for recommended action. If the game has not been used in set amount of time defined by **"time last played "** parameters (set by user: days/weeks/years), the management system will prompt the account holder, giving a status of the game (storage, last used, hours spent, add on-game purchases) and recommend appropriate action to the account holder, but not automatically clean storage up itself, so as to give the user complete control of their library. This application will take information gathered from the user's game history and compare it to games (new and old) from a database of games. It will then curate a list of games for the user based on developer, genre, data storage and compatibility, so that it fits with the user's taste and available storage. This will make it easier for the user to explore new titles and get through the backlog of games in their libraries. This also helps the application recommend games like what the user plays rather than just going off what has been purchased.

2 Scope

The proposed system, Play Vault Curator, will be a web-based and console-compatible application that enables users to organize and optimize their game libraries. It will offer features such as game time tracking, game recommendation algorithms, data storage optimization suggestions and integration with digital game stores databases. Key functionalities include:

- User authentication and permissions access
- Timing game play and recording dates of last accessed for games

• Data storage analysis and optimization suggestions

• Access to game data base through IGDB (internet game database)

• Access of user metrics through Steam API

• Track purchases associated with account

• Notification of new releases (including niche titles), game updates, and data consumption

• Organize game libraries (by title/developer/genre/last played)

• Extended feature (if time allows): Organize wish list and inform user on deals, new additions and current market price for listed games in wish list

• Extended feature (if time allows): Search the web for external hard drives compatible with the user's console or computer when prompted for "options for external space".

Application will need access to systems info: make, model and year.

This application will allow the user to easily find solutions for wasted space, keep track of subscriptions (if applicable) and will make it convenient to explore game titles new to the user.

3 User characteristics

3.1 Key users

- Hardcore/Active Gamers
  - Use the tool to optimize storage space, track gaming habits, manage subscriptions and discover niche/underplayed titles.
  - Subject Matter Experience: Moderate to Journeyman level of gaming experience to take full advantage of the recommendation algorithm. A large game library, time spent in the library, and user's friend's libraries improves algorithm tailoring. A Steam account is not required but will also assist in tailoring user experience.
  - Technological Experience: Novice to Moderate experience in computer usage. The user has to be able to act on notifications and navigate their own settings and libraries.
  - Attributes:
    - Age: Average age range targeted from 15 to 35, adolescents and young adults experienced with computer usage from a young age
    - Linguistic Skills: Assumes user is proficient in the English language
    - Platform: Assumes users are storing their video game library digitally and locally, i.e. stored on the local drive and not through game discs or cloud gaming

3.2 Secondary users

- Casual Gamers

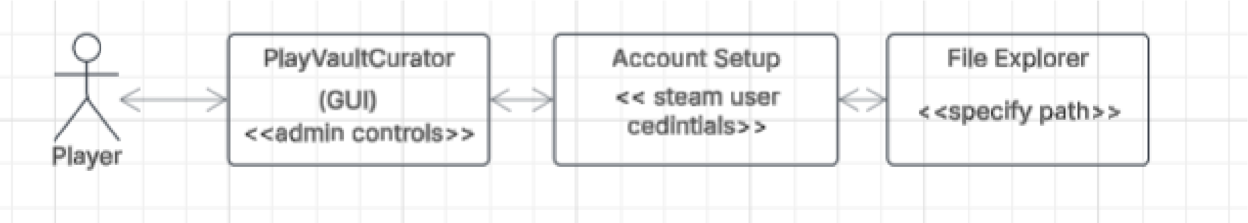
- Use: For organizing their game libraries and receiving recommendations for new games and game updates.
- Subject Matter Experience: Novice to Moderate level of gaming experience. Use the organizational tool to consider the user's preferences and to find new games.
- Technological Experience: Novice to Moderate experience in computer usage. The user has to be able to act on notifications and navigate their own settings and libraries.
- Attributes:
  - Age: Average age range targeted from 15 to 35, adolescents and young adults experienced with computer usage from a young age
  - Linguistic Skills: Assumes user is proficient in the English language
  - Platform: Assumes users are storing their video game library digitally and locally, i.e. stored on the local drive and not through game discs or cloud gaming

4 Product perspective

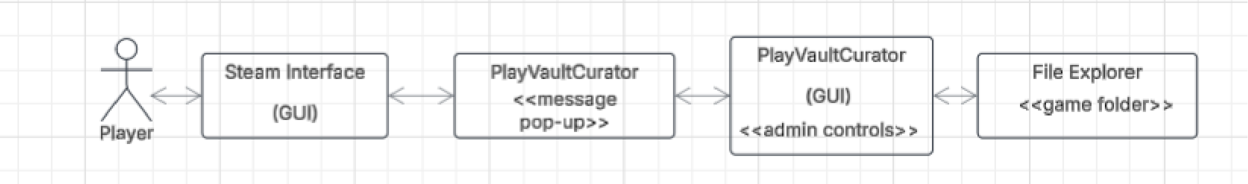
4.1 System Context

Define the system's relationship to users or other related systems. If the system is an element of a larger system, then identify the interfaces between the system covered by this SRS and the larger system. A block diagram showing the major elements of the larger system, interconnections, and external interfaces can be helpful. Below are two example block diagrams (with the system to be developed highlighted):

