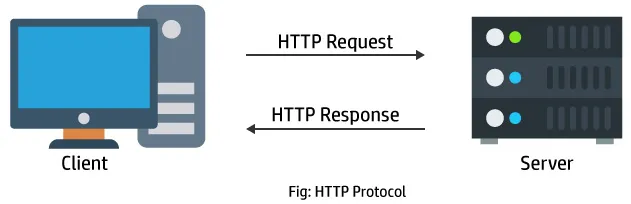
**4222COMP Introduction to Internet and Web Development - Coursework AS1**

**Section 1 – Background research**

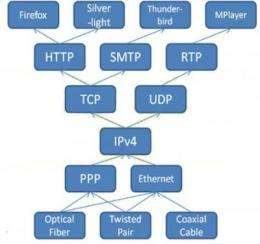
**1.1 Networking Technical Report: HTTP Protocol**

First, we must understand what a HTTP Protocol is, and what it does. The acronym HTTP stands for Hypertext Transfer Protocol. Hypertext is the text on the computer that directs the user to the desired information on request. The transfer protocol is a way for the computer to take this information and move it between computer systems through the internet. Hypertext documents store hyperlinks to other resources which the user can accessed by clicking on the link. Below is an example of machine-to-machine interactions through HTTP,



This image shows the 2 sides involved in the protocol, which is the client’s device, and the server. The first thing that must happen is the link between the 2 machines must be opened. This is the HTTP server. When the client requests information from their device, the request is sent to the server. An example of information that can be sent is a HTML file. Once the request has been processes, the requested data will be sent back to the client from the server. After this is complete, the link that was opened between the 2 machines is closed.

Next, we are looking at how the HTTP Protocol fits into the internet hourglass model. It is important that we understand what the hourglass model is, so we know how and why the HTTP Protocol fits into it. The internet hourglass is referring to the how the technologies that services use are linked together. Below is an example of the internet hourglass,



Here we can see the hourglass shape in which the model takes its name. The diagram shows us how the HTTP Protocol is near the top along with other protocols such as SMTP (Simple Mail Transfer Protocol), and RTP (Real-Time Transfer Protocol). Above the Protocols are specific applications, such as internet browsers and media players. Below HTTP we can see the diagram starts to narrow, as it moves into Transport Protocols such as TCP (Transmission Control Protocol), and UDP (User Datagram Protocol). When then move to the middle of the hourglass which is the Network protocols, such as IP (Internet Protocol). As the hourglass starts to expand out to data-link protocols such as ethernet and point-to-point protocols which in turn will branch into physical layer protocols.

We are now looking at how the HTTP Protocol is used for web communication. HTTP uses something called the request-response model. This method is used by computers in a network to communicate to each other, when the client-side computer sends a request for information, and the server responds with the requested information. When the request-response model is used over HTTP, it is typically done in a synchronous fashion, meaning that the connection between the 2 machines will be held open until there is a response, or until the timeout period ends. Generally web browsers call request and response, get and post respectively.

The HTTP Protocol is known as a stateless protocol, due to the fact each request it is given is executed independently with no knowledge of prior executed requests. This is done so that every request can be understood independently, and is the opposite of a stateful protocol, in which requests hold the same session state as previous requests. Sometimes stateless protocols might have decreased performance to stateful protocols due to an increased amount of repetitive data sent in multiple requests that can’t be reused by the server. However, this is made up for by the fact that stateless protocols improve on the visibility, reliability, and scalability.

The visibility of the protocol is improved because the monitoring system doesn’t have to look ahead at multiple requests, as it only has the current one to determine the nature of. The reliability is also more secure as it is far easier for the task to recover from any partial failures. The other positive that stateless brings is a greater scalability due to the fact the server doesn’t have to store multiple states between requests, allowing a quick free up of server space. These advantages make the stateless protocol a more suitable foundation for HTTP.

HTTP has a many commonly used methods, which include , put, patch, and delete, but the most important and primarily used are post, and get. We have already mentioned how these methods are part of the request-response protocol. Get is used to request data from a specified resource. Post is used for sending data to a server to create a resource. There are a few key technical differences between the 2 methods. The get method can be cached and bookmarked whereas the post method cannot. However, the post method doesn’t have any data length restrictions, whereas get has a maximum URL length of 2048 characters. Another restriction that get has that post doesn’t is the fact get only allows ASCII characters. Finally, it is important to know that post is a little more secure than get due to the fact that the parameters aren’t being stored in the browser history.

