Jack Melcher
67574625
EECS 118

Final Project Report

I. Introduction

This project was inspired by a Youtuber and Twitch streamer Sean Poole (aka Spoole). His philosophy is that a game's playtime in hours should be at minimum equivalent to its selling price, hence the saying "one dollar, one hour". For example, if a game costs $60, you should expect at least 60 hours of playtime. If a game doesn't meet this criterion, then it's recommended to not purchase said game. Not all gamers follow this idea, but it is a way to validate one's purchase of a game. With this webapp, you will be able to determine whether a game for sale on the Steam Store is "One Dollar, One Hour!!!".

II. Problem Statement (problem or problems you try to solve.)

A large majority video game databases provide information such as the title, publisher, developer, genre, platform, price, rating, and sales number, yet they lack an estimate of playtime. By using Steam Web API, I will be able to obtain data on all the games available in the Steam Store's catalogue and make an accurate estimation of whether a game is "one dollar, one hour".

III. Background (a survey of existing services; why are they not sufficient?)

There are several video game database websites. A majority of them provide the same data such as title, release date, genre, platform, developer, publisher, and etc. Very few list data on a game's price and its playtime. Few websites list one or the other, but not both. The websites listed below are ones that I considered using in building the webapp, as the essential data required for the webapp are a game's price and playtime.

Wikipedia (https://en.wikipedia.org/wiki/Main_Page)

Wikipedia being an online encyclopedia does include several pages of the full lists of video games released on various consoles and platforms. The information on any individual game is staggering; yet again it does not have the listed price or estimated playtime.

HowLongToBeat (http://howlongtobeat.com/)

HowLongToBeat is a video game database that also provides the estimated playtime inputted by users. The playtimes for games were accurate as there are many active members using the site. Some issues with this site however are the reliance on user input for data, price of a game isn't listed, and no API to easily access the data on the site.

Steam Database Calculator (https://steamdb.info/calculator/)

Steam Database Calculator allows a Steam user to estimate the worth of their Steam library as well as provide the dollar per hour value per game that the user owns. While the listed price and playtime is present, the problem is that this calculator is only based on a single user.

Steam Spy (https://steamspy.com/)

This database provides the data I need in order to perform the calculations for the webapp, such as estimated playtime and listed price, and other information about a game.

IV. Service Overview

The webapp has two functions. The first is to search for a game available on Steam. By typing keywords into the search bar and pressing enter or clicking the Search button, a user can search for a game. A table is then brought up with a list of games from the search results. The user can then click on the games unique ID listed in the table to perform the second function of the webapp. After clicking on the ID, a new set of tables is brought up with additional information about the game; the most important being the Price, One Dollar One Hour Score, and Dollar per Hour.

V. Design (UML preferred)

The webapp was designed using the standard Java Classes along with classes in the JSoup library to parse the information from the Steam Spy website.

