

Emerging Web Tech

by Sam Phoenix

FILE	2644_SAM_PHOENIX_EMERGING_WEB_TECH_93287_1109754292.DOC X		
TIME SUBMITTED	29-MAR-2018 05:20PM (UTC+0100)	WORD COUNT	5760
SUBMISSION ID	85828130	CHARACTER COUNT	48632

EMERGING WEB TECH

I. Part A: Understanding	2
A. Big Data Application	2
1. Volume	2
2. Velocity	2
3. Variety	3
4. Veracity	3
B. Hadoop Technology - HBase	3
II. Part B: Implementation	4
A. About the app	4
B. Problems encountered	4
III. References	11
IV. Appendix I: Code	12
A. Planner	12
1. TestServicePage.ts	12
2. Index.html	14
B. PlannerService	44
1. IMessageService.cs	44
2. MessageService.svc.cs	47
3. EmailMessage.cs	48
4. Global.asax.cs	50
5. Web.config	51

I. Part A: Understanding

This section discusses big data, how the four Vs apply in the context of a web application, and how it might benefit from one particular Hadoop technology.

A. Big Data Application

This section discusses the four Vs of big data collection as applied to Facebook.

1. Volume

Facebook began as a social network for Harvard students, and exploded in early 2000's when it opened up to the general public. It is now the most popular social media tool, collecting vast quantities of data daily. Although they have employed many methods to collect this data for years, it has only come to light in the last two weeks just how much they can collect from whom.

Their use of a logging API makes it possible to collect vast volumes of data across the internet, and the Messenger app on smart phones allows them to collect data on the user's friends. Collected data runs into terabytes, petabytes and even higher, and it all must go somewhere.

There are two ways to accommodate the growing volume of data; by scaling up and scaling out. Scaling up was done by continuing to upgrade one computer, increasing the storage capacity of the machine. This worked fine when the term "Big Data" was first coined in 1997[1], when "big" meant about 100GB; modern hard drives can now store up to 6TB or more. However, with data now running into PBs, this approach is no longer efficient or cost-effective.

The approach used nowadays is to scale out – using a cluster of hundreds of servers, each carrying their own fraction of the data being stored. Whenever more storage space needs to be added, the company can simply add another server to the cluster.  1

2. Velocity

Velocity refers to the speed and processing of data and is related to volume in terms of collecting of data. For Facebook, it refers to the rate of data being collected via their APIs as well as when users navigate the site itself.

It also refers to the processing rate of this data. Because it is collected in such large quantities at a constant rate, there needs to be some way of processing it to render it useful for marketing and other purposes.

A popular way to do this is by a concept known as **MapReduce**. The process involves conducting a search for keywords, mapping them to the fragments of data being stored and combining the results to provide a word count.  2

3. Variety

Variety refers to the different kinds of data that can be collected. Facebook allows not only textual posts, but also images and video uploads; the variations in languages spoken also affects variety of data, as well as the different image and video formats, and metadata (such as location and friends tagging) that can be attributed to all forms of posts that are shared.  3

4. Veracity

Veracity is a more recent addition to the concept of big data, referring to the uncertainty and unreliability of collected data. Facebook is among the companies for whom this is more problematic due to its nature.

Social media has come to serve three main purposes: to stay in touch with friends and family; to entertain; and to spread awareness of current events by circulating news articles. Journalism has evolved to take advantage of this[2], by looking for trending articles and seeking out witnesses, but journalists have also come up against the spreading of “fake news”. Not all news spread via social media is true, and this is the case for all data that is being collected.

As well as the problem of rumours, hoaxes, scams etc., data can become corrupted, or incomplete – users may not fill in all the available sections on their profile or may lock them down so as not to overshare information about themselves. In light of the recent revelation by Cambridge Analytica on how much data is being gathered, articles on how to do this[3] are surfacing in a bid to educate users on this issue.  4

B. Hadoop Technology - HBase

Hadoop is an open-source version of Google's propriety tool to manage big data. Hadoop stands for Highly Archived Distributed Object Oriented Programming and came into being in 2005[4]. It makes use of Hadoop Distributed File System (HDFS), of which HBase is one part.

HBase was developed by Apache for handling large quantities of data and is written in Java. Because of the fragmented, unstructured and varied nature of this data, structured database management such as SQL is not appropriate to manage it alone. Therefore, a technology known as NoSQL (Not only SQL) is used to process this data.

HBase is described as a “distributed, fault-tolerant and highly scalable, column-oriented, NoSQL database built on top of HDFS”[5]. Data is stored in tables with a primary key, but no relational information is stored. Instead, it is stored as a multi-dimensional map, each with its own timestamp and cell information. The mapping of this data is handled by a MapReduce program, which one study[5] found to be more efficient at handling data than MySQL.  5

II. Part B: Implementation

Discussed below is the app implementation using AJAX.

A. About the app

The app was originally designed to be a student budget planner, and the UI was developed with this in mind using Bootstrap. However, since getting the AJAX and web services to work with it proved tricky, I wound up using test code just to make sure the setup worked.

The test code is a simple message system, where the user can send, receive and modify messages. In production this would use a database; however, for test purposes it just writes to an array. The code for this is shown in Appendix I: Code.

B. Problems encountered

After spending some time building a working UI ready for the original project, getting AJAX to work with the relevant classes proved frustrating. Resorting to test code should have been an easy task, but it was not to be.



Problems with building the web services side were many and varied, sometimes with no valid reason. Some involved strange configuration errors, as shown in Figure 1 and Figure 2 below:

After some time debugging the Web.config file, removing duplicates and generally checking that all was well, the service page loaded fine, as shown below:

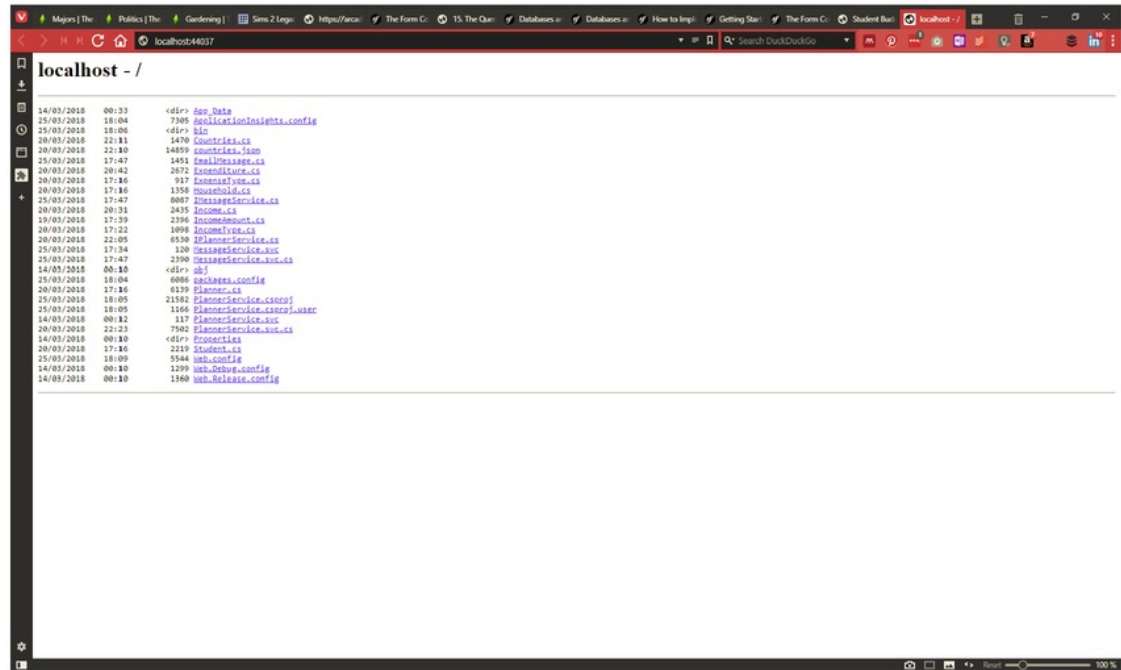


Figure 3: Services page finally loads

While AJAX finally worked, it also showed that the error function worked well:

