**Final Report**

**for**

**BiFrost Games**

**By**

**Curtis Naples**

**Devan Weber**

**Evan Plant**

**Pat Horler**

Contents

[Project Overview 3](#_Toc517479215)

[1. Project Specifications 3](#_Toc517479216)

[1.1 Function Requirements 3](#_Toc517479217)

[1.2 Non-Function Requirements 3](#_Toc517479218)

[1.3 Group Roles 3](#_Toc517479219)

[2. Usability Guide 4](#_Toc517479220)

[3. Problems and Challenges 12](#_Toc517479227)

[4. Skill Learned 13](#_Toc517479228)

[5. Proposed Changes 13](#_Toc517479229)

[6. Code Snippets 13](#_Toc517479230)

[7. Figures 18](#_Toc517479231)

[Figure 1 - UML Use Case Diagram 18](#_Toc517479232)

[Figure 2 – Data Model Diagram 19](#_Toc517479233)

[Figure 3 – Login page 19](#_Toc517479234)

[Figure 4 – Cart page 19](#_Toc517479235)

8. Summary………………………………………………………………………………………20

## Project Overview

We are creating a website that will be able to have video games posted and sold. A shopping cart will be implemented so customers can add multiple games to the cart and buy them in one transaction. The games will be able to be sorted by console and genre simultaneously. Customers must make an account to save their order history and be able to place orders.

## Project Specifications

### 1.1 Function Requirements

1. The customer must be able to browse items and select category filters.
2. An item must be able to be added to, edited, or removed from the shopping cart.
3. The customer can purchase the contents of the cart using a secure payment system.
4. The customer can have an account on the system.
5. The customer can view their purchase history.
6. The client can add products to the system.

### 1.2 Non-Function Requirements

1. The system will be built using apache web server
2. The front end will be coded using HTML, CSS and JavaScript
3. The middleware will be built with PHP
4. The database will be built using Oracle SQL
5. The system must provide a consistent look and feel across the user interface
6. The system will use free open source software
7. The system must be fully tested and running in 8 weeks

### 1.3 Group Roles

Curtis Naples Front-End Developer, Middle-ware Developer, Team Lead

Devan Weber Front-End Developer, Middle-ware Developer, Database/Back-end Developer

Evan Plant Database/Back-end Developer, Middle-ware Developer, Front-End Developer

Pat Horler Team Lead, Front-End Developer, Technical Writer

## Usability Guide

## Header

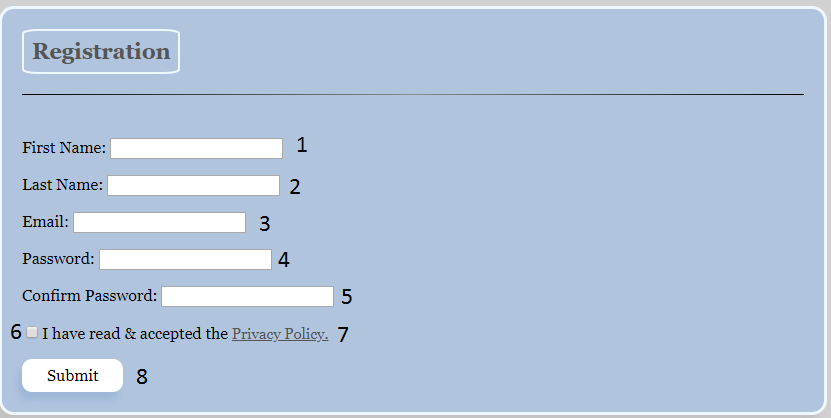


1. Opens the Navigation Sidebar
2. Header goes to products page
3. Log Out/Login to account

## Sidebar

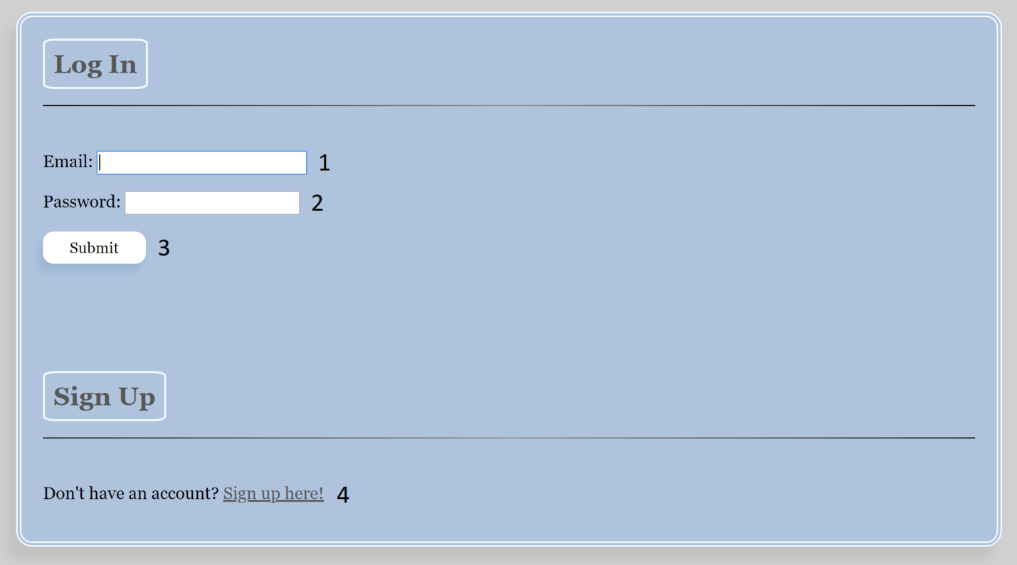
1. Closes the Sidebar
2. Goes to the products page
3. Add a new product (Admin Only)
4. View your Cart
5. View your Order History
6. Logout/Login to account

