

Team A
Randy Dinh
Jacob Francis
Scott Lee
Eri Kudo

Should I Buy It?

Link: <http://uga.jfrancis.us/index.html>

Intro

The idea came to us when we all agreed to avoid using social media apps. Social media apps, we felt, were being overused and simply weren't unique enough for the project as our group sought to create an app that did not involve the mainstream choice of social media. After some discussion, we all found that we shared the similar trait of having a Steam account, so we chose to revolve our term project based on that fact. As to what exactly we were going to make using the Steam API, we further progressed our discussion. Purchasing video games has become exceptionally easy and cheap, as all of the latest gaming systems have integrated online stores where users can browse and purchase game titles with the click of a button. With PC video game services such as Steam, new discounts to video games are added everyday and many users swiftly purchase these game simply if the title or appearance of the game looks appealing. There have also been cases where gamers purchase a game simply due to a price drop, or pre-purchasing games that have yet to be released because of the hype surrounding it. A new problem is that video gamers are purchasing games recklessly, and their PC video game collection quickly becomes larger than the player can ever hope to play. Combined with simple lack of free time to play every one of those games and funds that would just as quickly diminish, gamers today are running into a predicament of over-purchasing video games.

Development History

Our project started simple enough in concept and design early on. We decided on a system that could decide whether or not a user should buy a specific PC game based on a variety of properties of the game and reviews. We started out at first with the steam api thinking that it would fulfill all of our needs but later developments proved that steam web api was not efficient enough to for all of our needs. We had just found steamkit2 a C# database that had all of the information we need to create an effective website and had lined up a metacritic api to use in conjunction. However, we managed to locate an unlisted steam store api that just so happened to have everything that we needed to fulfill our needs. Since design phase was already out and after two meetings we had already designed our algorithm, the actual programming of the site was relatively short. The team was given their separate tasks and throughout two weeks a prototype of the site was up and ready to demo. Changes were later made to the algorithm to help narrow down some of the "Bad" games to the "Potential" games as too many games the group knew were lower in quality had gotten "Maybe" ratings on them when they should have been flat out "Bad." We had developed a majority of this project separately as one handled the algorithm, another designed the homepage, someone else worked on the database and the connection to the database, and lastly someone had to put it together and put it into the server. Everyone did roughly a fair share or work and the final product turned out nicely.

Properties of the Website

The website was developed using HTML, CSS, Javascript, and PHP with a MySQL database. The homepage was developed using Bootstrap and was tested using Apache and later on Jacob's server. Initially there was three pages for the questions necessary to run the algorithm, but through Ajax the page count was decreased to one. The algorithm itself was programmed on php and through various calculations taking apart each of the properties that were listed on the steam page. Although it should be noted that Steam is NOT a perfect system and there are numerous games that are missing essential properties that are factored into the algorithm that the system simply needs to function. Because of this, the system was designed around games that were "Most Likely" going to be asked such as indie games or popular "Triple A" games. Completely obscure indie games that are lacking in many different ways tended to skew the results in a multitude of ways and while the system can give relatively correct scores, the scores themselves are less accurate compared to the type of games we designed around. It should also be noted that there was no efficient of simply listing all games on steam into the website as simply linking the user to a source of AppIDs to be used for the game as simply searching for the game by name alone and listing all of it's data without the AppID is a project within itself and while it is possible to store games into our database with the AppID and name, it would be a poor representation as it won't be even a percent of the actual size of the steam video game library no matter how much time we put into it as the steam game library constantly grows every day and has been for at least a decade.

Functionality

Our website uses PHP to request JSON formatted data.

Problems

As mentioned before, our system itself is not perfect for generating a score for each individual game and telling the user in three different answers as to whether or not the user should buy the product. The primary reason as to why is that the Steam store app is not a perfect system. Because the Steam app is flawed, our "Should you buy it?" app is inherently flawed and while a majority of games we tested work fine (we tested dozens individually and possibly over a hundred different games the four of us combined), there are still potentially dozens, if not hundreds of games that can bring up minor issues with the system. It is simply not possible for our database to contain all these games as we would have to sort through hundreds of thousands of appIDs, check if they are games or not, then add them to database despite knowing that dozens, if not, hundreds of games being incompatible with our system due to simply the way they are inputted into a game (for example: if a game was part of a package deal and the only way to get said game was through the package deal, then the game would be missing some essential information such as package/method of purchase). Also there is a limit to how many API calls we can make in a short period of time before the website starts temporarily denying all requests for a short while (is also really slow when doing so).

Sample Input and Output

SHOULD I BUY IT?


Tired of wasting money on Steam games you'll never play?

Not sure what to do with all of your spare time and money?
We can decide for you!

