

Kalan Assignment 1

by Kalan Hughes

Submission date: 06-Dec-2021 03:38PM (UTC+0000)

Submission ID: 165135663

File name: 17861_Kalan_Jay_Hughes_Kalan_Assignment_1_382340_1549474858.docx (38.54K)

Word count: 3513

Character count: 21916

Assignment 1

Kalan Hughes

1902299

Bachelor of Science: Computing (Web Development)

Cloud-Based Web Services

10 December 2021

Table of Contents

Web Services Report..... 3

YouTube Video Link..... 5

References 5

Appendix 7

 index.html 7

 games.js 16

Web Services Report

This report is going to talk about three different web services. These web services are SOAP, REST, and GraphQL. Each web service will have its benefits and drawbacks talked about. Once all three web services have been talked about and have had their benefits and drawbacks discussed, they will be compared to see how they are similar and how the web services contrast with each other.

The first web service that will be talked about is SOAP. Simple Object Access Protocol, also known as SOAP, is an XML-based protocol that is used for accessing web services over HTTP. SOAP is used to exchange information between different computers by using the internet. SOAP helps to avoid complex development methods as it is an intermediate language which means that applications that are built on different programming languages are able to communicate with ease [1]. SOAP messages are unique as they are written entirely in XML which is not commonly seen in other frameworks. This makes SOAP platform and language independent [2]. There are some advantages of SOAP. The first advantage is that it is language and platform independent [3]. This means that SOAP web services are extremely versatile as they can be executed on any platform, and they can be written in any programming language. Another advantage of SOAP is that it works on the HTTP protocol [1]. The HTTP protocol is used by all of the web applications as it is the default protocol. This means that no customization is needed in order for it to run on the World Wide Web. The final advantage of SOAP that will be talked about is simplicity [4]. SOAP uses a simple XML format for its messages which helps to make it simple to use. This can be useful for beginners and novice users who are not familiar with more complex formats for messages. These are some advantages of SOAP. There are also some disadvantages of SOAP. The first disadvantage is that it is slow [4]. SOAP needs to be parsed due to the fact that it uses the XML format. SOAP is also lengthy which makes it a lot slower than other frameworks. Another disadvantage of SOAP is that it is WSDL dependent [3]. SOAP does not have standardized mechanisms to discover the service as it only uses WSDL. These are some disadvantages of SOAP.

The second web service that will be talked about is REST. REpresentational State Transfer, also known as REST, is a web standard based architecture which uses the HTTP Protocol [5]. There are four main HTTP methods that are used in REST architecture [5]. These are POST, GET, PUT, and DELETE [6]. An example of when you would use these methods, is when managing a database. You would use the POST method to add a new piece of data to the database. GET would be used to retrieve data from the database. PUT would be used to update data in the database. Finally, DELETE would be used to delete data from the database [5]. Much like SOAP, REST allows web applications that are built using different programming languages to communicate with each other. REST data is usually accessed through links on the web which are called Uniform Resource Identifiers, also known as URIs [7]. There are some advantages of REST. The first advantage is that it is language independent [8]. Much like SOAP, REST is language independent which means that developers can make web-based requests using their preferred language that uses HTTP. Another advantage of REST is its communication [8]. REST has good communication as it uses numerical HTTP status codes to detect any errors. This can help the user quickly and easily find and fix any errors. The final advantage of REST is that it is pervasive [8]. This is because it has uses in both server-side and client-side implementations. This is because developers can use REST-based frameworks on the server-side and developers on the client-side can use frameworks such as jQuery, Node.js, and much more. There are also some disadvantages of REST. The first disadvantage of REST is the architecture [8]. This is because developers can encounter limitations due to the architecture design. An example of one of these issues is long HTTP request headers, which can result in delays when loading a

webpage. Another disadvantage of REST is data over fetching and under fetching [8]. This is an issue because REST can return a lot of unusable data while also returning relevant information. This can increase the amount of time it takes for the required data to be returned. These are some disadvantages of REST.

