

# LeahCahillAssignmentOne

*by* Leah Cahill

---

|                |                                                                |                 |       |
|----------------|----------------------------------------------------------------|-----------------|-------|
| FILE           | 2254_LEAH_CAHILL_LEAHCAHILLASSIGNMENTONE_93287_1056836356.DOCX |                 |       |
| TIME SUBMITTED | 21-MAR-2018 10:11AM (UTC+0000)                                 | WORD COUNT      | 5332  |
| SUBMISSION ID  | 85196886                                                       | CHARACTER COUNT | 37998 |

# Emerging Web Technologies

ASSIGNMENT ONE

LEAH CAHILL (P140092)

## Table of Contents

|                                                         |    |
|---------------------------------------------------------|----|
| Part A (Understanding) .....                            | 2  |
| Volume .....                                            | 2  |
| Veracity .....                                          | 3  |
| Velocity .....                                          | 3  |
| Value .....                                             | 3  |
| Variety .....                                           | 4  |
| Hadoop Ecosystem.....                                   | 5  |
| Implementation .....                                    | 7  |
| Testing.....                                            | 10 |
| POST Method .....                                       | 10 |
| Second POST Method testing.....                         | 11 |
| GET Method .....                                        | 12 |
| Second GET Method .....                                 | 13 |
| DELETE Method.....                                      | 14 |
| Second DELETE Method .....                              | 15 |
| PUT Method .....                                        | 16 |
| Fetching data from XML file .....                       | 18 |
| References .....                                        | 20 |
| Appendix .....                                          | 22 |
| Code for client-side application- Webapplication4. .... | 22 |
| CorsService Code .....                                  | 31 |

## Part A (Understanding)

Big data is hugely important for the modern-day business and an example of an industry that makes use of big data applications is the banking and financial industry. In this section, big data and the following V's (volume, variety, velocity, value and veracity) will be discussed and how they relate to the banking and financial industry.

As banks involve millions of customers and their money, it is important to consider the security measures and ensure that they are secure to prevent crimes such as credit card and identity fraud or crimes such as money laundering. All personal details and information should be encrypted, and all transactions should be authorised. Big data prevents unauthorised transactions taking place and provides a certain level of security.

A more specific example of big data usage in the banking industry, is the Securities Exchange Commission (SEC) uses big data which allows them to monitor and catch illegal trading activities [1].

A specific example of finance-wise firms, American banks such as the Bank of America uses big data to enable them to understand their customers [2].

### Volume

Using big data, the systems can handle more volume when there are huge streams of data in comparison to the older systems, the older systems were not always able to process the ever-growing amount of data streaming in.

Big data enables banks to consider the huge streams of volume being generated every day and allows them to come up with a solution on where to store it all. The data will not be stored as a whole in one location, it will be broken up into parts and stored over several locations using distributed systems and then the data will be brought back together by software when required [3].

The data in big data applications is said to be much more accessible and this is helped by it being structured and organised. As there is so much of the data, searching and finding what you need on an older system would be complicated and time consuming, but big data infrastructure helps solve this issue.

Banks deal with millions of customers a day and have to be organised to deal with the volume of data they handle and how easily accessible it is when they require specific data and when they require it within a certain time frame, this leads to the point that big data is often relevant as it only contains the data needed [4].

The solution of looking after the company's data should be scalable as it is likely that the volume of data will grow, and the company should be prepared to grow with it. By using big data to analyse their current growth patterns, they could estimate the volume of data streaming and predict how fast it is likely to multiply to keep up and handle it [5] [6].

## Veracity

Veracity is the trustworthiness of the data in these applications and a customer should trust their banks as they are dealing with their finances.

As these systems process huge amounts of data, it has to be considered that not all of the data collected is going to be valuable or reliable in any sense. Analytics within the big data application help the company's focus on valid and accurate data [3].

The banks need to consider what data should be collected and how that data is going to be analysed and considered valuable to any decision making within the company, for example, if they are focusing on an account that they consider suspicious, then they need to collect data such as recent transactions and who owns the account. There is no need to collect data that is irrelevant and is no use to the company.

Considering the security of this data; big data can provide a level of security with secure infrastructures and can be used when the company is carrying out risk management. The banks can analyse and prioritize their risks by using big data to create scenario simulations or use pattern recognition to detect anything unordinary which may point to illegal activity such as fraud [7] [6]. Big data allows the banks to use prediction models and can search the huge amounts of data extensively and still create them in a relatively fast time.

The bank of America focused on their risk management with big data by moving from shared-services to a "Grid Computing" platform. This enables them to detect accounts that are considered high risk earlier than they would have with the older structure [8].

## Velocity

The velocity in big data is the frequency of incoming data that needs processing, and the company should consider how fast the data is coming in. Real-time data analysis and user behaviour analysis is used in banking. An example of this is if suspicious activity is recognised on anyone's account, they are then able to take action as soon as they gather the data rather than spend weeks waiting around until it is analysed, noticed and miss their chance to solve potentially illegal activity [9].

Banks can also use big data to analyse patterns regarding their services such as looking at patterns on interest rates and how it will affect the loans that they currently offer. The data would have to be analysed in an efficient time as interest rates are constantly changing and always have an impact on their rates, so they must stay up to date on the current trends and patterns [10].

## Value

The value of the data is important to consider because certain data may be used to make decisions for the company. Human errors such as data entry could affect the value of the data, if any of it entered is incorrect. Data redundancy is another factor to consider.

Analysing the data is valuable, as the Bank of America is able to discover and analyse other companies and predict how the future of the stock market may rise or fall. The value of this data is essential as they are able to prepare themselves for any changes [11].

Big data provides value to a company as banks are able to use big data to identify and create an in-depth customer profile. They can use this data in a positive way to personalise their marketing towards their targeted customers. An example of this is OCBC Bank in Singapore, OCBC bank focus on customer actions and aims to achieve a high standard of customer engagement and customer satisfaction [12].

Banks are able to analyse patterns in their branches to forecast situations such as whether a customer is close to closing their account or whether any branches could possibly close in the near future [12].

### Variety

The variety of big data relates to the structure of the application and whether it includes images, audio, whether it has a complete structure, is semi-structured or is unstructured.

The Bank of America uses big data to analyse the customer's details and access information such as their current credit cards or mortgage status and are able to make the decision to offer them deals based upon their situation, they have restructured their organisational structure to work with big data and allow it to become more effective within their decision making [2].

Big data is growing in the use of the banking industry and with ever-growing powerful machines and algorithms, banks are able to structure their systems to focus on areas such as financial analysis and really focus on identifying their customer's needs [13].

## Hadoop Ecosystem

Hadoop is an open source software library which helps companies gather data and allows them to store and analyse it in a way that is manageable. It is used within companies who use distributed computing and is applied to help the system become scalable [14].

It is called the Hadoop ecosystem because there are several modules that are associated that companies can use if it specifically works for them and their systems. Examples of Hadoop-related technologies are:

- Hive
- Pig
- Spark
- Zoo Keeper
- HBase

Hive is an open sourced project part of the Hadoop ecosystem. It is used to query data for analysis using Hive Query Language (HQL) and provides a database query interface.

Hive can integrate with the current existing SQL based tools and it is like SQL in some ways, however, it does have its real differences which will be discussed. Like SQL queries, applications can process Hive queries from a command line interface or open database connectivity [15].

Hive can only be used on data that has been formatted and structured. This may be time consuming if the data is not structured but the developer is also able to use features to make analysing the data easier once it is all sorted and structured and ready to go [16].