Continue »

About

With an endless selection of games on the Steam Store, it can be difficult to determine what's worth the money and what isn't. Use our website to help you make the right choice!



How does it work?


Give us a Steam game title and we will use factors such as price, ratings, and number of players to calculate a score telling you if you should buy the game.

Team A: CSCI 4300 Fall 2016

We start off on our home page where we have a button to proceed to the actual algorithm and questioning, and a short description of how to use our application and what our site is about.

SHOULD I BUY IT?

Enter the AppID for the Steam Game:

AppID: 


[Not sure? Click here!](#)

Team A: CSCI 4300 Fall 2016

The next step after clicking on the “Continue” button is to enter the App ID for the Steam game you were considering on purchasing. If you don’t know the game’s App ID, you can click on the link right below the text field to find out.

SHOULD I BUY IT?

Enter the AppID for the Steam Game:

AppID:
70 

[Not sure? Click here!](#)

First and Only Question:

Can you run this game?
Half-Life
System Requirements:
Minimum: 500 mhz processor, 96mb ram, 16mb video card, Windows XP, Mouse, Keyboard, Internet Connection
Recommended: 800 mhz processor, 128mb ram, 32mb+ video card, Windows XP, Mouse, Keyboard, Internet Connection



☒ Yes
☐ No
☐ I don't Know

Team A: CSCI 4300 Fall 2016

The moment we type in the App ID for the game we are interested in, the AJAX associated with our application will automatically generate the next step. The only question the user needs to answer before the final recommendation of whether or not he or she should buy the game is if the user’s computer meets the requirement based on the presented system requirements. Here in this example, the App ID of 70 was automatically detected as the game “Half-Life.”

SHOULD I BUY IT?

☒ Yes
☐ No
☐ I don't Know

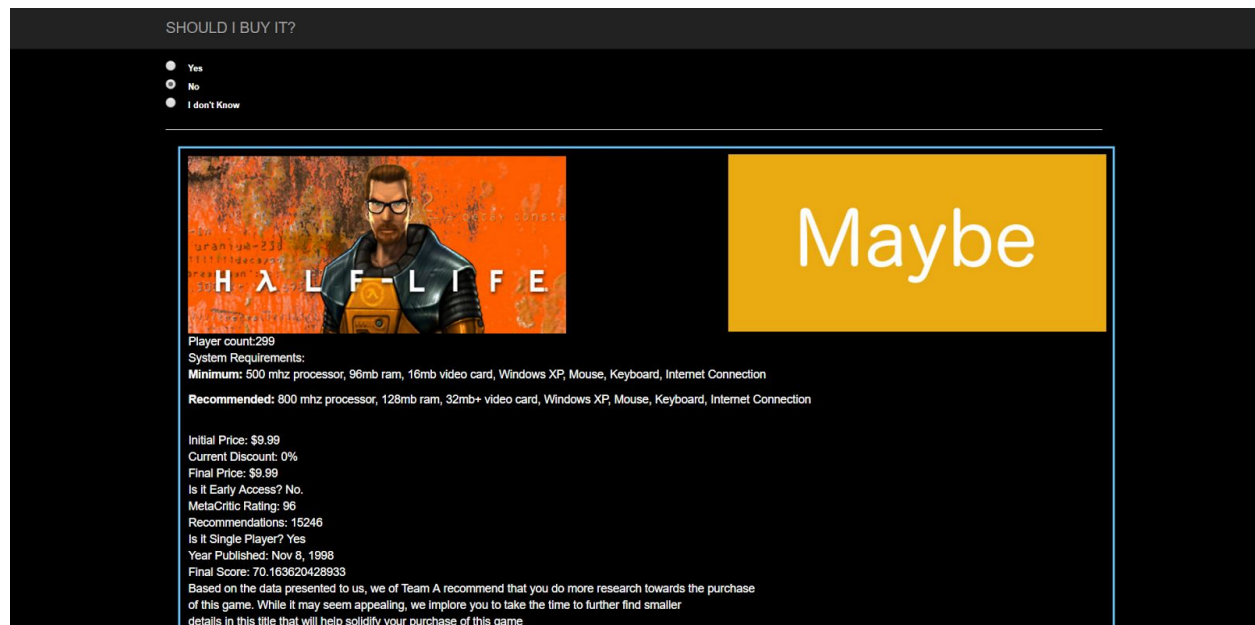


Player count:306
System Requirements:
Minimum: 500 mhz processor, 96mb ram, 16mb video card, Windows XP, Mouse, Keyboard, Internet Connection
Recommended: 800 mhz processor, 128mb ram, 32mb+ video card, Windows XP, Mouse, Keyboard, Internet Connection

Initial Price: \$9.99
Current Discount: 0%
Final Price: \$9.99
Is it Early Access? No.
MetaCritic Rating: 96
Recommendations: 15246
Is it Single Player? Yes
Year Published: Nov 8, 1998
Final Score: 80.385892096659
Based on the data presented to us, we of Team A recommend that you purchase this game!