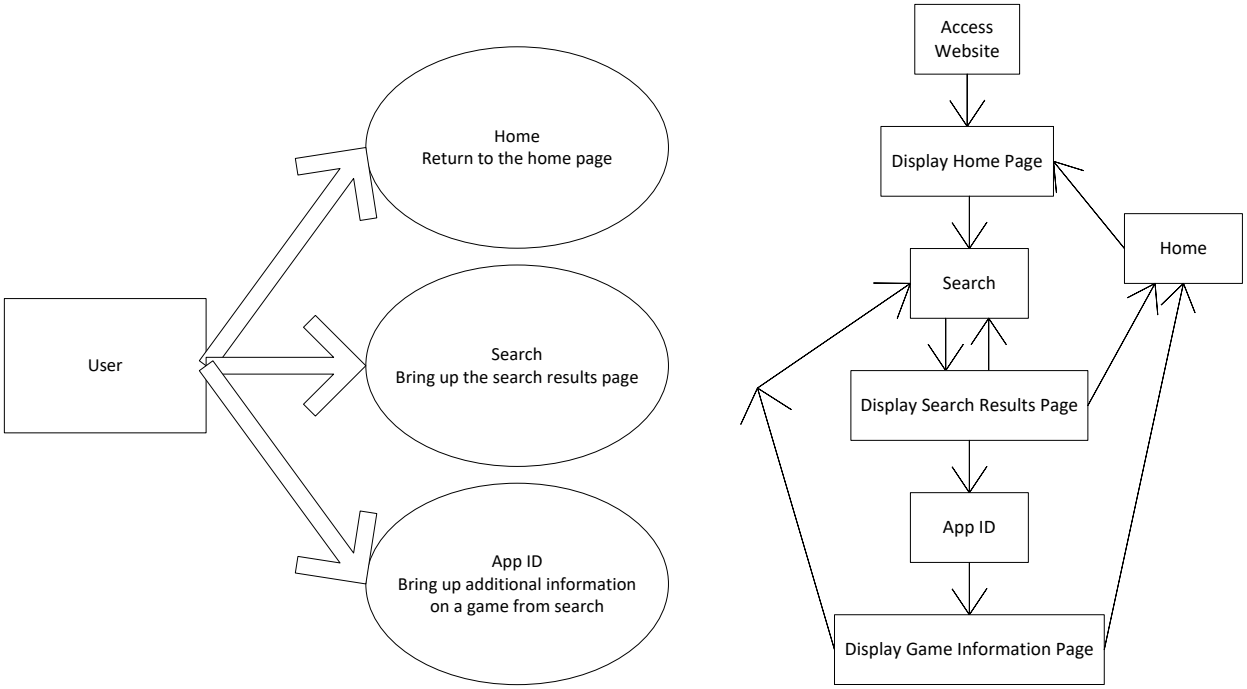

Figure 1: UML Use Case Diagram and Sequence Diagram

VI. User Interface

Below are screenshots of the user interface.

Figure 2: Home Page

Figure 2 shows the home page. It includes the webapp's title, a search bar, a cover image, and an about section, and a home button that will return you to this page. The user can search for a game by typing keywords into the search bar. The about section elaborates on inspiration for the webapp as well as sources for the cover image, where the data is obtained, and what values are from the database versus what is calculated by the webapp.