The final web service that will be talked about is GraphQL. GraphQL is a query language for APIs that is used to provide users with only the data that they requested [9]. Unlike REST, GraphQL only fetches the required data. This is because the data is declared which removes any unnecessary data from being received in the front end [10]. REST is used to give web services a structure whereas GraphQL is more suited towards retrieving elements [10]. GraphQL is a programming language and a database agnostic [10]. GraphQL grew in popularity due to the changing digital landscape. These changes included faster development cycles and increased mobile usage [10]. There are some advantages of GraphQL. The first advantage is that it has a faster performance compared to REST [10]. This is because you are able to choose the fields you want to query unlike REST which tends to send additional fields. You are also able to retrieve multiple entities in one request which can help to increase its efficiency. Another advantage is the longer API call limit threshold [10]. This is because GraphQL uses fewer queries than REST which means that it does not encounter call limits as quickly as REST. The final advantage of GraphQL that will be talked about is the amount of data returned in a query request [10]. GraphQL only sends the required data which means that the user does not have to spend a lot of time searching through data to find the specific data they want. There are also some disadvantages of GraphQL. The first disadvantage of GraphQL is the time it takes to analyse errors [10]. This is because it requires you to parse through messages to find the errors. This can take up a lot of time which can result in it taking a lot of time to solve errors. Another disadvantage of GraphQL is complex queries [10]. This is because the larger the data set, the larger the query will have to be to accommodate to the data. This can be difficult for developers as they would have to spend a lot of time creating the query depending on the size of the data set. These are some disadvantages of GraphQL.

Now that all three web services have been talked about and they have had their benefits and drawbacks discussed, they will now be compared to see how they are similar and how they contrast with each other. One similarity between SOAP and REST is that they are both language independent which means that they are both versatile as they can both be executed on any platform. They can also be written in any programming language. A similarity between GraphQL and REST is that they can both be used to query data. One difference between them is that GraphQL only queries the necessary data whereas REST tends to query unnecessary data with the required data. Another difference between REST and SOAP is that REST accesses data whereas SOAP carries out operations by using a standardized set of messaging patterns [11]. One similarity between SOAP and GraphQL is that they both access data through a single endpoint [12]. These are some of the similarities and differences between SOAP, REST, and GraphQL.

In conclusion, this report has talked about each web service and their benefits and drawbacks. The report has also talked about some similarities and differences between the three web services. All of the web services provide their own unique features which means that they are all useful and they can all be used depending on the situation. SOAP can be used for when you want to communicate and exchange data between two different computers over the internet. REST can be used for when you want to easily add, edit, retrieve, and delete data in a database. GraphQL can be used for when you want to create a query to get specific data from a data set.

YouTube Video Link

Below is the link for the YouTube video which demonstrates the webpage that has been created and the code that was written when creating the webpage.

<https://youtu.be/VjiSMbJijHQ>

References

- [1 A. Walker, "SOAP Web Services Tutorial: What is SOAP Protocol? EXAMPLE," 8 October 2021.
] [Online]. Available: <https://www.guru99.com/soap-simple-object-access-protocol.html>.
[Accessed 8 November 2021].
- [2 "What is SOAP?," [Online]. Available: https://www.tutorialspoint.com/soap/what_is_soap.htm.
] [Accessed 8 November 2021].
- [3 "SOAP Web Services," [Online]. Available: <https://www.javatpoint.com/soap-web-services>.
] [Accessed 8 November 2021].
- [4 "What is SOAP? Why is SOAP required? Advantages of SOAP," [Online]. Available:
] <https://www.careerride.com/SOAP-What-is-SOAP.aspx>. [Accessed 8 November 2021].
- [5 "RESTful Web Services - Introduction," [Online]. Available:
] https://www.tutorialspoint.com/restful/restful_introduction.htm. [Accessed 8 November 2021].
- [6 A. Walker, "RESTful Web Services Tutorial with REST API Example," 8 October 2021. [Online].
] Available: <https://www.guru99.com/restful-web-services.html>. [Accessed 8 November 2021].
- [7 Eric Jendrock, Ricardo Cervera-Navarro, et al., "What Are RESTful Web Services?," January 2013.
] [Online]. Available: <https://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html>. [Accessed 8 November 2021].
- [8 Kerry Doyle, Kevin Ferguson, Cameron McKenzie, "REST (REpresentational State Transfer)," January 2021. [Online]. Available:
] <https://searcharchitecture.techtarget.com/definition/REST-REpresentational-State-Transfer>.
[Accessed 8 November 2021].
- [9 "What is GraphQL?," 8 January 2019. [Online]. Available:
] <https://www.redhat.com/en/topics/api/what-is-graphql>. [Accessed 8 November 2021].
- [1 J. Mason, "What is GraphQL? (And is it Really Better than REST)," 10 November 2020. [Online].
0] Available: <https://www.coredna.com/blogs/what-is-graphql>. [Accessed 8 November 2021].