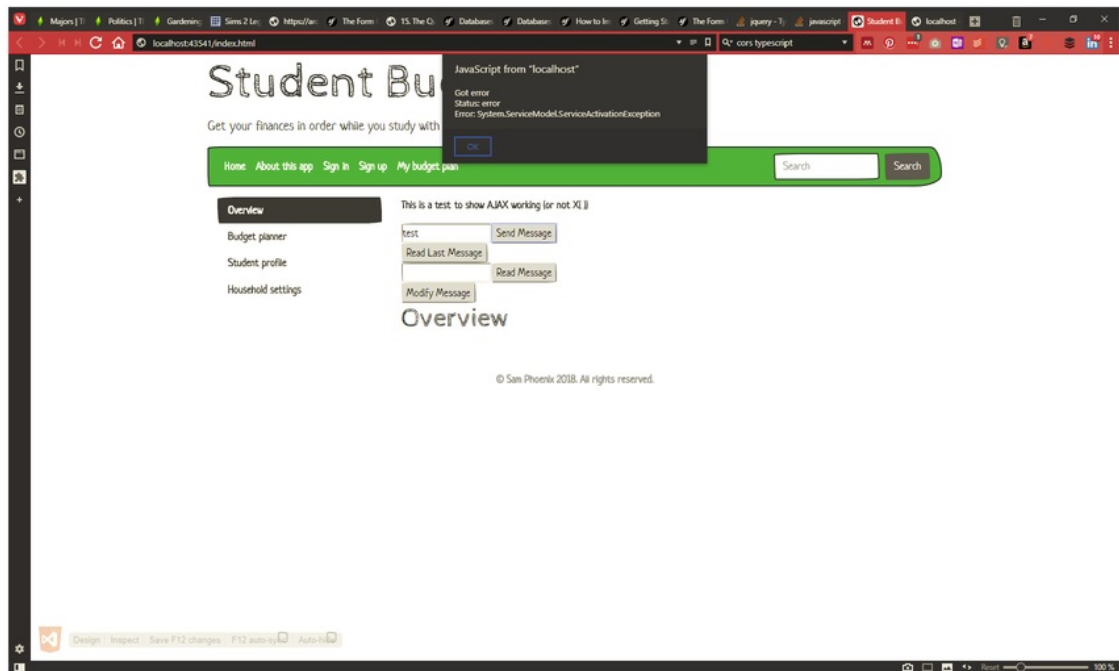


Figure 4: AJAX finally works, but gives an exception error

In trying to trace the problem, I used the browser's debugging tools:

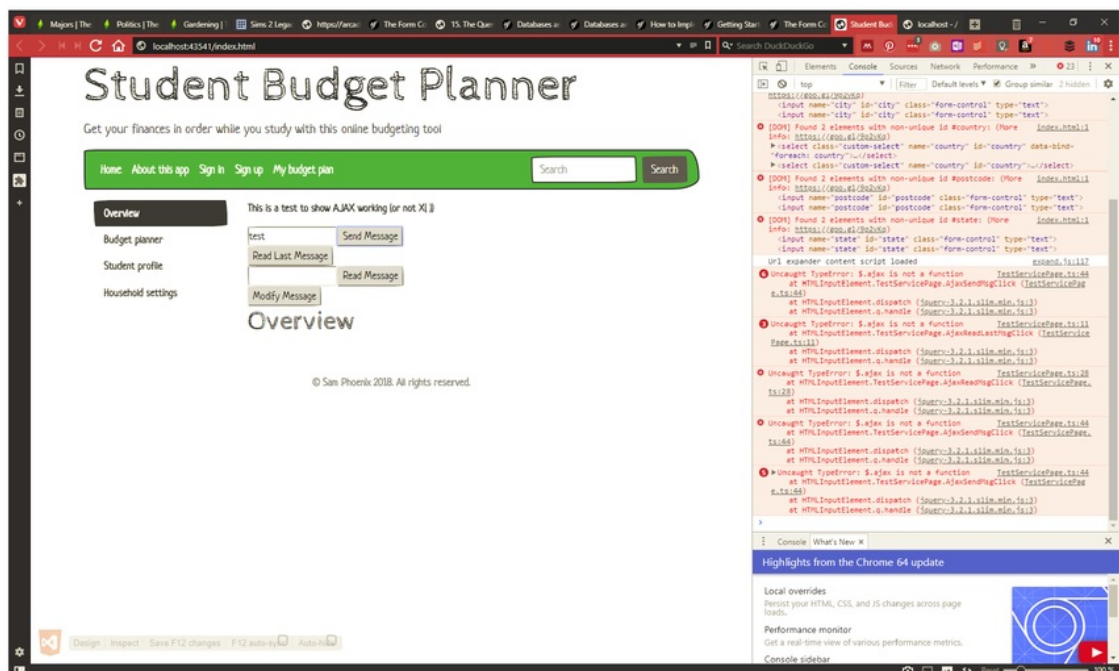


Figure 5: Using the browser's inspect - console tools for debugging

With no way of knowing where to look for this error message, I turned to Google and StackOverflow for help. One answer suggested checking the Windows Event Viewer -> Application Logs, which proved to be very useful:

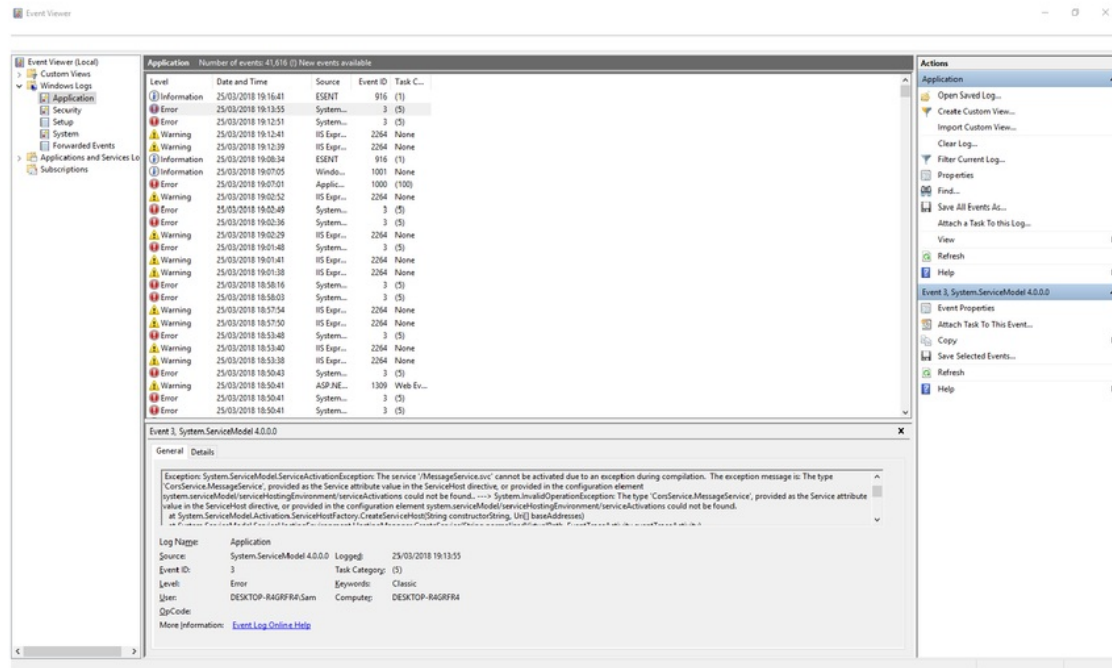


Figure 6: StackOverflow suggested checking the EventViewer for more information - which points to an odd error

The error message explained that the app was looking for a service that wasn't there. I had fixed that by renaming items throughout the service files and Web.config, but the original name was still being referenced somewhere.

With this in mind, I went through the files and renamed everything to the name it was looking for – and it worked:

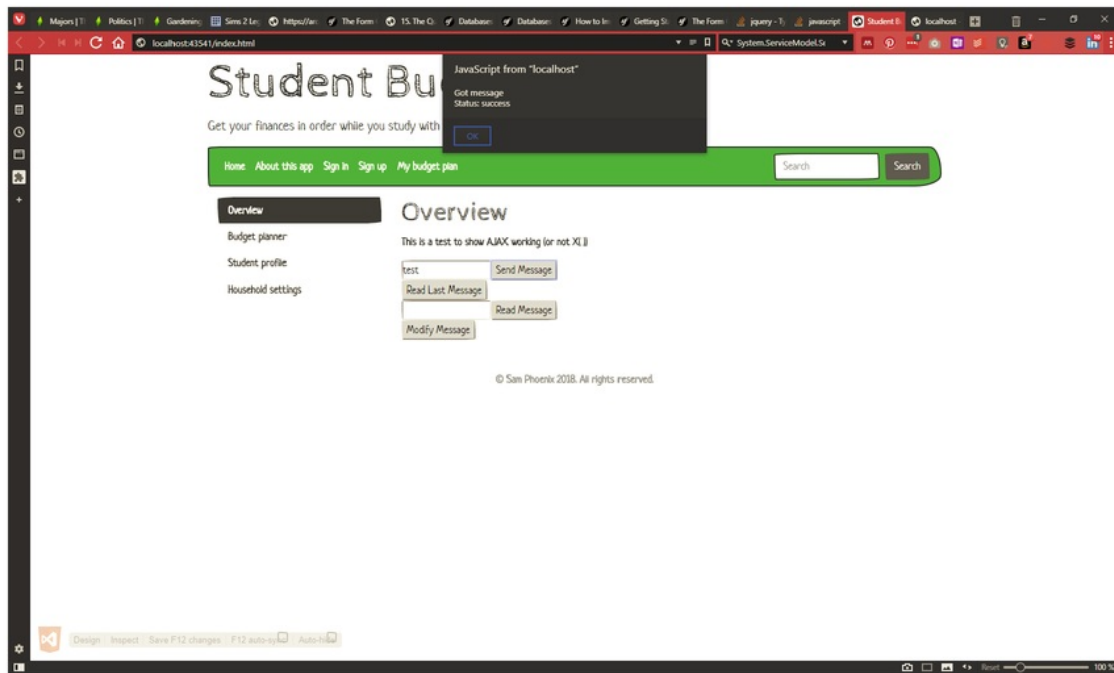


Figure 7: Sending message finally works

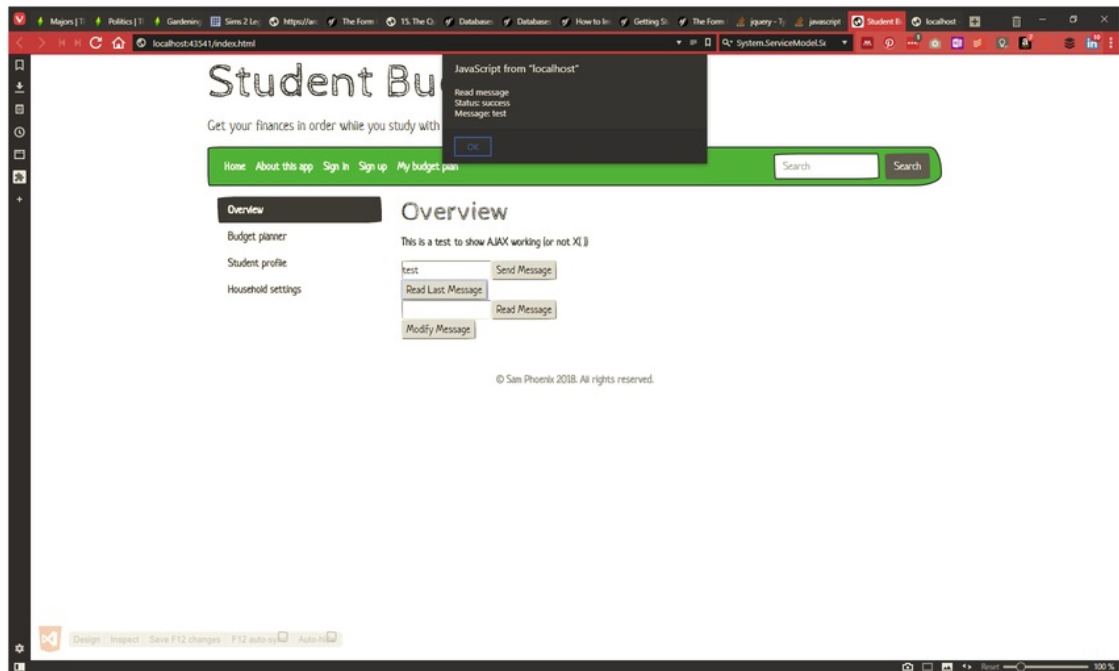


Figure 8: Reading the last message sent

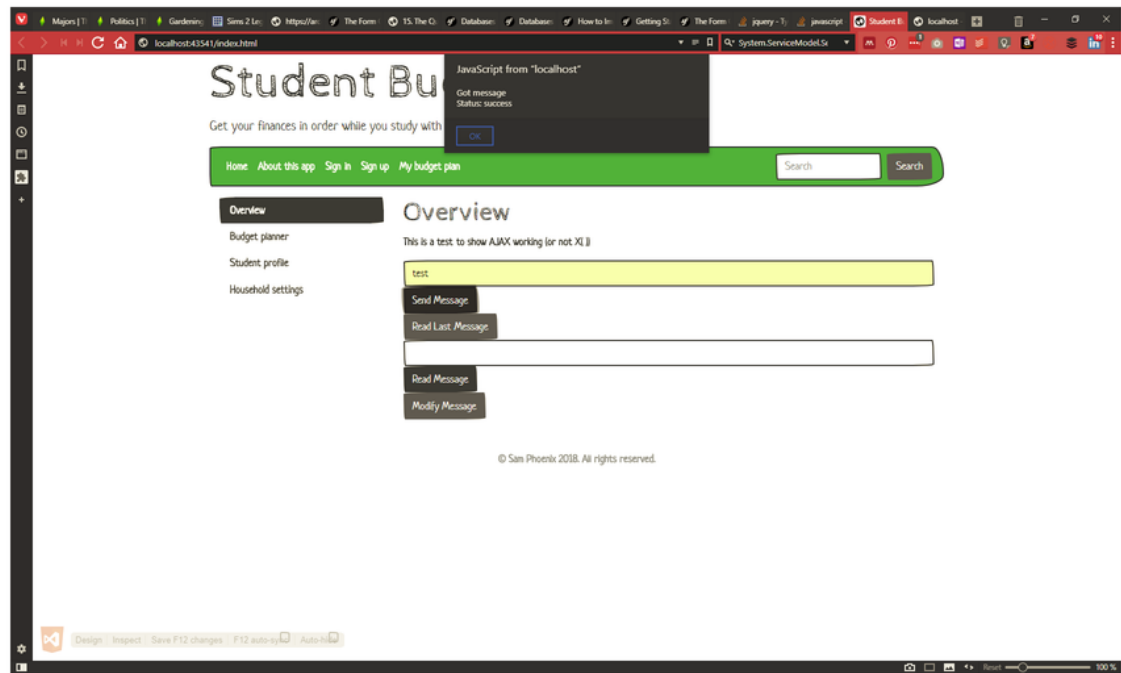


Figure 9: Prettified success form using Bootstrap - because after all this, it can at least look consistent

With more time, I would have liked to get this working with the custom fields that are a part of this app, including reading a `countries.json` file to populate the select list. I had originally considered Knockout.js for this purpose, but found it took longer to understand it enough to apply it in this context.

III. References

- [1] R. Dautov and S. Distefano, 'Quantifying volume, velocity, and variety to support (Big) data-intensive application development', in *2017 IEEE International Conference on Big Data (Big Data)*, 2017, pp. 2843–2852.
- [2] A. V. Parvathi and S. Karthika, 'Designing prototype for accessing Veracity using social sensors - Big data perspective', in *Proceedings of the 3rd IEEE International Conference on Advances in Electrical and Electronics, Information, Communication and Bio-Informatics, AEEICB 2017*, 2017, pp. 198–201.
- [3] 'Check your Facebook settings NOW - it's likely you're allowing third-party applications to access your data - MoneySavingExpert.com Team Blog'. [Online]. Available: https://www.moneysavingexpert.com/team-blog/2018/03/20/check-your-facebook-settings-now/?utm_source=MSE_Newsletter&utm_medium=email&utm_term=27-Mar-18-8427&utm_campaign=nt-oneliners-one&utm_content=17. [Accessed: 28-Mar-2018].
- [4] M. Manwal and A. Gupta, 'Big data and hadoop — A technological survey', in *2017 International Conference on Emerging Trends in Computing and Communication Technologies (ICETCCT)*, 2017, pp. 1–6.
- [5] M. N. Vora, 'Hadoop-HBase for large-scale data', in *Proceedings of 2011 International Conference on Computer Science and Network Technology, ICCSNT 2011*, 2011, vol. 1, pp. 601–605.

IV. Appendix I: Code

Below is the code related to the working AJAX implementation.