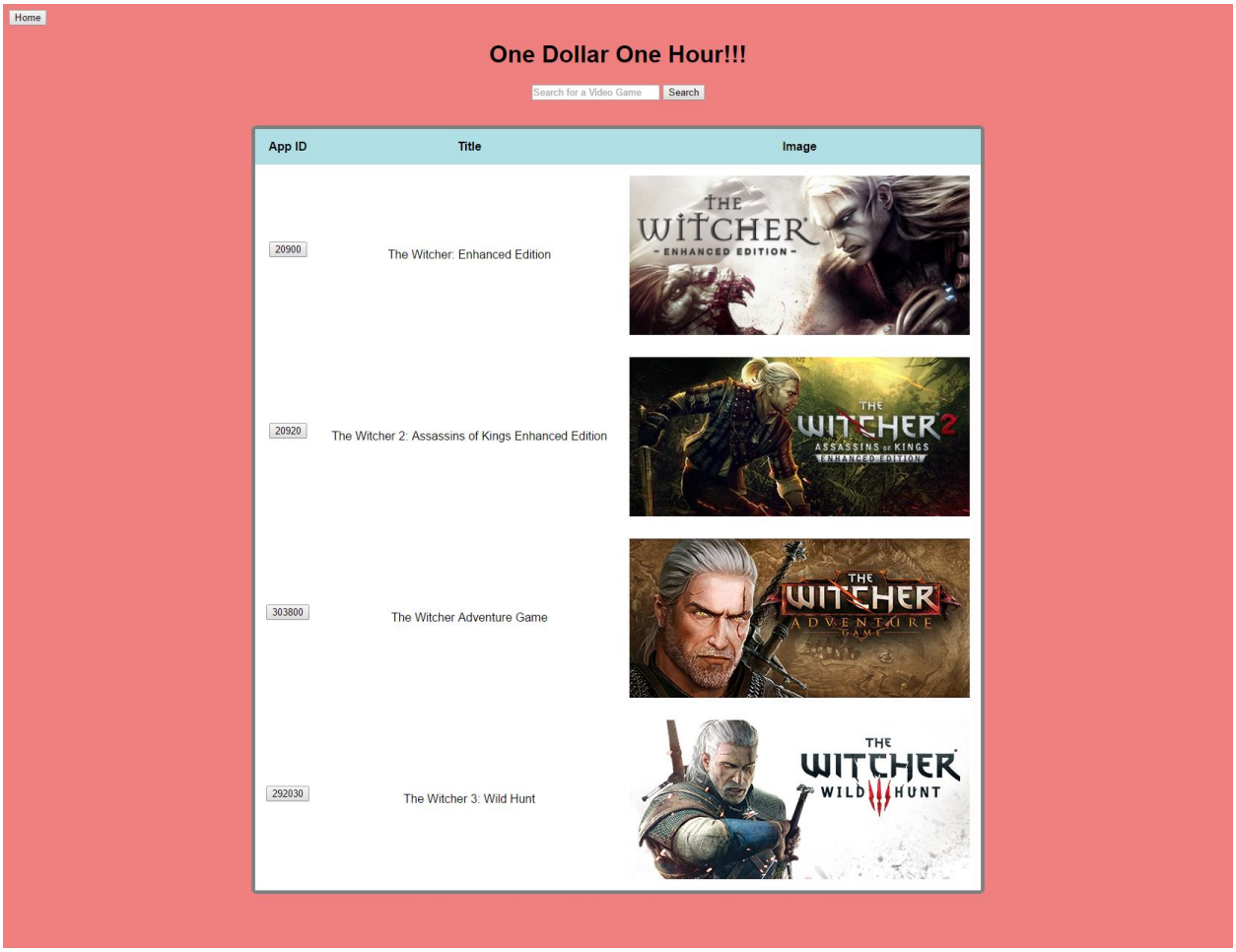

Figure 3: Search Results

Figure 3 shows an example of a search results table. The table includes a game's App ID, title, and cover image. The search results are based on the search results from Steam Spy. In this example, the keyword "witcher" was used. The APP IDs for each game are buttons that will bring up that game's additional information. The user can still search for games on this page or return to the home page by clicking on the home button.

Figure 4: Game Information

Figure 4 shows tables of a game's information. The first table shows the Price, One Dollar One Hour Score, Dollar per Hour, and Worth the Price. The second table shows the Average Playtime Total, Average Playtime in the Last Two Weeks, and Average Playtime in One Day. The third table shows the Metacritic Score, User Score, and Combined Score. The fourth table shows the Owners, Estimated Profit, Players, and Players in the Last Two Weeks. The fifth table shows the Player to Owner %, Active Player %, and Current Activity. The user can still search for games on this page or return to the home page by clicking on the home button.

VII. Testing

The following is the testing plan and elaboration of the webapp development.

1. Determine whether to use an API or parse from a website to gather the data needed for the webapp. I eventually decided on parsing Steam Spy using the JSoup
2. Practice with JSoup until I could use the websites searching method and parse data from the html of my search result.
3. Use the search results to reach a game of interest. From here, parse the data from the html with JSoup that was of interest for the webapp. Unfortunately, the data was all contained in a single html element that had to be further parsed using a custom method.
4. Code the custom method to parse the data and separate the names from their corresponding values. Both the names and values were printed separately to know they were indeed obtained properly.
5. Perform custom calculations on the parsed data and print the names and results of the calculations.
6. Determine the layout of the webapp. I decided to keep it simple, using basic CSS styles and placing all the data in tables. The pages then were centered and the data was organized tables based on their similarity to each other. Afterwards I added in a home button and a cover image.
7. Cleaned up the code and removed any unnecessary comments.

