**CSC 400 Final Report**

**GamerTrade: An Ecommerce Site for Buying/Selling Video Games.**

**By: Anuj Chadha**

1. **Introduction**

GamerTrade is an ecommerce website that allows consumers to sell video games that they already own or buy games from other users on the site. The site will allow users to create an account and either buy and/or sell video games from that account. The site will also have a recommendation system built in that will recommend video games to the user based on their buying history. GamerTrade is a website that aims to focus entirely on buying and selling only video games. I believe this will give users an easy time looking for video games as the site will only have users who focus on selling games and buying games. Sites such as eBay do not focus on selling any one product. eBay sells a vast array of products to their consumers. Our site will be only focused on selling one kind of

1. **Architecture**

GamerTrade will be a website that will be running from a server for users to access. The architecture is client-server, as the client will be the user’s PC that is accessing the website being hosted on the server. The backend of the site will be using flask for programing functions into the site and phpMyAdmin for database management. For the front end I will be using HTML and CSS to create the pages and style them, as well as JavaScript to program the front page.

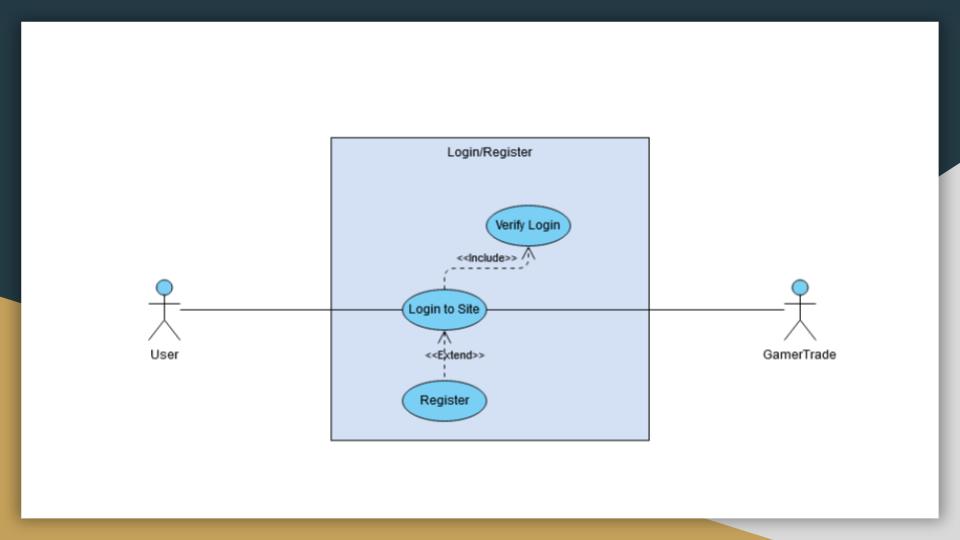
**Major Inputs and Outputs of the system**

**Inputs to website:**

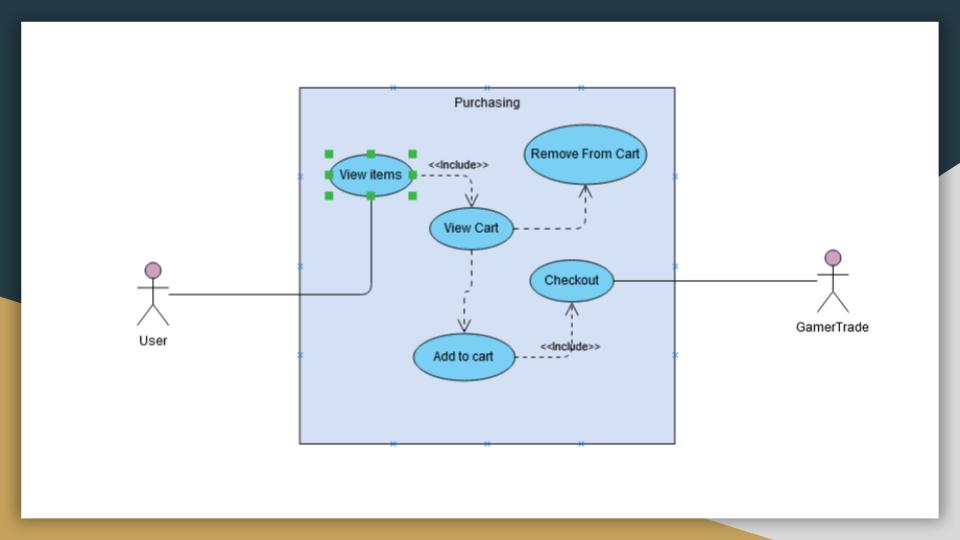
1. From the User: User will be able to input their information to create an account, login, list items to sell, and search for items to buy from the website. They can also use the recommendation feature of the site to get an item recommended to them based on their buying history.
2. From the Website: Website will input information from user to database such as registration information and items to be sold.

**Output to the User:**

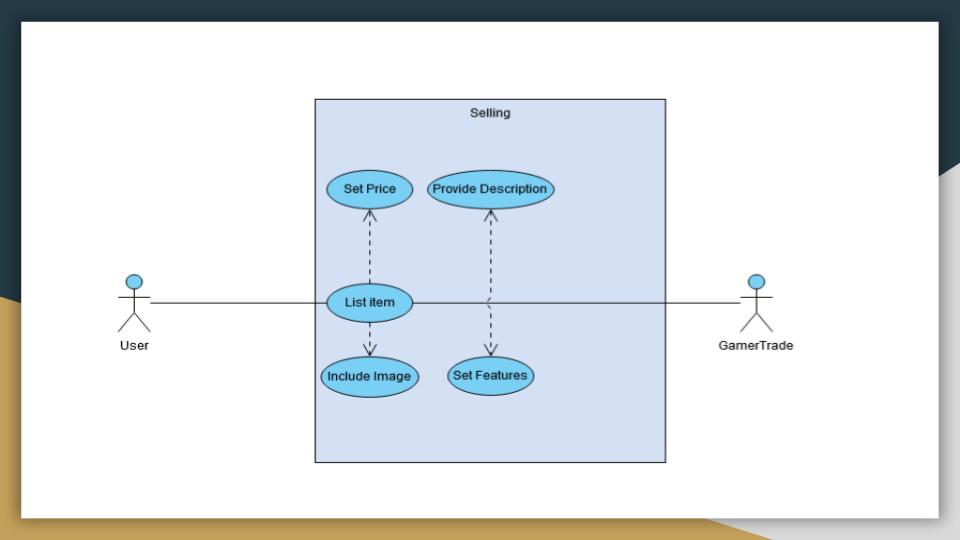
1. From Website to User: Website will display if the login was successful, items user is searching for, items the user is putting up to sell, and a recommendation made from the recommender based on user’s purchase history.
2. **Use Case**



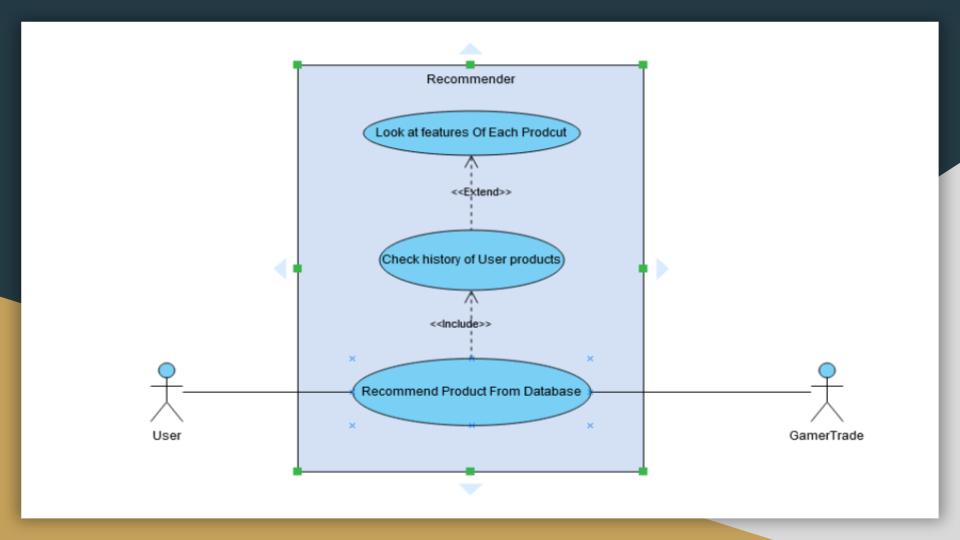
**User registering/logging in**



**User making a purchase**



**User putting item up for sell**



**User getting recommendation**

**Use Cases and Scenarios**

1. **User Accesses Website from domain name**

The user must first access the website to use any of its features. The site is fully hosted with a domain name called, “gamertrade.tk”.

**Scenario 1:**

User can access the website and will end up on the home page.

User can use the domain name and access the website’s home page.

**Scenario 2:**

User is unable to access the website.

User has trouble connecting to the website. This can occur because of issues with their own connection, or problems with the hosting server.

1. **User creates account**

To use the features of the website, users are required to create an account to access most of the site’s features. Users can still search and view item pages, but will be unable to make purchases, sell items, and use the recommendation system.

**Scenario 1:**

User can successfully create an account and is shown on the webpage that they have created an account successfully.

User can create an account and start utilizing the various features of the site.

**Scenario 2:**

User is unable to create an account.

User was unable to create an account because their username is taken, their email is taken, or their passwords did not match.

1. **User lists item to sell**

Users logged in will be able to put items up for sale. They are required to input details of their item to put it up for sale.

**Scenario 1:**

User can add item to sell and it enters the database.

User goes to add item, and from there adds information about their product such as the name, the price, a description, its genre, and an image of the product.

**Scenario 2:**

Item is unable to be added because required fields are not submitted.

User was not able to add for sale because not all the required fields were submitted.

1. **User searches for product**

Users can search for products using the search bar that is located on the top of every webpage on the website.

**Scenario 1:**

User can find item through searching by looking up keyword.

User can use the search bar to search for products. The search bar is located on the top of the page where the navigation bar is.

**Scenario 2:**

User is unable to find item and is told by the system that the item is unable to be found.

If the phrase the user inputted into the search bar is not found in the database, it will report to the user it was unable to find anything.

1. **User removes item from their selling list**

Users can remove items that they have put up for sale.

**Scenario 1:**

User can find the items they want to delete and can click on delete to remove it from the item database so it will not be shown in search.

The user will be able to see all items that are up for sale and click remove to delete those items.

**Scenario 2:**

User is unable to find their items because they either have no items listed for sale, or have an error connecting to the database.

User will be unable to delete any items because they have no items up for sale.

1. **User makes a purchase**

Users who have an account and are logged in can make purchases on the site.

**Scenario 1:**

User can see the purchase button and provide purchase information after clicking on purchase and finalize their purchase.

User should be able to see a purchase button when they are on the view page of the item they want to buy. After clicking the button, they should be sent to a page to input their payment information and make the purchase.

**Scenario 2:**

User is unable to see the purchase button because they are not logged in.

User is unable to see the purchase button because they are not logged in and therefore cannot make any purchases.

1. **User checks selling history**

Users can check all items that have been sold on the website and see who purchased the item, and where it needs to be shipped to.

**Scenario 1:**

User can see all their games they are trying to sell on the site.

Shows the items that have sold on your account. It displays the details on the item as well as the user who purchased the item and their shipping address.

**Scenario 2:**

User cannot see their list of games because they have no games that have sold.

User cannot see any items sold because they have not sold any items.

1. **User checks purchase history**

Users can look up their purchase history and see the details of the item they bought.

**Scenario 1:**

User can see their purchase history.

User can see each game they have purchased as well as details such as price, genre, product name, description, and the user you purchased the product from.

**Scenario 2:**

User is informed they have made no purchases and have no purchase history.

User has not made a purchase and because of this is unable to see any items. They are displayed a message that they do not have any purchases.

1. **User uses the recommendation feature of the site**

Users can obtain a recommendation from the site based on their buying history.

**Scenario 1:**

User can receive a recommendation from the site based on their purchasing history.

User clicks on recommendation from the user profile and are given a recommendation that is limited to 3, based on their most purchased genre, and the average price of that genre.

**Scenario 2:**

User is unable to receive a recommendation because they do not have any games purchased and are informed to make a purchase.

User is informed that they do not have any purchases and are unable to receive any recommendations.

**Scenario 3:**

User is unable to receive a recommendation because there are no games in the database that fit their recommendation profile. They are given genres to search so they can make a purchase.

User is informed of their most purchased genre and average paying price for that genre but are unable to receive a recommendation because the database does not have any games that fit their recommendation profile.

1. **Searching by genre**

Users can search by genre and see all games in our database depending on the genre.

**Scenario 1:**

User can click on the picture of the genre on the home page and search for all games in the database that are in the items database in that genre.

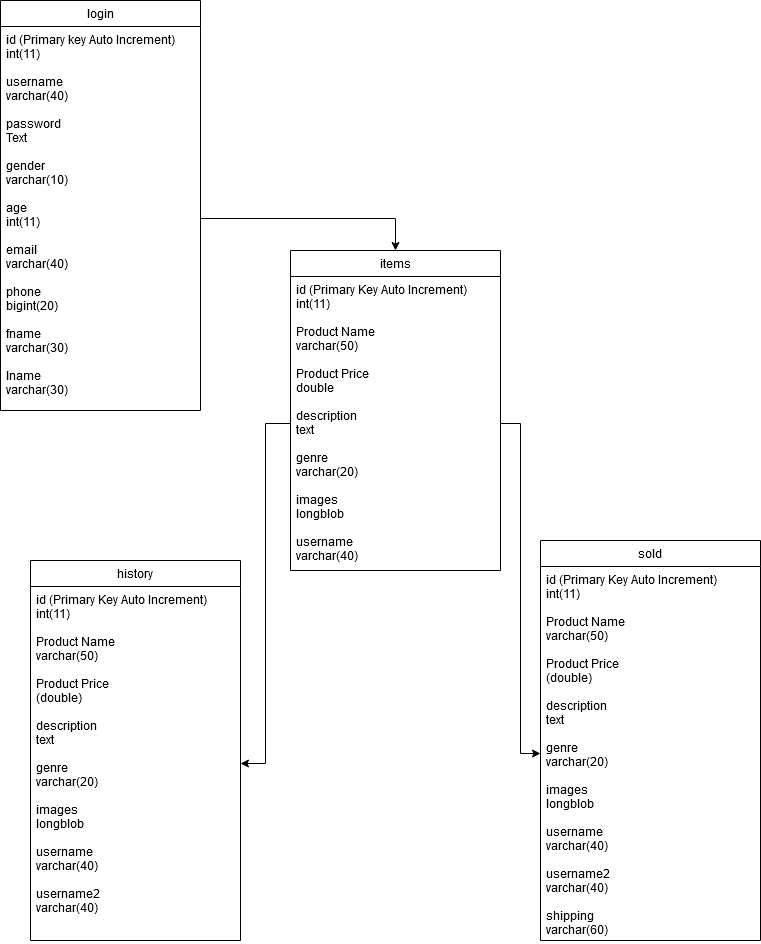
When a user clicks on one of the images on the homepage, they can see all the items on display with that selected genre.