A. Planner

1. TestServicePage.ts

```
/// <reference path="scripts/typings/jquery/jquery.d.ts" />

class TestServicePage {
    PageLoad(): void {
        // Button actions
        $("#sendMsgButton").click(this.AjaxSendMsgClick);
        $("#readLastMsgButton").click(this.AjaxReadLastMsgClick);
        $("#readMsgButton").click(this.AjaxReadMsgClick);
        $("#modifyMsgButton").click(this.AjaxModifyMsgClick);
    }

    private AjaxReadLastMsgClick = (e: JQueryEventObject): boolean => {
        $.ajax({
            cache: false,
            type: "GET",
            url: "http://localhost:44037/MessageService.svc/readmessage",
            data: '{}',
            contentType: "application/json; charset=utf-8",
            dataType: "json",
            success: this.AjaxMsgReadSuccessful,
            error: this.AjaxError
        });
        e.preventDefault();
        return false;
    }

    // Callback methods
    private AjaxReadMsgClick = (e: JQueryEventObject): boolean => {
        // Read data in REST - use GET
        $.ajax({
            cache: false,
            type: "GET",
            url: "http://localhost:44037/MessageService.svc/readmessage?id=" +
                $("#msgIdTextBox").val(),
            data: '{}',
            contentType: "application/json; charset=utf-8",
            dataType: "json",
            success: this.AjaxMsgReadSuccessful,
            error: this.AjaxError
        });
        e.preventDefault();
        return false;
    }

    private AjaxSendMsgClick = (e: JQueryEventObject): boolean => {
        // Add new data in REST - use POST
        $.ajax({
            cache: false,
```

```

        type: "POST",
        url: "http://localhost:44037/MessageService.svc/postmessage",
        data: '{"sender":"Sam", "recipient": "Bob", "message":"' +
$( "#msgTextBox" ).val() + "'}',
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        success: this.AjaxMsgSendSuccessful,
        error: this.AjaxError
    });
    e.preventDefault();
    return false;
}

private AjaxModifyMsgClick = (e: JQueryEventObject): boolean => {
    // Edit data in REST - use PUT
    $.ajax({
        cache: false,
        type: "PUT",
        url: "http://localhost:44037/MessageService.svc/putmessage?id=" +
$( "#msgIdTextBox" ).val(),
        data: '{ "messageChanges": { "sender":"Sam", "recipient": "Bob", "text":"' +
$( "#msgTextBox" ).val() + "' } }',
        contentType: "application/json; charset=utf-8",
        dataType: "json",
        success: this.AjaxMsgEditSuccessful,
        error: this.AjaxError
    });
    e.preventDefault();
    return false;
}

private AjaxMsgSendSuccessful = (json: any, status: string, req: JQueryXHR): void =>
{
    window.alert("Got message\n Status: " + status + "\n");
}

private AjaxMsgReadSuccessful = (json: any, status: string, req: JQueryXHR): void =>
{
    // "Message" here comes from the return type in IMessageService
    window.alert("Read message\n Status: " + status + "\nMessage: " + json.Message +
"\n");
}

private AjaxMsgEditSuccessful = (json: any, status: string, req: JQueryXHR): void =>
{
    // "Message" here comes from the return type in IMessageService
    let emailMessage = json.Modifications; // See interface file for "Modifications"
    return
    window.alert("Changes made: sender=" + emailMessage.sender + " message=" +
emailMessage.text + "\n");
}

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

$(function () {

```

```
let page: TestServicePage = new TestServicePage();
page.PageLoad();
// (new TestServicePage()).PageLoad();
});
```

2. Index.html

```
<!DOCTYPE html>

<html lang="en-gb">

<head>

  <title>Student Budget Planner</title>

  <meta charset="utf-8" />

  <meta http-equiv="X-UA-Compatible" content="IE=edge" />

  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

  <link href="css/bootstrap.min.css" rel="stylesheet" />

</head>

<body>

  <div class="container">

    <header>

      <h1 class="display-3">Student Budget Planner</h1>

      <p class="lead">

        Get your finances in order while you study with this online budgeting tool

      </p>

      <nav class="navbar navbar-expand-lg navbar-dark bg-success mb-3">

        <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarColor01" aria-controls="navbarColor01" aria-expanded="false" aria-
label="Toggle navigation" style="">

          <span class="navbar-toggler-icon"></span>

        </button>

        <div class="collapse navbar-collapse" id="navbarColor01">
```

```

        <ul class="navbar-nav mr-auto">
            <li class="nav-item active">
                <a class="nav-link" href="index.html">Home <span class="sr-
only">(current)</span></a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#">About this app</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#">Sign in</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#">Sign up</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#">My budget plan</a>
            </li>
        </ul>

        <form class="form-inline my-2 my-lg-0">
            <input class="form-control mr-sm-2" type="text" placeholder="Search">
            <button class="btn btn-secondary my-2 my-sm-0"
type="submit">Search</button>
        </form>
    </div>
</nav>
</header>

```

```

<!-- Begin page content -->

<main role="main" class="container">

  <div class="row">

    <div class="col-6 col-lg-3">

      <!-- Vertical nav -->

      <div class="nav flex-column nav-pills" id="v-pills-tab" role="tablist" aria-orientation="vertical">

        <a class="nav-link active" id="v-pills-home-tab" data-toggle="pill" href="#v-pills-home" role="tab" aria-controls="v-pills-home" aria-selected="true">Overview</a>

        <a class="nav-link" id="v-pills-planner-tab" data-toggle="pill" href="#v-pills-planner" role="tab" aria-controls="v-pills-planner" aria-selected="false">Budget planner</a>

        <a class="nav-link" id="v-pills-profile-tab" data-toggle="pill" href="#v-pills-profile" role="tab" aria-controls="v-pills-profile" aria-selected="false">Student profile</a>

        <a class="nav-link" id="v-pills-settings-tab" data-toggle="pill" href="#v-pills-settings" role="tab" aria-controls="v-pills-settings" aria-selected="false">Household settings</a>

      </div>

    </div>

    <!-- Nav content -->

    <div class="col-12 col-sm-6 col-lg-9">

      <div class="tab-content" id="v-pills-tabContent">

        <div class="tab-pane fade show active" id="v-pills-home" role="tabpanel" aria-labelledby="v-pills-home-tab">

          <h1>Overview</h1>

          <p>This is a test to show AJAX working (or not X( ))</p>

          <form id="1">

            <input id="msgTextBox" type="text" class="form-control" /><input id="sendMsgButton" class="btn btn-primary" value="Send Message" type="button" />

          </form>

        </div>

      </div>

    </div>

  </div>

</main>

```

```

        <br />

        <input id="readLastMsgButton" class="btn btn-secondary"
value="Read Last Message" type="button" />

        <br />

        <input id="msgIdTextBox" class="form-control" type="text" /><input
id="readMsgButton" class="btn btn-primary" value="Read Message" type="button" />

        <br />

        <input id="modifyMsgButton" value="Modify Message" class="btn btn-
secondary" type="button" />

    </form>

</div>

<div class="tab-pane fade" id="v-pills-planner" role="tabpanel" aria-
labelledby="v-pills-planner-tab">

    <!-- Budget planner section -->

    <h1>Budget planner</h1>

    <p>This is where you can define your total income and expenses, and
see how much you're left with.</p>

    <ul class="nav nav-tabs" id="myTab" role="tablist">

        <li class="nav-item">

            <a class="nav-link active" id="income-tab" data-toggle="tab"
href="#income" role="tab" aria-controls="income" aria-selected="true">Income</a>

        </li>

        <li class="nav-item">

            <a class="nav-link" id="outgoings-tab" data-toggle="tab"
href="#outgoings" role="tab" aria-controls="outgoings" aria-
selected="false">Outgoings</a>

        </li>

        <li class="nav-item">

```

```

        <a class="nav-link" id="results-tab" data-toggle="tab"
href="#results" role="tab" aria-controls="results" aria-selected="false">Results</a>

    </li>

</ul>

<!--Planner content-->

<div class="tab-content" id="myTabContent">

    <div class="tab-pane fade show active" id="income" role="tabpanel"
aria-labelledby="income-tab">

        <h3>Income</h3>

        <div class="row">

            <div class="col-md-4">

                <div class="dropdown">

                    <button class="btn btn-primary dropdown-toggle"
type="button" id="income" data-toggle="dropdown" aria-haspopup="true" aria-
expanded="false">

                        Display amounts per:

                    </button>

                    <div class="dropdown-menu" aria-labelledby="income">

                        <a class="dropdown-item" id="incomeWk"
name="incomeWk" href="#">Week</a>

                        <a class="dropdown-item" id="incomeMth"
name="incomeMth" href="#">Month</a>

                        <a class="dropdown-item" id="incomeQt"
name="incomeQt" href="#">Quarter</a>

                        <a class="dropdown-item" id="incomeTm"
name="incomeTm" href="#">Term</a>

                        <a class="dropdown-item" id="incomeYr"
name="incomeYr" href="#">Year</a>

                    </div>

                </div>

            </div>

        </div>
    </div>

```

```

</div>

<div class="col-md-3">

    Showing <span>monthly</span> amounts

</div>
</div>
<br />
<table class="table table-hover">
    <caption>
        Total income: £<span>1029.90</span>
    </caption>
    <thead>
        <tr class="table-success">
            <th scope="col" class="text-center">
                <div class="form-check">
                    <input type="checkbox" class="form-check-input"
id="selectAll" name="selectAll">
                    <label class="sr-only form-check-label"
for="selectAll">Select all</label>
                </div>
            </th>
            <th scope="col">Name</th>
            <th scope="col">Type</th>
            <th scope="col">Amount</th>
        </tr>
    </thead>
    <tbody>
        <tr>