VIII. Documented Source Code

index.jsp

```
<! Jack Melcher
        67574625
        10/27/2016
        EECS118
        Term Project
>

<%-- In the beginning of the file, import the library you need: --%>
<%@ page contentType="text/html;charset=UTF-8" pageEncoding="UTF-8" %>
<%@ page import="java.util.*"%>
<%@ page import="java.io.*"%>
<%@ page import="java.text.NumberFormat"%>

<!jsoup libraries for html parsing/cleaning>
<%@ page import="org.jsoup.*"%>
<%@ page import="org.jsoup.helper.*"%>
<%@ page import="org.jsoup.nodes.*"%>
<%@ page import="org.jsoup.parser.*"%>
<%@ page import="org.jsoup.safety.*"%>
<%@ page import="org.jsoup.select.*"%>

<html>
        <head>
                <title>One Dollar One Hour</title>
                <style>
                table{
                        margin: 0px auto;
                        border-collapse: separate;
                        border-spacing: 0;
                        border: 5px solid grey;
                        border-radius: 7px;

                }
                th, td {
                        padding: 15px;
                        margin: 0px auto;
                        text-align: center;
                        font-size: 100%
                }
                th {
                        background-color: powderblue;
                }
                td{
                        background-color: white;
                }
                h1, h2, h3, h4 {
```

```
                text-align: center;
        }
        h1, th, td{
                font-family: arial;
        }
        body{
                background-color: lightcoral;
        }
        a {
                text-decoration: none;
        }
        </style>
</head>


<body>
        <form method="post">
                <input type="submit" name="submit3" value="Home"/>
        </form>
        <h1>One Dollar One Hour!!!</h1>


        <!User provides keywords of a video game's title. When the search button is pressed, a list of video games containing the keywords should pop-up >


        <center>
        <form method="post">
                <input name="queryString" type=text placeholder="Search for a Video Game" />
                <input type="submit" name="submit1" value="Search"/>
        </form>
        </center>
        <br>
        <!Cover Image>
        <%
        if (request.getParameter("submit1") == null && request.getParameter("submit2") == null){
                out.println("<img src=\"H57bLkC.jpg\" style=\"display: block; margin: 0px auto; width: 50%;\">");
        }
        %>

        <%
        if (request.getParameter("submit1") != null) {
                Document doc;

                try {

                        // need http protocol
```

```
                                        doc =
Jsoup.connect("https://steamspy.com/search.php?s="+request.getParameter("queryString")).userAgent("Mozilla").g
et();

                                        //Table
                                        out.println("<table>");
                                        out.println("<tr>");
                                        out.println("<th>App ID</th>");
                                        out.println("<th>Title</th>");
                                        out.println("<th>Image</th>");
                                        out.println("</tr>");

                                        // get game titles
                                        Elements game_titles = doc.select("h3");

                                        // get all links
                                        Elements links = doc.select("a[href*=/app/]");

                                        //Print values from both Elements lists
                                        for(int i = 0; i < game_titles.size(); i++){
                                                out.println("<tr>");
                                                out.println("<td><form method=\"post\"><input name=\"submit2\"
type=\"submit\" value=\""+links.get(i).attr("href").substring(5)+"\"></form></td>");
                                                out.println("<td>"+game_titles.get(i).html()+"</td>");
                                                out.println("<td><img
src=\"http://cdn.akamai.steamstatic.com/steam/apps/"+links.get(i).attr("href").substring(5)+"/header.jpg\"></td>");
                                                out.println("</tr>");
                                        }

                                out.println("</table>");


                                } catch (IOException e) {
                                        out.println("Threw an error<br>");