Hive has its limitations of taking quite a while to process and gather results for queries, this means that it is not suitable to make decisions with real-time querying and it is unsuitable for transaction processing applications. As it takes time to process queries, Hive could be used to look at calculating certain trends over periods of time and gather information on patterns rather than the real time decision making scenario, it could also be used to create reports of data. Hive has the disadvantage of not being compliant with ACID and not being able to support update statements [17].

Hive has the advantage for SQL users as they will feel comfortable using Hive Query Language when it is very familiar to SQL and allows the user to create user-defined functions if a function is not available that they wish to use. Developers are likely to find working with Hive relatively easy as it does not require them to write multiple lines of complex code.

Hive can be compared to the other technologies in the Hadoop ecosystem such as Pig which is also used for data manipulation and queries. These technologies should always be researched when the developer is considering which one to choose as they have some major differences such as Hive using HQL and pig using a language called Pig Latin or consider that Pig is able to handle big data using a high-level language. The data does not have to be completely structured to use Pig, whereas mentioned above with Hive, it must be structured to use [16].

When it comes to storing data, there is also HBase, which is another part of the Hadoop Ecosystem, as mentioned above that Hive is not for real-time data streaming, this could be a reason for a

developer to choose HBase over hive as it is used for real-time data streaming and especially when there is a large amount of data [18] These comparisons with the other technologies within the Hadoop ecosystem shows that each technology helps a company in a specific way to process, analyse and store data depending upon their needs and requirements.



## Implementation

The webpage created was a fictional dental practice which includes multiple sections where the user can interact with it. It includes a section where the user can view the treatments available interactively by buttons rather than displaying all the information on the page at once. Another section that allows the user to post a message, read it back, change or delete it. There is also a small section that shows a text file upon a button click and lastly, another section that allowed a user to post, read and delete their name.

### Basic Features implemented

```
$(function () {  
    $.ajaxSetup({ cache: false });  
  
    // ajax button  
    $("#Button1").click(AjaxTestbuttonClick);  
  
    // Handle submit  
    $("#form2").submit(DoSubmit);  
  
});  
  
function AjaxTestbuttonClick(e: JQueryEventObject)  
    //Load HTML  
    $("#loadedText").load("text2.txt");  
  
    e.preventDefault();  
    return false;  
  
}  
  
function DoSubmit(e: JQueryEventObject) {  
    let text = "read it";  
    text += $("#textBox").val(); // Read Text  
    $("#textBox").val(text); // Write it Back
```

#### Loading a simple text file with a submit button

```
<section id="one">  
    <form id="form2">  
        <input id="Button1"  
type="button" value="Click to check  
updates" />  
        <div  
id="loadedText"><p></p></div>  
    </form>  
</section>
```

Click to check updates

Dr Andrews will not be in until further notice, ring 07807912550 for more information.

The text loads when the button is clicked on by the user.

Fetching information from an XML file with AJAX.

treatments.xml

form2.ts

views.ts

HtmlPage1.html

Mess

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <HF>
3 <treatments>
4   <name>General Check Up</name>
5   <description> - - </description>
6   <price> £9. </price>
7   <doctor>
8     Dr Andrews.
9   </doctor>
10  <pagenum>
11    1 of 4
12  </pagenum>
13 </treatments>
14 <treatments>
15   <name>Bridges</name>
16   <description> Fixing and replacing a missing tooth.</descrip
17   <price> From £39. </price>
18   <doctor>
19     Dr L Jones.
20   </doctor>
21   <pagenum>
22     2 of 4
```

Treatments.xml. Left: screenshot of the XML file showing the syntax of XML

Code snippet of Views.ts showing the AJAX getting the elements by ID tags and displaying the element next to the appropriate label

WebApplication4

viewFunction

```
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
```

localhost:62651/HtmlPage1.html

Dental Surgery

Treatments at our dental practice

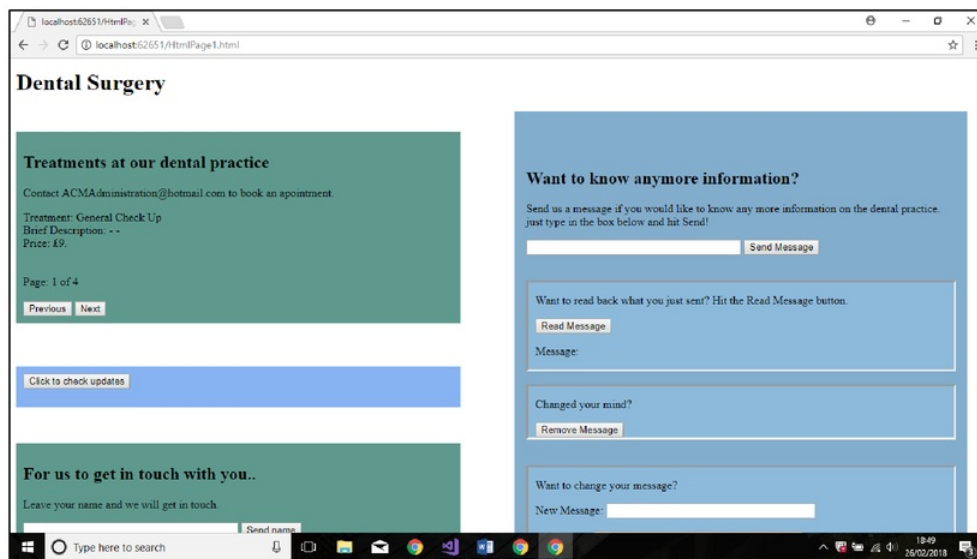
Contact ACMAAdministration@hotmail.com to book an appointment.

Treatment: Bridges  
Brief Description: Fixing and replacing a missing tooth.  
Price: From £39.

Page: 2 of 4

Previous Next

The text displayed in the homepage on the right. The functions forward() and back() called allows the user to navigate through the four treatments without refreshing the content on the whole page each time.



Screenshot of the basic webpage created. The content was divided into its own sections, so it was easy to display and understand what part of the page does what.

AJAX methods were used in this project: Example below of the Typescript and AJAX code below to link the buttons with the service. - a standard error message was set when the code does not work.

```
private AjaxSendMessageClick = (e: JQueryEventObject): boolean => {
    //Add new data in REST
    $.ajax({
        cache: false,
        type: "POST",
        url: "http://localhost:63514/MessageService.svc/postname",
        data: '{"Id": "1", "age": "21", "name": "' + $("#nameTextBox").val() +
    "" }',
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        success: this.AjaxSendMessageSuccessful,
        error: this.AjaxError
    });
    e.preventDefault();
    return false;
}

private AjaxSendMessageSuccessful = (json: any, status: string, req: JQueryXHR): void
=> // GOT success
{
    window.alert("Name sent.");
}