**Scenario 2:**

There are no games that are in the database for that genre.

If there are no games that in the database that fit the genre it will not display any games.

1. **Structural Design**



**UML diagram of website**

1. **Data**

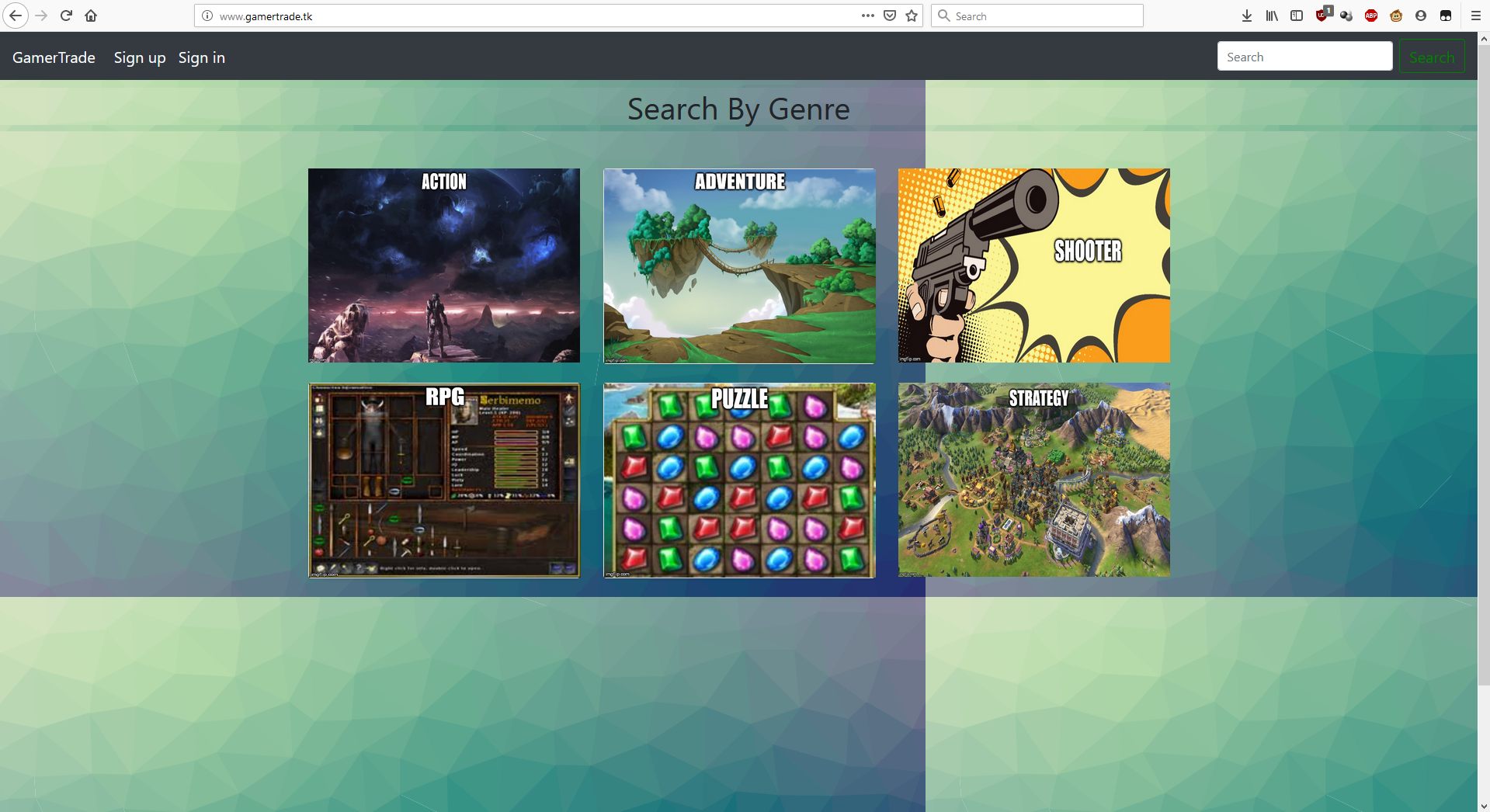
Data for the website is stored inside of the gamertrade database. All persistent data being used by the website is stored into the database into different tables. Queries are executed when they are called from the PHP file. To fully create the website, I needed to use twenty-nine PHP files and one four tables in our database, and eight images are used in the website. When data is called, we store it into an array and execute it depending on its position in the array. Most of the site is run on the back end of the server. PHP files are used to display the front end of the site, the HTML/CSS, and PHP is used to execute queries that display the data that is stored in the database.

For the twenty-nine files, six of the files are used to search based on genre. For example, one files will perform a search just for action games, and another file will perform a search for only puzzle games. One of the files is used to keep track of the user’s session and store that into a variable. Three of the files are used as HTML forums but are being treated as PHP files so that the search bar functions, and the nav bar will change based on if the user is logged in or not. The rest of the files are PHP scripts that perform queries and display information once clicked on them, such as the recommendation.php, remove.php, or itemadd.php.

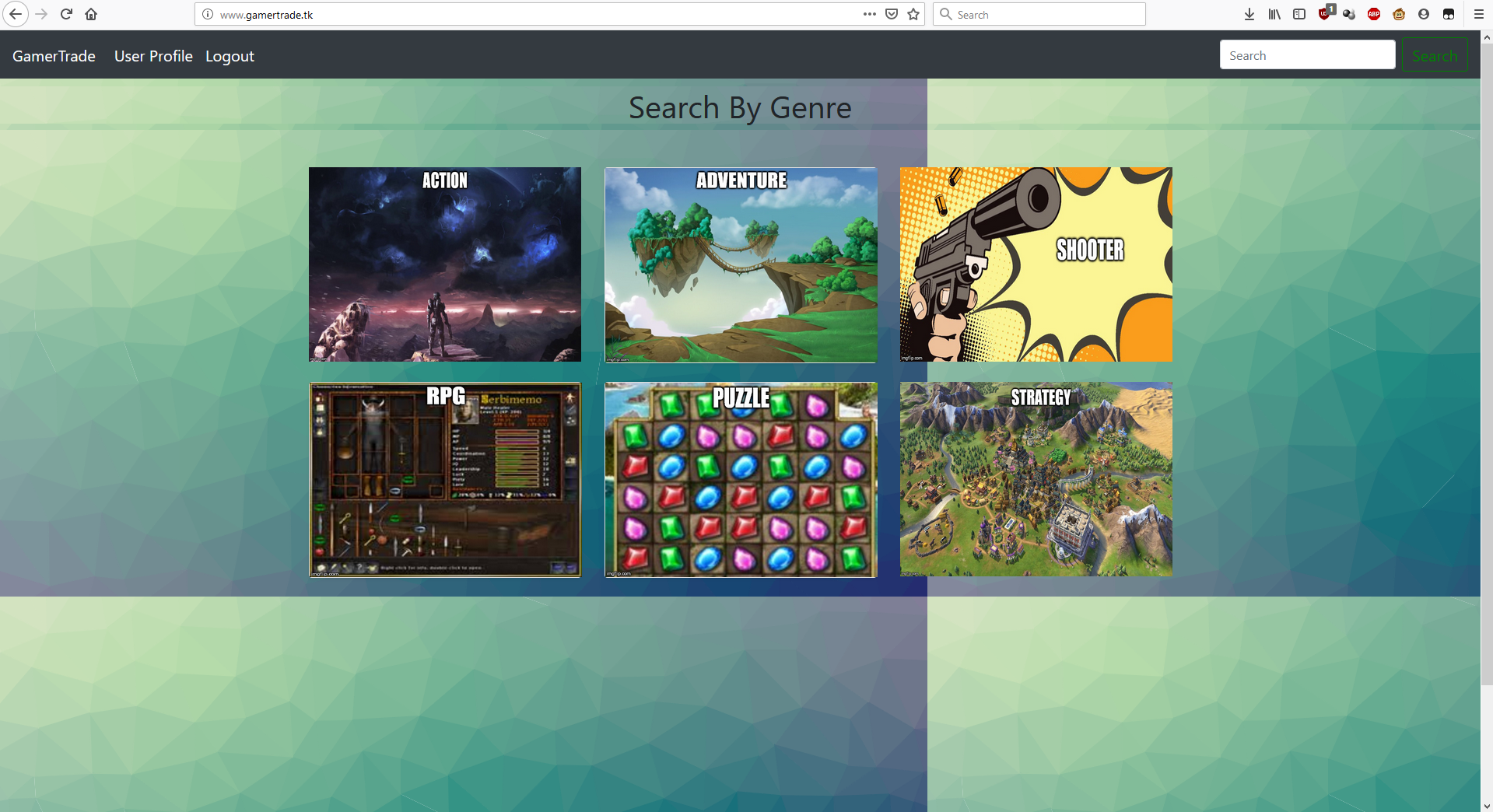
1. **Implementation**

The core algorithms of this website were the home page, genre searching, login/registration, recommendation system, user profile, add/remove items, purchase items, and view purchase/sold history. The next section will cover the code used for each of the features and how they were implemented.

1. **Home Page**

****

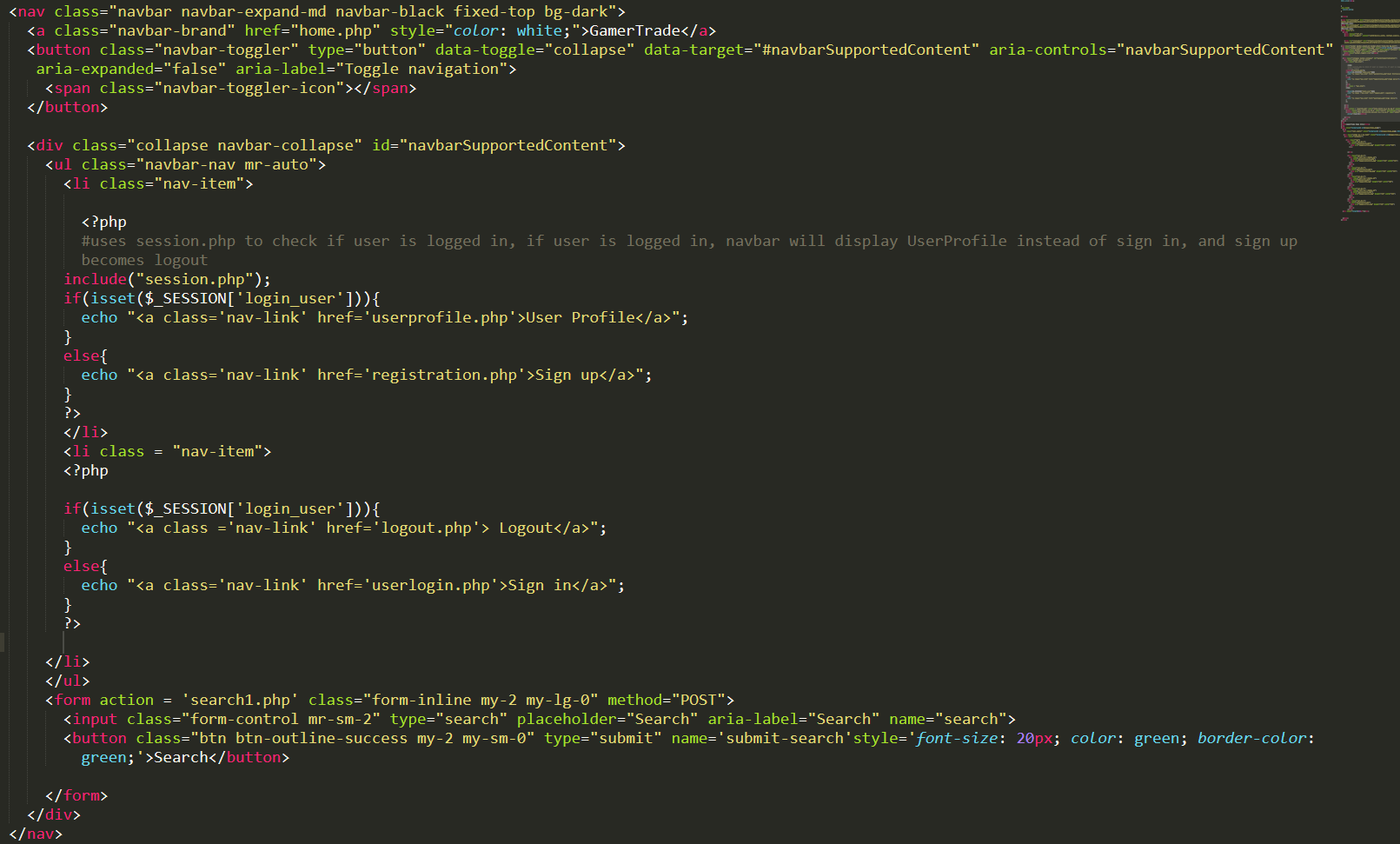
**User not logged into system**

****

**User logged in**

The homepage for GamerTrade was made to be user friendly and allow the user to easily access the core features of the site. To do this, Bootstrap, CSS, HTML, and PHP were needed to make a user-friendly experience. The figure below will show the home page of the website when the user is not logged in, and the picture below that will show the homepage when the user is logged in.