## Registration



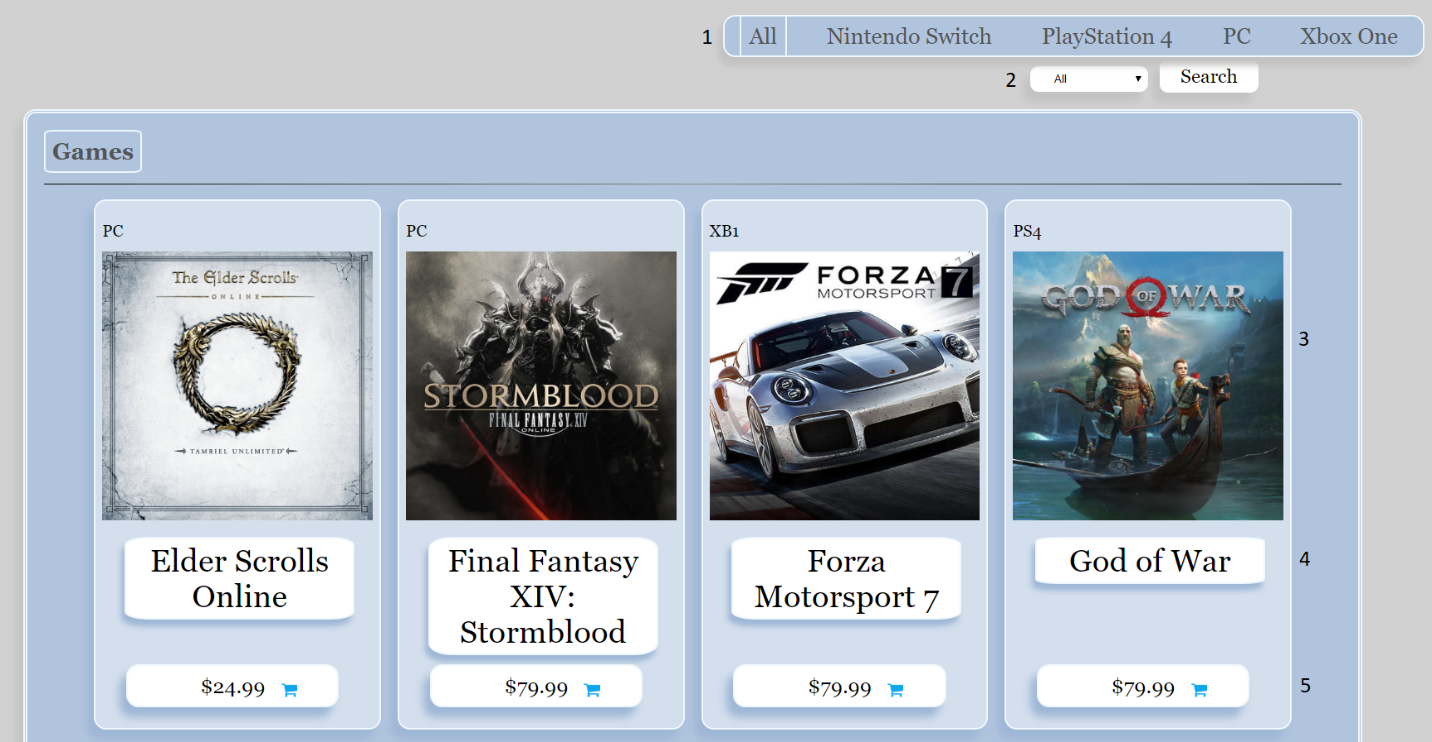
1. Textbox to enter your first name – Required
2. Textbox to enter your last name – Required
3. Textbox to enter your email – Required and Validated
4. Textbox to enter your password – Required
5. Textbox to confirm your password – Required
6. Checkbox to confirm that you have accepted the Privacy Policy – Required
7. Privacy Policy link, if clicked will open the Privacy Policy for you to read
8. Submit button, all fields will be validated when this is pressed and if everything is good, your information and account will be saved into the database and you will be redirected and automatically logged in. If anything is missing or validation fails, the page will alert you to what is needed.

## Login



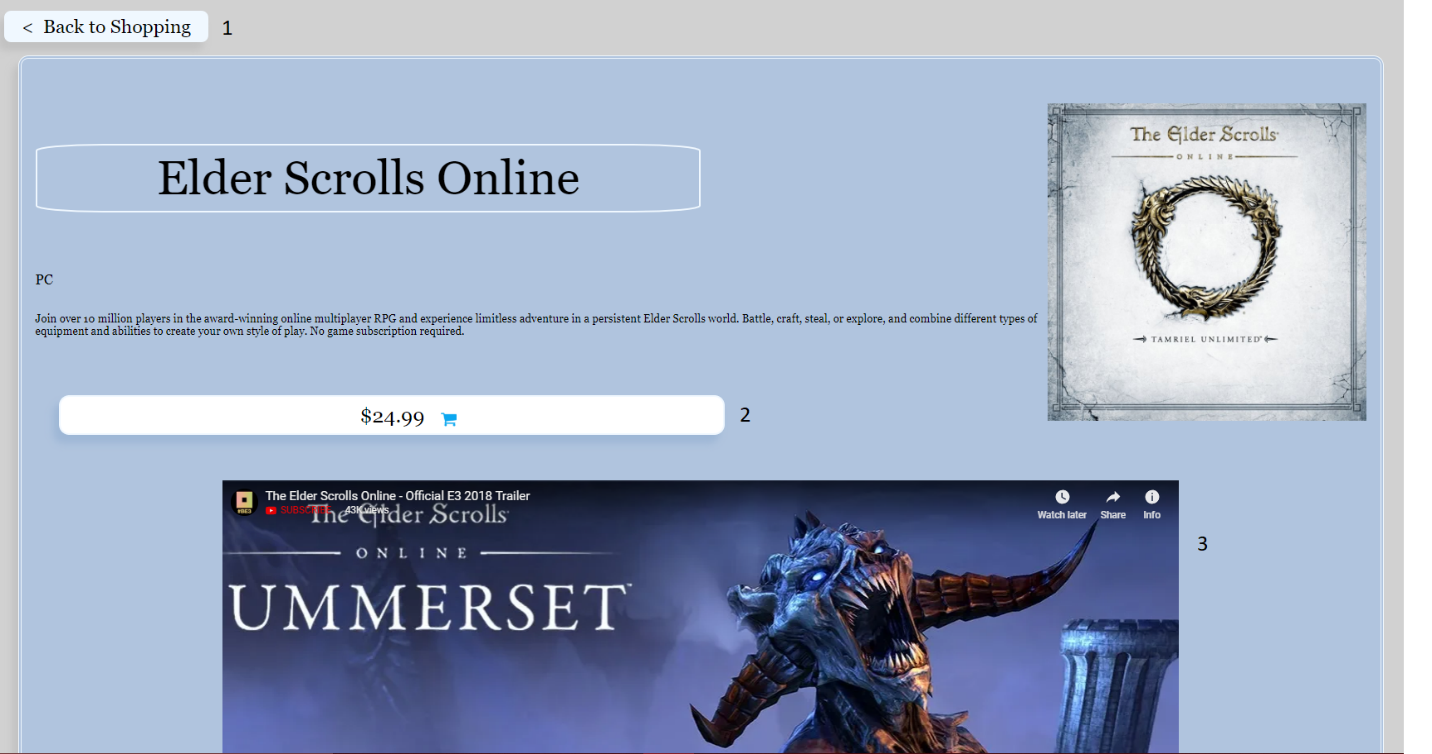
1. Textbox to enter your login email – Required and Validated
2. Textbox to enter your login password – Required
3. Submit button, your login info will be sent and validated against the info stored in the database. If the submitted info is incorrect, you will be given, and error message and you can try to login again. If the submitted info is correct, you will be redirected to an appropriate page. If your Privacy Policy is out of date, you will be prompted to agree to a new one before the login completes.
4. Link to redirect you to the Registration page, where you can create a new account.