private AjaxError = (req: JQueryXHR, status: string, error: string): void => {
    window.alert("Got error\n status:" + status + "\n error:" + error + "\n");
}
```

## Testing

### POST Method

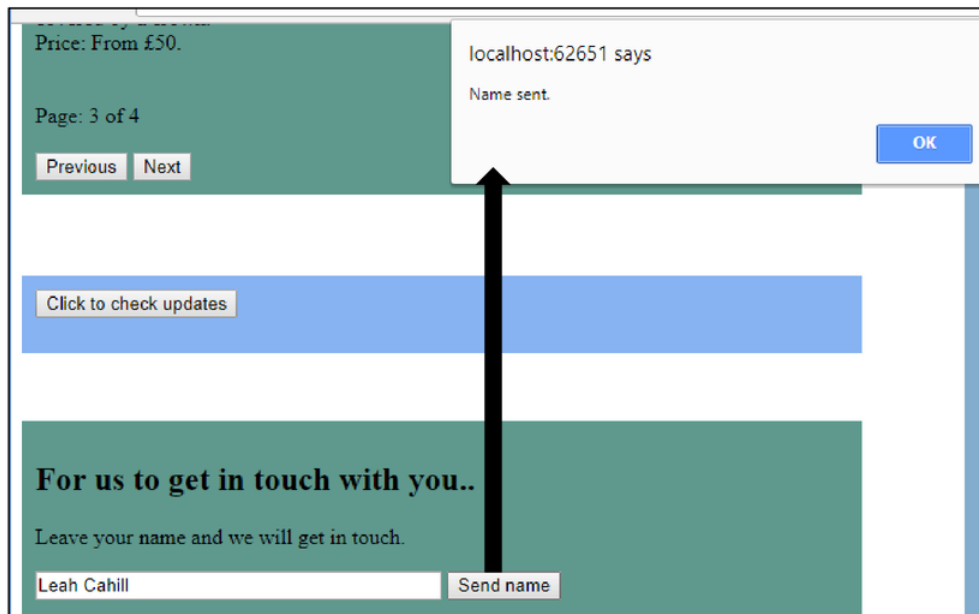


Screenshot showing the alert message when "Send Message" is clicked and the users message is sent successfully. POST Method used here as shown in the code snippet below that is in myService.ts

```
private AjaxSendMsgclick = (e: JQueryEventObject): boolean => {
    //Add new data in REST
    $.ajax({
        cache: false,
        type: "POST",
        url: "http://localhost:63514/MessageService.svc/postme",
        data: '{"sender" : "Leah", "recipient": "Jerry", "message": "Hello"}',
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        success: this.AjaxSendMsgSuccessful,
        error: this.AjaxError
    });
    e.preventDefault();
    return false;
}

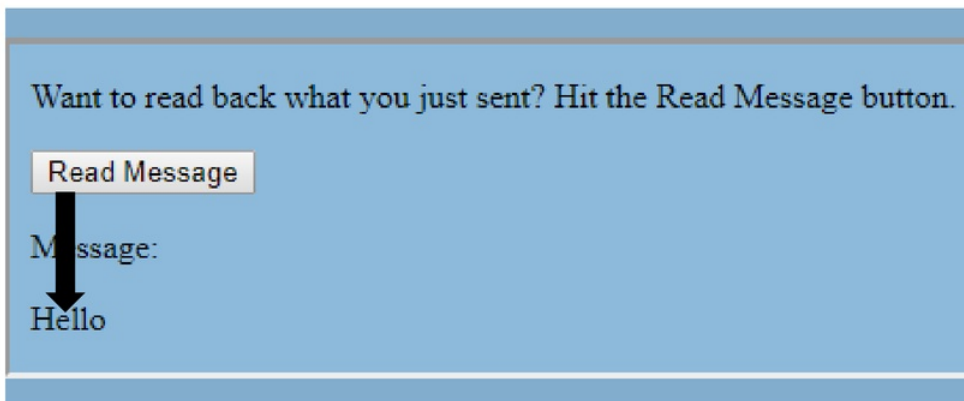
private AjaxSendMsgSuccessful = (json: any, status: string, responseText: string) => {
    window.alert("Messaged Sent");
}
```

## Second POST Method testing



The POST AJAX method was used here in the application, but this was accessed from myService22.ts

## GET Method



Here, it displays the message that was posted when the user clicks on "Read Message" button. The screenshot shows the Json object displayed in the HTML after the user have entered a message, clicked "Send Message" then "Read Message". The GET method was used here and is shown in the code snippet below.

The Json object is displayed when the GET is successful and is displayed in the HTML by  
"document.getElementById("msg").innerHTML = Json.Message;

```
private AjaxReadMsgclick = (e: JQueryEventObject): boolean => {
    $.ajax({
        cache: false,
        type: "GET",
        url: "http://localhost:63514/MessageService.svc/readmessage?ID=" + $("#msgIdText").val(),
        data: '{}',
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        success: this.AjaxReadMsgSuccessful,
        error: this.AjaxError
    });
    e.preventDefault();
    return false;
}
private AjaxReadMsgSuccessful = (json: any, status: string, req: JQueryXHR): void => {
    // window.alert("Read message\n status:" + status + "\n The message was: " + json.Message);
    document.getElementById("msg").innerHTML = json.Message;
}

private AjaxModifyMsgClick = (e: JQueryEventObject): boolean => { // simple one
    $.ajax({
```

**For us to get in touch with you..**

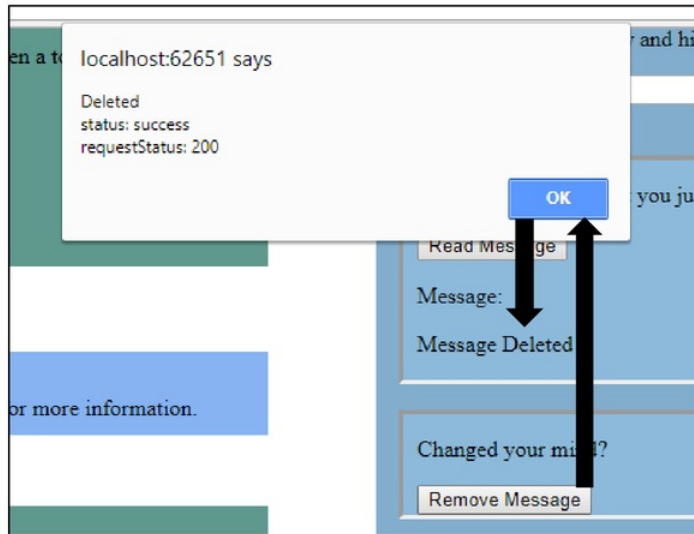
Leave your name and we will get in touch.

Leah Cahill

Thank you:  
Leah Cahill

Again, displaying the name in the HTML when the GET method has successfully received the users input.

## DELETE Method



Screenshot showing the alert when "Remove Message" button is clicked. The AJAX DELETE method used here.

The text in the html file also changes to inform user it is deleted.

The code is shown below for this DELETE method and shows in the screenshot how the HTML changes when successful.

```
private AjaxDeleteClick = (e) => {
    $.ajax({
        cache: false,
        type: "DELETE",
        url: "http://localhost:63514/MessageService.svc/deletemessage",
        data: '{}',
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        success: this.AjaxDeleteSuccessful,
        error: this.AjaxError
    });
    e.preventDefault();
    return false;
};