[1 A. ALTVATER, "SOAP vs. REST: The Differences and Benefits Between the Two Widely-Used Web
1] Service Communication Protocols," 14 March 2017. [Online]. Available:
<https://stackify.com/soap-vs-rest/>. [Accessed 8 November 2021].

[1 P. Logan, "REST, SOAP, GraphQL — Gesundheit!," 22 September 2018. [Online]. Available:
2] <https://medium.com/better-practices/rest-soap-graphql-gesundheit-6544053f65cf#:~:text=REST%2C%20SOAP%2C%20GraphQL%20%E2%80%94%20Gesundheit%21%201%20REST.%20REST,time%20now.%20...%205%20GraphQL%20vs.%20REST.%20>. [Accessed
8 November 2021].

Appendix

index.html

```
<html>
```

```
<head>
```

```
<link rel="stylesheet" href="themes/kalan1.min.css" /> <!-- These stylesheets and scripts allow  
themes and features to be added to the webpage -->
```

```
<link rel="stylesheet" href="themes/jquery.mobile.icons.min.css" />
```

```
<link rel="stylesheet" href="http://code.jquery.com/mobile/1.4.5/jquery.mobile.structure-  
1.4.5.min.css" />
```

```
<script src="jquery-3.6.0.js"></script>
```

```
<script src="games.js"></script>
```

```
<script src="http://code.jquery.com/jquery-1.11.1.min.js"></script>
```

```
<script src="http://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.js"></script>
```

```
<style>
```

```
#pageheading {  
text-align: center;  
font-size: 80px;  
}
```

```
#mybutton.ui-btn { background-color : #79b9fb}
```

```
#leftpanel {  
background-color: #c5e0fc;  
}
```

```
.maintext{  
background-color: #99CCFF;  
text-align: center;  
}
```

```
.footer{
background-color: #99CCFF;
text-align: center;
}

hr{
background-color: Black;
margin-bottom: 40px;
}

</style> <!--This style tag allows things on the webpage to be styled in a certain way-->
```

```
<script language="JavaScript" type="text/javascript">

var myimg1=new Image(); <!-- This section adds four images that will be used in the function below-->
>

var myimg2=new Image();
var myimg3=new Image();
var myimg4=new Image();

myimg1.src="images/godofwar.jpg"; <!--These are the four images that have been added to the
webpage-->

myimg2.src="images/ghost.jpg";
myimg3.src="images/spiderman.jpg";
myimg4.src="images/thelastofus.jpg";

function changeimage(theimage,replacementimg,theimage) <!-- This function changes the image that
is displayed once the mouse rolls over them-->
{

theimage.src=replacementimg.src;

theconsolemessage=document.getElementById("conmessage")
thesecondmessage=document.getElementById("secmessage")
```

```
        if(thestate==1) <!--These are the messages that are displayed with the images-->
        {
            theconsolemessage.innerHTML="Ghost of Tsushima [2]"
        }
        else if(thestate==2)
        {
            theconsolemessage.innerHTML="God of War [1]"
        }
        else if(thestate==3)
        {
            thesecondmessage.innerHTML="The Last of Us [4]"
        }
        else{
            thesecondmessage.innerHTML="Spider-Man [3]"
        }
    }
</script>
</head>

<body>

<h1 id="pageheading">Video Games Database</h1> <!-- This adds a heading to the webpage-->

<div data-role="main" class="maintext"> <!-- This adds a navigation menu to the webpage-->
<h2>Navigation Menu</h2>
<a href="#leftpanel" id="mybutton" class="ui-btn ui-btn-inline">Open Panel</a> <!--Adds a button
that opens the panel when clicked-->
</div><br><br>
```

```
<div data-role="panel" id="leftpanel" data-display="push"> <!-- This adds the panel to the webpage-->
```

```
<h2>Information about the video games data in the database</h2> <!--This is the content that is displayed in the panel-->
```

```
<p>The database contains a lot of video games data. In the database, there are 100 video games.<br><br>
```

Each one of these video games has a unique 'Game Id' which allows you to easily identify each video game.