## Products



1. Select console category
2. Select Sub-Category
3. Image links to individual product page
4. Title links to individual product page
5. Price and Cart adds the item to your cart

## Product

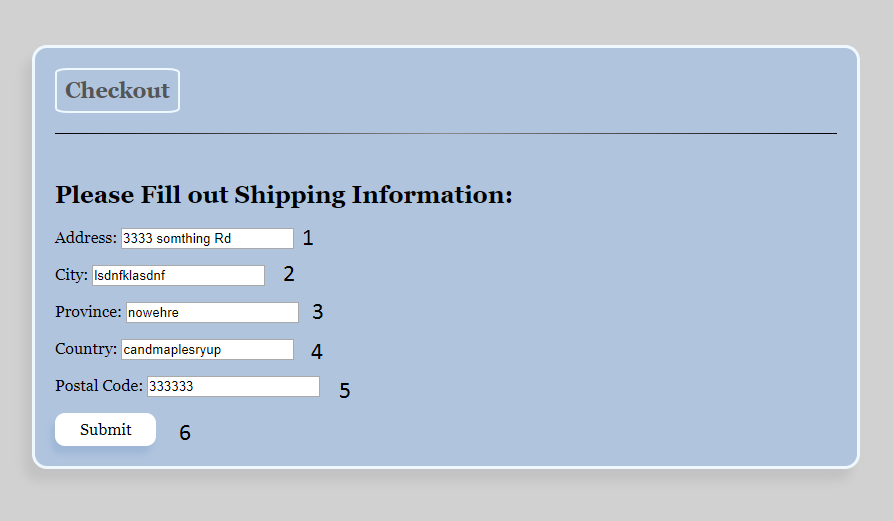


1. Go back to the products page button
2. Price and Cart adds the item to your cart
3. YouTube trailer for the game

## Cart

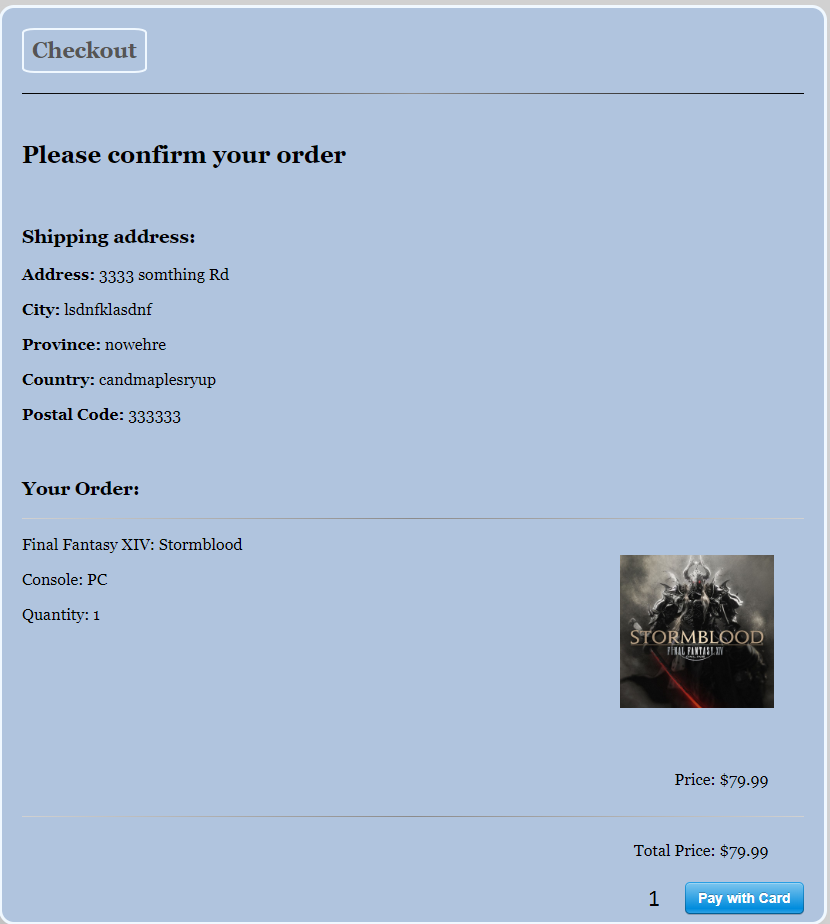
1. Add Quantity
2. Clear all contents from cart
3. Go to checkout
4. Back to products page
5. Link to product details page
6. Remove Item from cart
7. Link to product details page
8. Reduce Quantity

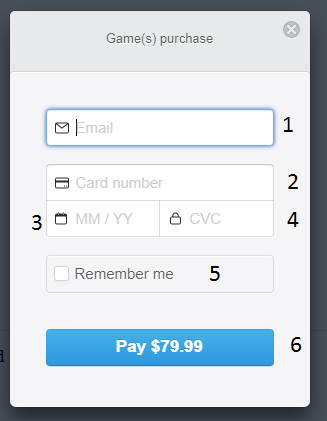
## Checkout



1. Textbox to enter your Address – Required and Validated
2. Textbox to enter your City – Required
3. Textbox to enter your Province – Required
4. Textbox to enter your Country – Required
5. Textbox to enter your Postal Code – Required
6. Submit button to enter your information and move to page 2 of checkout. When this button is pressed, if you never entered an address before, it will be saved to the database and next time you visit this page the textboxes will be filled automatically. If you have entered a address before and if any of the textboxes have different text than what is saved, the database will be updated.