private AjaxDeleteSuccessful = (json, status, req) => {
    window.alert("Deleted\nstatus: " + status + "\nrequestStatus: " + req.status);
    document.getElementById("msg").innerHTML = "Message Deleted";
};
```

## Second DELETE Method

**For us to get in touch with you..**

Leave your name and we will get in touch.

Leah Cahill

---

Thank you:

Name CLEARED



Another DELETE method used here and informs the user that the name has been cleared.

## PUT Method



Want to change your message?

New Message:

Message ID:

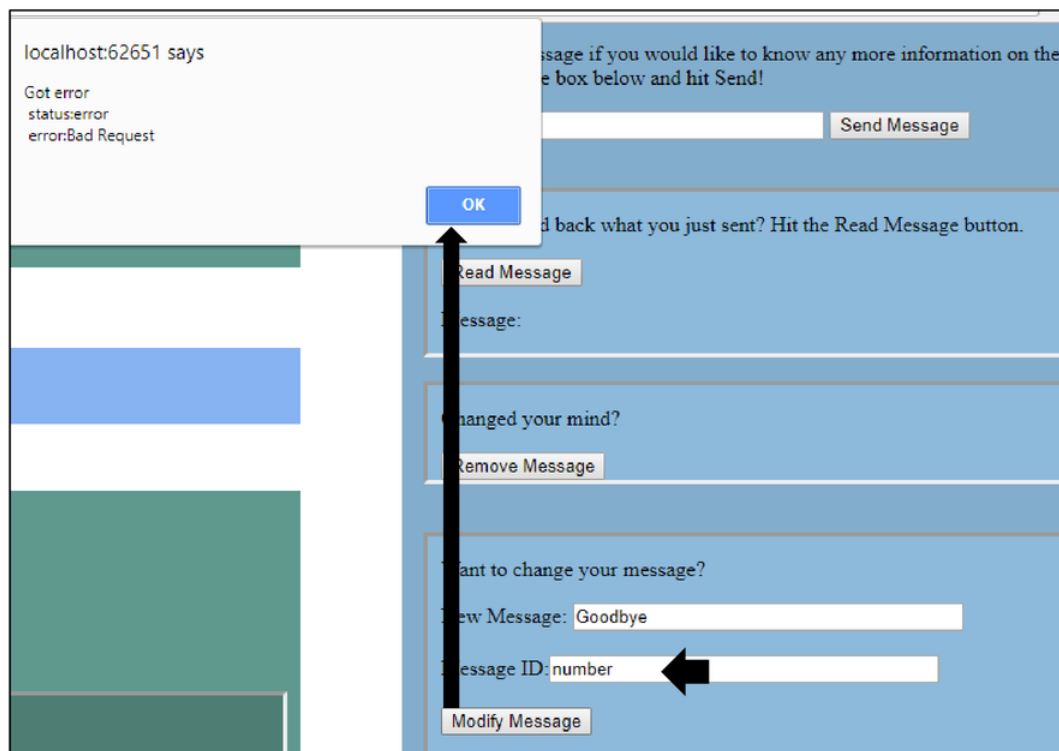
↓

Successful.Changes have been made

The PUT method was used here which allows the user to change their message when they type in an appropriate ID with it. The PUT method is shown in the screenshot below.

```
private AjaxModifyMsgClick = (e: JQueryEventObject): boolean => { // simple one
    $.ajax({
        cache: false,
        type: "PUT",
        url: "http://localhost:63514/MessageService.svc/putmessage?ID=" + $("#msgIdTextBox").val(),
        data: '{"sender": "Leah", "recipient": "ME", "message": "' + $("#msgTextBox2").val() + "'",
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        success: this.AjaxModifyMsgSuccessful,
        error: this.AjaxError
    });
    e.preventDefault();
    return false;
}

private AjaxModifyMsgSuccessful = (json: any, status: string, req: JQueryXHR): void => { //Modify
    let done = json.ChangesComplete; // See interface file for ChangesComplete
    // window.alert("Successful. Changes have been made");
    document.getElementById("msg2").innerHTML = "Successful.Changes have been made";
}
```

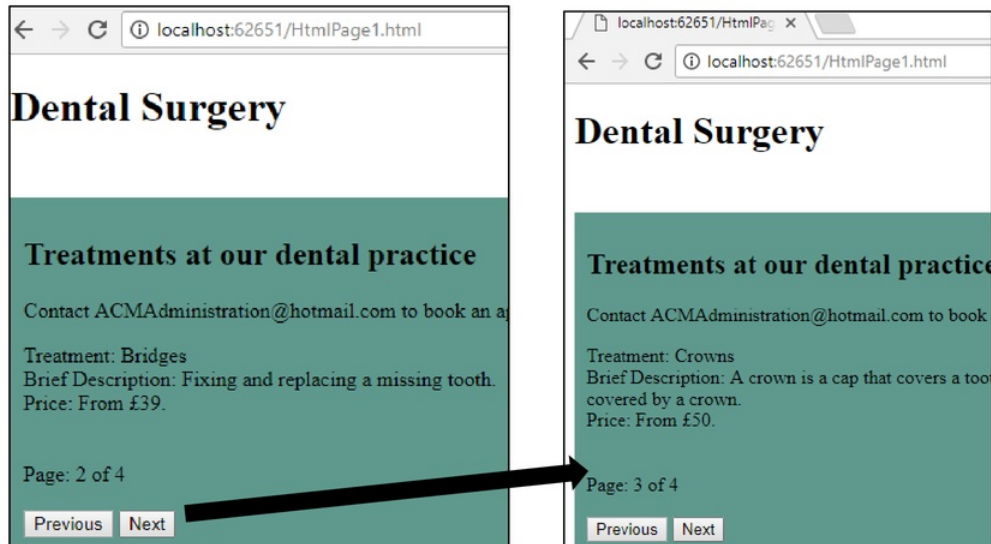


This screenshot shows how the page alerts the user with an error message when the message ID entered is invalid.

Code below shows the error message in the typescript file for when the AJAX requests return an error instead of success message.

```
private AjaxError = (req: JQueryXHR, status: string, error: string): void => {
    window.alert("Got error\n status:" + status + "\n error:" + error + "\n");
}
```

Fetching data from XML file



Screenshot showing how the user can browse information in a section of the page using "Next" and "Previous" buttons, without the entire webpage reloading. This information was saved in an XML file and fetched with AJAX.

Code below showing the function that displays the XML on the HTML page.

```
var i = 0, len;
displayTreatments(i);

function displayTreatments(i) {
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) {
            viewFunction(this, i);
        }
    };
    xmlhttp.open("GET", "treatments.xml", true);
    xmlhttp.send();
}