HTTP has many status codes, which are used as standard responses to identify problems that may arise when a resource cannot load. Status codes are also referred to as browser error codes or internet error codes. Every status code is grouped into a specific number group for the type of code it is. Below is a list of these codes,

Chart

Description automatically generated

1xx- Response given when the request was received, and the process is continuing

2xx- Response when the request was a success, and it has been accepted

3xx- Response given when further action needs to be completed

4xx- Response when the request cannot be fulfilled due to bad syntax in the request

5xx- Response when the server cannot fulfil a valid request

An example of a common status code is 404, which is used when the server can’t find the requested page. It is important to note that the status codes used in HTTP are unique and are not the same as other error codes, even if they may share similar numbers.

The HTTP Protocol also defines the concept of cookies. It does this in 4 simple components. The first of which is the cookie header line of the HTTP response message. Next, is the cookie header line in the next HTTP request message. From here the cookie file that is kept on the client-side is managed by the user’s internet browser. The final component is the back-end storage system which is used by a web-sites server host. Cookies were invented due to the stateless nature of the HTTP Protocol. Having cookies available allows stateful information to be remembered in a stateless protocol.

**1.2: Web Technology Technical Report: Session Management in PHP:**

As previously discussed, the HTTP Protocol is a stateless protocol, which is inconvenient when working on the internet as any changes we make won’t be saved. To overcome this, we use PHP sessions, which will help us by storing user information across multiple web pages. The server tracks the user’s information by recording the user’s unique information such as the users IP address. Each user is assigned a unique session ID automatically by the session. The session will last from when the user opens the web page until the browser is closed by the user.

Sessions are mainly used when we need to store information as we move through pages until the website is closed by the user. The type of website where we would most commonly see this is in shopping websites. In these situations the type of information that carries over from page to page is things like product name product codes, and prices of products etc. This is done by tracking the unique session ID the user was given once they opened the website.

To start a session using PHP we user the session\_start() function. This must be declared before the HTML tags. Each user is given a unique number for every new session. If you want to check this session id you can do so with the following code,

Echo session\_id();

The session\_id() is useful as we can also use it to start a session with a custom id. If we are doing this it is important to remember that this should be done before we start the session. To do this we would simply put the id number of our choice in the brackets of the function like this,

session\_id(id number of choice);

We can set variables using the $\_SESSION which is a PHP global variable. There are several types of variables but the main ones that are used are strings, Booleans, integers, and arrays among others. We need to give this variable a value, which can be done using the following line of code,

$\_SESSION[“name”] = “Charlie”;

If we want the variable to be relayed back to us, we us the line of code,

echo “Your name is “ . $\_SESSION [“name”];

The next step for us is to update the variable. This is just as simple as opening the session and giving the variable a new value, for example,

$\_SESSION[“name”] = “Eoin”;

As we have opened the session it is important that we remember to close the session with the session\_unset() function, which will remove all the global session variables, and the session\_destroy() function, which will destroy the current session. If we put all of this code together it should look something like this.

<?php

session\_start();

?>

<!DOCTYPE html>

<html>

<body>

<?php

$\_SESSION[“name”] = “Eoin”;

?>

<?php

session\_unset();

session\_destroy();

?>

</body>

</html>

**Section 2 – Server-side Design**

**2.1: Data Requirements Class Diagram:**

Our next step is to create a class diagram which will show the server-side representation of our application data. To describe all of the data we will be putting into our system we are using classes, which will be represented with a rectangular box in the diagram. We can also connect classes to other classes with relationships. The class can be split into 3 sections, the top will be the name of the class, the middle will be where we put all of the variables, and the bottom is where we put the methods. Below is an example of a class,

A picture containing rectangle

Description automatically generated

Inside the class we use variables to identify each instance of a piece of data which contains values that describe the data. In the example above we can see that variables can also be referred to as attributes, and also properties. It is important to format the variables correctly. To write a variable first you must name the variable which must start with a lowercase letter, followed by a colon, and finally by the data type. For example one that we might use in our project could be, +nameOfGame: string.

At the bottom of the class, we put the methods, which can also be called functions, or called functions or operations. Methods allow us the specify the behavioural features of any class. Methods also must start with a lowercase character followed by a set of brackets. For example in our project we might want a method to set the price of the game we are reviewing, which could be done by typing this, +setPrice():.