## Checkout 2



1. When clicked, Stripe will open to start the payment process

# 

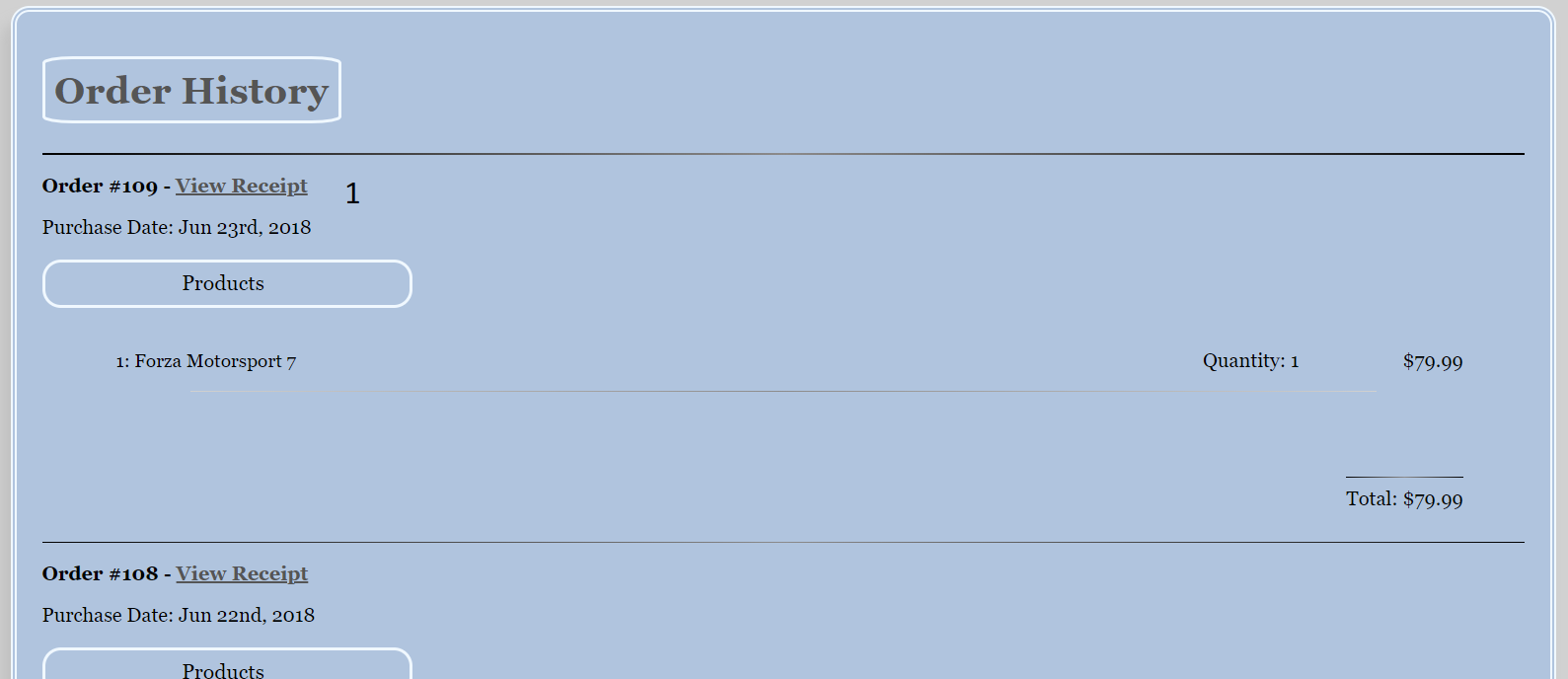
## Stripe Payment

1. Enter email for stripe receipt – Required and Validated
2. Enter your credit card number – Required and Validated
3. Enter the expiry for your credit card – Required
4. Enter CVC – Required
5. Check this to have stripe remember this information, you will have to sign up to do this though
6. Confirm and pay the amount listed.