```

```

        <th scope="row" class="text-center">
            <div class="form-check">
                <input type="checkbox" class="form-check-input"
id="select1" name="select1">
                <label class="sr-only form-check-label"
for="select1">Select</label>
            </div>
        </th>
        <td>Maintainence loan</td>
        <td>Student finance</td>
        <td>£357.14</td>
    </tr>
    <tr>
        <th scope="row" class="text-center">
            <div class="form-check">
                <input type="checkbox" class="form-check-input"
id="select2" name="select2">
                <label class="sr-only form-check-label"
for="select2">Select</label>
            </div>
        </th>
        <td>Maintainence grant</td>
        <td>Student finance</td>
        <td>£290.25</td>
    </tr>
    <tr>
        <th scope="row" class="text-center">
            <div class="form-check">

```

```

        <input type="checkbox" class="form-check-input"
id="select3" name="select3">

        <label class="sr-only form-check-label"
for="select3">Select</label>

    </div>

</th>

<td>Working Tax Credit</td>

<td>State benefit</td>

<td>£382.51</td>

</tr>

</tbody>

</table>

    <button class="btn btn-outline-success" id="addIncome"
name="addIncome" data-toggle="modal" data-target="#Income">Add income</button>
    <button class="btn btn-outline-danger" id="removeIncome"
name="removeIncome">Remove income</button>

</div>

<div class="tab-pane fade" id="outgoings" role="tabpanel" aria-
labelledby="outgoings-tab">

    <h3>Outgoings</h3>

    <div class="row">

        <div class="col-md-4">

            <div class="dropdown">

                <button class="btn btn-primary dropdown-toggle"
type="button" id="expenses" data-toggle="dropdown" aria-haspopup="true" aria-
expanded="false">

                    Display amounts per:

                </button>

                <div class="dropdown-menu" aria-labelledby="expenses">

                    <a class="dropdown-item" href="#">Week</a>

```

```

        <a class="dropdown-item" href="#">Month</a>
        <a class="dropdown-item" href="#">Quarter</a>
        <a class="dropdown-item" href="#">Term</a>
        <a class="dropdown-item" href="#">Year</a>
    </div>
</div>
</div>
<div class="col-md-3">
    Showing <span>monthly</span> amounts
</div>
</div>
<br />
<table class="table table-hover">
    <caption>
        Total outgoings: £<span>1029.90</span>
    </caption>
    <thead>
        <tr class="table-danger">
            <th scope="col" class="text-center">
                <div class="form-check">
                    <input type="checkbox" class="form-check-input"
id="selectAll" name="selectAll">
                    <label class="sr-only form-check-label"
for="selectAll">Select</label>
                </div>
            </th>
            <th scope="col">Name</th>

```

```

        <th scope="col">Type</th>
        <th scope="col">Amount</th>
        <th scope="col">Who pays?</th>
    </tr>
</thead>
<tbody>
    <tr>
        <th scope="row" class="text-center">
            <div class="form-check">
                <input type="checkbox" class="form-check-input"
id="select1" name="select1">
                <label class="sr-only form-check-label"
for="select1">Select</label>
            </div>
        </th>
        <td>Rent</td>
        <td>Rent</td>
        <td>£357.14</td>
        <td>Just me</td>
    </tr>
    <tr>
        <th scope="row" class="text-center">
            <div class="form-check">
                <input type="checkbox" class="form-check-input"
id="select2" name="select2">
                <label class="sr-only form-check-label"
for="select2">Select</label>
            </div>

```

```

        </th>
        <td>Electric</td>
        <td>Utilities</td>
        <td>£290.25</td>
        <td>Everyone</td>
    </tr>
    <tr>
        <th scope="row" class="text-center">
            <div class="form-check">
                <input type="checkbox" class="form-check-input"
id="select3" name="select3">
                <label class="sr-only form-check-label"
for="select3">Select</label>
            </div>
        </th>
        <td>Food shopping</td>
        <td>Groceries</td>
        <td>£382.51</td>
        <td>Just me</td>
    </tr>
</tbody>
</table>

    <button class="btn btn-outline-success" data-toggle="modal"
id="addExpense" name="addExpense" data-target="#Expense">Add expense</button>
    <button class="btn btn-outline-danger" id="removeExpense"
name="removeExpense">Remove expense</button>

</div>

    <div class="tab-pane fade" id="results" role="tabpanel" aria-
labelledby="results-tab">

```

```

        <h3>Results</h3>

    </div>

</div><!--/planner content-->


</div><!--/planner section-->

<!--Profile section-->

<div class="tab-pane fade" id="v-pills-profile" role="tabpanel" aria-
labelledby="v-pills-profile-tab">

    <h1>Student profile</h1>

    <p>

        Please complete as much of this section as possible. The choices
        entered here will be used in budget calculations.

    </p>

    <div id="accordion">

        <form>

            <!-- General details -->

            <div class="card">

                <div class="card-header">

                    <a class="card-link" data-toggle="collapse" data-
parent="#accordion" href="#generalDetails">

                        General details

                    </a>

                </div>

                <div id="generalDetails" class="collapse show">

                    <div class="card-body">

                        <fieldset>

                            <div class="form-group">

                                <label for="studentID">Student ID</label>

```

```

        <input name="studentID" id="studentID" class="form-
control" aria-describedby="studentIDHelp" type="text" />

        <small id="studentIDHelp" class="form-text text-
muted">

            This is the ID number assigned by your college or
            university

        </small>
    </div>
    <div class="form-group">
        <label for="fName">First name</label>
        <input name="fName" id="fName" class="form-control"
type="text" />
    </div>
    <div class="form-group">
        <label for="sName">Last / family name</label>
        <input name="sName" id="sName" class="form-
control" type="text" />
    </div>
    <div class="form-group">
        <label for="studyPlace">Place of study</label>
        <select class="custom-select" name="studyPlace"
id="studyPlace">
            <option selected value="">-- Where in the UK are
            you studying? --</option>
            <option value="england">England</option>
            <option value="scotland">Scotland</option>
            <option value="wales">Wales</option>
            <option value="ireland">N Ireland</option>
        </select>
    </div>

```

```

        </div>
    </fieldset>
</div>
</div>
</div>

<!-- Home address -->
<div class="card">
    <div class="card-header">
        <a class="collapsed card-link" data-toggle="collapse" data-
parent="#accordion" href="#homeAddress">
            Home address
        </a>
    </div>
    <div id="homeAddress" class="collapse">
        <div class="card-body">

            <fieldset>
                <div class="form-group">
                    <label for="hadd1">Address 1</label>
                    <input name="hadd1" id="hadd1" class="form-control"
type="text" />
                </div>
                <div class="form-group">
                    <label for="hadd2">Address 2</label>
                    <input name="hadd2" id="hadd2" class="form-control"
type="text" />
                </div>
            </fieldset>
        </div>
    </div>
</div>

```

```

type="text" />
    <div class="form-group">
        <label for="hcity">City</label>
        <input name="hcity" id="hcity" class="form-control"
control" type="text" />
    </div>
    <div class="form-group">
        <label for="hpostcode">Postcode</label>
        <input name="hpostcode" id="hpostcode" class="form-
type="text" />
    </div>
    <div class="form-group">
        <label for="hstate">State / Province</label>
        <input name="hstate" id="hstate" class="form-control"
type="text" />
    </div>
    <div class="form-group">
        <label for="hcountry">Country</label>
        <select class="custom-select" name="hcountry"
id="hcountry" data-bind="foreach: country">
            <option data-bind="text: name"></option>
        </select>
    </div>
</fieldset>
</div>
</div>
</div>

```

```

<!-- Account details -->
<div class="card">
  <div class="card-header">
    <a class="collapsed card-link" data-toggle="collapse" data-
parent="#accordion" href="#accountDetails">
      Account details
    </a>
  </div>
  <div id="accountDetails" class="collapse">
    <div class="card-body">
      <fieldset>
        <div class="form-group">
          <label for="email">Email address</label>
          <input class="form-control" type="email" name="email"
id="email" />
        </div>
        <div class="form-group">
          <label for="password">New password</label>
          <input class="form-control" type="password"
name="password" id="password" />
        </div>
        <div class="form-group">
          <label for="cPword">Confirm password</label>
          <input class="form-control" type="password"
name="cPword" id="cPword" />
        </div>
        <div class="form-group">