function viewFunction(xml, i) {
    var xmlDoc = xml.responseXML;
    var notetext = xmlDoc.getElementsByTagName("treatments");
    len = notetext.length;
    document.getElementById("showT").innerHTML =
        "Treatment: " +
        notetext[i].getElementsByTagName("name")[0].childNodes[0].nodeValue +
        "<br>Brief Description: " +
        notetext[i].getElementsByTagName("description")[0].childNodes[0].nodeValue +
        "<br>Price: " +
        notetext[i].getElementsByTagName("price")[0].childNodes[0].nodeValue +
        "<br><br>Page:" +
        notetext[i].getElementsByTagName("pagenum")[0].childNodes[0].nodeValue;
}
```

Summary of interactions on page.

The screenshot shows a web browser displaying a page with two main sections. The left section, titled 'Treatments at our dental practice', contains text about a dental practice and a 'Click to check updates' button. The right section, titled 'Want to know anymore information?', contains a form for sending a message, a 'Read Message' button, a 'Remove Message' button, and a 'Modify Message' button. Annotations with arrows point to various elements on the page, describing the AJAX interactions:

- Fetching data from an XML file**: Points to the 'Click to check updates' button.
- Sending message. POST method in AJAX.**: Points to the 'Send Message' button.
- Reading message. GET method in AJAX.**: Points to the 'Read Message' button.
- Delete message. DELETE method in AJAX.**: Points to the 'Remove Message' button.
- Modify message. PUT method in AJAX.**: Points to the 'Modify Message' button.
- Reading name. GET method in AJAX.**: Points to the 'Display name entry' button.
- Delete name. DELETE method in AJAX.**: Points to the 'Remove name?' button.
- Sending name. POST method in AJAX.**: Points to the 'Send name' button.
- Displaying a simple text file with a button click.**: Points to the 'Click to check updates' button.

## References

- [1] M. Gaitho, "How Applications of Big Data Drive Industries," Simplilearn, 20 December 2017. [Online]. Available: <https://www.simplilearn.com/big-data-applications-in-industries-article>. [Accessed 11 February 2018].
- [2] T. Groenfeldt, "Banks Use Big Data To Understand Customers Across Channels," Forbes, 11 June 2013. [Online]. Available: <https://www.forbes.com/sites/tomgroenfeldt/2013/06/11/banks-use-big-data-to-understand-customers-across-channels/#4214573c3218>. [Accessed 18 March 2018].
- [3] B. Marr, "Big Data: The 5 Vs Everyone Must Know," 6 March 2014. [Online]. Available: <https://www.linkedin.com/pulse/20140306073407-64875646-big-data-the-5-vs-everyone-must-know>. [Accessed 18 March 2018].
- [4] D. Karr, "What is Big Data? What Are the Benefits of Big Data?," MarTech, 8 March 2017. [Online]. Available: <https://martech.zone/benefits-of-big-data/>. [Accessed 11 February 2018].
- [5] J. Tee, "Maximizing and managing big data with SOA middleware," The Server Side, [Online]. Available: <http://www.theserverside.com/feature/Handling-the-four-Vs-of-big-data-volume-velocity-variety-and-veracity>. [Accessed 11 February 2018].
- [6] Mauricio, "The role of big data in the banking industry," Big Data Made Simple, 03 May 2016. [Online]. Available: <http://bigdata-madesimple.com/role-big-data-banking-industry/>. [Accessed 18 March 2018].
- [7] D. A. Huber, "Big data: Potentials from a risk management perspective," Banking Hub, 2 July 2014. [Online]. Available: <https://www.bankinghub.eu/banking/finance-risk/big-data-potentials-from-a-risk-management-perspective>. [Accessed 11 February 2018].
- [8] J. B. O. A. S. d. R. S. K. L. C. V. C. Jean Coumaros, "Big Data Alchemy: How can Banks Maximize the Value of their Customer Data?," [Online]. Available: [https://www.capgemini.com/consulting-se/wp-content/uploads/sites/38/2017/08/big-data-customer-analytics-in-banks\\_0\\_0.pdf](https://www.capgemini.com/consulting-se/wp-content/uploads/sites/38/2017/08/big-data-customer-analytics-in-banks_0_0.pdf). [Accessed 18 March 2018].
- [9] "How does Big Data help with financial risk management?," icar, 17 March 2016. [Online]. Available: <https://www.icarvision.com/en/how-does-big-data-help-with-financial-risk-management->. [Accessed 11 February 2018].
- [10] "The Impact of Big Data in Market Risk Management," analyticsweek, 3 October 2015. [Online]. Available: <https://analyticsweek.com/content/the-impact-of-big-data-in-market-risk-management/>. [Accessed 11 February 2017].
- [11] R. Kade, "Here Are Bank of America's Revelations of the Future of Big Data," Smart Data Collective, 8 September 2017. [Online]. Available: <https://www.smartdatacollective.com/bank-america-revelations-future-of-big-data/>. [Accessed 18 March 2018].

- [12] D. C. H. R. S.-P. M. Yook-Pei Shee, "Big data in banking for marketers," [Online]. Available: <https://www.evry.com/globalassets/insight/bank2020/bank-2020---big-data---whitepaper.pdf>. [Accessed 18 March 2018].
- [13] "The force awakens: Big Data in banking," Fin Extra, 01 March 2016. [Online]. Available: <https://www.finextra.com/newsarticle/28541/the-force-awakens-big-data-in-banking>. [Accessed 18 March 2018].
- [14] "Welcome to Apache™ Hadoop®!", Apache Hadoop, 18 December 2017. [Online]. Available: <http://hadoop.apache.org/>. [Accessed 2017 February 11].
- [15] "Apache Hive," IBM Analytics, [Online]. Available: <https://www.ibm.com/analytics/hadoop/hive>. [Accessed 11 February 2018].
- [16] H. T. Team, "Hive vs Pig- When to use Apache Hive and When Apache Pig," HDFS Tutorial, 29 July 2017. [Online]. Available: <https://www.hdfstutorial.com/blog/hive-vs-pig/>. [Accessed 11 February 2018].
- [17] S. Neumann, "Hive vs. HBase," Xplenty, 26 May 2014. [Online]. Available: <https://www.xplenty.com/blog/hive-vs-hbase/>. [Accessed 11 February 2018].
- [18] "Hive vs.HBase–Different Technologies that work Better Together," DeZyre, 7 December 2016. [Online]. Available: <https://www.dezyre.com/article/hive-vs-hbase-different-technologies-that-work-better-together/322>. [Accessed 11 February 2018].

## Appendix

Code for client-side application- Webapplication4.

### HtmlPage1.html

```
<!DOCTYPE html>
<html>
<head>

    <link href="StyleSheet1.css" rel="stylesheet" />
    <title></title>
    <meta charset="utf-8" />
</head>
<body>
    <h1> Dental Surgery </h1>
    <section id="two">
        <form id="form1">
            <h1> Want to know anymore information?</h1>

            <p>
                Send us a message if you would like to know any more information on
the dental practice.
                <br /> just type in the box below and hit Send!
            </p>
            <input id="msgTextBox" type="text" />
            <input id="sendMsgButton" value="Send Message" type="button" />
            <br /><br /><br />

            <div class="inset">
<p>Want to read back what you just sent? Hit the Read Message button.</p><input
id="readMsgButton" value="Read Message" type="button" />
                <p>Message: </p><p id="msg"></p>
            </div>
            <br />
            <div class="inset"> <p> Changed your mind?</p><input id="deleteMsgButton"
value="Remove Message" type="button" />
                <br />
            </div>
            <br /><br />
            <div class="inset">
                <p>Want to change your message?</p>
                <label>New Message:</label> <input id="msgTextBox2" type="text" /><br /><br />
                <label>Message ID:</label><input id="msgIdTextBox" type="text" /><br /><br />
                <input id="simpleModifyMsgButton" value="Modify Message" type="button"/>
                <p id="msg2"></p>
                <br /><br />
            </div>
        </form>

    </section>

    <section id="three">
        <h2>Treatments at our dental practice</h2>
        <p> Contact ACMAAdministration@hotmail.com to book an apointment.</p>
        <div id="showT"></div><br />
        <input type="button" onclick="back()" value="Previous" />
        <input type="button" onclick="forward()" value="Next" />

    </section>
```

```

<section id="one">
  <form id="form2">
    <input id="Button1" type="button" value="Click to check updates" />
    <div id="loadedText"><p></p></div>
  </form>
</section>

<section id="three">
  <h1>For us to get in touch with you..</h1>
  <p>Leave your name and we will get in touch.</p>
  <input id="nameTextBox" type="text" /><!--<input id="nameIdTextBox"
type="text" />-->
  <input id="sendnameButton" value="Send name" type="button" />
  <br /><br />
  <div class="inset1">
    <input id="readnameButton" value="Display name entry" type="button" />
  <input id="deletenameButton" value="Remove name?" type="button" />
  <p>Thank you: </p><p id="namep"></p>
  <br /><br /><br />
</div>

</section>

<script src="scripts/jquery-3.3.1.js"></script>
<script src="myService.js"></script>
<script src="myService22.js"></script>
<script src="views.js"></script>
<script src="form2.js"></script>
</body>
</html>

```

#### myService.ts