Another important note we must make is the visibility of the variables and methods. As we can see before each variable and method example, we have put a + sign. This shows us that that particular variable or method is public and therefore can be accessed by other classes. If we were to use a – sign this would be the opposite, and the variables and methods would be only accessible in that particular class. Another sign that can be used is the # sign which means that particular variables or methods that can only be accessed by the same class or it’s subclasses. For our project it is best to use the + to keep the variables and methods the same so that they can be carried across pages.

It is now important that we understand the links between the classes, which are called relationships. The first relationship that we should know is inheritance relationships, which shows us which subclasses belong to which class. The subclasses inherit the variables from the class but can also have their own variables added to them. Any change in the main class will apply itself to all connected subclasses. Below is the arrow that we use to show inheritance,

Text

Description automatically generated with low confidence

The next relationship we need to know is association. We use association when there is no dependency between the 2 classes, and the relationship is purely a basic association. We represent this with a simple line and a word used to describe the simple relationship, for example in our project a relationship that could be an association is the class for all of the games and the class for a specific game,

has



Another important relationship is an aggregation relationship. This is a relationship where a class can exist and has a relationship with another class, but if the first class was deleted the second class could still exist independently. An example of this is if we had an animal such as a sheep, and a separate class of a herd of sheep. If we deleted the herd of sheep class the sheep could still exist independently. The line to show this relationship is shown below,

A picture containing logo

Description automatically generated

The opposite of an aggregation relationship is a composition relationship. Going back to the previous example used, this means that if the sheep class didn’t exist, then the heard of sheep wouldn’t be able to exist independently. This is shown with this line,



For all relationships we can also use multiplicity. Multiplicity helps us to set constraints on the number of relationships. This is simply done by typing the number we want to set on the line of the relationship. We can also use multiplicity to show we want at least a certain amount of a relationship, for example if we wanted at least on we could type, 1..\* .

We now need to create our own class diagram for the server-side design. The classes that we will be using for the diagram are

-Games

-Reviews

-Ranking

-Retail

-User Profiles

Timeline

Description automatically generatedBelow is the diagram I have made on Microsoft Visio,

From the diagram I have made we can see there are 6 classes, which include games, user profile, retail info, reviews, user reviews, and external reviews. I started off with the game class, which includes the data for all 10 games I have chosen. This includes the name of the game, the game ID, the games genre, the age rating, the publisher studio, the games description, and finally the games price. There is also a few functions that are included such as getID(), so we know which game to fetch the correct data for, getRank(), which will find the users rank of the games, and getGameDetails, were the system will fetch the games ID and display the information for it.

The next class I created was the retail info class. This class holds the data for the games ID, the games price, the games retailers, and the websites for these game retailers. The methods I have included are methods for getting the price of the game and getting the retail websites for the games. I have connected this class to the games class using association, as the games have retail information.

Next are the classes I have made for reviews. For the reviews superclass I have added variables for the games ID, the games ranking, and the average external scoring for the games. The 2 methods I have included are getID() so we know which game to fetch the reviews and scores for, and getAverageScore() so the average score of the external reviewers for the game can be shown. I chose to make this a superclass as I felt that the user reviews and external reviews classes share similar variables and methods, so I made them subclasses and joined them with an inheritance relationship.

The similar methods and variables aren’t enough however, so for the user reviews class I have also included variables for the user’s profile, the users game ranking, the users scoring of the games, and the users reviews of the games. The methods I have added in this class is setUserReview(), which saves the users review after they make it, setUserScore(), which will save the score the user gives for the games, and setUserRanks(), which will save the game ranking the user gives.

Similarly the external review class also needs other variables and methods on top of the ones shared. The variables I have added to this class are the name of the external reviewer, the review left by this reviewer, the average score of the reviewers, and the websites the users made these reviews. For the methods I have added methods for getting the external score and getting the external reviews. I have used get methods here as opposed to the set methods used in the user reviews as the user is the one adding the reviews whereas we already have the external reviews and are just fetching them.

The last class I have made is the user profile class. This class has variables for the username of the client, the profile ID associated with the username, and the users ranking of the games. The methods I have used are getUserID(), which finds the ID of the user, and getUserRank(), which finds the game ranking the user has set. This class holds a composition relationship with the users review class. The user review class cannot exist without the user’s class, as without a user there is nobody to leave a user review.