```

```

        <label for="acType" aria-
describedby="accountHelp">Account type: </label>

        <div class="btn-group btn-group-toggle" data-
toggle="buttons">

            <label class="btn btn-primary">

                <input type="radio" name="acType"
id="individual" value="individual" autocomplete="off"> Individual

            </label>

            <label class="btn btn-primary active">

                <input type="radio" checked="" name="acType"
id="household" value="household" autocomplete="off"> Household

            </label>

        </div>

        <small id="accountHelp" class="form-text text-muted">

            If you are the only household member, choose
            "Individual"; otherwise choose "Household"

        </small>

    </div>

</fieldset>

</div>

</div>

</div>

<div class="row mt-3">

    <div class="col-md-4">

        <button class="btn btn-primary" id="stSave" name="stSave"
type="submit">Save</button>

    </div>

    <div class="col-md-4">

        <button class="btn" type="reset">Reset</button>

```

```

        </div>

    </div>

</form>

</div><!-- /profile accordion -->

</div><!--/profile section-->

<!--Household section-->

<div class="tab-pane fade" id="v-pills-settings" role="tabpanel" aria-
labelledby="v-pills-settings-tab">

    <h1>Household settings</h1>

    <p>

        Please complete as much of this section as possible. The choices
        entered here will be used in budget calculations.

    </p>

    <form>

        <div id="accordion">

            <!-- General details -->

            <div class="card">

                <div class="card-header">

                    <a class="card-link" data-toggle="collapse" data-
parent="#accordion" href="#householdDetails">

                        General details

                    </a>

                </div>

                <div id="householdDetails" class="collapse show">

                    <div class="card-body">

                        <fieldset>

                            <div class="form-group">

```

```

        <label for="householdName">Household name</label>
        <input name="householdName" id="householdName"
class="form-control" aria-describedby="householdNameHelp" type="text" />
        <small id="householdNameHelp" class="form-text text-
muted">
            Give your household a name so you can identify it
            later
        </small>
    </div>
    <div class="form-group">
        <label for="members">List of members</label>
        <table class="table table-hover">
            <caption>Number of students in household:
            <span>3</span></caption>
            <thead>
                <tr>
                    <th scope="col" class="text-center">
                        <div class="form-check">
                            <input type="checkbox" class="form-
check-input" id="selectAll">
                            <label class="sr-only form-check-label"
for="selectAll">Select all</label>
                        </div>
                    </th>
                    <th scope="col">First name</th>
                    <th scope="col">Last name</th>
                    <th scope="col">Email</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>
                        <div class="form-check">
                            <input type="checkbox" class="form-
check-input" id="selectAll">
                            <label class="sr-only form-check-label"
for="selectAll">Select all</label>
                        </div>
                    </td>
                    <td>
                        <input type="text" class="form-control" />
                    </td>
                    <td>
                        <input type="text" class="form-control" />
                    </td>
                    <td>
                        <input type="text" class="form-control" />
                    </td>
                </tr>
            </tbody>
        </table>
    </div>

```

```

</thead>
<tbody>
  <tr>
    <th scope="row" class="text-center">
      <div class="form-check">
        <input type="checkbox" class="form-
check-input" id="select1" name="select1">
        <label class="sr-only form-check-label"
for="select1">Select</label>
      </div>
    </th>
    <td>Mark</td>
    <td>Otto</td>
    <td>@mdo</td>
  </tr>
  <tr>
    <th scope="row" class="text-center">
      <div class="form-check">
        <input type="checkbox" class="form-
check-input" id="select2" name="select2">
        <label class="sr-only form-check-label"
for="select2">Select</label>
      </div>
    </th>
    <td>Jacob</td>
    <td>Thornton</td>
    <td>@fat</td>
  </tr>

```

```

        <tr>
            <th scope="row" class="text-center">
                <div class="form-check">
                    <input type="checkbox" class="form-
check-input" id="select3" name="select3">
                    <label class="sr-only form-check-label"
for="select3">Select</label>
                </div>
            </th>
            <td colspan="2">Larry the Bird</td>
            <td>@twitter</td>
        </tr>
    </tbody>
</table>

    <a href="#" class="btn btn-outline-success" data-
toggle="modal" data-target="#addStudent">Add student</a> <a href="#" class="btn btn-
outline-danger">Remove student</a>

</div>
</fieldset>
</div>
</div>
</div>
<!-- Term address -->
<div class="card">
    <div class="card-header">
        <a class="collapsed card-link" data-toggle="collapse" data-
parent="#accordion" href="#termAddress">
            Term address

```

```

        </a>
    </div>

    <div id="termAddress" class="collapse">
        <div class="card-body">
            <fieldset>
                <div class="form-group">
                    <label for="tadd1">Address 1</label>
                    <input name="tadd1" id="tadd1" class="form-control"
type="text" />
                </div>
                <div class="form-group">
                    <label for="tadd2">Address 2</label>
                    <input name="tadd2" id="tadd2" class="form-control"
type="text" />
                </div>
                <div class="form-group">
                    <label for="tcity">City</label>
                    <input name="tcity" id="tcity" class="form-control"
type="text" />
                </div>
                <div class="form-group">
                    <label for="tpostcode">Postcode</label>
                    <input name="tpostcode" id="tpostcode" class="form-
control" type="text" />
                </div>
                <div class="form-group">
                    <label for="tstate">State / Province</label>

```

```

type="text" />
<input name="tstate" id="tstate" class="form-control"

</div>
<div class="form-group">
  <label for="tcountry">Country</label>
  <select class="custom-select" name="tcountry"
id="tcountry">
    <option selected value="">-- Choose country --
  </option>
    <option value="england">England</option>
    <option value="scotland">Scotland</option>
    <option value="wales">Wales</option>
    <option value="ireland">N Ireland</option>
  </select>
</div>
</fieldset>

</div>
</div>
</div>
<fieldset>
  <div class="row mt-3">
    <div class="col-md-4">
      <button class="btn btn-primary" id="hhSave"
name="hhSave" type="submit">Save</button>
    </div>
    <div class="col-md-4">
      <button class="btn" type="reset">Reset</button>

```

```

        </div>
    </div>
</fieldset>
</div>
</form>

</div>
</div>
</div>
</div>

</main>

<footer class="footer text-center mt-5">
    <div class="container">
        <span class="text-muted">&copy; Sam Phoenix 2018. All rights
reserved.</span>
    </div>
</footer>
</div><!--/container-->

<!-- Form modals -->
<!-- Add / edit income -->

<div class="modal fade" id="Income" tabindex="-1" role="dialog" aria-
labelledby="addIncomeLabel" aria-hidden="true">
    <div class="modal-dialog modal-dialog-centered" role="document">
        <div class="modal-content">
            <form>

```

```

<div class="modal-header">
    <h5 class="modal-title" id="addIncomeLabel">Add income</h5>
    <button type="button" class="close" data-dismiss="modal" aria-
label="Close">
        <span aria-hidden="true">&times;</span>
    </button>
</div>
<div class="modal-body">
    <fieldset>
        <div class="form-group">
            <label for="incomeType">Income type</label>
            <select class="form-control" name="incomeType" id="incomeType">
                <option></option>
            </select>
        </div>
        <div class="form-group">
            <label for="incomeName">Income name</label>
            <input type="text" id="incomeName" name="incomeName"
class="form-control" placeholder="Give a name for this form of income" />
        </div>
        <div class="form-group">
            <label for="incomeFreq">How often is this paid?</label>
            <select class="form-control" name="incomeFreq" id="incomeFreq">
                <option value="Weekly">Weekly</option>
                <option value="Monthly">Monthly</option>
                <option value="Quarterly">Quarterly</option>
                <option value="Termly">Termly</option>
            </select>
        </div>
    </fieldset>
</div>

```

```

        <option value="Yearly">Yearly</option>
    </select>
</div>

<div class="form-group">
    <label for="incomeAmount">Amount paid</label>

    <div class="input-group">
        <div class="input-group-prepend">
            <div class="input-group-text">£</div>
        </div>

        <input type="text" class="form-control" name="incomeAmount"
id="incomeAmount" />
    </div>

</div>

</fieldset>

</div>

<div class="modal-footer">
    <button type="button" class="btn btn-secondary" data-
dismiss="modal">Close</button>

    <button type="submit" name="incomeSave" id="incomeSave" class="btn
btn-primary">Save changes</button>
</div>

</form>

</div>

</div>

</div>