## Checkout 3

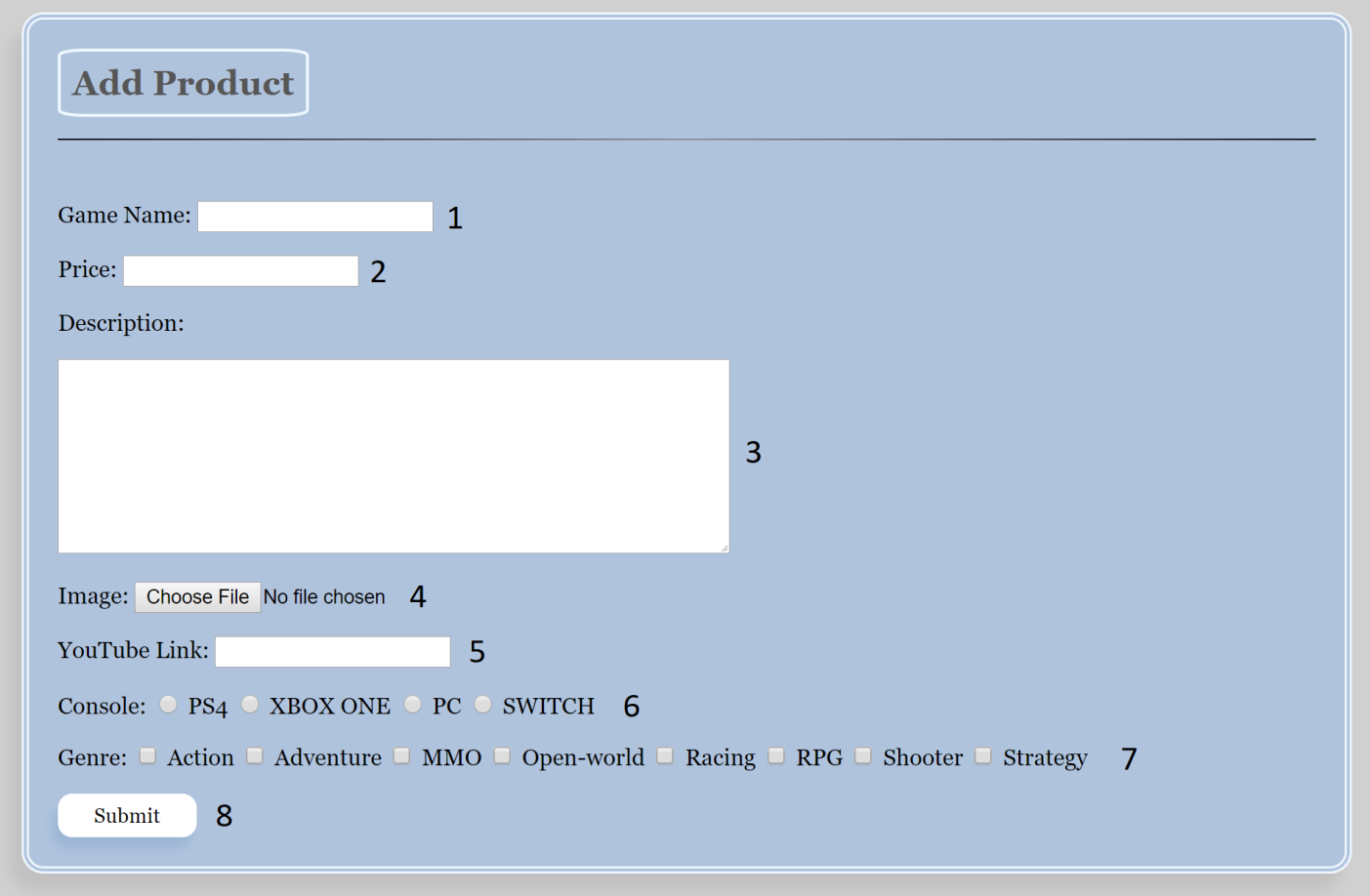


## Order History

****

1. Link to view individual receipt

## Add Product



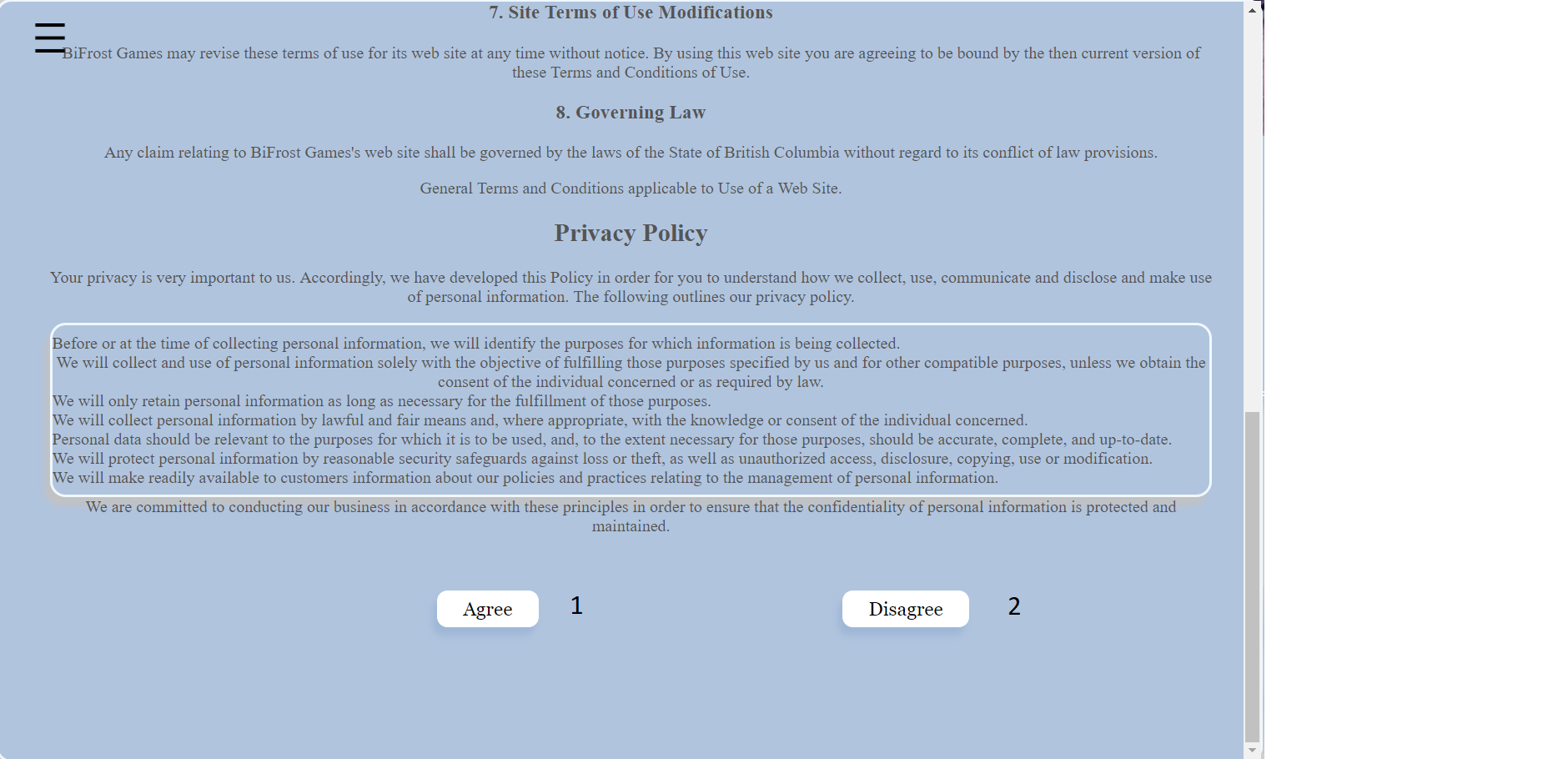
1. Textbox to enter the game’s title – Required
2. Textbox to enter the price of the game – Required and Validated
3. Text field to enter the game’s description
4. Button used to select and upload the cover picture for the game
5. Textbox to enter the YouTube video ID to feature with the game
6. Radio buttons to select the platform for the game – Required
7. Checkboxes to select the applicable genres for the game – At least 1 selection is required
8. Submit button. The info will be added to the database, and you will be given appropriate confirmation messages. If the game already exists for the selected console, it will not be added, and you will be given an error message.

