Vapor, A PC Game Library Manager

Steven Gray and Andy Lum

Abstract

Vapor will be a minamalist game library manager to help gamers keep track of all of their games across various launchers. This proposal document is the bulk of what we've done so far.

1. Introduction

In the modern era of PC gaming, its too easy to lose track of games. Many of the biggest games are spread across multiple launchers, while a lot of smaller games find themselves disconnected from all of them. Even the launchers that allow for users to add games often require them to search for each game individually. This makes it easier than ever to simply forget you have games downloaded, even if you keep them organized.

That's where Vapor comes in. Vapor will allow for a user to keep up with every game they have on their computer with minimal upkeep on their end.

1.1. Background

Increasing popularity of PC gaming has lead to a unique situation where many of the biggest developers want to create their own storefronts and game libraries to maximize the amount of profit they make on a game sale. This becomes a hassle for gamers who simply want to play their games without having to sift through multiple different programs to find where their game is hiding. This issue is further exacerbated by storefronts without their own launchers, or programs managed by storefronts/developers to , leading to a collection of stray games.

1.2. Impacts

At the very least, our goal is to make game management easier and more flexible for the user. We want to create an environment where someone can avoid wasting time sifting through folders or launchers just to find what game they want to play, or avoid wasting money buying games they already own because they simply can't find what they're looking for.

1.3. Challenges

The immediate challenges this project brings up include determining what the name of a particular game is, finding the launcher that the game is associated with, and figuring out how to work with online databases.

2. Scope

The initial goal for this project is to create a functional interface that allows for a user to have all of their games accessible and playable through a single library manager. This would include:

- A main menu with all of their games available and playable.
- An options menu that would allow a user to add new games or treat certain folders as library folders (alongside an automatically created library folder handled by the program itself).
- The ability to remove a game or library folder from the library manager or delete them altogether
- The ability to refresh the library at any time, to find newly added games or remove recently removed ones.
- The ability to open the location where a game is stored

A few possible stretch goals could be:

- Icons that allow for a user to discern where the games are from, such as Steam, Battle.net, or a user defined Library folder.
- A store page that allows a user to search/browse popular storefronts like Steam, GOG, and Humble.
- Allow the user to create and manage groups of games

Use Case ID	Use Case Name	Primary Actor	Complexity	Priority
1	Add item to cart	Shopper	Med	1
2	Checkout	Shopper	Med	1

TABLE 1. SAMPLE USE CASE TABLE

2.1. Requirements

As part of fleshing out the scope of your requirements, you'll also need to keep in mind both your functional and non-functional requirements. These should be listed, and explained in detail as necessary. Use this area to explain how you gathered these requirements.

2.1.1. Functional.

- User needs to have a locally stored list of games on the computer - including their locations and any necessary assets such as cover art or icons
-

2.2. Use Cases

Use Case Number: 1
Use Case Name: Add item to cart
Description: A shopper on our site has identified an item they wish to buy. They will click on a "Add to Cart" button. This will kick off a process to add one instance of the item to their cart.

1) User navigates to page listing desired item
2) User left-clicks on "Add to Cart" button.
3) User cart is updated to reflect the new item, this also updates the current total.

Termination Outcome: The user now has a single instance of the item in their cart.

Alternative: Item already exists in the cart

1) User navigates to page listing desired item
2) User left-clicks on "Add to Cart" button.
3) User cart is updated to reflect the new item, showing that one more instance of the existing item has been added. This also updates the current total.

Termination Outcome: The user now has multiple instances of the item in their cart.

Use Case Number: 2
Use Case Name: Checkout
Description: A shopper on our site has finished shopping. They will click on a "Checkout" button. This will kick off a process to calculate cart total, any taxes, shipping rates, and collect payment from the shopper.

2.3. Interface Mockups