```

```

<!-- Add / edit expense -->

```

```

<div class="modal fade" id="Expense" tabindex="-1" role="dialog" aria-
labelledby="addExpenseLabel" aria-hidden="true">

  <div class="modal-dialog modal-dialog-centered" role="document">

    <div class="modal-content">

      <form>

        <div class="modal-header">

          <h5 class="modal-title" id="addExpenseLabel">Add expense</h5>

          <button type="button" class="close" data-dismiss="modal" aria-
label="Close">

            <span aria-hidden="true">&times;</span>

          </button>

        </div>

        <div class="modal-body">

          <fieldset>

            <div class="form-group">

              <label for="expenseType">Expense type</label>

              <select class="form-control" name="expenseType"
id="expenseType">

                <option></option>

              </select>

            </div>

            <div class="form-group">

              <label for="expenseName">Expense name</label>

              <input type="text" class="form-control" placeholder="Give a name for
this form of expense" />

            </div>

            <div class="form-group">

              <label for="expenseFreq">How often is this paid?</label>

```

```

<select class="form-control" name="expenseFreq" id="expenseFreq">
  <option value="Weekly">Weekly</option>
  <option value="Monthly">Monthly</option>
  <option value="Quarterly">Quarterly</option>
  <option value="Termly">Termly</option>
  <option value="Yearly">Yearly</option>
</select>
</div>
<div class="form-group">
  <label for="expenseAmount">Amount paid</label>
  <div class="input-group">
    <div class="input-group-prepend">
      <div class="input-group-text">£</div>
    </div>
    <input type="text" class="form-control" name="expenseAmount"
id="expenseAmount" />
  </div>
  <small>Please enter the <b>total</b> amount due, no matter how you
split the cost (more on that below) </small>
</div>
<div class="form-group">
  <label for="isCommunal">Who foots the bill - everyone or just
you?</label>
  <select class="form-control" name="isCommunal" id="isCommunal">
    <option value="1">Everyone</option>
    <option value="0">Just me</option>
  </select>
</div>

```

```

        </fieldset>

    </div>

    <div class="modal-footer">

        <button type="button" class="btn btn-secondary" data-
dismiss="modal">Close</button>

        <button type="submit" name="expenseSave" id="expenseSave" class="btn
btn-primary">Save changes</button>

    </div>

</form>

</div>

</div>

</div>

<!-- Add / edit student -->

<div class="modal fade" id="addStudent" tabindex="-1" role="dialog" aria-
labelledby="addStudentLabel" aria-hidden="true">

    <div class="modal-dialog modal-dialog-centered" role="document">

        <div class="modal-content">

            <form>

                <div class="modal-header">

                    <h5 class="modal-title" id="addStudentLabel">Add student</h5>

                    <button type="button" class="close" data-dismiss="modal" aria-
label="Close">

                        <span aria-hidden="true">&times;</span>

                    </button>

                </div>

                <div class="modal-body">

                    <div class="form-group">

```

```

        <label for="studentSearch">Find a student to add to your
household:</label>

        <div class="input-group">

            <input type="search" class="form-control" aria-label="Search
students">

            <div class="input-group-append">

                <button class="btn btn-outline-secondary" id="studentSearch"
name="studentSearch" type="button">Search</button>

            </div>

        </div>

    </div>

</div>

<div class="modal-footer">

    <button type="button" class="btn btn-secondary" data-
dismiss="modal">Close</button>

    <button type="button" id="saveStudentHH" name="saveStudentHH"
class="btn btn-primary">Save changes</button>

</div>

</form>

</div>

</div>

</div>

<!-- Scripts -->

<script src="https://code.jquery.com/jquery-3.3.1.min.js" integrity="sha256-
FgpCb/KJQILNfOu91ta32o/NMZxltwRo8QtmkMRdAu8="
crossorigin="anonymous"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"
integrity="sha384-

```

```
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
crossorigin="anonymous"></script>
```

```
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"
integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYI"
crossorigin="anonymous"></script>
```

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

```
</body>
```

```
</html>
```

B. PlannerService

1. IMessageService.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Serialization;
using System.ServiceModel;
using System.ServiceModel.Web; // WebInvoke(), WebGet()
using System.Text;

/*
 * REST Web service follows four basic design principles:
 * + Use HTTP methods explicitly.
 * + Be stateless.
 * + Expose directory structure-like URIs.
 * + Transfer XML, JavaScript Object Notation (JSON), or both.
 *
 * REST asks developers to use HTTP methods explicitly and in a way that's
 * consistent with the protocol definition. This basic REST design principle
 * establishes a one-to-one mapping between Create, Read, Update, and Delete
 * (CRUD) operations and HTTP methods. According to this mapping:
 * + To create a resource on the server, use POST.
 * + To retrieve a resource, use GET.
 * + To change the state of a resource or to update it, use PUT.
 * + To remove or delete a resource, use DELETE.
 *
 * POST and PUT are expected to have request bodies.
 * GET, PUT and DELETE can have parameters specified in URI used.
 *
 * Stateless -- client (browser) must maintain any state required
 *
 * REST Web service URIs should be intuitive to the point where they are easy to guess.
 * Think of a URI as a kind of self-documenting interface that requires little,
 * if any, explanation or reference for a developer to understand what it points to and
 * to derive related resources. To this end, the structure of a URI should be
 * straightforward,
 * predictable, and easily understood e.g.
 * http://www.myservice.org/discussion/topics/{topic} where the user would specify
 * some {topic} to be searched in a database.
```

```

*
* Different content can be returned as required by the client.
*/
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.
    // NOTE: You can use the "Rename" command on the "Refactor" menu to change the
    // interface name "IMessageService" in both code and config file together.
    [ServiceContract]
    public interface IMessageService
    {
        // UriTemplates can be used with POST, GET, DELETE and PUT.
        // URI templates allow you to map a URI or a set of URIs to a service operation.
        // e.g. say we have a C# method called SendMessage(), using the
        // UriTemplate="/postmessage" allows MessageService.svc/postmessage
        // to be used instead of MessageService.svc/SendMessage allowing us to
        // change the method name later without modifying the service "seen" by
        // the user.
        // Note: If you want to use a URL instead of MessageService.svc, you need to
        // map a URL route in ASP.Net.
        // For GET, DELETE and PUT, you may also specify parameters to be read from the
        URL.
        // UriTemplate = "SendMessage?thesender={sender}&themessage={message}"

        // Use WebInvoke for all methods other than GET (although it will also work for
        GET, use WebGet instead).
        [OperationContract]
        [WebInvoke(Method = "POST",
the service
// Specifies the HTTP method
// operation is mapped to.

        BodyStyle = WebMessageBodyStyle.Wrapped,
requests and
// Controls whether to wrap
// responses sent to and
received from
// the service operation i.e.
whether parameters
// or returns are written
within
// infrastructure-provided
XML elements.
// Req'd, but has no bearing in
JSON

        RequestFormat = WebMessageFormat.Json,
messages are formatted.
// Controls how request

        ResponseFormat = WebMessageFormat.Json,
messages are formatted.
// Controls how response

        UriTemplate = "/postmessage")]
// e.g. MessageService.svc/postmessage
// instead of
// MessageService.svc/SendMessage
// POST is expected to have request body containing data
void SendMessage(string sender, string recipient, string message); // CREATE

```

```

// Use WebGet for GET methods.
[OperationContract]
[WebGet(BodyStyle = WebMessageBodyStyle.Wrapped,
RequestFormat = WebMessageFormat.Json,
ResponseFormat = WebMessageFormat.Json,
UriTemplate = "/readmessage")]
[return: MessageParameter(Name = "Message")] // Result is returned as
a JSON name/value pair
// By default, the name is the method name
// with "Result" appended
// e.g. "ReadLastMessageResult":"some string"
// Change this name to "hide" this method
// "Message":"some string". Use this name in
// JavaScript to recover the returned value.

string ReadLastMessage(); // READ

[OperationContract]
[WebGet(BodyStyle = WebMessageBodyStyle.Wrapped,
RequestFormat = WebMessageFormat.Json,
ResponseFormat = WebMessageFormat.Json,
UriTemplate = "/readmessage?id={msgId}")] // GET can pass parameters in
URL,
// (no request body)
[return: MessageParameter(Name = "Message")]
string ReadMessage(int msgId); // READ

// Out of interest, these will also work for a GET:
//
//[WebInvoke(ResponseFormat = WebMessageFormat.Json,
//           BodyStyle = WebMessageBodyStyle.Wrapped)]
//
//[WebInvoke(Method = "GET",
//           BodyStyle = WebMessageBodyStyle.Wrapped,
//           RequestFormat = WebMessageFormat.Json,
//           ResponseFormat = WebMessageFormat.Json)]
//

// More info here:
// http://msdn.microsoft.com/en-us/library/bb943471.aspx

[OperationContract]
[WebInvoke(Method = "PUT",
BodyStyle = WebMessageBodyStyle.Wrapped,
RequestFormat = WebMessageFormat.Json,
ResponseFormat = WebMessageFormat.Json,
UriTemplate = "/putmessage?id={msgId}")] // Read userId from URL, PUT
is expected to have a
// request body containing data (last argument is read
// from body).
[return: MessageParameter(Name = "Modifications")]
EmailMessage ModifyMessage(int msgId, EmailMessage messageChanges); // UPDATE