## Footer



1. Opens the Terms of Service Agreement side bar

## Terms of Service Side Bar



1. Agree sets your agreement to yes and closes side panel
2. Disagree sets your agreement to no and logs you out

## Problems and Challenges

Problem – Changing the quantity without reloading the page

Solution – Learned how to use ajax to change quantity with a server-side script. Explained in code snippets

Problem – Git conflicts when pushing your work

Solutions – Ensuring that the most recent version of master was pulled and rebased in branch before work begins

Problem – Git problems with temp files made by Microsoft when file is open

Solutions – Close all windows before pushing to git

Problem – Adopting change requests into "completed" code

Solution – Re-think the code logic to effectively accommodate new and existing functionality

## Skill Learned

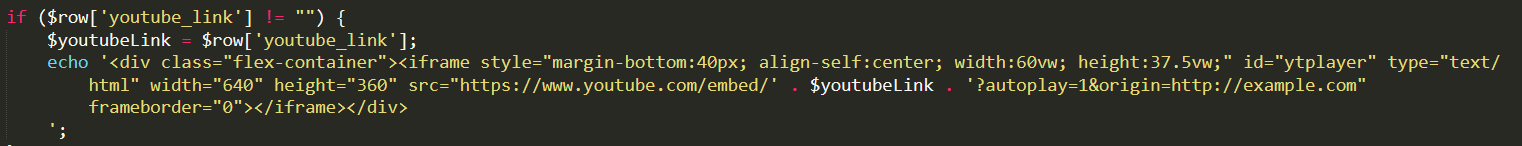
* Learned how to use ajax with jQuery to run server sided scripts
* Learned how to use git properly and how to coordinate the team using it
* Planning and time management
* Re-using code to make tasks easier
* How to work on code together with another teammate.
* How to organize project tasks with MS Project.
* How to design code to be more modular (i.e. using header/footer.php files).
* How to integrate external code (i.e. Stripe).
* How to use php to create dynamic web pages.
* How to use php to interact with a database.

## Proposed Changes

* Plan out the design/pages needed more carefully.
* Next time I would make cleaner code from the beginning and put comments explaining what the code does as I do it rather than later.
* Switch roles more systematically to gain experience with every role.

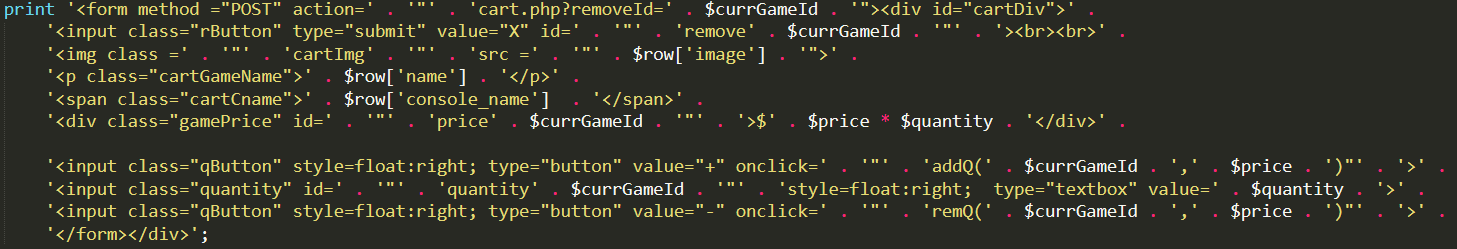
## Code Snippets

### Php loaded YouTube player



This php generates a YouTube player if a video id is inserted into the DB. The video id is retrieved from the database and is inserted into the iframe.