```

class myService {
  PageLoad(): void {
    $("#sendMsgButton").click(this.AjaxSendMsgclick);
    $("#readLastMsgButton").click(this.AjaxReadLastMsgclick);
    $("#readMsgButton").click(this.AjaxReadMsgclick);
    $("#simpleModifyMsgButton").click(this.AjaxModifyMsgClick);
    $("#deleteMsgButton").click(this.AjaxDeleteClick);
  }

  private AjaxSendMsgclick = (e: JQueryEventObject): boolean => {
    //Add new data in REST
    $.ajax({
      cache: false,
      type: "POST",
      url: "http://localhost:63514/MessageService.svc/postmessage",
      data: '{"sender" : "Leah", "recipient": "Jerry", "message" : "' +
$("#msgTextBox").val() + '" }',
      contentType: "application/json; charset=utf-8",
      dataType: "json",
      success: this.AjaxSendMsgSuccessful,
      error: this.AjaxError
    });
    e.preventDefault();
    return false;
  }
}

```

```

private AjaxSendMsgSuccessful = (json: any, status: string, req: JQueryXHR): void
=> // GOT success message
{
    window.alert("Messaged Sent");
}

private AjaxReadLastMsgclick = (e: JQueryEventObject): boolean => {
    $.ajax({
        cache: false,
        type: "GET",
        url: "http://localhost:63514/MessageService.svc/readmessage",
        data: '{}',
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        success: this.AjaxReadLastMsgSuccessful,
        error: this.AjaxError
    });
    e.preventDefault();
    return false;
}
private AjaxReadLastMsgSuccessful = (json: any, status: string, req: JQueryXHR):
void => { // READ success message
    // window.alert("Last message\n status:" + status + "\n The message was: " +
    json.Message + "\n"); // Message comes from return type in Imessageservice
    document.getElementById("msgprevious").innerHTML = json.Message;
}

private AjaxReadMsgclick = (e: JQueryEventObject): boolean => {
    $.ajax({
        cache: false,
        type: "GET",
        url: "http://localhost:63514/MessageService.svc/readmessage?ID=" +
$("#msgIdTextBox").val(),
        data: '{}',
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        success: this.AjaxReadMsgSuccessful,
        error: this.AjaxError
    });
    e.preventDefault();
    return false;
}
private AjaxReadMsgSuccessful = (json: any, status: string, req: JQueryXHR): void
=> { //READ ALL success message
    // window.alert("Read message\n status:" + status + "\n The message was: " +
    json.Message + "\n"); // Message comes from return type in Imessageservice
    document.getElementById("msg").innerHTML = json.Message;
}

private AjaxModifyMsgClick = (e: JQueryEventObject): boolean => {
    $.ajax({
        cache: false,
        type: "PUT",
        url: "http://localhost:63514/MessageService.svc/putmessage?ID=" +
$("#msgIdTextBox").val(),
        data: '{"sender" : "Leah", "recipient": "ME", "message" : "' +
$("#msgTextBox2").val() + '" }',

```

```

        contentType: "application/json; charset=utf-8",
        dataType: "json",
        success: this.AjaxModifyMsgSuccessful,
        error: this.AjaxError
    });
    e.preventDefault();
    return false;
}

private AjaxModifyMsgSuccessful = (json: any, status: string, req: JQueryXHR):
void => { //Modify success message
    let done = json.ChangesComplete;
    // window.alert("Successful. Changes have been made");
    document.getElementById("msg2").innerHTML = "Successful.Changes have been
made";
}

private AjaxDeleteClick = (e) => {
    $.ajax({
        cache: false,
        type: "DELETE",
        url: "http://localhost:63514/MessageService.svc/deletemessage",
        data: '{}',
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        success: this.AjaxDeleteSuccessful,
        error: this.AjaxError
    });
    e.preventDefault();
    return false;
};

private AjaxDeleteSuccessful = (json, status, req) => {
    window.alert("Deleted\nstatus: " + status + "\nrequestStatus: " + req.status);
    document.getElementById("msg").innerHTML = "Message Deleted";
};

private AjaxError = (req: JQueryXHR, status: string, error: string): void => {
    window.alert("Got error\n status:" + status + "\n error:" + error + "\n");
}

}
$(function () {
    let page: myService = new myService
    page.PageLoad();
    //(new Service()).PageLoad();
});

```

#### myService22.ts

```

class myService2 {
    PageLoad(): void {

        $("#deletenameButton").click(this.AjaxDelete2Click);
        $("#sendnameButton").click(this.AjaxSendNameclick);
        $("#readnameButton").click(this.AjaxreadNameclick);
    }
}

```

```

private AjaxSendNameClick = (e: JQueryEventObject): boolean => {
    //Add new data in REST
    $.ajax({
        cache: false,
        type: "POST",
        url: "http://localhost:63514/MessageService.svc/postname",
        data: '{"Id" : "1", "age": "21", "name" : "' + $("#nameTextBox").val() +
'" }',
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        success: this.AjaxSendNameSuccessful,
        error: this.AjaxError
    });
    e.preventDefault();
    return false;
}

private AjaxSendNameSuccessful = (json: any, status: string, req: JQueryXHR): void
=> // GOT success
{
    window.alert("Name sent.");
}

private AjaxreadnameClick = (e: JQueryEventObject): boolean => {
    $.ajax({
        cache: false,
        type: "GET",
        url: "http://localhost:63514/MessageService.svc/readname",
        data: '{}',
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        success: this.AjaxReadNameSuccessful,
        error: this.AjaxError
    });
    e.preventDefault();
    return false;
}

private AjaxReadNameSuccessful = (json: any, status: string, req: JQueryXHR): void
=> { // READ success message
    // window.alert("Last message\n status:" + status + "\n The message was: " +
json.Message + "\n"); // Message comes from return type in Imessageservice
    document.getElementById("namep").innerHTML = json.Name;
}

private AjaxDelete2Click = (e) => {
    $.ajax({
        cache: false,
        type: "DELETE",
        url: "http://localhost:63514/MessageService.svc/deletename",
        data: '{}',
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        success: this.AjaxDelete2Successful,
        error: this.AjaxError
    });
    e.preventDefault();
}

```

```

        return false;
    };

    private AjaxDelete2Successful = (json, status, req) => {
        window.alert("Deleted\nstatus: " + status + "\nrequestStatus: " + req.status);
        document.getElementById("namep").innerHTML = "Name CLEARED";
    };

    private AjaxError = (req: JQueryXHR, status: string, error: string): void => {
        window.alert("Got error\n status:" + status + "\n error:" + error + "\n");
    }
}

$(function () {
    let page: myService2 = new myService2
    page.PageLoad();
    //(new Service()).PageLoad();
});

```

### Form2.ts

```

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

    // ajax button
    $("#Button1").click(AjaxTestbuttonClick);

    // Handle submit
    $("#Form2").submit(DoSubmit);
});

function AjaxTestbuttonClick(e: JQueryEventObject) {
    //Load HTML
    $("#loadedText").load("text2.txt");

    e.preventDefault();
    return false;
}

function DoSubmit(e: JQueryEventObject) {
    let text = "read it";
    text += $("#textBox").val(); // Read Text
    $("#textBox").val(text); // Write it Back
    // e.preventDefault();
    // return false;
}

```

### Views.ts