**2.2: Examples of the class data:**

For the next section we are focusing on examples of the class data from the previous section. First we need 2 examples from the game list which will be FIFA 22, and Rocket League for the Xbox One Series X which we will use throughout all of the relevant classes. Using Microsoft excel the classes will be derived and shown in able form. Below is the tables,

Graphical user interface, application

Description automatically generated

We can see here all of the classes followed by the variables, and then 2 examples of each. Starting off with the games class, we can see all the data for FIFA 22 and Rocket League. For now the data isn’t finalised but is a good placeholder for the data we will actually use on the website. The Retail info class is very similar, as the only class it has association with is the games class.

The reviews class is important to see here, as we can see from the tables that both the External reviews and User reviews classes start off with the same data. This is because as discussed in 2.1, they share the same superclass which Is the reviews class. The inheritance relationship becomes more apparent here as we can see how the External reviews and user reviews classes inherit the data. Finally, the User profile class is small, and in the table it isn’t very apparent that it shares a composition relationship with the games class.

**2.3: Site Map**

The focus is now on the functionality of our application. Site maps are a good way to show how the functionality is split and the purpose of the pages. Here is the site map made on Microsoft Visio, showing each individual PHP page,

Diagram

Description automatically generated

The site map shows all of the individual PHP pages that will be used within the website. It all starts at the top with the home page icon, which will be the page the client is greeted with when the first open the website. From here the website will provide the client with links to access to the other pages. From the home page the client is provided links which can take them to either the User Profile, User Review, Games, or External Review pages, via the correct link. This is completed through GET/POST as the user sends a request for the selected page to the server, and the server posts the correct data back in the form of the selected PHP page. The request handling code we can use in this situation are the doGet() and doPost() methods.

In the case of the User Profile, User Review, Games, and Game info pages, they also have another link inside them to bring you to a further page. These further pages are linked from these pages instead of the home page as they have more relevance to these pages. This is also done through the GET/POST methods. It is also important to have a link on every page that will bring the client back to the home page.

We can also create a site map that ignores the GET/POST method and shows the pages linked without it. Below is what the site map would look like in that case,

Diagram

Description automatically generated

The website would work the same but would do so using different methods to the GET/POST.

**2.4: Server-Side Logic Activity Diagram:**

The next diagram we will be working with is an activity diagram which we will be using to show how the workflow of the server generates a page for presenting the editorial recommendation and score of the highest scoring game. Once again we will be using Microsoft Visio to complete the diagram. It is important to consider how we are making the diagram from the Server-Side perspective and show all possible steps they could take. Below is this diagram,

Diagram

Description automatically generated

The diagram starts at the moment the user opens the external review page. The server will display the page, and so the next step is for the user to work out the top scoring game. The system will get all the external game scores and find the game ID associated with the highest score. The user then will select if they would like to select the games external reviews or if they would like to see the external scores for the game.

If the user chooses to see the reviews then the system must process this, and then get the external reviews for the game. Once the system has got the external reviews for the game the system will display the reviews. The other option is for the user to select the game scores, in which case the system will get the external score, before it then goes to work out the average of the scores. After it works this out it will display it on the page for the user. After either of the score or reviews are displayed the diagram comes to a stop.

**Section 3 – Client-side Design**

**3.1: Client-Side Logic Activity Diagram:**

Moving on from focusing on the server side of the system, its time to look at the client-side aspects. Again we will be making an activity diagram using Microsoft Visio, but this time it will be from the client’s point of view, which can be seen below,

Diagram

Description automatically generated

We start off in the diagram by the user selecting the game reviews page. From here the system is given the decision to check if the user is logged in. If not, the user will be brought to the login page where they will need to login before continuing. If the user enters their login details and they are incorrect the system has another decision, whether or not the user’s login details are correct or incorrect. If incorrect the user will be given an error message and be brought back to re-enter their details. If correct then the user will be brought to the reviews page, which they would’ve already be brough to if they were already logged in.

From the user reviews page, there is another decision that needs made. We need to know if the user has already made a game ranking or if they are yet to do so. If they have already made their game ranking then they will be shown their current ranking. From here the user is given the option to change their ranking if they wish to do so. If they don’t wish to change their ranking then the diagram will come to a stop, but if they do, they will be brought to the same process as the user would have been brought to if they hadn’t made a ranking already in the previous decision. After the user has changed the ranking the diagram will come to a stop.

The activity diagram was useful in showing us the behavioural model that is being executed by the system. It helps plan out what should happen when the system responds to the user and what data should be sent out. It has also helped to simply the process by laying it all out and making it more clear, which can lead to overall improvement in the system in the future.