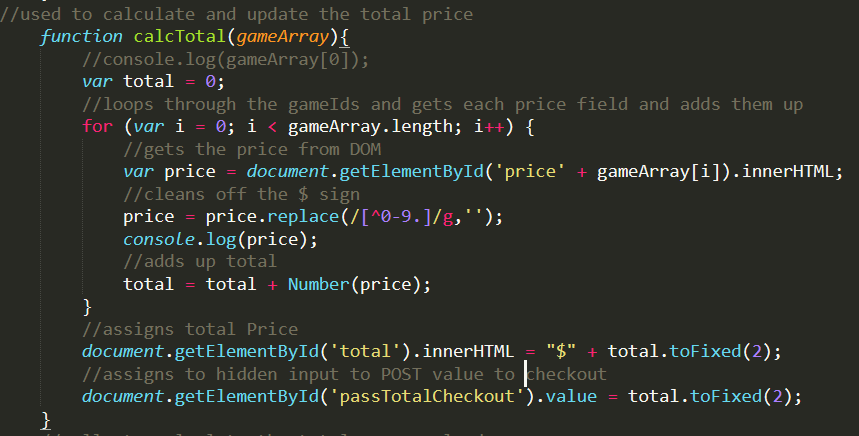
### Updating Quantity



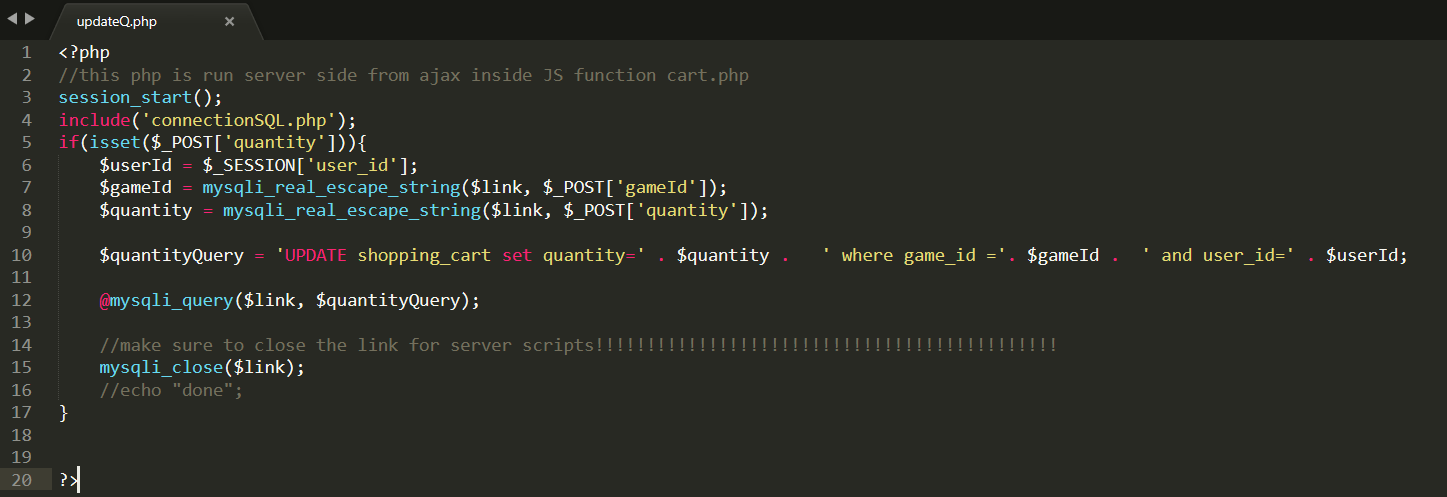
Using dynamic naming with php variables for fields that need to be change made it easy to keep track of and pass variables. Input buttons for quantity call to a JavaScript function as seen below. It passes the game Id and price to the function.



This function uses the game ID and DOM elements to retrieve the current quantity, then it increments the quantity. The price for that item is updated and then a calcTotal() function is called to loop through and add item prices to update the total price. Finally, an ajax function is called that runs a server sided script to update the database.



This function loops through the carts price elements and adds them up to get the total to be updated on the page and passed to a hidden input for POST to cart.



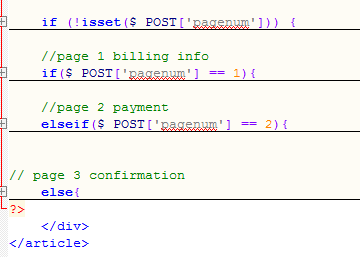
This is the server sided script that is called by the ajax function. This is updating the quantity dynamically in the database without reloading the users page.

### Login Redirect



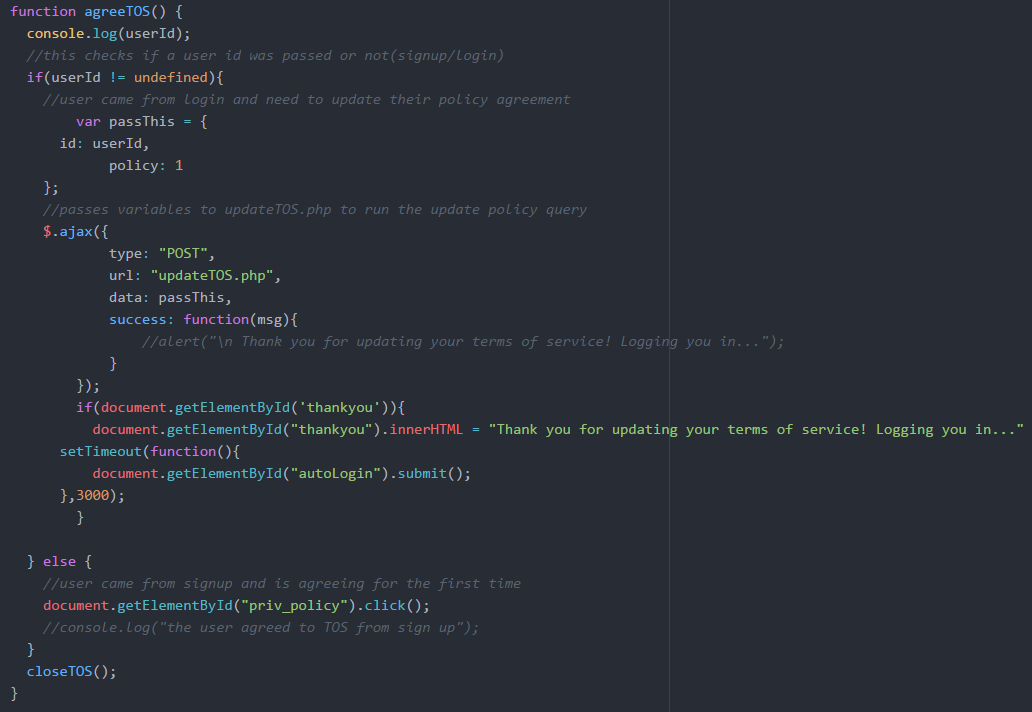
This is a function used on our login page. It is setup to take a URL to direct to, a message to print to the screen, and a value of time, in milliseconds, to wait before redirecting. Different conditions are programmed to call this function with different values. For instance, if a user just created an account, they already have an automatic confirmation and redirection to the login page, so we don't want them delayed for a second time.