This allows you to easily retrieve a video game as you only need to enter its 'Game Id' to retrieve the rest of the information about the game.

Each video game in the database also has a 'Game Title'. This allows you to easily identify a game based off of its name.

This allows you to easily find a video game in the database as you can look for its title. You will then be able to see the

rest of the information about the game.

Each video game in the database also has a 'Game Description' which allows you to easily identify what the game is about.

This allows you to easily find a video game that you are interested in as you can read the description for each game until you find

a game that you like. Once you find a video game, you can easily read the rest of the information about the game.

Finally, each video game has a 'Genre Id' which allows you to easily identify the genre for each video game in the database.

This allows you to easily find video games which fit your genre preference. Once you find a video game based on your preferred genre,

you can easily find the rest of the information about the game.

This is all of the information about the video games data in the database. Click the 'Close Panel' button below to close the panel.

```
<br>
```

```
</p>
```

```
<a id="panelcancel" data-rel="close" data-swipe=close="true" >Close Panel</a> <!--Adds a button that closes the panel when clicked-->
```

```
</div>
```

```
<div class="maintext"> <!--First paragraph on webpage-->
```

```
<h2> About The Webpage</h2>
```

```
<p>This webpage contains information about video games data in a database. There is a side panel which contains information about the data that is stored in the database.
```

```
The section below talks about the importance of the webpage.
```

```
Below that, there are sections which allow you to add data to the database, edit data in the database, delete data in the database, and retrieve a certain piece of data in the database.
```

```
You will also be able to retrieve all of the data in the database by clicking the buttons at the bottom of the page.</p><br><br>
```

```
</div>
```

```
<div class="maintext"> <!--Second paragraph on webpage-->
```

```
<h2>Importance of The Webpage</h2>
```

```
 <!--changes the image when the mouse
rolls over it-->
```

```
<span id="conmessage">...</span> <!--Displays a message for the corresponding image-->
```

```

```

```
<span id="secmessage">...</span>
```

```
<p>This webpage is important as it allows you to easily manage the data that is in the video games database. The webpage allows you to manage the data in many ways.
```

```
The first way is by adding a new game. This is because there is a section where you can add in the details for a new game. Once the 'Add Game' button is clicked, the game will then be added to
```

```
the database. Another way the webpage allows you to manage the data is by allowing you to edit data that is in the database. This is because there is a section where you can edit existing data in the database.
```

```
The third way that the webpage allows you to manage the database is by easily allowing you to delete a piece of data by entering the 'Game Id'.
```

The final way that the webpage allows you to manage the database is by retrieving data in the database by entering the 'Game Id'. All of the data in the database can be retrieved by clicking the 'Get All Games' button.

This shows that the webpage allows you to easily manage the database in many different ways which is why the webpage is important.

</p>

</div>

<div class="maintext"> <!--This is where the webservice features are located-->

<h2>Games Database</h2>

<form id="postdata-form"> <!--This form allows the user to POST data-->

<h3>Add New Game</h3>

<p> This section of the webpage will allow you to add new games to the database.