                                        e.printStackTrace();
                                        out.println(e.getMessage());
                                }
                        }
                        %>

                        <!When a game is selected by the user from the above list the following will happen>


                        <!Function B>
                        <%
                        if (request.getParameter("submit2") != null) {
                                Document doc;
```

```
try {

        // need http protocol
        doc =
Jsoup.connect("https://steamspy.com/app/"+request.getParameter("submit2")).userAgent("Mozilla").get();
        //out.println("Connected to the website <br>");

        // get game titles
        Elements game_titles = doc.select("h3");

        // get the image
        Elements images =
doc.select("img[src*="+request.getParameter("submit2")+"]");
        //out.println(images.get(0).toString());
        //out.println("<br><br>");

        //Print values from both Elements lists
        String values = doc.select("p").text();


        //Pulled Data
        //out.println("Pulled Data<br>");
        //Price
        int index = values.indexOf("Price:");
        String pricestring = "";
        //out.println(index);
        int i = 0;
        if(index != -1){
                while(i < 2){
                        //out.print(values.charAt(index));
                        if(i > 0){
                                pricestring += values.charAt(index);
                        }
                        index++;
                        if(values.charAt(index) == ' '){
                                i++;
                        }
                }
                //out.println("<br>");
        }
        float price = 0;
        //out.println(pricestring+"<br>");
        //out.println(pricestring.replace('$',' ')+"<br>");

        Scanner scanner = new Scanner(pricestring.replace('$',' '));
        if(scanner.hasNextFloat()){
                price = scanner.nextFloat();
        }
        //out.println("Price: $"+price+"<br>");
```

```
//Playtime Total
index = values.indexOf("Playtime total:");
String averageplaytimestring = "";
//out.println(index);
i = 0;
if(index != -1){
	while(i < 3){
		//out.print(values.charAt(index));
		if(i > 1){
			averageplaytimestring += values.charAt(index);
		}
		index++;
		if(values.charAt(index) == ' '){
			i++;
		}
	}
	//out.println("<br>");
}
int averageplaytimehour = 0;
int averageplaytimemin = 0;
float averageplaytime = 0;
//out.println(averageplaytimestring+"<br>");

scanner = new Scanner(averageplaytimestring.replace(':',' '));
if(scanner.hasNextInt()){
	averageplaytimehour = scanner.nextInt();
}
if(scanner.hasNextInt()){
	averageplaytimemin = scanner.nextInt();
}
averageplaytime = averageplaytimehour + ((float)averageplaytimemin/60);
//out.println("Average Playtime Total:
"+String.format("%.2f",averageplaytime)+" hours<br>");


//Playtime last 2 weeks
index = values.indexOf("Playtime in the last 2 weeks:");
String averageplaytime2weeksstring = "";
//out.println(index);
i = 0;
if(index != -1){
	while(i < 7){
		//out.print(values.charAt(index));
		if(i > 5){
			averageplaytime2weeksstring +=
values.charAt(index);
		}
```

```
                                        index++;
                                        if(values.charAt(index) == ' '){
                                                i++;
                                        }
                                }
                                //out.println("<br>");
                        }
                        int averageplaytime2weekshour = 0;
                        int averageplaytime2weeksmin = 0;
                        float averageplaytime2weeks = 0;
                        //out.println(averageplaytime2weeksstring+"<br>");

                        scanner = new Scanner(averageplaytime2weeksstring.replace(':',' '));
                        if(scanner.hasNextInt()){
                                averageplaytime2weekshour = scanner.nextInt();
                        }
                        if(scanner.hasNextInt()){
                                averageplaytime2weeksmin = scanner.nextInt();
                        }
                        averageplaytime2weeks = averageplaytime2weekshour +
((float)averageplaytime2weeksmin/60);
                        //out.println("Average Playtime in the Last Two Weeks:
"+String.format("%.2f",averageplaytime2weeks)+" hours<br>");


                        //Metacritic Score
                        index = values.indexOf("Metascore:");