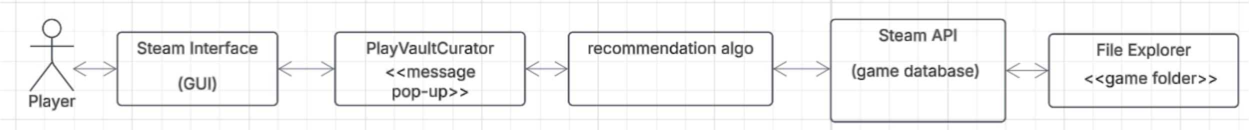
(PlayVault Curator plugin-in is installed and user is linking to Steam account with specified library path)



(PlayVault Curator plugin-in is already installed and player is installing game through Steam)



(PlayVault Curator plugin-in is already installed and player receives pop-up recommendation)



4.2 User interfaces

Specify the required characteristics of each interface between the software product and its users. Keep in mind that we are specifying requirements, so we here only care about the requirements on interfaces (this is NOT the place to put user interface designs by the software developers). For each requirement on an interface, it may include:

- It is required for the system to provide pop-up message in bottom left of screen when user attempts to download game and/or storage size needs attention.
- It is required to provide buttons on pop-up, one is a link to redirect user to system admin controls, the other to dismiss the message.
- It is required for the system to prompt users for Steam credentials to access game library and associated information.
- It is required for system to prompt user for file path, so PVC can access, and mange said pathway
- It is required that the system allows users to remove and organize games from their local library.
- It is required that the system allows users to filter and sort recommendations (within admin controls), based on various criteria (e.g., genre, developer, storage requirements).

4.3 Software interfaces

- The system must use the JRE

4.4 Deployment requirements

- N/A

5 Assumptions and Dependencies

List each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but any changes to these factors can affect the requirements in the SRS.

We are dependent on the Steam API for a large amount of our planned features, but will work with dummy data to ensure there is a functional program that may need some adaptation to use the Steam API but won’t break without it.

We are assuming that the user is on a Windows device but believe that being on a Mac shouldn't change the running of the program but would have to be tested later.

We assume the user is connected to the internet for the most functionality. There is offline capability, but the program will be limited.

6 Specific requirements

This is a specification, a designer should be able to read this spec and build the system without bothering the customer again.

- *Specify all of the software requirements to a level of detail sufficient to enable designers to design a software system to satisfy those requirements.*
- *Specify all of the software requirements to a level of detail sufficient to enable testers to test that the software system satisfies those requirements.*
- *Each requirement should be uniquely identifiable for traceability. Usually, they are numbered R.1, R.1.1, R.1.2.1 etc. so that each can be cross-referenced in other documents.*
- *Each requirement should also be testable. Avoid imprecise statements like, "The system shall be easy to use." What is easy to use?*

Use proper terminology:

- *A required, must have feature: The system shall…*
- *An optional, nice-to-have feature that may never make it to implementation: The system may…*

Avoid over-constraining your design. Do not require specific software packages, etc., unless the customer specifically requires them.

Avoid examples: Don't say things like, "The system shall accept configuration information such as name and address." The designer doesn't know if that is the only two data elements or if there are 200. List every piece of information that is required so the designers can build the right UI and data tables.

6.1 System Functional Requirements

- As a gamer, I want this program to monitor and manage my storage, so that I have freed up space to add new games to my library.
- As a developer, I want this program to allow the user to be able to change "time last played" and "free up space" parameters so that the user has a tailored experience to their changing preferences.
- As a gamer, I want this program to recommend games based off my game history, console/computer compatibility and my friends game libraries so that I can discover new games popular in the gaming community.
- As a developer, I want this program to tell the difference between game execution files and other local execution files so that the user only receives relevant storage management recommendations.
- As a gamer, I want to be able to access administrative controls offline, so that I can receive storage management assistance even when wifi connection fails
- As a gamer, I would like this program to provide solutions to app-related issues such as proper setup, compatibility, find a specific feature, etc..., so that I have an easier time navigating this tool.

6.2 Logical Database Requirements

Specify the logical requirements for any information that is to be placed into a database, including data entities and their relationships, and Integrity constraints.

If the customer provided you with data models, those can be presented here. ER diagrams (or static class diagrams) can be useful here to show entity relationships. Remember a diagram is worth a thousand words of confusing text.