The home page for the site changes based on if the user is logged into the website. This is done by checking if the user is logged into the session. If the user is not logged into the website, sign up and sign in button are shown on the navigation bar. But if the user is logged into the session, the navigation bar will instead show user profile for sign up, and log out, for sign in. Clicking on user profile will redirect the user to their user profile and clicking on the sign out will destroy the current and log the user out.



**Code for navigation bar**

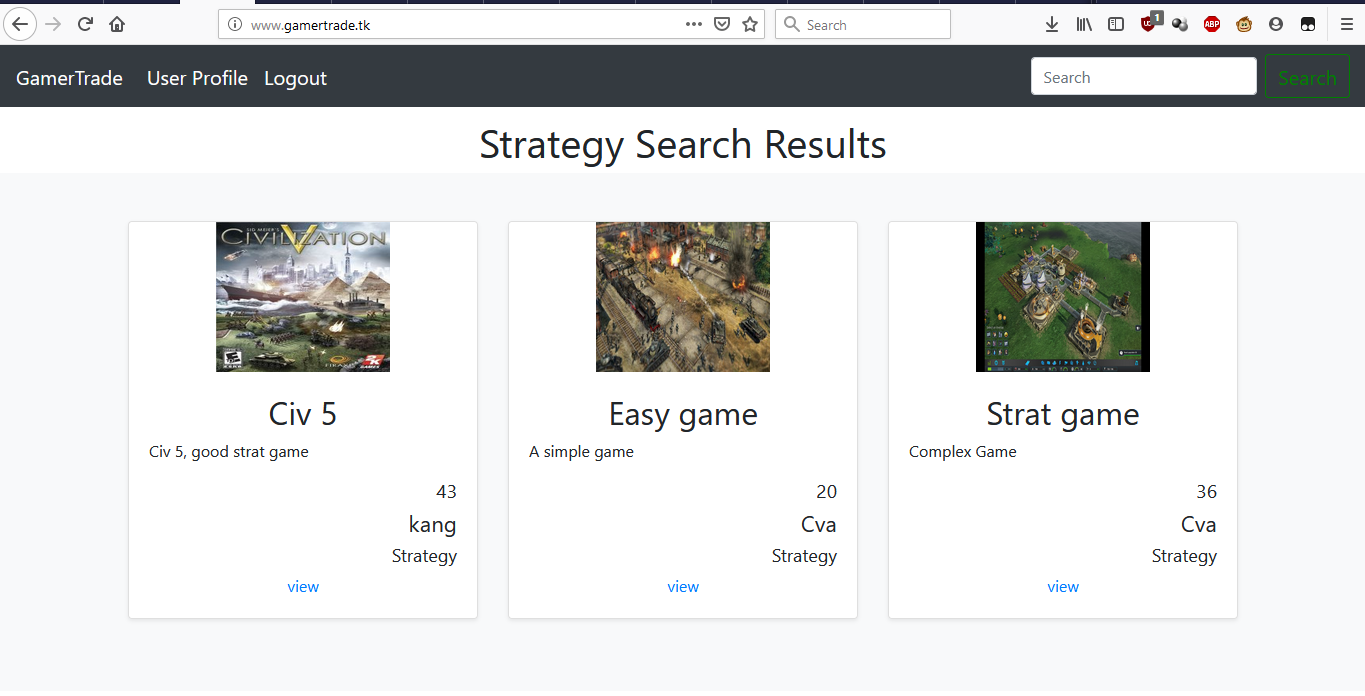
The image above is the code used for the navigation bar of the website. The navigation bar was made using Bootstrap. However, the navigation bar was added to every made to make sure the website was neat looking and more professional. PHP was used to make the navigation bar change depending on the user being logged in or not. This is done by checking the current login session. If there is a user logged into the session, show user profile and redirect the user to the user profile page when they click on it as well as change login to logout. If the user is not logged in, display sign up or sign in so the user will be able to create an account, or login to their existing account. The navigation bar also contains the search bar. The search bar works by taking the string the user inputted and putting it through a form that is sent to the search1.php form.



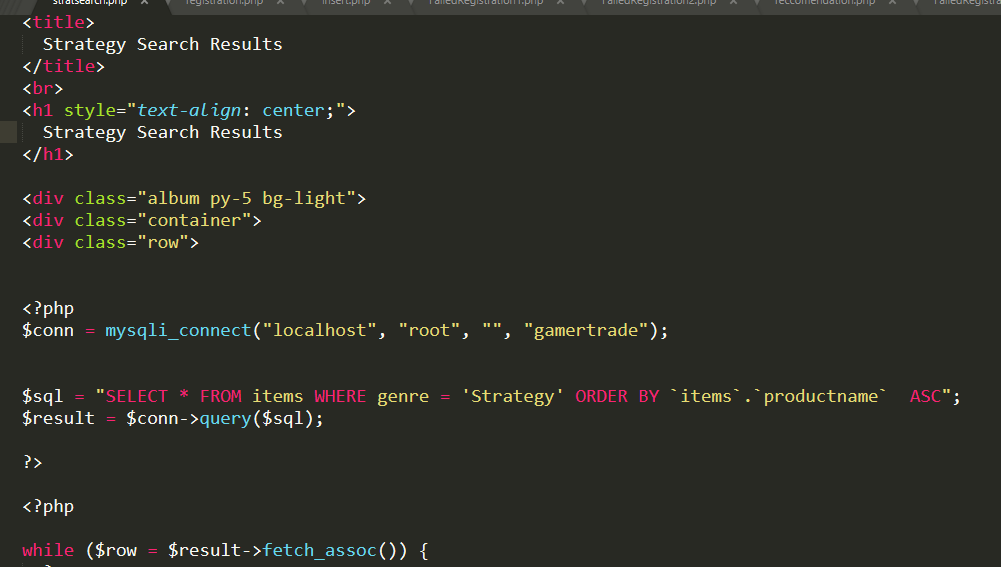
**Code for homepage**

The image above shows the code used to create the homepage’s feature to genre search and the background. The background is done by using CSS to take an image and use that for the background. The genre searching is done by using Bootstrap’s class to create a container and separate the images on the homepage by rows and columns. Within each image we set the image to the size of 250x350 and make each image a hyperlink. By clicking on the image, you are able to search by genre. This is done six times for each genre that our site supports, and each genre has its own PHP script to search for the genre.

1. **Genre Searching**



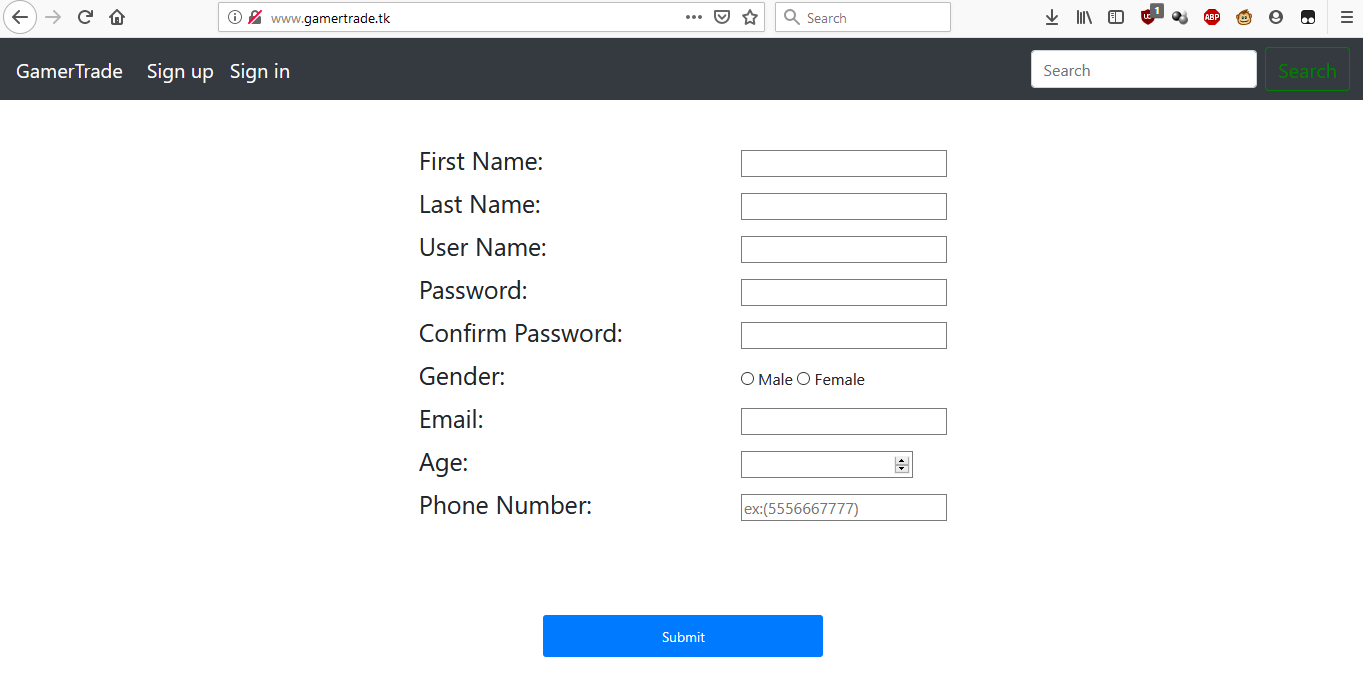
**Search results of searching by the strategy genre**



**Code for searching by genre**

The image above shows the search results of searching by genre from the homepage and the image below that shows the code used. To perform a genre search, the user must first click on the image they want to search. This calls the PHP script that will select all the contents in a row within the items database where the genre matches the genre clicked on by the user. For this example, it shows “Strategy”, but depending on the script called it will change the genre called. For example, if it was searching for all action games the genre would change to “Action”. After performing the query, a while loop is used to display the results of the search.