                        String metascorestring = "";
                        //out.println(index);
                        i = 0;
                        if(index != -1){
                                while(i < 2){
                                        //out.print(values.charAt(index));
                                        if(i > 0){
                                                metascorestring += values.charAt(index);
                                        }
                                        index++;
                                        if(values.charAt(index) == ' '){
                                                i++;
                                        }
                                }
                                //out.println("<br>");
                        }
                        int metascore = 0;
                        //out.println(metascorestring+"<br>");

                        scanner = new Scanner(metascorestring.replaceAll("%",""));
                        if(scanner.hasNextInt()){
                                metascore = scanner.nextInt();
```

```
}
//out.println("Meta Score: "+metascore+"%<br>");

//User Score
index = values.indexOf("Userscore:");
String userscorestring = "";
//out.println(index);
i = 0;
if(index != -1){
	while(i < 2){
		//out.print(values.charAt(index));
		if(i > 0){
			userscorestring += values.charAt(index);
		}
		index++;
		if(values.charAt(index) == ' '){
			i++;
		}
	}
	//out.println("<br>");
}
int userscore = 0;
//out.println(userscorestring+"<br>");

scanner = new Scanner(userscorestring.replaceAll("%",""));
if(scanner.hasNextInt()){
	userscore = scanner.nextInt();
}
//out.println("User Score: "+userscore+"%<br>");


//Total Number of Owners
index = values.indexOf("Owners:");
String ownersstring = "";
//out.println(index);
i = 0;
if(index != -1){
	while(i < 2){
		//out.print(values.charAt(index));
		if(i > 0){
			ownersstring += values.charAt(index);
		}
		index++;
		if(values.charAt(index) == ' '){
			i++;
		}
	}
	//out.println("<br>");
}
```

```
int owners = 0;
//out.println(ownersstring+"<br>");

scanner = new Scanner(ownersstring.replaceAll(",",""));
if(scanner.hasNextInt()){
	owners = scanner.nextInt();
}
//out.println(owners+"<br>");
//out.println("Owners: "+ownersstring+"<br>");



//Total Number of Players
index = values.indexOf("Players total:");
String playertotalstring = "";
//out.println(index);
i = 0;
if(index != -1){
	while(i < 3){
		//out.print(values.charAt(index));
		if(i > 1){
			playertotalstring += values.charAt(index);
		}
		index++;
		if(values.charAt(index) == ' '){
			i++;
		}
	}
	//out.println("<br>");
}
int playerstotal = 0;
//out.println(playertotalstring+"<br>");

scanner = new Scanner(playertotalstring.replaceAll(",",""));
if(scanner.hasNextInt()){
	playerstotal = scanner.nextInt();
}
//out.println(playerstotal+"<br>");
//out.println("Players: "+playertotalstring+"<br>");


//Players in last 2 weeks
index = values.indexOf("Players in the last 2 weeks:");
String players2weeksstring = "";
//out.println(averageplaytimeindex);
i = 0;
if(index != -1){
	while(i < 7){
		//out.print(values.charAt(index));
```

```
                    if(i > 5){
                        players2weeksstring += values.charAt(index);
                    }
                    index++;
                    if(values.charAt(index) == ' '){
                        i++;
                    }
            }
            //out.println("<br>");
}
int players2weeks = 0;
//out.println(players2weeksstring+"<br>");

scanner = new Scanner(players2weeksstring.replaceAll(",",""));
if(scanner.hasNextInt()){
        players2weeks = scanner.nextInt();
}
//out.println(players2weeks+"<br>");
//out.println("Players in the Last Two Weeks: "+players2weeksstring+"<br>");

//out.println("<br>");
//out.println("Calculated Data<br>");

//Calculated Data
float oneDollarOneHourScore = averageplaytime - price;
float dollarPerHour = price / averageplaytime;
float estimatedProfit = price * owners;
float activeplayerrate = (float)players2weeks / (float)playerstotal * 100;
float averageplaysession = averageplaytime2weeks / 14;
float overallrating = 0;
if(metascore == 0){
        overallrating = userscore;
}
else if(userscore == 0){
        overallrating = metascore;
}
else{
        overallrating = (metascore + userscore) / 2;
}
float playerpercent =  (float)playerstotal /  (float)owners *100;


String currentactivity;
//out.println("Current Activity: ");
if(players2weeks >= 2000000) {
        currentactivity = "Extremely High";
}
else if(players2weeks >= 1000000) {