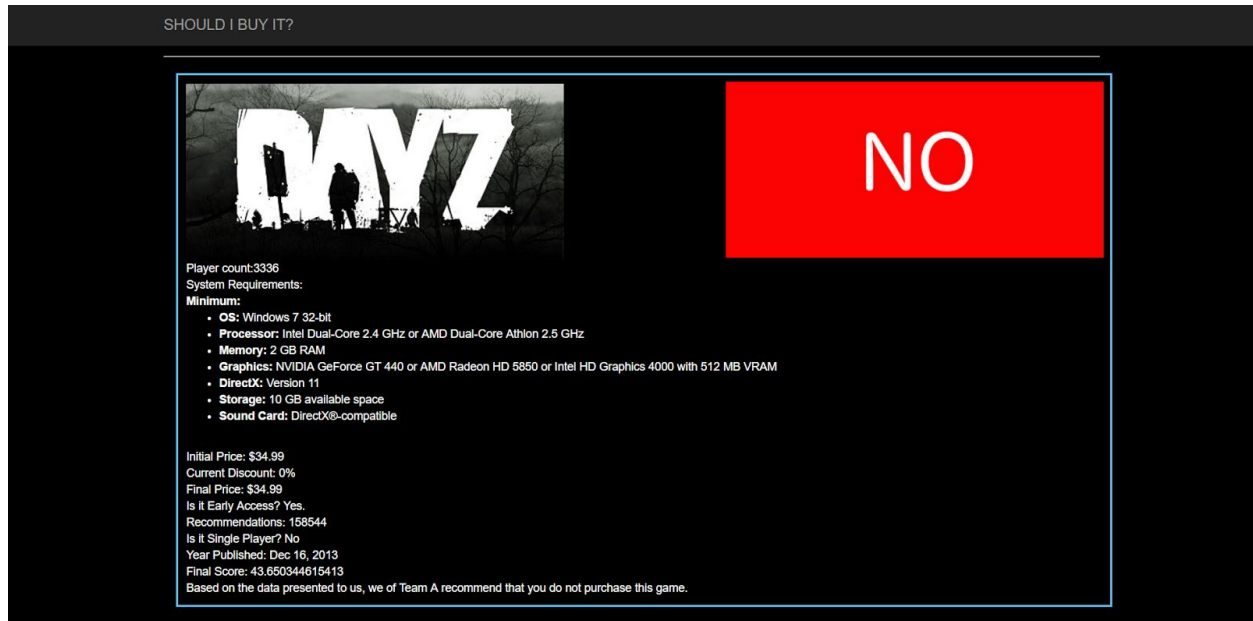
Here in this “Half-Life” example after answering “Yes” to the system requirements question, we get the recommendation that we should indeed purchase this game based on the presented game specifications.

Based on our application's algorithm, the game "Half-Life" received a score of 80.39% (rounded). Since the final score was above 75%, the result was an absolute "YES," and since we also answered that we met the requirements to run the game, no penalty was applied to the final score.



This is an example of if we were to change our answer to the system requirements to "No," which also automatically changes the web page dynamically in real time through the use of AJAX. Since now our system requirements don't meet with the game's system requirements, the final score receives a penalty, dropping it down to 70.16%. In our algorithm, a final score between 70-75% is a "MAYBE," and we suggest the user to conduct further research on the game as to whether or not he or she should buy the game because there are many other factors that can come into play that would be impractical for us to account for. One example would be if the user was planning on purchasing a more powerful computer in the near future, we would not be able to account for such a factor.

For the sake of completion but not shown in the image, an answer of "I Don't Know" results in a score of 76.94% which is another final rating of "YES," a middle-ground score in between choosing "Yes" and "No" for the system requirements.



As a final example, this is our page with a final recommendation of “NO.” The game used as an example here is “DayZ” with the answer of “Yes” to the system requirements. The reason this game received a low score of 43.65% is because it is an early access game, as shown as one of the game specifications. We all agreed as a group that just about every early access game on Steam turns out to be terrible because the developers of these early access games never finish completing the game and stop updating the game to fix all sorts of bugs and glitches, so the user reviews are filled with negativity. Due to that fact, we had our algorithm penalize the final score for all early access games.

Note: We added an additional changes when we finished the database and added a drop down menu to automatically fills the appID field for the game via MySQL database on the server.