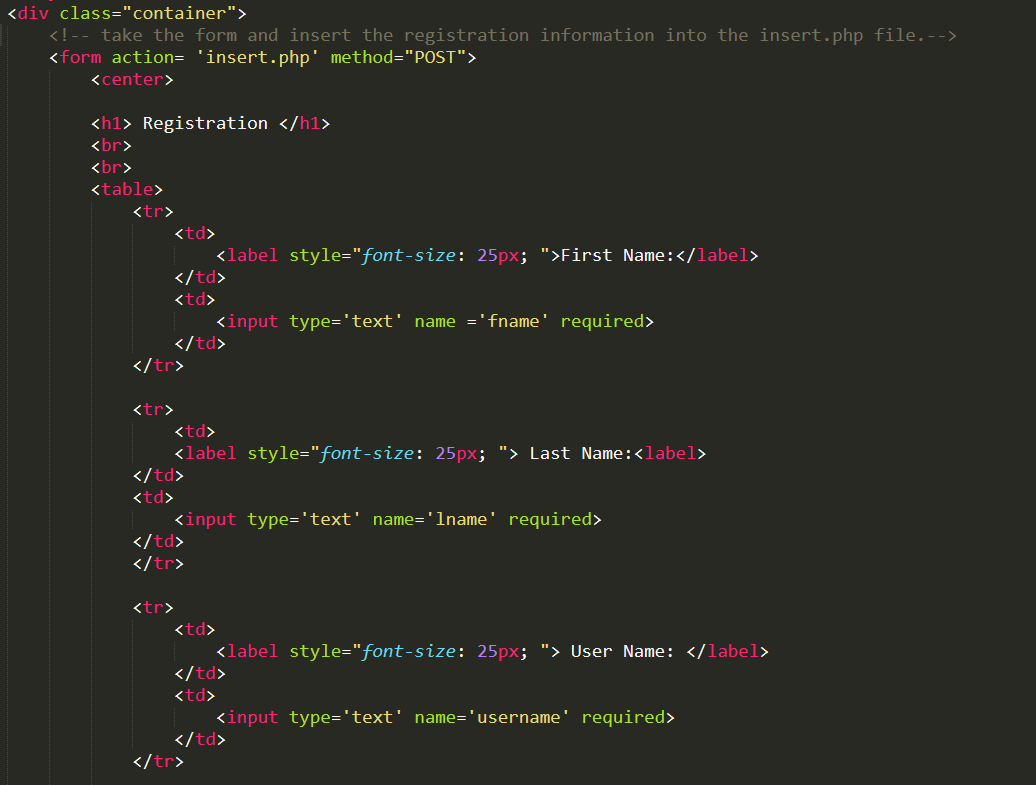
1. **Registration/Login**

****

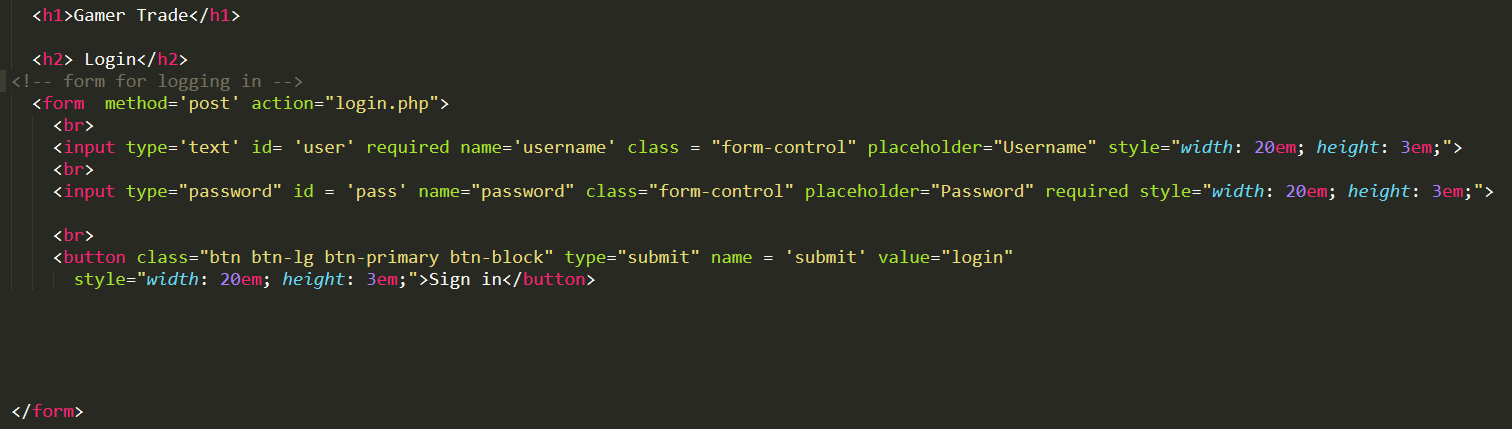
**Registration form**



**Login form**

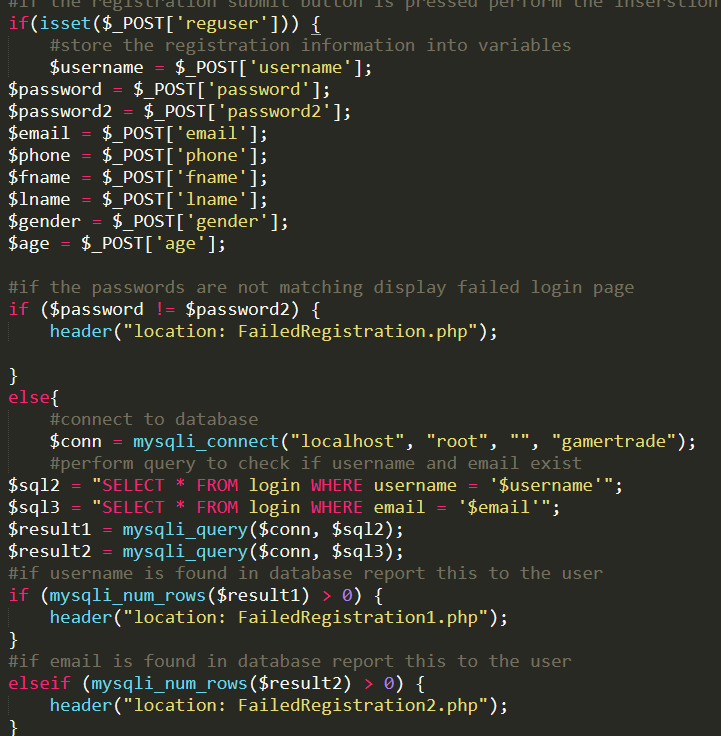
****

**Code for registration form**

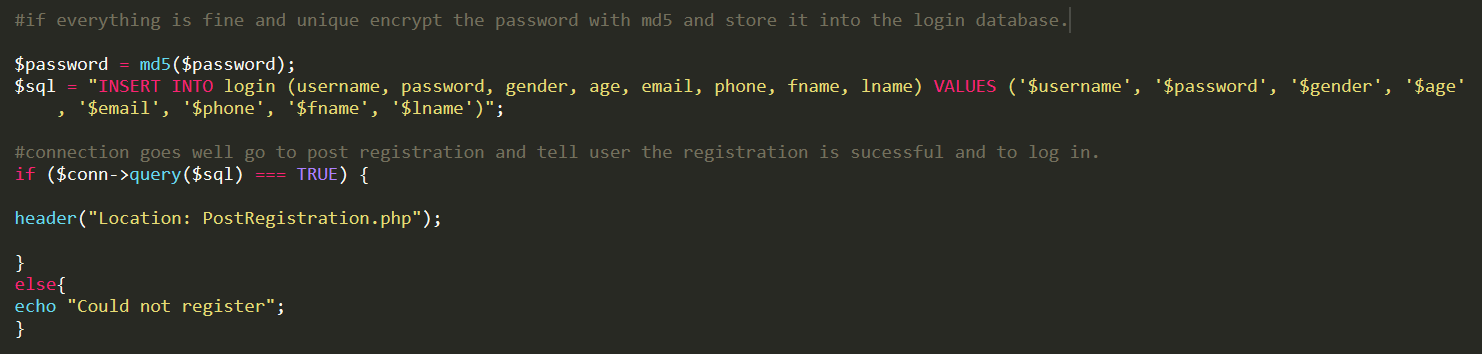
****

**Code for login form**

The images above show the login and registration forms of the site and the code for each form. The forums are made using HTML and CSS and sent into a form that is then processed using PHP and inserted into the login table in the database. To make sure data is being inserted into the table, the required HTML tag is used to make sure the submit button will only work if the form is filled out.

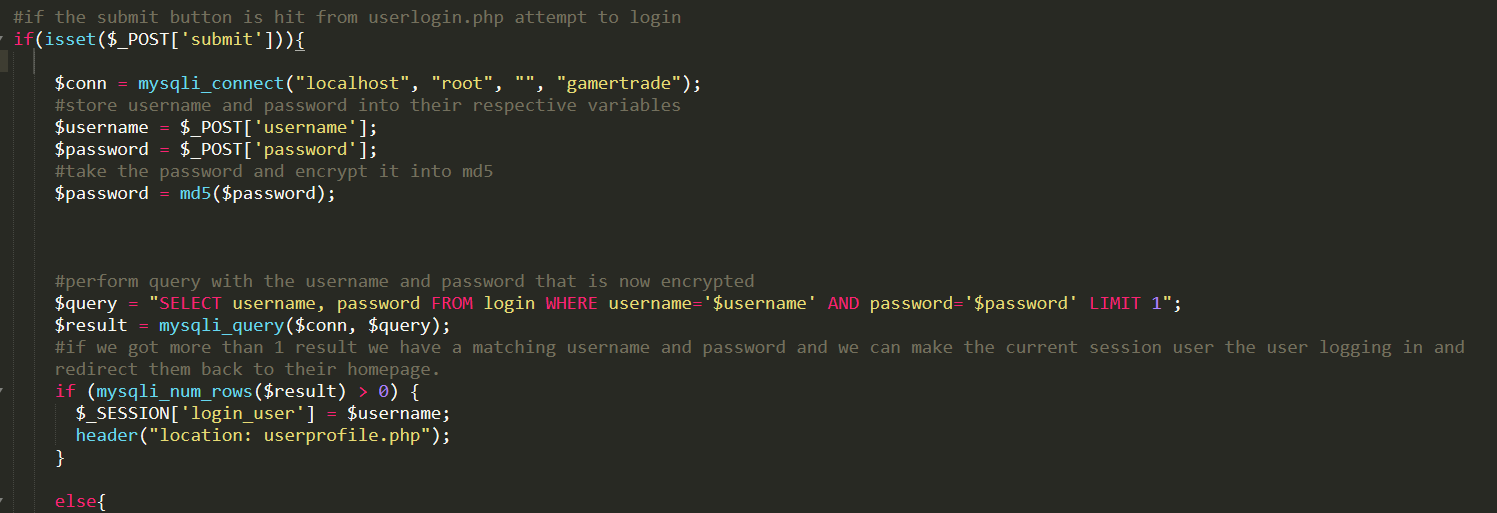


**Code for registration (1)**

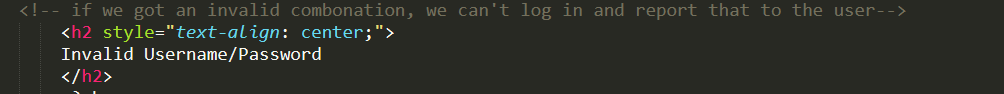


**Code for registration (2)**

To finalize the registration for the user the values put into the form are sent into the PHP script and stored into their own variables. A check is done to see if the passwords match. If the passwords do not match, the user is redirected to a different page and tells them that the passwords do not match. If they do match, a query is performed to see if the username and email are taken. If they are taken the user is redirected to a page and told they must choose a new email or username. If both the username and password are not taken the password is encrypted using PHP’s built in function with md5 and inserted into the database. The user is then redirected to a page informing them they have successfully made their account.



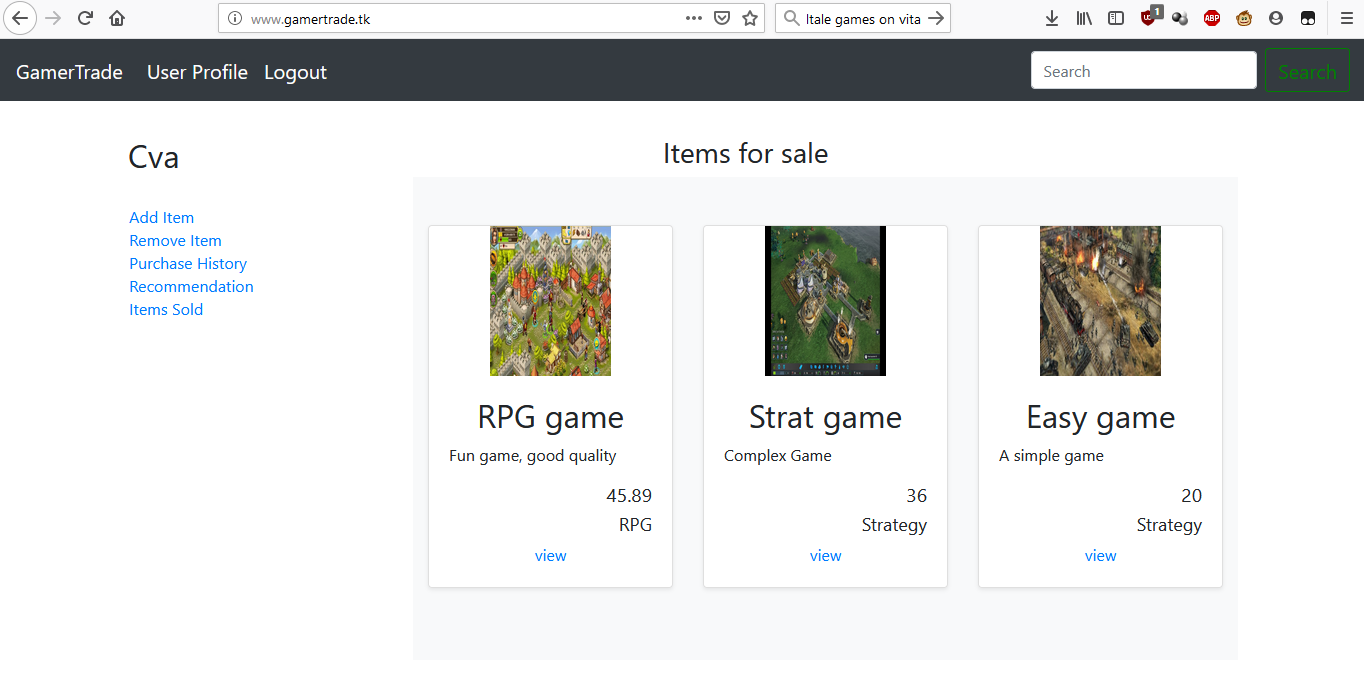
**Code for login (1)**

****

**Code for login (2)**

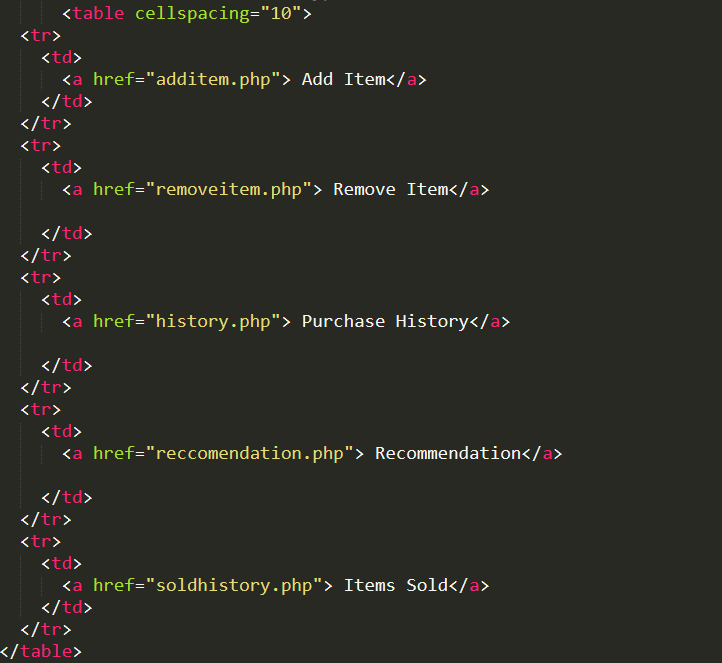
To log the user into the website the information the user inputted into the form is put into the php script. The username and password are stored into their own variables and the password the user used for the login is also encrypted with md5. This is done because the password is stored in the database encrypted and not in plain text so it must be encrypted to compare them to see if it was valid. If the result obtained in the query is more than one store the username into the session and redirect the user to their profile.

1. **User Profile**

****

**User profile**

The image above shows the user’s profile. The profile page shows the items the user is currently selling, allows the user to check out their purchase pages, as well as access the features of the site such as adding, removing, history for buying, history for selling, and recommendations.