[OperationContract]
[WebGet(BodyStyle = WebMessageBodyStyle.Wrapped,
RequestFormat = WebMessageFormat.Json,
ResponseFormat = WebMessageFormat.Json,
UriTemplate = "/readall?recipient={recipient}&keyword={keyWord}")]

```

```

[return: MessageParameter(Name = "AllMessages")]
EmailMessage[] ReadAllMessages(string recipient, string keyword);    // READ

[OperationContract]
[WebInvoke(Method = "DELETE",
    BodyStyle = WebMessageBodyStyle.Wrapped,
    RequestFormat = WebMessageFormat.Json,
    ResponseFormat = WebMessageFormat.Json,
    UriTemplate = "/deletemessage?id={msgId}")]    // Read userId from URL
(No request body)
void DeleteMessage(int msgId);    // DELETE

// Handle Raw JSON
// http://stackoverflow.com/questions/2297903/generic-wcf-json-deserialization
//[OperationContract]
//[WebInvoke(Method = "POST",
//    BodyStyle = WebMessageBodyStyle.Bare,
//    RequestFormat = WebMessageFormat.Json,    WARNING: This didn't
work
//    ResponseFormat = WebMessageFormat.Json)]
//void Test(System.IO.Stream rawJSON);
}
}

```

2. MessageService.svc.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 // Changing to something else didn't work, despite changing
throughout web.config! Don't ask me why :(
{
    // 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>();

        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;
    }

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

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

    public EmailMessage[] ReadAllMessages(string recipient, string keyWord)
    {
        return _messages.ToArray();
    }

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

```

3. EmailMessage.cs

```

using System;
using System.Collections.Generic;

```

```

using System.Linq;
using System.Web;

using System.Runtime.Serialization;    // <-- Add this for DataContract

namespace CorsService
{
    // Use a data contract as illustrated in the sample below to add composite types to
    // service operations.
    [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; }
        }
    }
}

```

4. Global.asax.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.SessionState;

namespace CorsService
{
    public class Global : System.Web.HttpApplication
    {
        protected void Application_Start(object sender, EventArgs e)
        {
        }

        protected void Session_Start(object sender, EventArgs e)
        {
        }

        protected void Application_BeginRequest(object sender, EventArgs e)
        {
            // Can check the "Origin" header in the HTTP Request to determine
            // if request is allowed.
            // Access-Control-Allow-Origin = Accept request for resources.
            // Echo back the origin details from the request or * for a
            // public resource
            HttpContext.Current.Response.AddHeader("Access-Control-Allow-Origin", "*");

            if (HttpContext.Current.Request.HttpMethod == "OPTIONS")
            {
                //These headers are handling the "pre-flight" OPTIONS call sent by the
                browser
                HttpContext.Current.Response.AddHeader("Access-Control-Allow-Methods",
                "GET, POST, DELETE, PUT");
                HttpContext.Current.Response.AddHeader("Access-Control-Allow-Headers",
                "Authorization, Origin, Content-Type, Accept, X-Requested-With");

                // Set Max-Age to 0 to allow us to see preflight check, should be
                something like 1728000 to cache the check response
                HttpContext.Current.Response.AddHeader("Access-Control-Max-Age", "0");

                HttpContext.Current.Response.End();
            }

            //if (credentialsSupported == true)
            //{
            //    HttpContext.Current.Response.AddHeader("Access-Control-Allow-
            Credentials", "true");
            //}
        }

        protected void Application_AuthenticateRequest(object sender, EventArgs e)
        {
        }
    }
}
```

```

    }

    protected void Application_Error(object sender, EventArgs e)
    {
    }

    protected void Session_End(object sender, EventArgs e)
    {
    }

    protected void Application_End(object sender, EventArgs e)
    {
    }
}
}

```

5. Web.config

```

<?xml version="1.0" encoding="utf-8"?>
<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>
            <!-- Add endpointBehaviors here !! (Mike)
            http://msdn.microsoft.com/en-us/library/ms733932.aspx
            -->
            <endpointBehaviors>
                <behavior name="AjaxBehavior">
                    <!-- name: used to identify behaviour below-->
                    <!-- <webHttp/> enables the Web programming model with WCF, by default returns
XML-->
                    <!--<enableWebScript/> Alternatively, specify ASP.AJAX approach to return
dynamic JavaScript -->
                    <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="true" />
    </behavior>
</serviceBehaviors>
</behaviors>

<!-- Add services here !! (Mike) -->
<services>

    <!-- <service> the "name" is the C# class that implements the interface (or
contract) -->
    <service name="CorsService.MessageService">
        <!--
Can have multiple endpoints for service, so that you can access it in multiple
ways.

        "address":
Specifies the service's Uniform Resource Identifier (URI), which can be an
absolute
address or one that is given relative to the base address of the service.
You will need to add a URL route (see ASP MapPageRoute) to get to work.
If set to an empty string, it indicates that the endpoint is available at the
base address e.g. http://localhost for ASP application

        "behaviorConfiguration":
Link to behaviour specified above

        "binding":
The binding specified determines the type of transport, security and encoding
used, and whether
reliable sessions, transactions, or streaming is supported or enabled.

        "contract":
Specifies the C# interface that defines the contract.
-->
        <endpoint address="" behaviorConfiguration="AjaxBehavior"
binding="webHttpBinding" contract="CorsService.IMessageService" />
        <!--
NOT REQD when hosting in IIS
If self hosting, can set up baseAddresses
... self hosting = write code to execute service yourself
e.g. using a console application.
        <host>
            <baseAddresses>
                <add baseAddress="http://localhost:44037/" />
            </baseAddresses>
        </host>-->
    </service>
</services>

    <protocolMapping>
        <add binding="basicHttpsBinding" scheme="https" />
    </protocolMapping>
    <serviceHostingEnvironment aspNetCompatibilityEnabled="true"
multipleSiteBindingsEnabled="true" />
</system.serviceModel>

```

```

<system.webServer>
  <modules runAllManagedModulesForAllRequests="true">
    <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>

<runtime>

  <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">

    <dependentAssembly>

      <assemblyIdentity name="System.Diagnostics.DiagnosticSource"
publicKeyToken="cc7b13ffcd2ddd51" culture="neutral" />

      <bindingRedirect oldVersion="0.0.0.0-4.0.2.0" newVersion="4.0.2.0" />

    </dependentAssembly>

    <dependentAssembly>

      <assemblyIdentity name="System.Text.Encodings.Web"
publicKeyToken="cc7b13ffcd2ddd51" culture="neutral" />

      <bindingRedirect oldVersion="0.0.0.0-4.0.1.1" newVersion="4.0.1.1" />

    </dependentAssembly>

    <dependentAssembly>

      <assemblyIdentity name="Newtonsoft.Json" publicKeyToken="30ad4fe6b2a6aeed"
culture="neutral" />

      <bindingRedirect oldVersion="0.0.0.0-11.0.0.0" newVersion="11.0.0.0" />

    </dependentAssembly>

    <dependentAssembly>

      <assemblyIdentity name="System.Collections.Immutable"
publicKeyToken="b03f5f7f11d50a3a" culture="neutral" />

      <bindingRedirect oldVersion="0.0.0.0-1.2.2.0" newVersion="1.2.2.0" />

    </dependentAssembly>

  </assemblyBinding>

</runtime>
</configuration>

```

Emerging Web Tech

GRADEMARK REPORT

FINAL GRADE

56/100

GENERAL COMMENTS

Instructor

Big data application. (20) 13

Discussion of Hadoop technology (20) 12

Quality of website (Typescript) (20) 11

Single page application (20) 10

Dynamic content using AJAX (15) 8

Software testing (5) 2

Sometimes the discussion of four V's, provides an interesting example but not enough explanation of their importance. Discussion of HBase starts off OK, but both trials off when it starts to getting interesting and is a bit too brief.

Typescript used with a REST service, although not complete.

Structure of single page app is a bit basic.

PAGE 1

PAGE 2



Comment 1

Ref



Comment 2

I'm not convinced this concept is directly related to the velocity of the data being received.



Comment 3

You make a nice point you could have expanded on the nature of this data and the implications to "BigData"



Comment 4

Quite a good example, but you needed to provide a little more evaluation of the implications.



Comment 5

This is a bit brief.



Comment 6

But you did have a working version to start with.

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

PAGE 39

PAGE 40

PAGE 41

PAGE 42

PAGE 43

PAGE 44

PAGE 45

PAGE 46

PAGE 47

PAGE 48

PAGE 49

PAGE 50

PAGE 51

PAGE 52

PAGE 53