```
var i = 0, len;
displayTreatments(i);

function displayTreatments(i) {
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) {
            viewFunction(this, i);
        }
    };
    xmlhttp.open("GET", "treatments.xml", true);
    xmlhttp.send();
}

function viewFunction(xml, i) {
    var xmlDoc = xml.responseXML;
    var notetext = xmlDoc.getElementsByTagName("treatments");
    len = notetext.length;
    document.getElementById("showT").innerHTML =
        "Treatment: " +
        notetext[i].getElementsByTagName("name")[0].childNodes[0].nodeValue +
        "<br>Brief Description: " +
        notetext[i].getElementsByTagName("description")[0].childNodes[0].nodeValue +
        "<br>Price: " +
        notetext[i].getElementsByTagName("price")[0].childNodes[0].nodeValue +
        "<br><br><br>Page:" +
        notetext[i].getElementsByTagName("pagenum")[0].childNodes[0].nodeValue;
}

function forward() {
    if (i < len - 1) {
        i++;
        displayTreatments(i);
    }
}

function back() {
    if (i > 0) {
        i--;
        displayTreatments(i);
    }
}
```

### Treatments.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<HF>
    <treatments>
        <name>General Check Up</name>
        <description> - - </description>
        <price> £9. </price>
        <doctor>
            Dr Andrews.
        </doctor>
        <pagenum>
            1 of 4
        </pagenum>
    </treatments>
```

```

<treatments>
  <name>Bridges</name>
  <description> Fixing and replacing a missing tooth.</description>
  <price> From £39. </price>
  <doctor>
    Dr L Jones.
  </doctor>
  <pagenum>
    2 of 4
  </pagenum>
</treatments>
<treatments>
  <name>Crowns</name>
  <description> A crown is a cap that covers a tooth, usually when a tooth is
broken, it is covered by a crown.</description>
  <price> From £50. </price>
  <doctor>
    Dr Andrews.
  </doctor>
  <pagenum>
    3 of 4
  </pagenum>
</treatments>
<treatments>
  <name>Fillings</name>
  <description> Repairs holes in teeth cause by decay. </description>
  <price> From £19 </price>
  <doctor>
    Dr Andrews.
  </doctor>
  <pagenum>
    4 of 4
  </pagenum>
</treatments>
</HF>

```

#### Text2.txt.

Dr Andrews will not be in until further notice,  
ring 07807912550 for more information.

#### Stylesheet1.css

```

ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
}

li a {
  text-align: center;
  padding: 14px 16px;
}

#two {
  background-color: #83ADCC;
  float: right;
  width: 600px;
}

```

```

padding: 1em;
height: 800px;
color: black;
padding-top: 60px;
margin-right: 5px;
}

#one {
background-color: #87B3F2;
max-width: 600px;
/*float:right;*/
padding: 1em;
max-height: 600px;
color: black;
margin-top: 60px;
padding: 10px;
}

#three {
background-color: #5F998E;
max-width: 600px;
/*float:right;*/
padding: 1em;
max-height: 600px;
color: black;
margin-top: 50px;
padding: 10px;
}

#three {
background-color: #5F998E;
/*float: left;
max-width: 600px;
margin-left: 50px;
max-height: 600px;
color: black;
margin-top: 10px;*/
}

input[type=text] {
width: 50%;
box-sizing: border-box;
height: 30%;
}

.inset {
border-style: inset;
padding-left: 10px;
background-color: #8DBADB;
}

.inset1 {
border-style: inset;
padding-left: 10px;
background-color: #4E7D74;
}

.msg {
font-weight: bold;
}

```

## CorsService Code

### Web.Config

```
<?xml version="1.0"?>
<configuration>

  <appSettings>
    <add key="aspnet:UseTaskFriendlySynchronizationContext" value="true"/>
  </appSettings>
  <system.web>
    <compilation debug="true" targetFramework="4.5.2"/>
    <httpRuntime targetFramework="4.5.2"/>
    <httpModules>
      <add name="ApplicationInsightsWebTracking"
type="Microsoft.ApplicationInsights.Web.ApplicationInsightsHttpModule,
Microsoft.AI.Web"/>
    </httpModules>
  </system.web>
  <system.serviceModel>
    <behaviors>
      <!-- Changes ive made -->
      <endpointBehaviors>
        <behavior name="AjaxBehaviour">
          <webHttp/>
        </behavior>
      </endpointBehaviors>
      <serviceBehaviors>

        <behavior>
          <!-- To avoid disclosing metadata information, set the values below to false
before deployment -->
          <serviceMetadata httpGetEnabled="true" httpsGetEnabled="true"/>
          <!-- To receive exception details in faults for debugging purposes, set the
value below to true. Set to false before deployment to avoid disclosing exception
information -->
          <serviceDebug includeExceptionDetailInFaults="false"/>
        </behavior>
      </serviceBehaviors>
    </behaviors>
    <!-- my changes -->
    <services>
      <service name="CorsService.MessageService">
        <endpoint
          address=""
          behaviorConfiguration="AjaxBehaviour"
          binding="webHttpBinding"
          contract="CorsService.IMessageService"/>
      </service>
    </services>

    <protocolMapping>
      <add binding="basicHttpsBinding" scheme="https"/>
    </protocolMapping>
    <serviceHostingEnvironment aspNetCompatibilityEnabled="true"
multipleSiteBindingsEnabled="true"/>
  </system.serviceModel>
  <system.webServer>
    <modules runAllManagedModulesForAllRequests="true">
      <remove name="ApplicationInsightsWebTracking"/>
    </modules>
  </system.webServer>
</configuration>
```

```

        <add name="ApplicationInsightsWebTracking"
type="Microsoft.ApplicationInsights.Web.ApplicationInsightsHttpModule,
Microsoft.AI.Web"
        preCondition="managedHandler"/>
    </modules>
    <!--
        To browse web app root directory during debugging, set the value below to
true.
        Set to false before deployment to avoid disclosing web app folder information.
    -->
    <directoryBrowse enabled="true"/>
    <validation validateIntegratedModeConfiguration="false"/>
</system.webServer>

</configuration>

```

### IMessageService.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;

namespace CorsService
{
    // NOTE: You can use the "Rename" command on the "Refactor" menu to change the
interface name "IService1" in both code and config file together.
    [ServiceContract]
    public interface IMessageService
    {
        //create
        [OperationContract]
        [WebInvoke(Method = "POST",
        BodyStyle = WebMessageBodyStyle.Wrapped,
        RequestFormat = WebMessageFormat.Json,
        ResponseFormat = WebMessageFormat.Json,
        UriTemplate = "/postmessage")]

        void SendMessage(string sender, string recipient, string message);

        [OperationContract]
        [WebGet(
        BodyStyle = WebMessageBodyStyle.Wrapped,
        RequestFormat = WebMessageFormat.Json,
        ResponseFormat = WebMessageFormat.Json,
        UriTemplate = "/readmessage")]
        [return: MessageParameter(Name = "Message")] //return Json
        "message":"...message text..."
        string ReadLastMessage();

        [OperationContract]
        [WebGet(
        BodyStyle = WebMessageBodyStyle.Wrapped,
        RequestFormat = WebMessageFormat.Json,
        ResponseFormat = WebMessageFormat.Json,
        UriTemplate = "/readmessage?id={msgID}")]
        [return: MessageParameter(Name = "Message")]

```

```

string ReadMessage(int msgID);

[OperationContract]
[WebInvoke(Method = "PUT",
    BodyStyle = WebMessageBodyStyle.Wrapped,
    RequestFormat = WebMessageFormat.Json,
    ResponseFormat = WebMessageFormat.Json,
    UriTemplate = "/putmessage?id={msgID}")]
[return: MessageParameter(Name = "Modifications")]
EmailMessage ModifyMessage(int msgID, EmailMessage messageChanges);