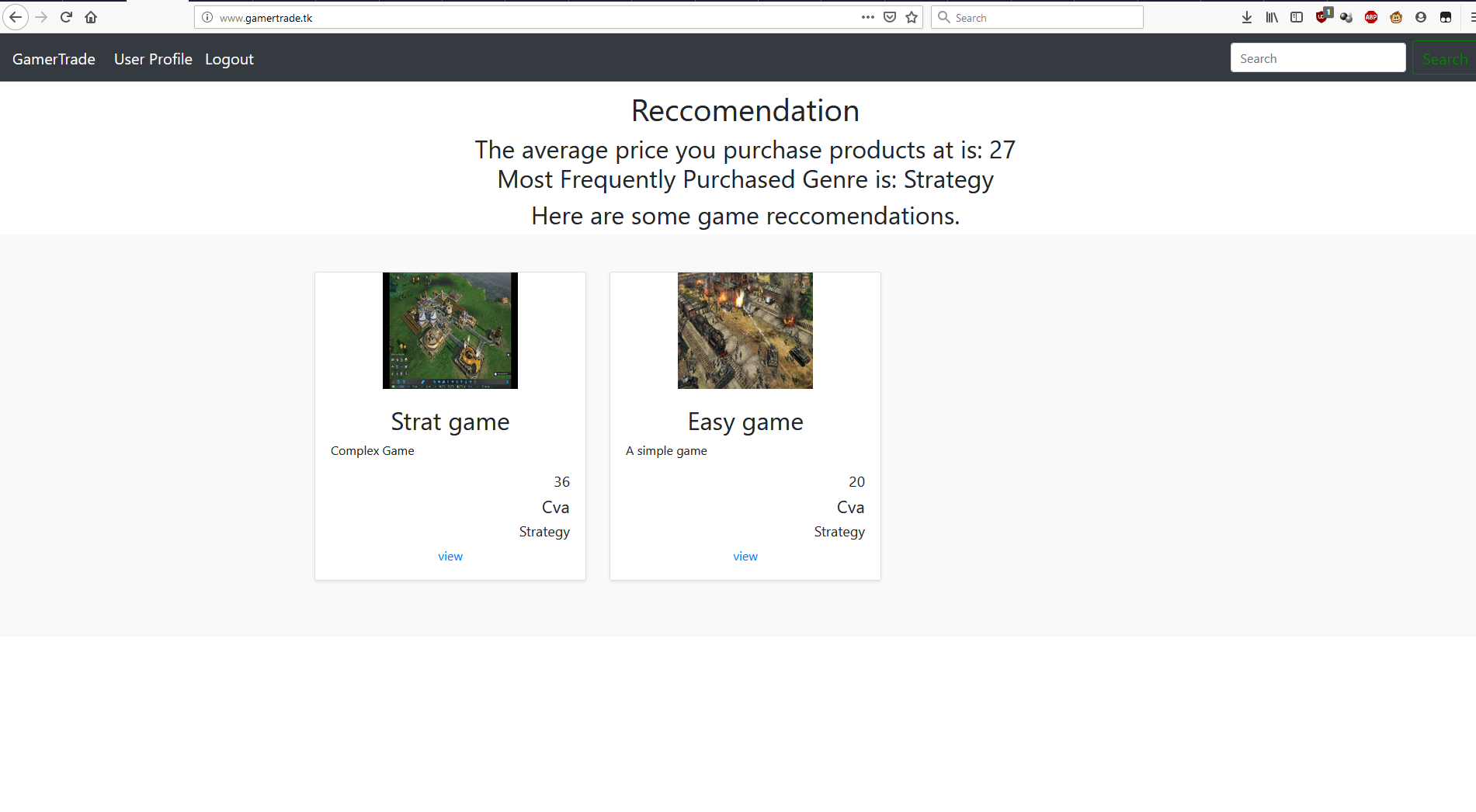
**Code for user profile (1)**



**Code for user profile (2)**

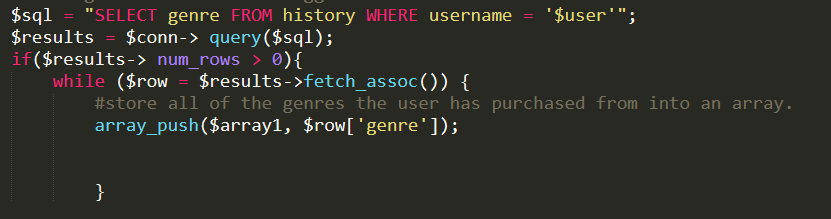
The first image of the code shows the code used to show the table that allows the user to access the features of their user profile from different links. The second image shows the code used to display the items in the database that the user is selling. This is done by performing a query and showing all the rows where the username matches the logged in user.

1. **Recommendation System**

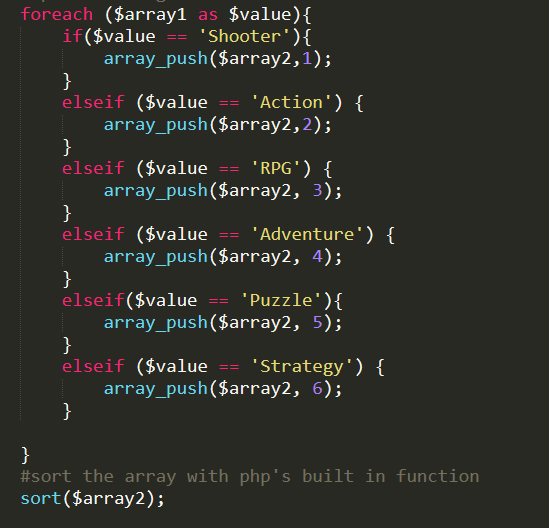


**Recommendation system providing a recommendation**

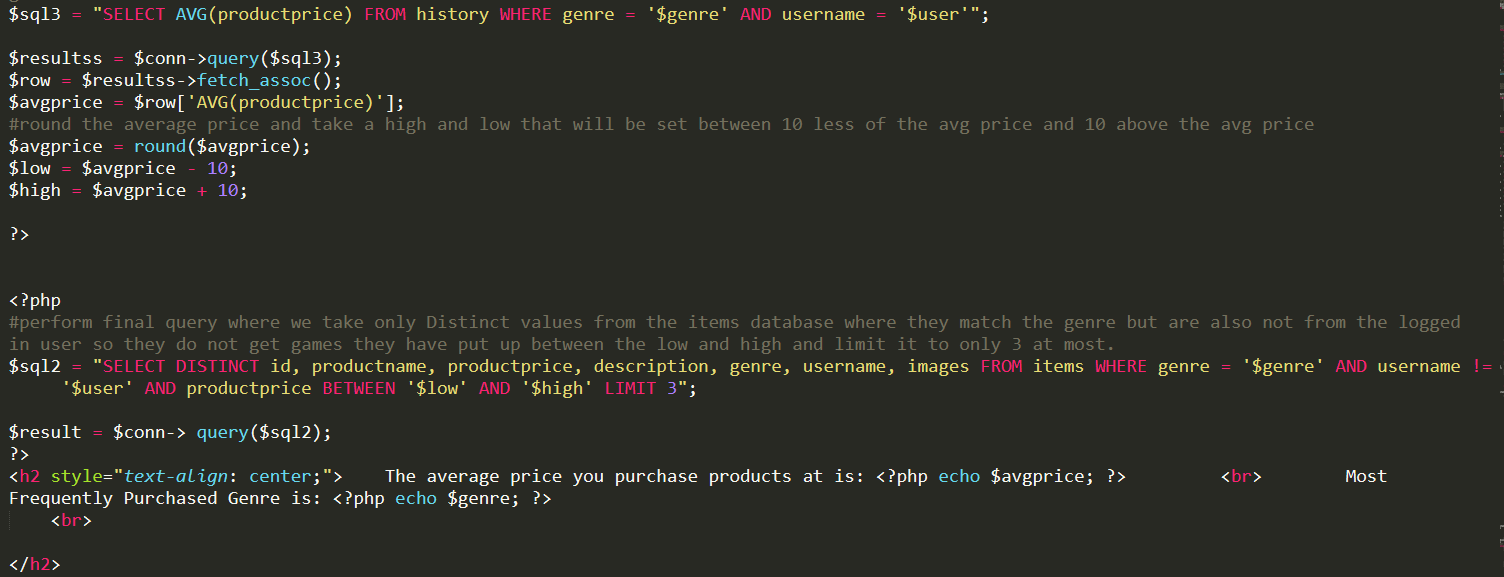
The picture above shows a user getting a recommendation based on their purchase history. The user is given the recommendation based on the average price of their most purchased genre. The recommendation is limited to three games at most and they cannot be recommended games they have put up for sale.



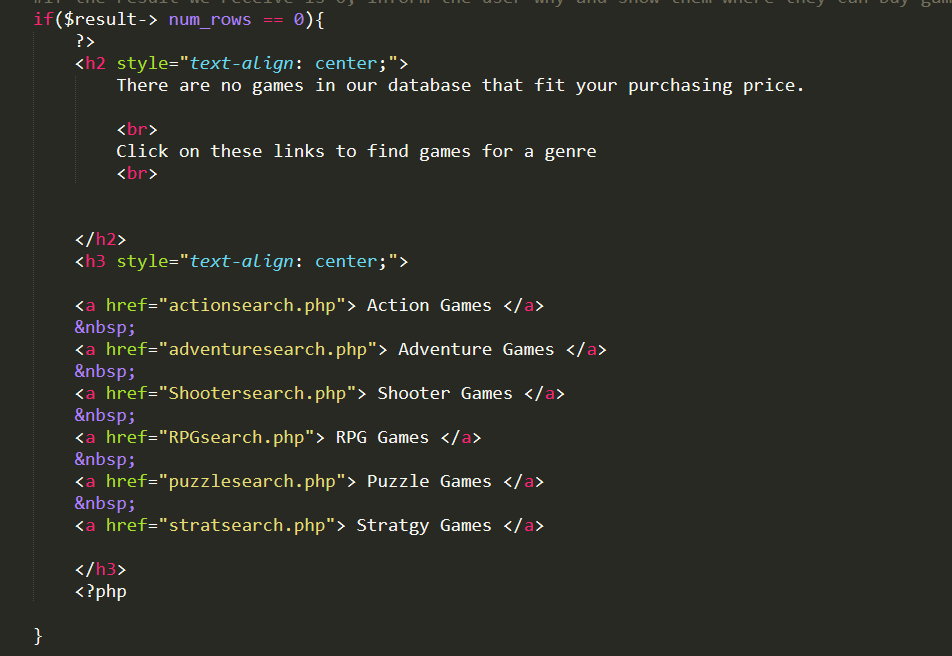
**Code for recommendation system (1)**



**Code for recommendation system (2)**



**Code for recommendation system (3)**

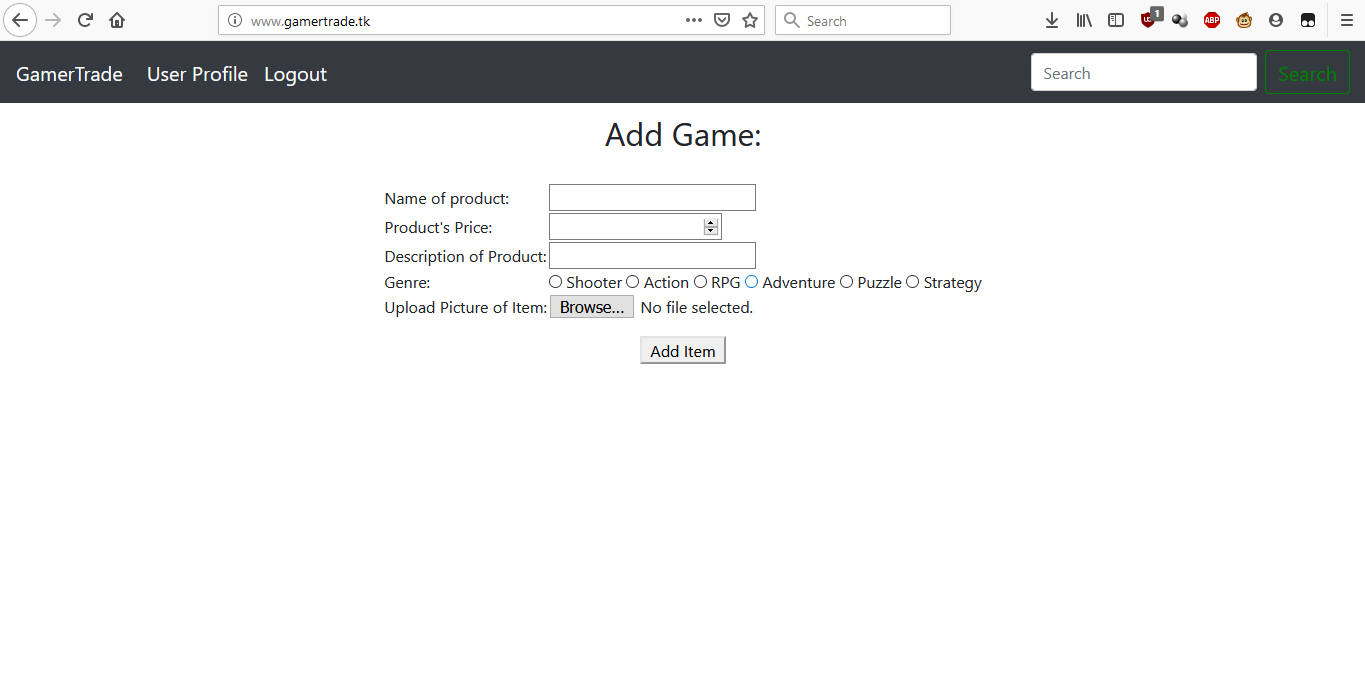


**Code for recommendation system (4)**

The code above is the code used for the recommendation system. The code works by taking the logged in user and looking for the genre in the history table. If at least one result is found from their purchase history, the genre is stored into an array. After being stored into an array, the genres are converted to an integer based on their genre and stored into a different array and sorted numerically. After they are sorted the array is counted using PHP’s count\_values() function to get the most frequently purchased genre.

After obtaining the most purchased genre a query is performed where the average price is taken based on the most purchased genre from that user. The high and low are the price are the average price reduced by ten and added by ten. A query is performed after that that takes the distinct product information based on the most purchased genre for items that are not being sold by the logged in user and the price is between the low and high of the average buying price. If there is nothing in the database that fits the user’s recommendation profile, show the user where they can start buying games.