Entities:

- User:
  - Steam ID:
  - System Type(mac/windows)
- Game:
  - Title
  - Size
  - Developer
  - publisher
  - genre
  - release date
  - ratings
- Developer/Publisher(optional)
  (If many games share the same developer or publisher to avoid redundancy

Relationships:

- One too many games per user
- Games can have one to many genres
- Publishers/Developers can have one to many games

Entity Integrity:

- **User Table / User Session Data**
- **User Identification:**
  - **Generic Session Identifier:**
    - For users who do not attach their Steam account, generate a temporary session-based identifier (e.g., SessionID) used only during the active session.
    - **Constraint:** This identifier is ephemeral and will not be stored permanently after the session ends.
  - **SteamID (Optional):**
    - If a user chooses to attach their Steam account, capture the SteamID.
    - **Constraint:** When provided, the SteamID must be unique. Only store the SteamID if the user opts in.
-

Game Table

- **Primary Key:**
  - **GameID:** A unique identifier for each game.
    - **Constraint:** Must be unique and not null.
- **Title:**
  - **Constraint:** Must not be null.
- **Size:**
  - **Constraint:** Must not be null.
- **Developer:**
  - Can either be a text field in the Game table or a foreign key referencing a separate Developer table (if you choose to normalize further).
  - **Constraint:** Must not be null if it's a direct attribute, or if using a foreign key, the referenced record must exist.
- **Publisher:**
  - Similar to Developer: can be a direct attribute or a foreign key referencing a separate Publisher table.
  - **Constraint:** Must not be null if it's a direct attribute, or if using a foreign key, the referenced record must exist.
- **Genre:**
  - Since games can have one to many genres, consider using a join table (e.g., GameGenres) where each record links a GameID with a GenreID from a Genre table.
  - **Constraint:** Each game must be linked to at least one genre (enforced via application logic or database constraints in the join table).
- **Release Date:**
  - **Constraint:** Should follow a standard date format; can be null if not applicable.
- **Ratings:**
  - **Constraint:** May have validation rules (e.g., a range between 0 and 10); can be null if ratings are not provided.

Developer/Publisher Table (Optional, for normalization)

- **Primary Key:**
  - **DeveloperID/PublisherID:** A unique identifier for each developer or publisher.
    - **Constraint:** Must be unique and not null.
- **Name:**
  - **Constraint:** Must not be null.

6.3 Software System Attributes

Specify non-functional requirements, that is, the required attributes of the software product.

For each requirements in this section, make sure it is testable

- As a gamer, I want all pop-up notifications to be 1in. X 1in, in the bottom right corner of my screen, so that it doesn't take up screen space.
- As a non-tech savvy gamer, I want to be able to effectively use this plug-in within 10 minutes of training, so that I can easily use this with my steam account.
- As a developer, I want the buttons in pop-up notification to hyperlink to the exact window in admin controls or pathway in File Explorer based off the content of the alert message to make it convenient to address for the user.
- As a gamer, I want pop-up alerts to only last 5 seconds before leaving my screen, so that my window is uncluttered.

6.3.1 Usability

Usability requirements for the software system include measurable effectiveness and satisfaction criteria in specific contexts of user interactions.

- The System shall accurately retrieve and display game information ensure that users can see relevant details (title, size) without error
- The System shall, for users who link their Steam accounts, correctly sync their game library and provide accurate storage insights.
- The System shall, for users who don't want to link their steam account, be able to access a preloaded set of games without functionality limitations.
- The System shall, provide a clean and organized UI
- Searching, filtering and sorting games should be fast and responsive
- The application should load and process user actions with 30seconds(for calculating the storage)
- The System shall have a dark theme so it doesn't hurt users eyes.
- Users should receive meaningful feedback (such as confirmations, loading indicators, or error messages) to keep them informed of actions taken.

6.3.2 Performance

Specify both the static and the dynamic numerical requirements placed on the software or on human interaction with the software as a whole.

- *Static numerical requirements (capacity) may include the following the number of simultaneous users to be supported; amount and type of information to be handled.*
- *Dynamic numerical requirements may include, for example, the numbers of transactions and tasks and the amount of data to be processed within certain time periods for both normal and peak workload conditions.*

The performance requirements should be stated in measurable terms. For example, 95 % of the transactions shall be processed in less than 1 second.

- Play Vault Curator will run natively on the user’s device as long as they have the provided files installed. Therefore, there is no capacity, and it runs as a single user interface.
- Play Vault Curator does not require concurrent access to a network.

- It can handle hundreds of installed games that will be sorted, ranked, and recommended.

- Each game includes:
  - A title: string
  - Size in GB: double
  - Price in dollars: double
  - Total playtime in hours: integer
  - Days since last played: integer
  - If the game is Multiplayer: boolean
  - Flag showing whether it is owned: boolean
  - Developer: string
  - Publisher: string
  - Genre: string
  - release date: string
  - Ratings: string
  - Developer/Publisher: string
  - Deletion Rank: double

- Play Vault Curator shall handle up to 500 games in 2 seconds whether it is to rank, read, or recommend them to the user.
- 90% of processes should take less than 2 seconds to complete. 4 seconds maximum.

6.3.3 Reliability/Dependability

The system will be designed to function without internet access or linking to a Steam account and the system will have limited functionality. The system will come packaged with a small offline database containing similar metrics to Steam’s game database, i.e. genre and size.

6.3.4 Security

- N/A

6.3.5 Maintainability

If the User wants to update their offline usability, then they would have to redownload a small file containing the most common games and their sizes.

The rest of the Functionality should be there on release so the only maintenance required would be to our updater if we have one and reliably accessing Steam or other game libraries.