### Checkout structure



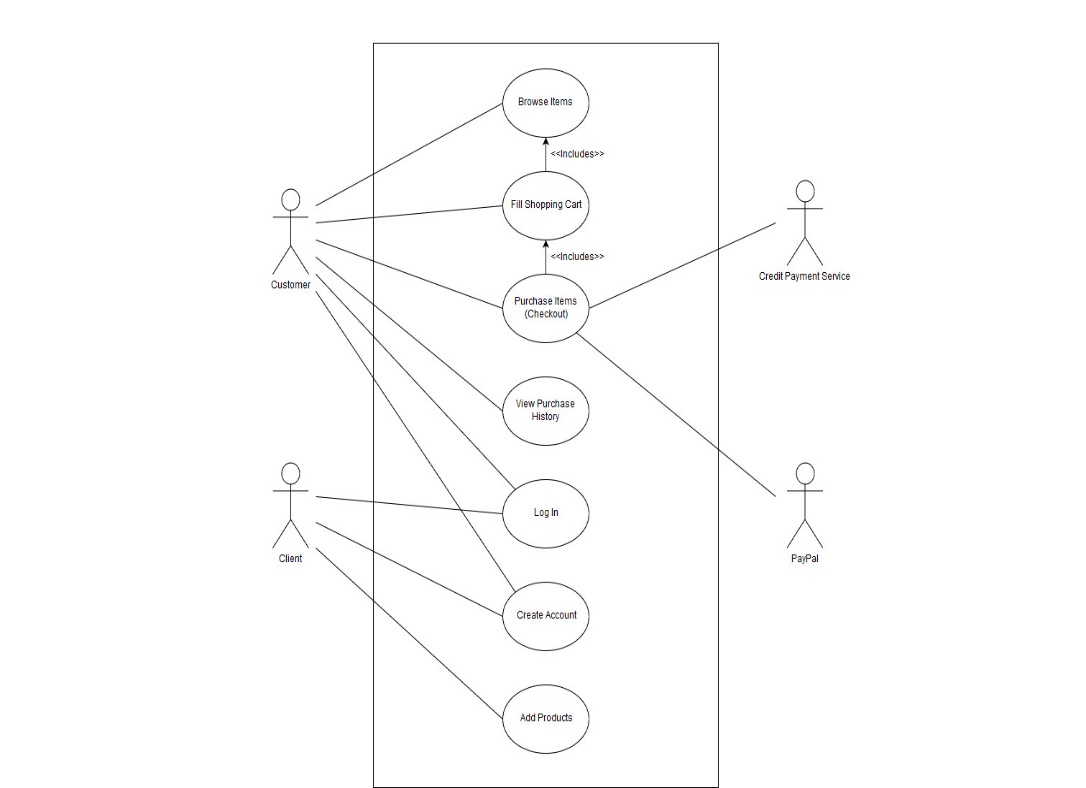
I’m proud of this code because the checkout is only one “page” but it acts like three. Using pagenum, which is set to 1 at first, the if statement checks to see what number it is, since it starts at 1 it will display the first page. When you click submit on the first page, pagenum becomes set to 2 and the page refreshes. The if statement then goes to the elseif checking to see if pagenum is 2, it then displays page 2. After you pay with stripe, pagenum is set to 3 which means the else statement at the end comes into effect showing the confirmation page.

### Terms of Service

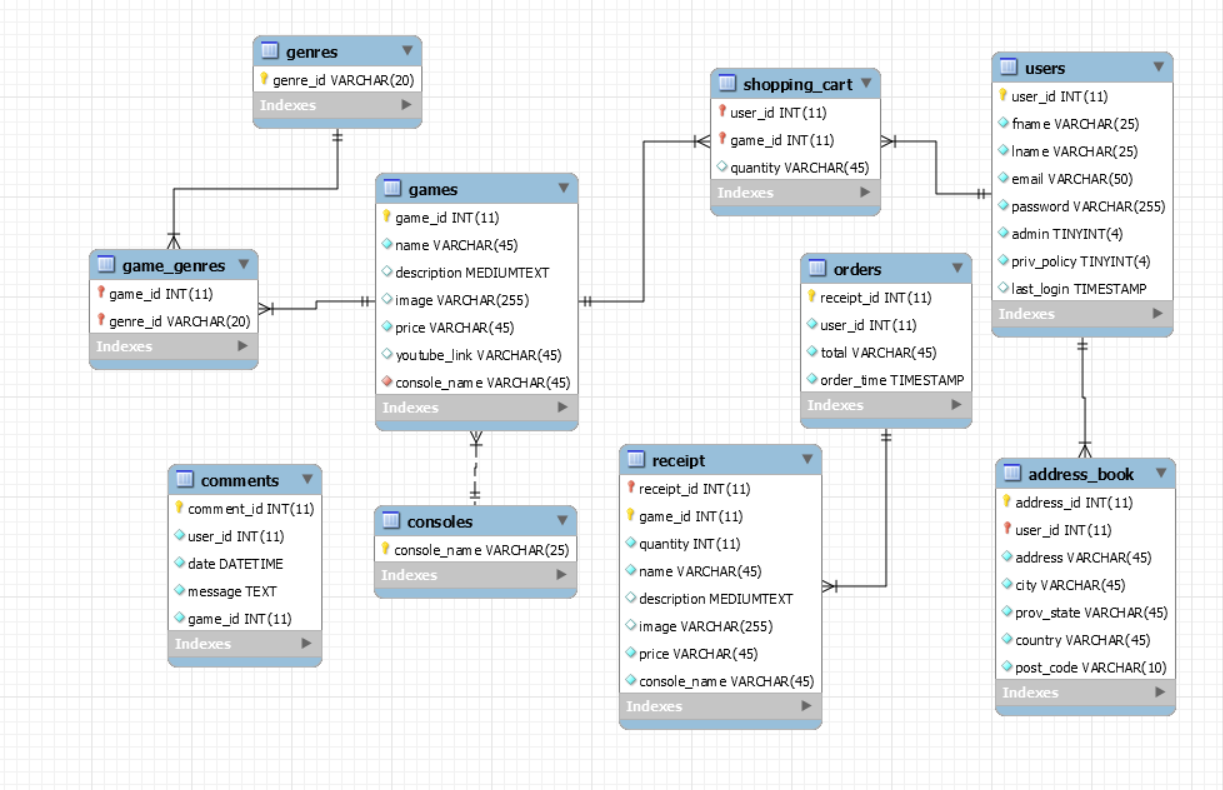


This function is for the terms of service/privacy policy, which we decided to put into the side navigation for a cleaner look. If accessed via the signup page, the agree function will auto click the TOS checkbox for users. The agree function also updates the policy attribute on the back-end using ajax.

## Figures

**

### Figure 1 - UML Use Case Diagram



### Figure 2 – Data Model Diagram



### Figure 3 – Login page



### Figure 4 – Cart page

## Summary

To summarize, BiFrost Games is a fully functional video game retail website that satisfies the list of requirements and expectations outlined for this course (“our client”). With a working sign up, log in, shopping cart, checkout, payment system, order history, and admin privileges; BiFrost Games has over 2,500 lines of code to give it a consistent functionality, clean look, and easy-to-use feel.

Some of the problems and challenges we faced were quite difficult at times, but with persistence and the co-operation of our team, we managed to solve them, typically learning a new skill set at it’s solution. In retrospect, there is a few things we would do differently, and will likely implement them during the next project any of us are a part of in the future.