        currentactivity = "Very High";
```

```
}
else if(players2weeks >= 500000) {
		currentactivity = "High";
}
else if(players2weeks >= 250000) {
		currentactivity = "Moderately High";
}
else if(players2weeks >= 100000) {
		currentactivity = "Moderate";
}
else if(players2weeks >= 10000) {
		currentactivity = "Moderately Low";
}
else if(players2weeks >= 1000) {
		currentactivity = "Low";
}
else if(players2weeks > 0) {
		currentactivity = "Very Low";
}
else{
		currentactivity = "Abandoned";
}
//out.println(currentactivity+"<br>");


String worthprice;
//out.println("Worth the Price? ");
if(oneDollarOneHourScore >= 100) {
		worthprice = "Unquestionably, Yes";
}
else if(oneDollarOneHourScore >= 75) {
		worthprice = "Truely, Yes";
}
else if(oneDollarOneHourScore >= 50) {
		worthprice = "Absolutely, Yes";
}
else if(oneDollarOneHourScore >= 25) {
		worthprice = "Definitely, Yes";
}
else if(oneDollarOneHourScore >= 0) {
		worthprice = "Yes";
}
else if(oneDollarOneHourScore >= -10) {
		worthprice = "Almost";
}
else if(oneDollarOneHourScore >= -20) {
		worthprice = "Questionable";
}
else if(oneDollarOneHourScore >= -30) {
```

```
                worthprice = "No";
            }
            else if(oneDollarOneHourScore >= -40) {
                worthprice = "Spend your money elsewhere";
            }
            else{
                worthprice = "Please do not spend your money on this game";
            }


            String output = "" +
            "<table>"+
                "<tr>"+
                    "<th>"+game_titles.get(0).html()+"</th>"+
                    "<th>"+"<a
href=https://store.steampowered.com/app/"+request.getParameter("submit2")+"> Steam Store</a>"+"</th>"+
                "</tr>"+
                "<tr>"+
                    "<td colspan=\"2\">"+"<img
src=\"http://cdn.akamai.steamstatic.com/steam/apps/"+request.getParameter("submit2")+"/header.jpg\">"+"</td>"+
                "</tr>"+
            "</table>"+
            "<br>"+
            "<table>"+
                "<tr>"+
                    "<th>Price</th>"+
                    "<th>One Dollar One Hour Score</th>"+
                    "<th>Dollar per Hour</th>"+
                    "<th>Worth the Price?</th>"+
                "</tr>"+
                "<tr>"+
                    "<td>$"+String.format("%.2f",price)+"</td>"+

        "<td>"+String.format("%.2f",oneDollarOneHourScore)+"</td>"+
                    "<td>$"+String.format("%.2f",dollarPerHour)+"</td>"+
                    "<td>"+worthprice+"</td>"+
                "</tr>"+
            "</table>"+
            "<br>"+
            "<table>"+
                "<tr>"+
                    "<th>Average Playtime Total</th>"+
                    "<th>Average Playtime in the Last Two Weeks</th>"+
                    "<th>Average Playtime in One Day</th>"+
                "</tr>"+
                "<tr>"+
                    "<td>"+String.format("%.2f",averageplaytime)+"
hours</td>"+
```

```
                                        "<td>"+String.format("%.2f",averageplaytime2weeks)+"
hours</td>"+
                                        "<td>"+String.format("%.2f",averageplaysession)+"
hours</td>"+
                                    "</tr>"+
                                "</table>"+
                                "<br>"+
                                "<table>"+
                                    "<tr>"+
                                        "<th>Metacritic Score</th>"+
                                        "<th>User Score</th>"+
                                        "<th>Combined Score</th>"+
                                    "</tr>"+
                                    "<tr>"+
                                        "<td>"+metascore+"%</td>"+
                                        "<td>"+userscore+"%</td>"+
                                        "<td>"+overallrating+"%</td>"+
                                    "</tr>"+
                                "</table>"+
                                "<br>"+
                                "<table>"+
                                    "<tr>"+
                                        "<th>Owners</th>"+
                                        "<th>Estimated Profit</th>"+
                                        "<th>Players</th>"+
                                        "<th>Players in the Last Two Weeks</th>"+
                                    "</tr>"+
                                    "<tr>"+
                                        "<td>"+ownersstring+"</td>"+

        "<td>$"+NumberFormat.getNumberInstance(Locale.US).format(estimatedProfit)+"</td>"+