1. **Add/Remove Items**

****

**User adding an item**

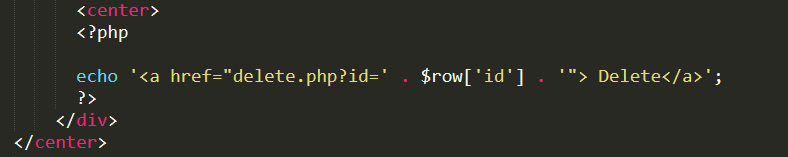
****

**User removing an item**

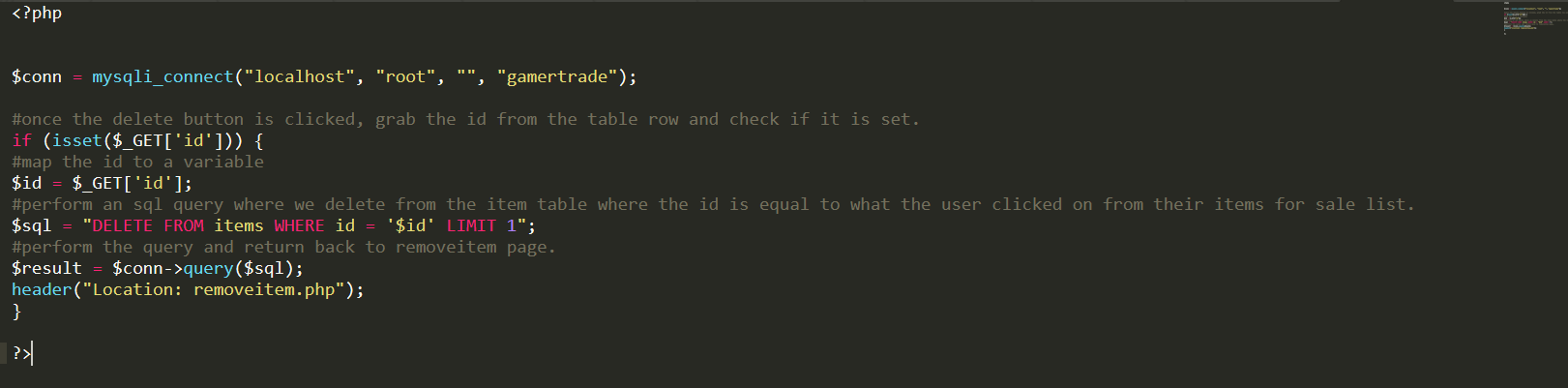
The images above show the user adding an item and, on the page, to remove any item they have put up from the database. Adding items are done through a form that takes the information of the product and insert that information into the items table. Removing items is done through a PHP script that shows all the items the user has for sale and if the user clicks delete on the item they want to remove, the item is deleted from the database using the delete.php script.



**Code for adding items**



**Code from remove item form**

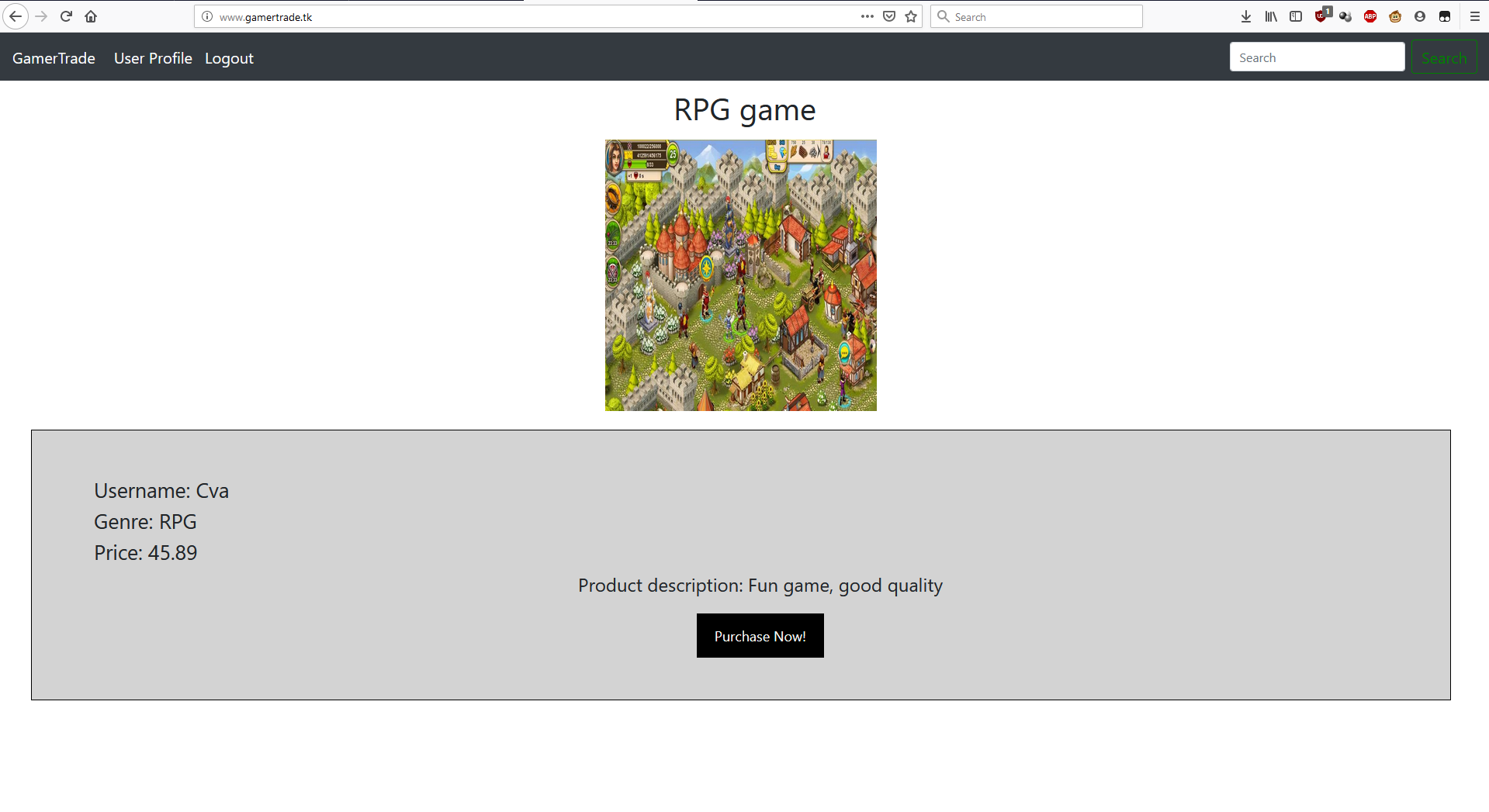


**Code for removing items**

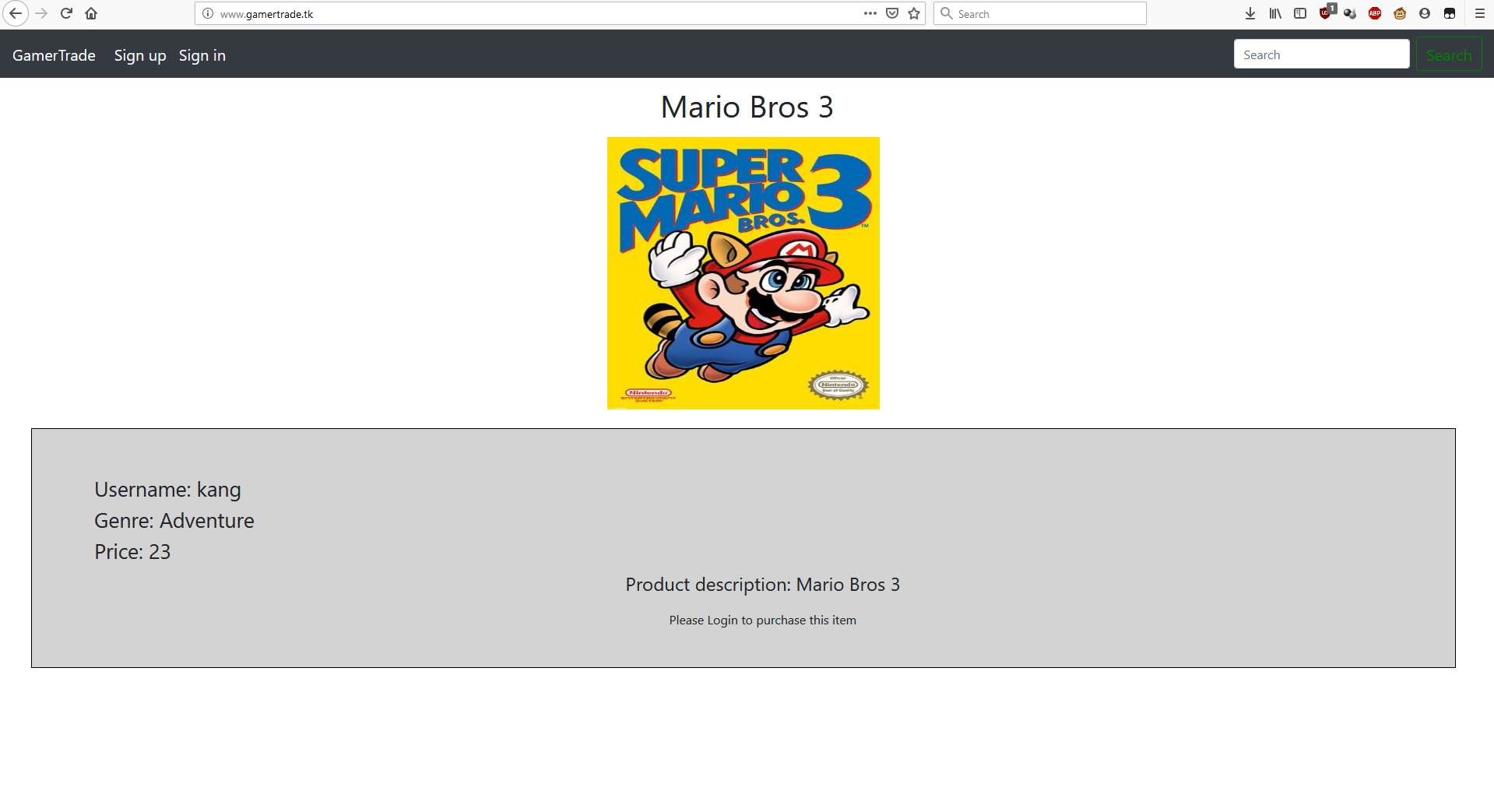
The first image shows the code used for adding items. The information from the HTML form are stored into variable. The image data has slashes added to it and is stored as a file type. If the database can connect it will insert the information into the database and redirect to the user profile.

The second two images show the code used for deleting items. From the remove item page, when the user clicks delete, it takes the id of the item and sends it to the delete PHP script. This script is the second image and that takes the id and performs a query that deletes from the items database from the id that was taken from the remove item page. After deleting the item, it redirects the user back to the remove item page. The item should then be missing from the page because it was removed.