</p>

<label for="title-text">Game Title:</label>

<input id="title-text" type="text" />

<label for="body-text">Game Description:</label>

<input id="body-text" type="text" />

<label for="user-id-text">Genre Id:</label>

<input id="user-id-text" type="text" />

<input id="ajaxpost-button" type="button" value="Add Game" />

<div id="postresults">Results...</div>

</form>

<hr>

</div>

<div class="maintext"> <!--This form allows the user to PUT data-->

```
<form id="putdata-form">
    <h3>Edit Existing Game</h3>
    <p> This section of the webpage will allow you to edit existing games in the
database. </p>
    <label for="title-text">Game Title:</label>
    <input id="title-text" type="text" /><br/>
    <label for="body-text">Game Description:</label>
    <input id="body-text" type="text" /><br/>
    <label for="user-id-text">Genre Id:</label>
    <input id="user-id-text" type="text" /><br/>
    <input id="ajaxput-button" type="button" value="Edit Game" />
    <div id="putresults">Results...</div>
</form>
<hr>
</div>
```

```
<div class="maintext"> <!--This form allows the user to DELETE data by id-->
<form id="deletedata-form">
    <h3>Delete Existing Game</h3>
    <p> This section of the webpage will allow you to delete existing games in the
database. </p>
    <label for="id-text">Game Id:</label>
    <input id="id-text" type="text" /><br/>
    <input id="ajaxdelete-button" type="button" value="Delete Game" />
    <div id="deleterresults">Results...</div>
</form>
<hr>
</div>
```

```
<div class="maintext"> <!--This form allows the user to GET data by id-->
<form id="retrievedata-form">
    <h3>Retrieve Existing Game</h3>
    <p> This section of the webpage will allow you to retrieve games from the database.
</p>
    <label for="getid-text">Game Id:</label>
    <input id="getid-text" type="text" /><br/>
    <input id="ajaxgetbyid-button" type="button" value="Retrieve Game" />
    <div id="getbyidresults">Results...</div>
</form>
<hr>
</div>
```

```
<div class="maintext"> <!--This form allows the user to Get all of the data in the database-->
<form id="getdata-form">
    <h3>Retrieve All Games</h3>
    <p> This section of the webpage will allow you to retrieve all of the games from the
database. </p>

    <input id="ajaxget-button" type="button" value="Get All Games" />
    <input id="ajaxget-p-button" type="button" value="Get All Games Using Promise" />
    <div id="loadtext">Results...</div>
</form>
<hr>
</div>
```

<div class="footer"> <!--This is the footer-->

<h2>References</h2>

<p>[1] A. Palumbo, "God of War Director on Why Single Player Games Are Awesome," 25 March 2018. [Online]. Available: <https://wccfttech.com/god-of-war-director-single-player-games/>. [Accessed 6 November 2021].

[2] R. Ramsey, "Feature: Ghost of Tsushima - 7 Things We Want from Future Updates," 6 August 2020. [Online]. Available: https://www.pushsquare.com/news/2020/08/feature_ghost_of_tsushima_-_7_things_we_want_from_future_updates. [Accessed 6 November 2021].

[3] S. Barker, "New Spider-Man PS4 Trophies Reveal More About First DLC," 20 October 2018. [Online]. Available: https://www.pushsquare.com/news/2018/10/new_spider-man_ps4_trophies_reveal_more_about_first_dlc. [Accessed 6 November 2021].

[4] "The Last of Us Wallpaper HD," 2 April 2017. [Online]. Available: <https://wallup.net/the-last-of-us-2/>. [Accessed 6 November 2021].

</p>

<h4>© Kalan 2021</h4>

</div>

</body>

</html>

games.js

```
$(function ()    // Functions for the buttons
```

```
{
```

```
    $.ajaxSetup({cache:false});
```

```
    $("#ajaxpost-button").click(function(e) //POST button
```

```
    {
```

```
        SendPost();
```

```
        e.preventDefault();
```

```
        return false;
```

```
    });
```

```
    $("#ajaxput-button").click(function(e)  //PUT button
```

```
    {
```

```
        PutPost();
```

```
        e.preventDefault();
```

```
        return false;
```

```
    });
```

```
    $("#ajaxdelete-button").click(function(e)    //DELETE button
```

```
    {
```

```
        DeletePost();
```

```
        e.preventDefault();
```

```
        return false;
```

```
    });
```

```
$("#ajaxgetbyid-button").click(function(e)    //GET by id button
{
    GetPost();
    e.preventDefault();
    return false;
});
```

```
$("#ajaxget-button").click(function(e)  //GET all button
{
    GetPosts();
    e.preventDefault();
    return false;
});
```

```
$("#ajaxget-p-button").click(function(e) //GET all with promise button
{
    GetPostsUsingPromise();
    e.preventDefault();
    return false;
});
});
```

```
function SendPost()
{
    var data = {
        title: $("#title-text").val(),
        body: $("#body-text").val(),
        userId: parseInt($("#user-id-text").val())
    };

    $.ajax({
        cache: false,
        type: "POST",
        url: "https://jsonplaceholder.typicode.com/posts",
        data: JSON.stringify(data),
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        success: GotPostResult,
        error: (req, status, error) => window.alert("AJAX error")
    });
}
```

```
function GotPostResult(json, status, req)
{
    var html = "";
    console.log(json);

    html += "<b>Game Id:</b> "+json.id + "<br/>"
        + "<b>Game Title:</b> "+json.title + "<br/>"
        + "<b>Game Description:</b> "+json.body + "<br/>"
        + "<b>Genre Id:</b> "+json.userId + "<br/>";

    $("#postresults").html(html);
}
```

```
}
```

```
function PutPost()
```

```
{
```

```
    var data = {
```

```
        title: $("#title-text").val(),
```

```
        body: $("#body-text").val(),
```

```
        userId: parseInt($("#user-id-text").val())
```

```
    };
```

```
    $.ajax({
```

```
        cache: false,
```

```
        type: "PUT",
```

```
        url: "https://jsonplaceholder.typicode.com/posts/1",
```

```
        data: JSON.stringify(data),
```

```
        contentType: "application/json; charset=utf-8",
```

```
        dataType: "json",
```

```
        success: (json, status, req) => $("#putresults").html("<p>Successfully Edited</p>"),
```

```
        error: (req, status, error) => window.alert("AJAX error")
```

```
    });
```

```
}
```

```
function DeletePost()
```

```
{
```

```
    var postId = parseInt($("#id-text").val());
```

```
    $.ajax({
```

```

        cache: false,
        type: "DELETE",
        url: `https://jsonplaceholder.typicode.com/posts/${postId}`,
        data: "",
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        success: (json, status, req) => $("#deleteresults").html("<p>Successfully Deleted</p>"),
        error: (req, status, error) => window.alert("AJAX error")
    });
}