**3.2: Application Master Page:**

For our next task we need to present the pages that make up our application, which we will do so using wireframe. By using wireframe we can make a conceptual layout which can show what information we are getting and how it functions, how it should be laid out, and how the data flows. The software we will be using to complete this wireframe is Pencil. Below is the wireframe for the application master page,

Diagram

Description automatically generated

To start we assumed that a 16:9 ratio widescreen display is used, so the page is 800:450 in pixels in size. On the top of the page I have added a header, that contains the title of the page, and also links to other pages of the website. These pages are the User profile, user reviews, and external review pages. The only thing that can’t be accessed through the header is the game pages, as these can be accessed through the game carousel. The game carousel is a useful way for us to show all of the games available on the website in an aesthetically pleasing way.

We also have the game ranking on the page, so that the user can see their very own ranking of the games as soon as they go on to the website. It will also act as a link if clicked on to bring the user to the user reviews page so they can edit their ranking if they wish to do so. Finally at the bottom of the page there is a footer which will give the user some essential information such as the privacy information for the website, but also the author (myself) and the about section.

**3.3: Navigation Systems:**

Having a navigation system in the program is very important as it allows the user to move throughout the website to all of the different pages that we have created. I feel like the best way to do this is to have a navigation bar with all the different pages the user can find all of the different web pages. The user will know if they want to go to any of the pages, they will be available through this navigation bar. Below I have highlighted the navigation bar in the master page,

Diagram

Description automatically generated

I feel like the navigation works best just below the page header as it is more visible and clear to the user. The navigation bar could also be in the form of a burger menu if the user was on a phone, and the screen was narrower. The icon for this is below,

Shape

Description automatically generated with low confidence

When the user clicks this icon then a drop-down menu with all of the pages from the navigation bar.

**3.4: Example Application Page Designs:**

The first page we will be showing the structure and layout for is the game details view. As previously discussed this can be accessed through the game carousel on the master page. Once the user has clicked on a game, they will be brought to a page that looks like this,

Diagram

Description automatically generated

The page shares a similar header and footer to the master page, however the external review in the navigation bar has been replaced with the home page link, so the user can return back to the master page. The page has the game title and the cover art of the game, with a brief description and info of the game. Near the bottom of the page I have added the retail info for the game which includes some basic details about the game’s retailer and an external link to take the user to purchase the game.

The main focus of the page is obviously the reviews which are in the middle of the page. The external review is to the left, and includes the reviewer’s name, and their review off the game, along with the link to the reviewer’s website. To the right of this we can see the scores from the external review, the games ranking, and the average score from the users. To the right of the score is the users review, which can be added through the users review page, which includes the reviewer’s username and their review of the game.

The next layout we want to see is the ranking page, which can be found through the master page. We get to this page by clicking on the ranking on the page, which shows the full ranking in less detail. Here is the page,

A picture containing qr code

Description automatically generated

I have made the ranking in a pyramid shape as I feel that’s a good way to show the ranking in a more unique way. Each of the boxes have the games ranking, along with its title and the average user score of the game. The page again also has the header and footers, but the navigation bar has an extra tab for the home page.

**References:**

Annonomous. (). Introduction to HTTP. Available: https://www.w3schools.in/http-tutorial/intro/. Last accessed 1/03/2022.

Georgia Institute of Technology. (2011). How the Internet architecture got its hourglass shape and what that means for the future. Available: [https://phys.org/news/2011-08-internet-architecture-hourglass-future.html. Last accessed 1/03/2022](https://phys.org/news/2011-08-internet-architecture-hourglass-future.html.%20Last%20accessed%201/03/2022).

Annonomous. (). HTTP Request Methods. Available: https://www.w3schools.com/tags/ref\_httpmethods.asp#:~:text=The%20two%20most%20common%20HTTP%20methods%20are%3A%20GET%20and%20POST. Last accessed 1/03/2022.

Annonomous. (2022). Using HTTP cookies. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies. Last accessed 1/03/2022.

Sajal Soni. (2021). How to Use Sessions and Session Variables in PHP. Available: https://code.tutsplus.com/tutorials/how-to-use-sessions-and-session-variables-in-php--cms-31839. Last accessed 5/03/2022.

Lucid Chart Youtube (2017). UML Class Diagram Tutorial. Available: <https://www.youtube.com/watch?v=UI6lqHOVHic> Last accessed 10/03/2022.