[OperationContract]
[WebInvoke(Method = "PUT",
    BodyStyle = WebMessageBodyStyle.Wrapped,
    RequestFormat = WebMessageFormat.Json,
    ResponseFormat = WebMessageFormat.Json,
    UriTemplate = "/simpleputmessage?id={msgID}")]
[return: MessageParameter(Name = "ChangesComplete")]
bool ModifyMessage1(int msgID, string sender, string recipient, string
message);

[OperationContract]
[WebInvoke(
    Method = "DELETE",
    BodyStyle = WebMessageBodyStyle.Wrapped,
    RequestFormat = WebMessageFormat.Json,
    ResponseFormat = WebMessageFormat.Json,
    UriTemplate = "/deletemessage")]

void DeleteMessage(int id);

[OperationContract]
[WebInvoke(Method = "POST",
    BodyStyle = WebMessageBodyStyle.Wrapped,
    RequestFormat = WebMessageFormat.Json,
    ResponseFormat = WebMessageFormat.Json,
    UriTemplate = "/postname")]

void SendName(string ID, string name, int age);

[OperationContract]
[WebGet(
    BodyStyle = WebMessageBodyStyle.Wrapped,
    RequestFormat = WebMessageFormat.Json,
    ResponseFormat = WebMessageFormat.Json,
    UriTemplate = "/readname")]
[return: MessageParameter(Name = "Name")] //return Json "Name"
string Readname();

[OperationContract]
[WebInvoke(
    Method = "DELETE",
    BodyStyle = WebMessageBodyStyle.Wrapped,
    RequestFormat = WebMessageFormat.Json,
    ResponseFormat = WebMessageFormat.Json,
    UriTemplate = "/deletename")]

void DeleteName(int id)  {}

```

### Customer.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Runtime.Serialization;

namespace Customer
{
    [DataContract(Name = "Name", Namespace = "")]

    public class Customers
    {
        // Attributes
        private string _name = "";
        private int _age = 0;
        private string _id;

        // Constructors

        public Customers()
        {
            // Do Nothing
        }

        public Customers(string ID, string name, int age)
        {
            _id = ID;
            _name = name;
            _age = age;
        }

        // Properties

        [DataMember(Name = "id", Order = 1)] // Set name returned to JSON
        public string ID
        {
            get { return _id; }
            set { _id = value; }
        }

        [DataMember(Name = "age", Order = 2)] // Set name returned to JSON
        public int Age
        {
            get { return _age; }
            set { _age = value; }
        }

        [DataMember(Name = "name", Order = 3)]
        public string Name
        {
            get { return _name; }
            set { _name = value; }
        }
    }
}
```

### EmailMessage.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Runtime.Serialization;

namespace CorsService
{
    [DataContract(Name = "Message", Namespace = "")]

    public class EmailMessage
    {
        // Attributes
        string _sender = "";
        string _recipient = "";
        string _message = "";

        // Constructors

        public EmailMessage()
        {
            // Do Nothing
        }

        public EmailMessage(string sender, string recipient, string message)
        {
            _sender = sender;
            _recipient = recipient;
            _message = message;
        }

        // Properties

        [DataMember(Name = "sender", Order = 1)] // Set name returned to
JSON public string Sender
        {
            get { return _sender; }
            set { _sender = value; }
        }

        [DataMember(Name = "recipient", Order = 2)]
        public string Recipient
        {
            get { return _recipient; }
            set { _recipient = value; }
        }

        [DataMember(Name = "text", Order = 3)]
        public string MessageText
        {
            get { return _message; }
            set { _message = value; }
        }
    }
}
```

### MessageService.svc.cs

```
using Customer;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web;
using System.Text;

namespace CorsService
{
    // NOTE: You can use the "Rename" command on the "Refactor" menu to change the
    // class name "Service1" in code, svc and config file together.
    // NOTE: In order to launch WCF Test Client for testing this service, please
    // select Service1.svc or Service1.svc.cs at the Solution Explorer and start debugging.
    public class MessageService : IMessageService
    {
        private static List<EmailMessage> _messages = new List<EmailMessage>();
        private static List<Customers> _names = new List<Customers>();

        public void SendMessage(string sender, string recipient, string message)
        {
            EmailMessage msg = new EmailMessage(sender, recipient, message);
            _messages.Add(msg);
        }

        public string ReadLastMessage()
        {
            if (_messages.Count == 0)
                return "";

            return _messages[_messages.Count - 1].MessageText;
        }

        [return: MessageParameter(Name = "Message")]
        public string ReadMessage(int msgID)
        {
            try
            {
                EmailMessage msg = _messages[msgID];
                return msg.MessageText;
            }
            catch // (ArgumentOutOfRangeException e)
            {
                return "Bad Id";
            }
        }

        [return: MessageParameter(Name = "Modifications")]
        public EmailMessage ModifyMessage(int msgID, EmailMessage messageChanges)
        {
            try
            {
                EmailMessage msg = _messages[msgID];

                msg.Sender = messageChanges.Sender;
                msg.Recipient = messageChanges.Recipient;
                msg.MessageText = messageChanges.MessageText;

                return msg;
            }
            catch { }
        }
    }
}
```

```

    }
    catch // (ArgumentOutOfRangeException e)
    {
        return new EmailMessage("Broken", "Broken", "Broken");
    }
}

[return: MessageParameter(Name = "ChangesComplete")]
public bool ModifyMessage1(int msgID, string sender, string recipient, string
message)
{
    EmailMessage msg = _messages[msgID];
    msg.Sender = sender;
    msg.Recipient = recipient;
    msg.MessageText = message;
    return true;
}

public void DeleteMessage(int id)
{
    try
    {
        _messages.RemoveAt(id);
    }
    catch // (ArgumentOutOfRangeException e)
    {
    }
}

public void SendName(string id, string name, int age)
{
    Customers nme = new Customers(id, name, age);
    _names.Add(nme);
}

[return: MessageParameter(Name = "Name")]
public string Readname()
{
    if (_names.Count == 0)
        return "";

    return _names[_names.Count - 1].Name;
}

public void DeleteName(int id)
{
    try
    {
        _names.RemoveAt(id);
    }
    catch // (ArgumentOutOfRangeException e)
    {
    }
}
}
}

```

# LeahCahillAssignmentOne

## GRADEMARK REPORT

### FINAL GRADE

76/100

### GENERAL COMMENTS

#### Instructor

Big data application. (20) 18

Discussion of Hadoop technology (20) 16

Quality of website (Typescript) (20 ) 16

Single page application (20) 15

Dynamic content using AJAX (15 ) 8

Software testing (5 ) 3

Very good analysis of the four Vs of big data. Hive described in suitable details, perhaps you could have focused more on it rather than spreading the discussion into other areas.

A number of AJAX concepts explored, and REST service provided. Single page application works OK but I would liked to have seen the use of REST being a bit more than just for simple Customer data.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5



#### Comment 1

What do these terms mean?

PAGE 6

PAGE 7



## Comment 2

I would have preferred more detail on Hlve rather than a discussion of multiplie technologies.

PAGE 8

---

PAGE 9

---



## Comment 3

Nice XML example

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

---

PAGE 24

---

PAGE 25

---

PAGE 26

---

PAGE 27

---

PAGE 28

---

PAGE 29

---

PAGE 30

---

PAGE 31

---

PAGE 32

---

PAGE 33

---

PAGE 34

---

PAGE 35

---

PAGE 36

---

PAGE 37

---

PAGE 38

---