```

```

function GetPost()
{
    var postId = parseInt($("#getid-text").val());

    $.ajax({
        cache: false,
        type: "GET",
        url: `https://jsonplaceholder.typicode.com/posts/${postId}`,
        data: "",
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        success: GotGetResultById,
        error: (req, status, error) => window.alert("AJAX error")
    });
}

```

```

function GotGetResultById(json, status, req)

```

```

{
    var html = "";

    html += "<b>Game Id:</b> "+json.id + "<br/>"
        + "<b>Game Title:</b> "+json.title + "<br/>"
        + "<b>Game Description:</b> "+json.body + "<br/>"
        + "<b>Genre Id:</b> "+json.userId + "<br/>";

    $("#getbyidresults").html(html);
}

function GetPosts()
{
    $.ajax({
        cache: false,
        type: "GET",
        url: "https://jsonplaceholder.typicode.com/posts",
        data: "",
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        success: GotAllPosts,
        error: (req, status, error) => window.alert("AJAX error")
    });
}

function GotAllPosts(json, status, req)
{
    var html = "";
    console.log(json);

```

```

    for(i in json)
    {
        html += "<b>Game Id:</b> "+json[i].id + "<br/>"
            + "<b>Game Title:</b> "+json[i].title + "<br/>"
            + "<b>Game Description:</b> "+json[i].body + "<br/>"
            + "<b>Genre Id:</b> "+json[i].userId + "<br/><br/>";
    }

    $("#loadtext").html(html);
}

```

```

function GetPostsUsingPromise()
{
    $.ajax({
        cache: false,
        type: "GET",
        url: "https://jsonplaceholder.typicode.com/posts",
        data: "",
        contentType: "application/json; charset=utf-8",
        dataType: "json"
    }).done((json, status, req) => {
        var html = "";
        console.log(json);
        for(i in json)
        {
            html += "<b>Game Id:</b> "+json[i].id + "<br/>"
                + "<b>Game Title:</b> "+json[i].title + "<br/>"
                + "<b>Game Description:</b> "+json[i].body + "<br/>"
                + "<b>Genre Id:</b> "+json[i].userId + "<br/><br/>";
        }
    });
}

```

```
    }  
    $("#loadtext").html(html);  
  }).fail((req, status, error) => {  
    window.alert("AJAX error");  
  });  
}
```

Kalan Assignment 1

GRADEMARK REPORT

FINAL GRADE

90/100

GENERAL COMMENTS

Instructor

Ajax & REST (45): 40

Written Research (45): 41

Demo (10): 9

Feedback:
=====

Quite a nice report, you could have identified characteristics for comparison.

Quite a nice site, using the fake webservices to create a video games information website.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23