                                        "<td>"+playertotalstring+"</td>"+
                                        "<td>"+players2weeksstring+"</td>"+
                                    "</tr>"+
                                "</table>"+
                                "<br>"+
                                "<table>"+
                                    "<tr>"+
                                        "<th>Player to Owner %</th>"+
                                        "<th>Active Player %</th>"+
                                        "<th>Current Activity</th>"+
                                    "</tr>"+
                                    "<tr>"+
                                        "<td>"+String.format("%.2f",playerpercent)+"%</td>"+
                                        "<td>"+String.format("%.2f",activeplayerrate)+"%</td>"+
                                        "<td>"+currentactivity+"</td>"+
                                    "</tr>"+
                                "</table>";
```

```
                        out.println(output);

                    } catch (IOException e) {
                        out.println("Threw an error<br>");
                        e.printStackTrace();
                        out.println(e.getMessage());
                    }

                    out.println("<br><br>");
                }


                %>

                <!When a game is selected by the user from the above list the following with display>


                <!This is the Formatted Website>



                <!About Section>
                <br><br><br><br>
                <b>About:</b><br>
                This project was inspired by a Youtuber and Twitch streamer Sean Poole (aka Spoole).<br>
                His philosophy is that a game's price should at minimum be equivalent to its estimated
playtime.<br>
                If a game doesn't meet this requirement, then you should not purchase said game.<br>
                With this webapp, you will be able to determine whether a game is One Dollar One Hour.<br>
                The following values are obtained from the website <a href=https://steamspy.com/> Steam
Spy</a>:<br>
                Price, Average Playtime Total, Average Playtime in the Last Two Weeks, Metacritic Score, User
Score, Owners, Players, Players in the Last Two Weeks<br>
                The following values are calculated by the webapp:<br>
                One Dollar One Hour Score, Dollar per Hour, Worth the Price?, Average Playtime in One Day,
Combined Score, Estimated Profit, Player to Owner %, Active Player %, Current Activity<br>
                Cover image obtained from the following <a
href=https://www.reddit.com/r/funhaus/comments/2wjwgh/spools_life_tips_001/> reddit post</a>.<br>
                This webapp only applies to video games available on Valve's <a
href=http://store.steampowered.com/> Steam Store</a>.<br>

        </body>
</html>
```

IX. Deployment

The project contains one JSP file and one JPG image. The JSP file runs on Apache Tomcat Server, with the JSoup library included in the lib folder. The JSP file was coded in the Sublime text editor. Everything that I wanted to implement in the webapp is working as intended. The site can be accessed from a web browser at:

http://128.195.204.85:8081/JackMelcher/index.jsp

X. Conclusion and Future Work

From the term project, I learned how to code basic HTML, how to utilize Java code in a JSP file to generate a HTML webpage, how to access a website and parse data using an HTML parser, and how to design the layout of a website using basic CSS.

In terms of what I learned in terms of the development, I learned that parsing data from another website isn't the best solution for making a webapp. The data I want access to and perform calculations on are available in an API called Steam Web API. An issue with obtaining a key for the API is that I needed a domain name which I don't own.

In the future and if I were able to have access to the API, I would be able to provide more accurate estimations on playtime hours, game profit, and a range of hours players have spent playing the game. I would have also avoided the need to perform addition methods on data parsed from the site I used.

In terms of design, I could spend additional time making a more streamline and modern layout for the website. A design more akin to Google's material design of their websites and services would be something I would like to make.

Overall, the webapp is a solution to the problem I was interested in solving and I can take what I learned and apply it to other web development projects.