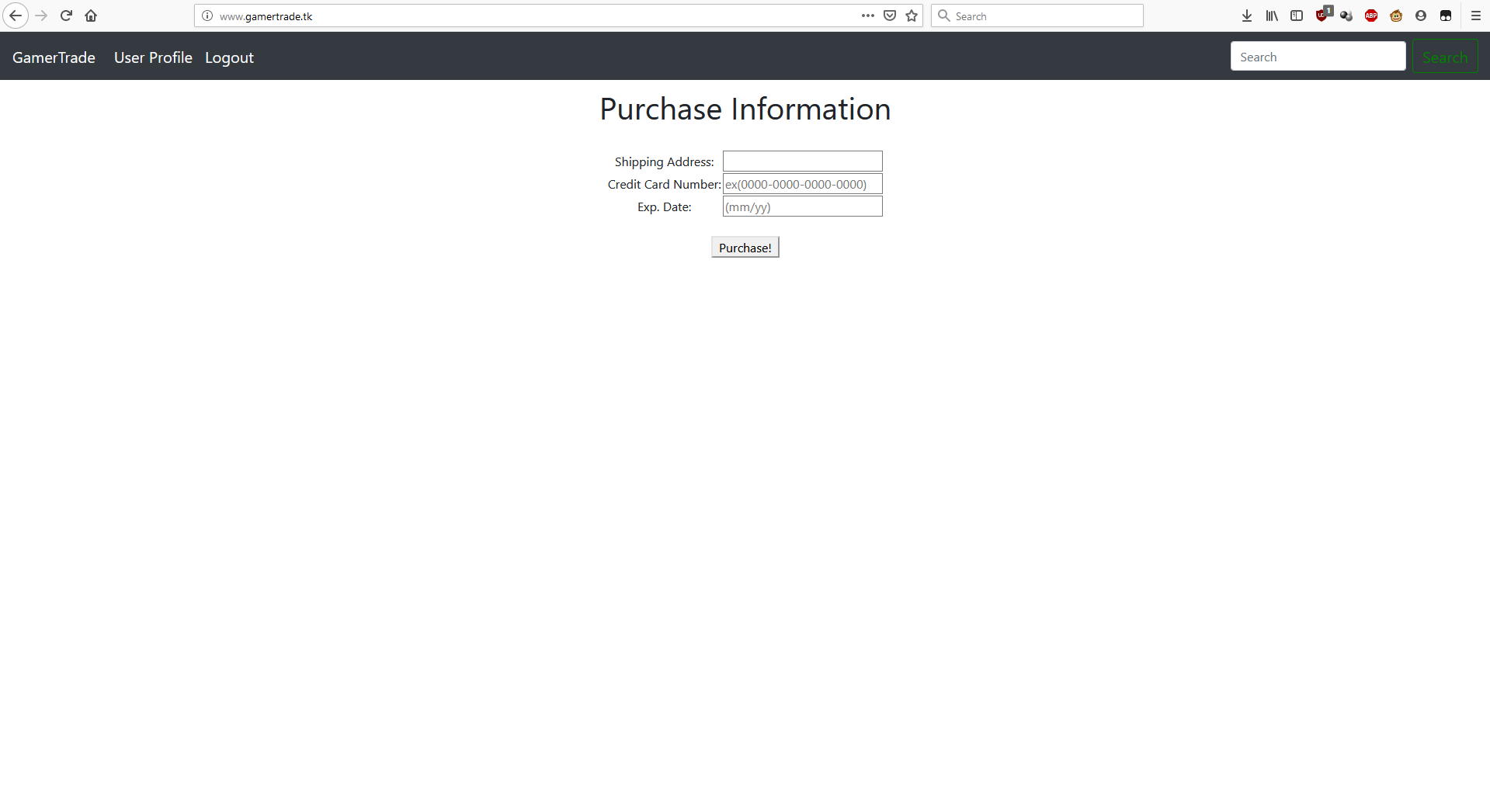
1. **Purchasing Items**

****

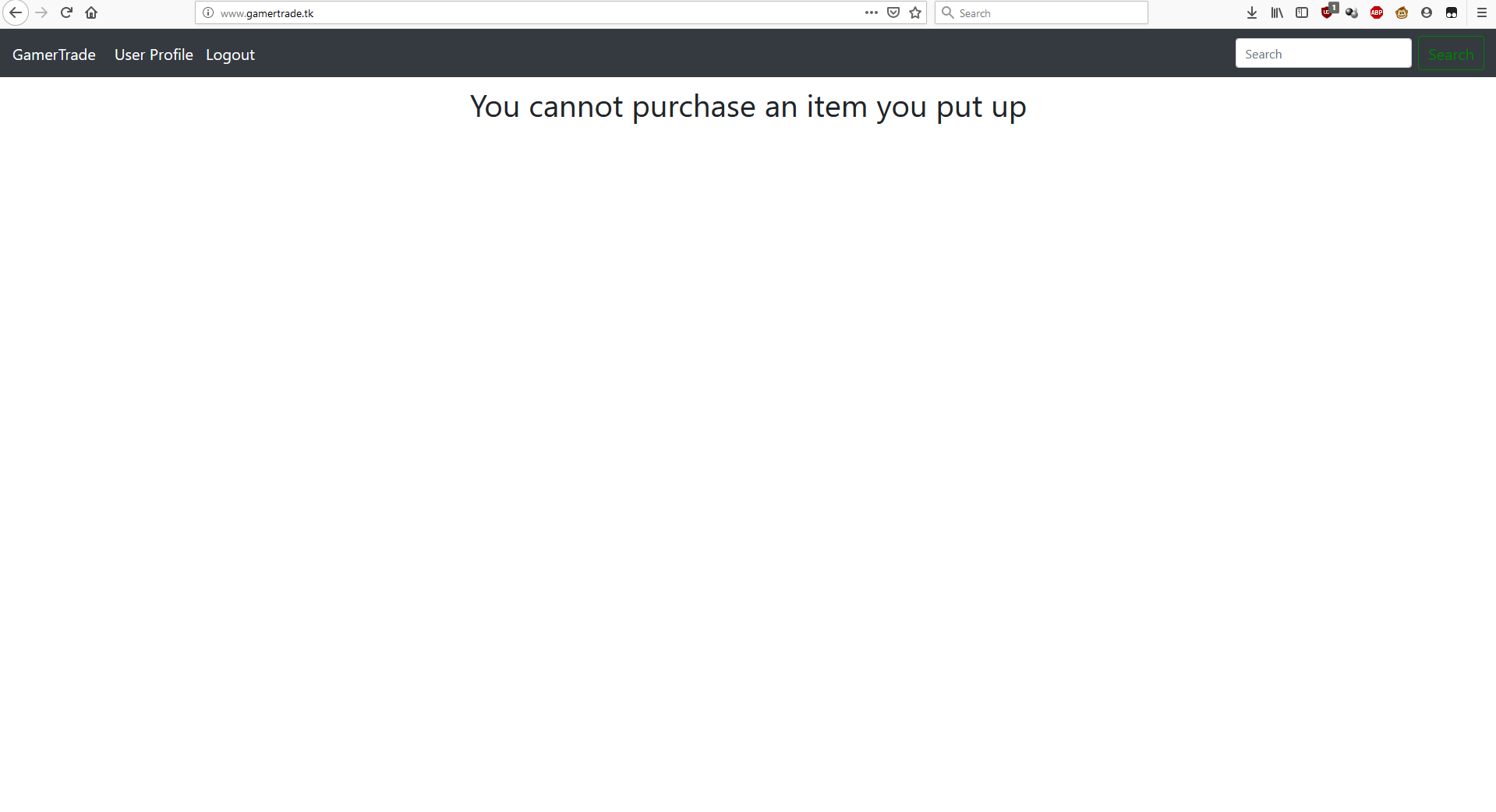
**Viewing page for item while logged in**

****

**Viewing page for item while not logged in**

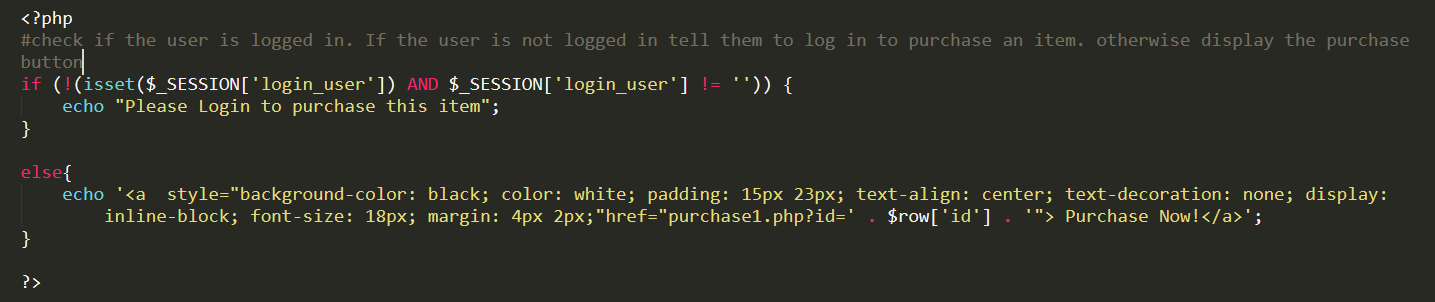
****

**Purchase information page**

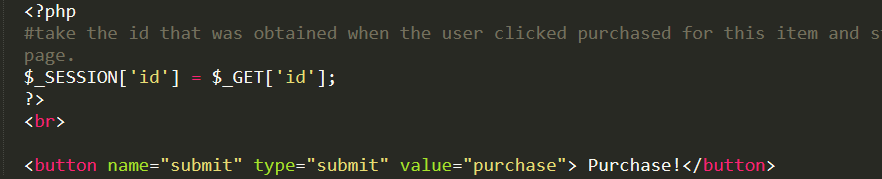
****

**Trying to buy an item you put up**

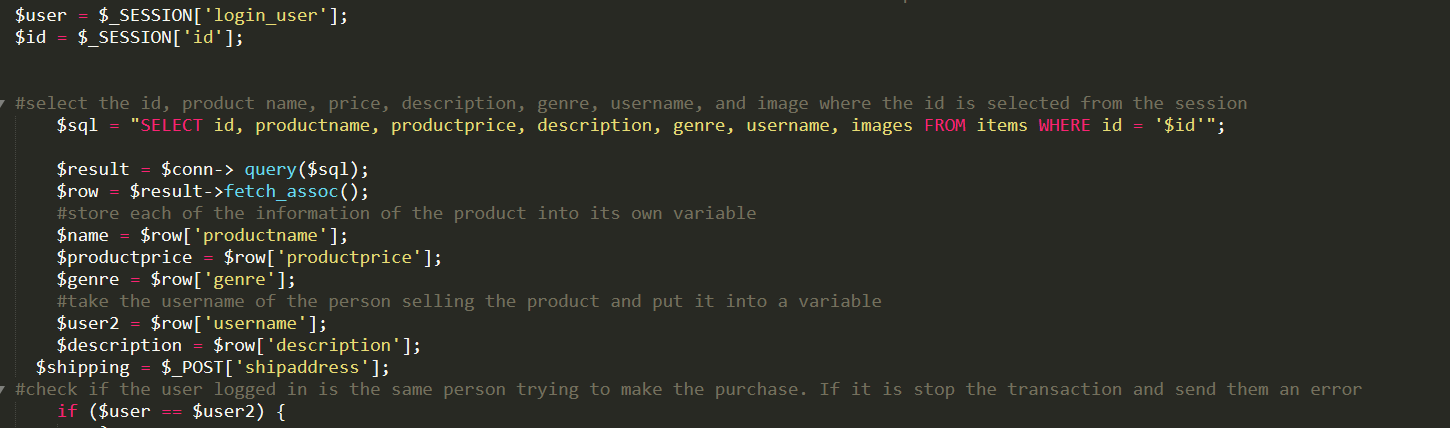
The images above show the process of purchasing an item from GamerTrade. When the user views the page, they must be logged in to make a purchase If they are not logged in, they are told to log in and will not be able to see the purchase button. Once the user clicks on the button the user must input their purchase information such as their shipping address and credit card information. The user is then redirected to their purchase history page. If the user is trying to buy an item they put up, they will not be allowed to purchase it and told they cannot buy items they have put up.



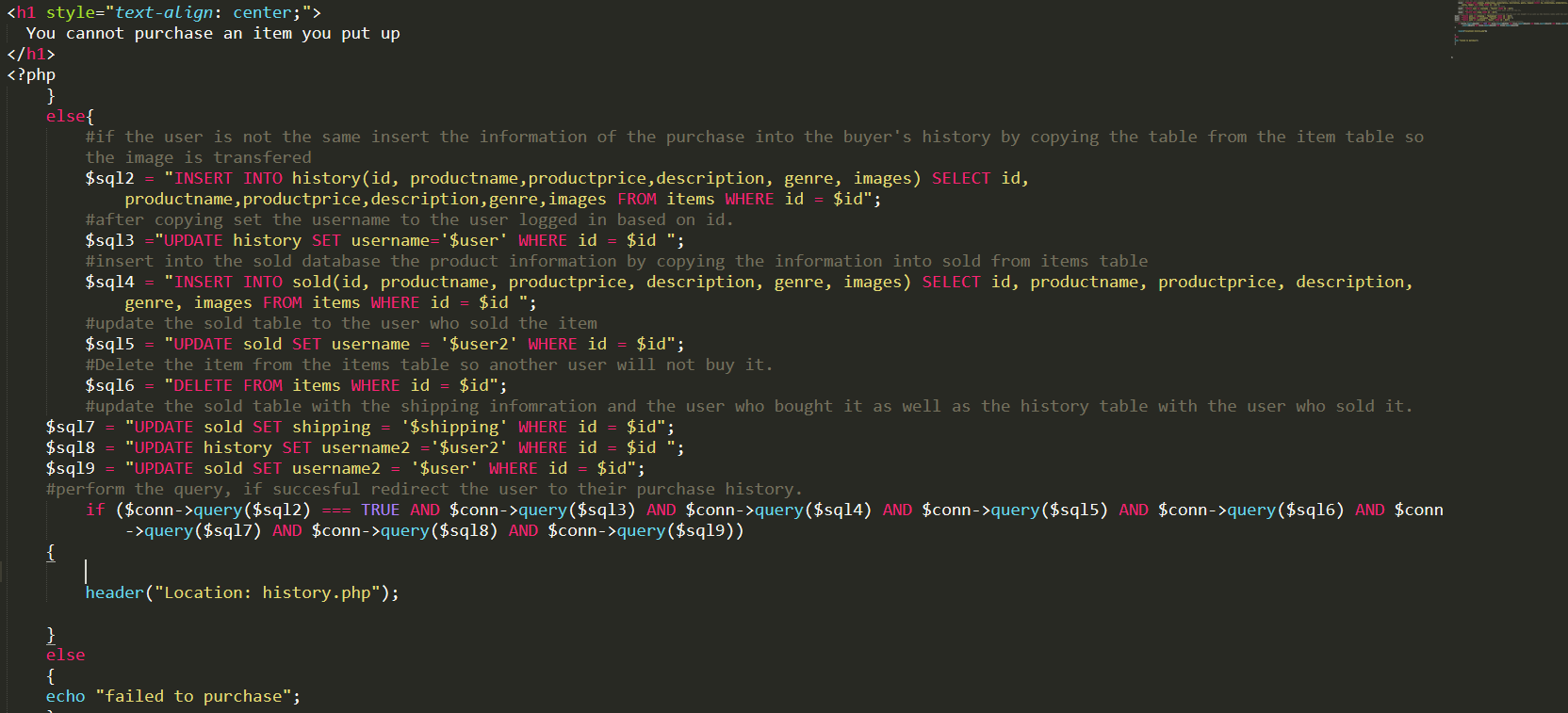
**Code for viewing page**



**Code for purchase information page**



**Code for finalizing purchase (1)**

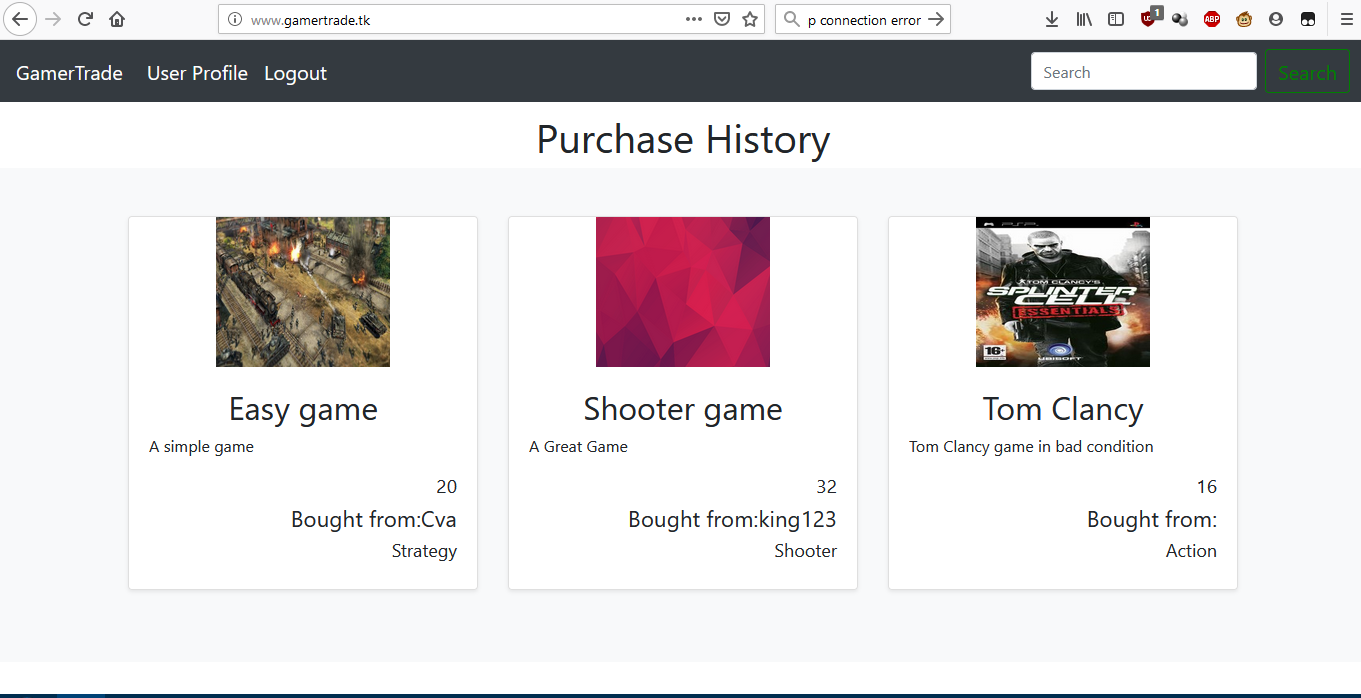


**Code for finalizing purchase (2)**

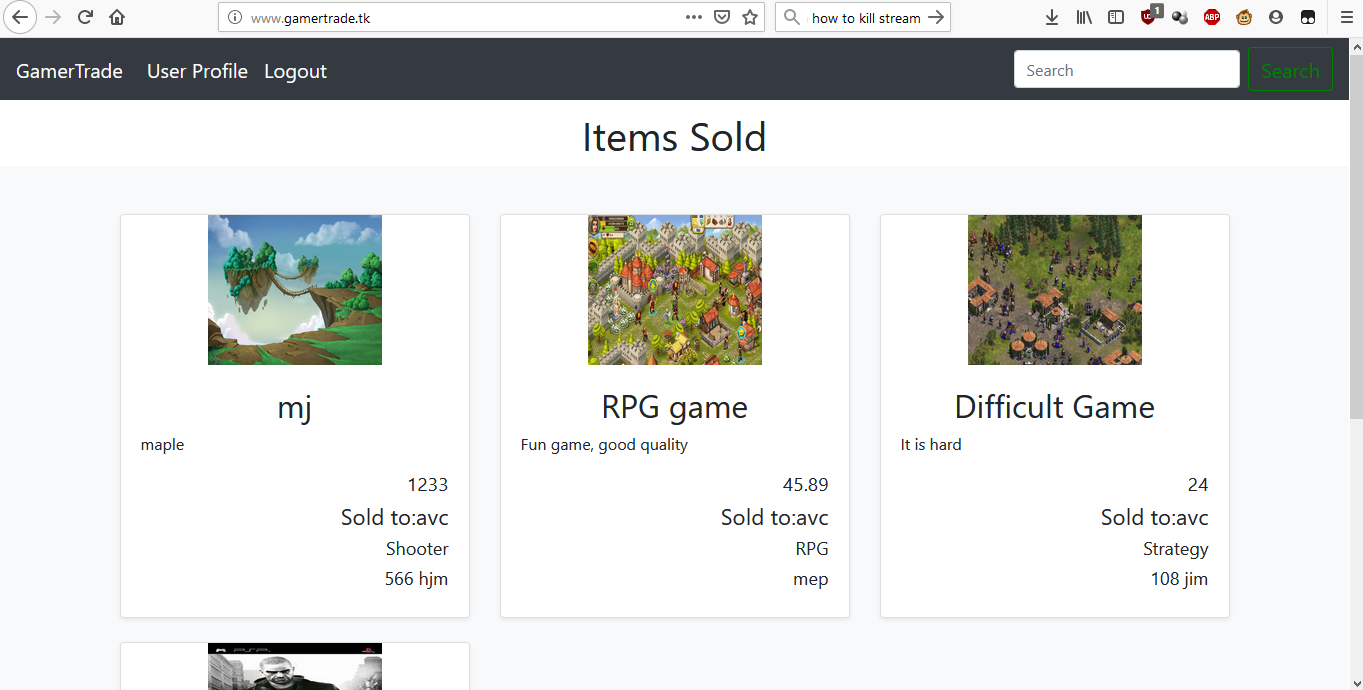
The first image shows the code for the viewing page. Depending on if the user is logged into the session will determine if they are able to purchase. This is done by checking if the user is logged into the session. If they are logged in, display the purchase button that will take the id from the product and send it to the next purchase page. After the user inputs their purchasing information the id is stored into the session so that it will be sent to the next PHP script that will perform the purchase.

The last two images are the final parts of the payment system. The first thing that is done is to store the username into the session and all the information of the product and the shipping information into variables. The user who is purchasing the item and the user who has put the item up is stored into their own variables. A check is done to see if the user who put up the item is trying to purchase their own item. If they are, post a message that they cannot and stop the purchase from happening. If they are not the same, perform a query that will insert into the history table the product information and update it with the username of the user who is making the purchase and the user who sold the item. After that, insert into the sold table the product information and insert the user who bought the item and the user who sold the item. Perform the query and if successful redirect the user to their history. If not, report to the user that the purchase failed.

1. **Purchase History and Sold History**

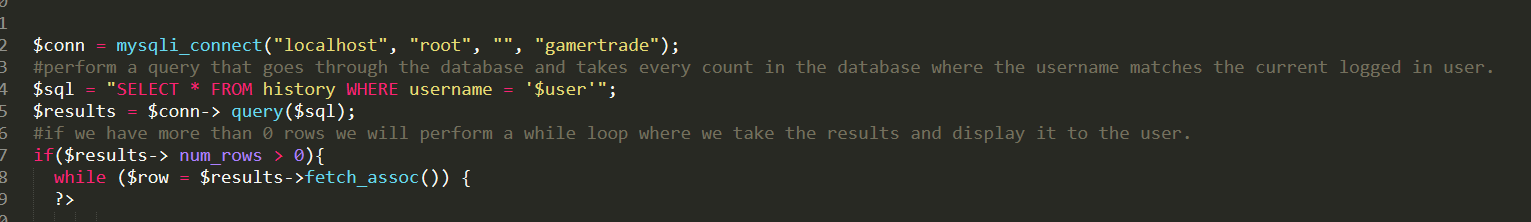
****

**User looking in their purchase history**

****

**User looking at the items they have sold**

The two images above show the purchase history and sold history features of the site. These features show the user their own history for all purchases they have made on the site, as well as details on the products they have purchased. It also shows the user who they bought the product from. The sold history feature shows the products that the user has successfully sold.



**Code for purchase history**



**Code for sold history**

The two images above show the code used for the purchase history and sold history. The code for both features is similar because they both perform a simple query that displays all the information from the sold table and purchase history to the user depending on which user is logged in during the session.

1. **State of Implementation**

**Feature 1: Homepage**

Feature is complete. Homepage is created and all the features on the page work. User is able to search for items from the navbar, sign up, or sign into the website.

**Feature 2: Genre Searching**

Genre searching works as intended. Users can search by genre from the home page by clicking on the image of the genre they want to search by.

**Feature 3: Registration/Login**

Registration and Login work perfectly and have been tested extensively. Encryption on password will be improved on in later versions. Validation works as intended, and the website does output to the user if their username is taken, the email is taken, or their passwords do not match.

**Feature 4: User Profile**

The user profile of the website works as intended. The profile page shows the user all the items they have put for sale and allows them to access the other features of the site such as adding items, removing items, purchase history, selling history, and recommendations. **Feature 5: Recommendation System**

The recommendation system of the website is working as intended but requires to be improved on later. The current recommendation system can provide a recommendation to the user based on their own buying history and will check if the user is able to receive a recommendation based on their recommendation profile. If they are unable to receive one, they are informed by the system.

**Feature 6: Add/Remove Items**

Adding and removing items works in the system. The system validates to check if the user has filled in everything in the item form. Once everything is filled in the user can hit the add item button to add the item to the items table of the database. Removing items also works and allows the user to easily delete items they have put up on the website and delete it from the items database.

**Feature 7: Purchasing**

The purchasing feature of the site is working as intended but is not complete as I have not added credit card processing to finalize the transaction between users and I have not added a proper shopping cart to allow users to buy multiple items. Now the user can still make a purchase and have the information added to their purchase history. The seller has the information stored into their sold history.

**Feature 8: Purchase/Sold history**

Purchase and sold history work as intended. Users can see all the items they have sold and all the items they have purchased. They can see the details of the items they have bought and the user they bought it from. The seller can see the details of the items sold as well as who bought it, and the shipping address to ship the item to.

1. **Testing and Evaluation**
2. **Testing**

To test this project, I used Xampp to host a server and connected to it locally from my IP address. This allowed me to debug and check the code. However, only I was able to check the website as it was not being hosted publicly. It also lacked a domain name and showed the IP address of my computer when accessed. To stop this a domain name had to be purchased and the server for the website needed to be hosted publicly from my computer. Testing the site now, I can access the website from anywhere by using the domain name from an IP outside of the network of the hosting server. I do need to do more testing on the server, as it does not have the security in mind to protect both the server that is hosting it, and the users using the site.

1. **Usability of System**

The website now is usable. Users can log in and place items to sell and they are able to receive recommendations. However, the site does not have the ability to perform purchases as finding a credit card processing company to process payments requires money to set the system up. I have also not implemented stronger security. For the security aspect, I used md5 encryption because it was built into PHP. Better encryption algorithms that are standard for all applications, especially in ecommerce tend to cost money and must be purchased from cyber security companies to use. The site also requires proper hosting. Now, the website is being hosted from my own PC and can only be accessed when I have the Xampp server started. Because of this, the website isn’t on 24/7. The recommendation system of the website needs to be improved on. The system does provide personalized recommendation, but I would improve on the way recommendations are provided to the user. For example, I would use the cosine similarity formula to find similarity between the user’s products and other products on the system. Another way to improve it would be to include more features for recommending. A feature that could be included is, if the user liked the item they purchased, and only consider items that they bought, and also enjoyed.

1. **Project Objectives**

For this project, I believe I have fulfilled all the objectives I have placed before myself. The system if fully working and there are no bugs that have been discovered from testing. The site does require more polish in terms of design, proper around the clock hosting, proper security encryption algorithms to improve security, and a payment system. For the design component I would take user feedback on how the system feels to use and improve on their feedback. For hosting, I would search for reputable hosting companies and pay them to host my server. For a security algorithm I would look into better security algorithms and implement them into the system to improve the security of the website. For a payment system I would contact PayPal or a credit card processing company to handle the transaction information for the website. I would have them handle this because they have the proper security on their backend for handling sensitive information such as account numbers.

1. **Lessons Learned**

When I first started, I did not plan to include too many features for a site and focus more on a recommendation system. However, as I began building more of the site, I learned much more was needed to create a proper site for ecommerce. Because of this I added more features like viewing sold items creating a proper purchasing system, and better searching tools. If I do this project again, I will plan my features out better to make it easier on implementation.

I also learned to better plan my tables because of this project. Because I did not plan my databases as well, I had to make multiple changes to my database and my tables. Originally, I had less columns in my table, but as I incorporated more features into my project, I needed to add more columns to the tables to make these features work. For example, I originally was not going to make a sold history feature. Because I added the feature, I needed to create a new table in the database. If I had planned my project better, I would not have needed to add a new table to my database. If I did this project again, I would use my newly learned knowledge on databases to plan my tables better.

1. **Version 2 Features**
2. **Improved Recommendation System**

On my second version of this website I would like to improve the recommendation system of the website. I believe that by incorporating more features into the recommendation system. Some features I would like to incorporate are, user enjoying the product, and considering items the user is selling. I would also use a cosine similarity formula to find similarity between products to make a more accurate recommendation for the user.

1. **Comment System**

Another feature I plan to add for my next version is a comment system that allows the user to make comments on item pages. This allows other users to see what other users think of a product and comment on the product.

1. **Shopping Cart**

In the final version a shopping cart would be implemented to make purchases more fluid and easier. It is also standard for ecommerce sites to have a shopping cart built in.

1. **Mobile App Companion**

One more feature I would like to try to add is a mobile app companion for the website. This would make using the website on a mobile device easier and much faster to make purchases than using the website on mobile devices.

1. **Professional Development and Lifelong Learning Requirement**

The project went well for me. I learned a lot about website design, database management, using the database to store and display information, styling web pages using CSS and Bootstrap, integrating PHP with HTML and CSS to display data in a proper format. Some of the things, such as HTML, CSS, and MySQL I have learned about in school from my classes. For example, I learned about HTML and CSS from CSC 235, and I learned about MySQL and database design from CSC 335. and I learned about to learn about these various features I needed to use various online sources to learn more to work on this project.

To learn PHP, I used different online sources on PHP’s manual to learn more about PHP. I had to also learn how to combine PHP and HTML together. This can be very difficult as combining multiple languages together can easily lead to syntax errors. However, doing it correctly can make the webpage looks much more professional and output data from the database in a presentable manner.

I also learned from this assignment how web hosting works. To make a proper website, it must be hosted so that users can access the website from anywhere. Normally you can pay for hosting services from a company to host your site for you. However, I did not have the funds for this, so I decided to host the server myself. To do this I had to watch videos and find guides that explain how to forward ports so that I can host my server. I also needed to obtain a domain name and port forward the IP to the domain name. A domain name is what a user will see when they access the site. If a domain name isn’t used it will just show the IP address it is logging into. Domain names normally cost money, but I had to research how to get a domain name for free. These steps were critical to making a hosted website.

This project has also taught me database creation and incorporating it into a project. My classes before this point only taught me how to do queries. But after undergoing this project I learned a lot more on how queries work. I learned how to use a query to display information from the database to show information on the webpage. I also learned how to make more complex queries and more of the interesting keywords used in queries to make more complex query statements. I also learned more about how tables connect with one another in databases and using primary keys.

Overall, I have learned a lot from this assignment. I have learned how to host servers, how to use databases, and how to use PHP. What I have learned from this project will help me in the future when I obtain a job in Computer Science and use what I have learned on future projects.

Website Name: <http://www.gamertrade.tk/>

GitHub: <https://github.com/ChadhaAnuj>