

Mobile-enabled website utlising REST services

by Richard Lamnea

FILE	11200_RICHARD_LAMNEA_MOBILE-ENABLED_WEBSITE_UTLISING_REST_SERVICES_308223_1780823991.D OCX (3.76M)		
TIME SUBMITTED	28-JAN-2021 02:50PM (UTC+0000)	WORD COUNT	3952
SUBMISSION ID	143611718	CHARACTER COUNT	26294

University of Wales Trinity St David

School of Applied Computing

Mobile-enabled website utilising REST services (JSON)

Web and Mobile Technologies

ACCA5031

Richard Lamnea

1802749

10/01/2021

ABSTRACT

This document discusses implementation of a JSON REST service and its application within a real-world scenario.

TABLE OF CONTENTS

I. INTRODUCTION	1
II. IMPLEMENTATION	1
A. SCREENSHOTS OF THE APPLICATION	1
III. METHODOLOGY	3
A. ANNOTATED CODE	3
IV. CONCLUSION	15
V. REFERENCE LIST	15

I. INTRODUCTION

Students were required to create an application using jQuery mobile and REST services using JSON. In this case the application utilizes jQuery mobile for the user interface and JavaScript & jQuery to implement 3 components. The first component of the application uses location data stored within a database to request atmospheric data from an external API. The second component requests an input from the user and then requests data from an external API from the user input. The third component requests the data from the external API and displays this data on the application.

II. IMPLEMENTATION

The application is created using HTML and jQuery Mobile framework for the user interface. The application also utilises jQuery and JavaScript to consume data from a REST API and display the data within the application [1]. The application requires the user to login prior to the components being activated which can be seen in Fig 1. When a user registers the application requests the user's location which is used in the first JavaScript component which can be seen in Fig 2. A user will need to log in which can be seen in Fig 3 and 4. Once a user has logged into the application, the index.html will then display atmospheric data from the user's location data which was input when registering which can be seen in Fig 5. The application utilizes jQuery mobile user interface panels to create a navigation menu which can be seen in Fig 6. jQuery mobile was used as this framework as it allows the use of single page applications and uses an AJAX navigation system to load pages into the DOM when requested [2]. This navigation requests the user to input a location and then requests JSON data from the REST API to be consumed by the application and then displays the requested information onto the homepage which can be seen in Fig 7. The navigation menu also allows the user to request forecast information of the weather for 3 hours, 6 hours, 9 hours and 12 hours in advance which can be seen in Fig 8 and 9.

A. SCREENSHOTS OF THE APPLICATION

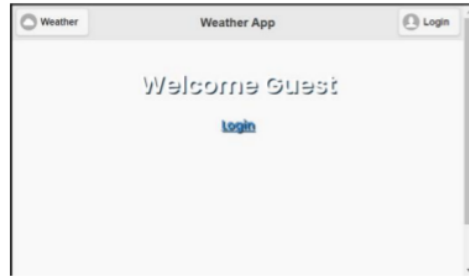


Figure 1 - Index.html.

Figure 2 - Registration form.

Figure 3 - Login form.



Figure 4 - logged in.



Figure 5 - Component 1, User logged in to application.

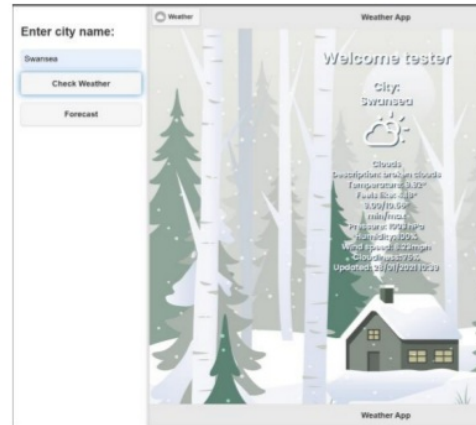


Figure 7 - Component 2, user input and display data on page.

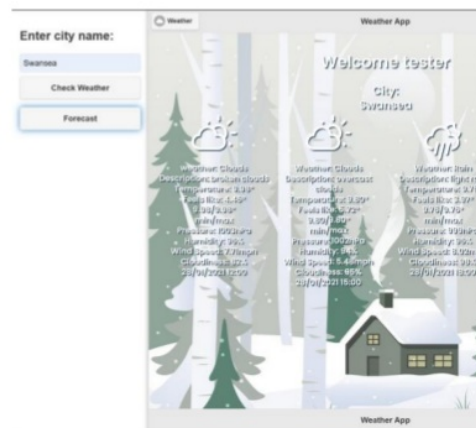


Figure 8 - Component 3, display forecast information.

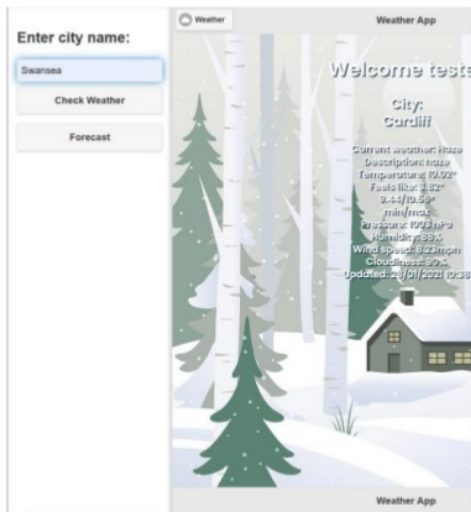


Figure 6 - jQuery mobile user interface.

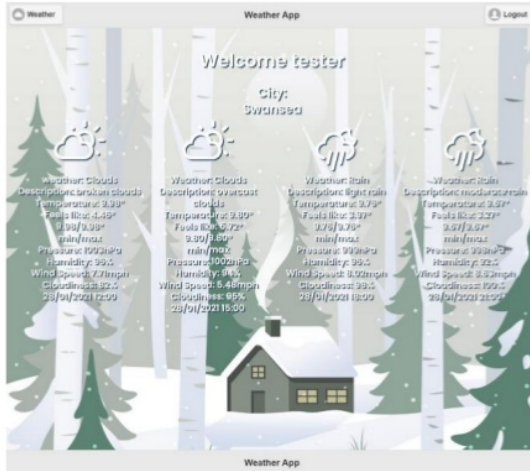


Figure 9 - Component 3 implementation.

III. METHODOLOGY

The first component uses the login session variable which was created using a PHP library. This saved time and effort when creating my own bespoke solution and provided basic login and registration functionality required. This component also utilizes jQuery to consume JSON from the OpenWeatherMap REST API [3]. Once consumed the component validates and converts the data to be displayed on the homepage. This component calculates which background image to display on the homepage dependent on the time and date of the REST data. If the season is winter and the time of day is night, the program will display a night-time winter image. If the season is summer and the time of day is day, the program will display a day-time summer image.

The second component is like the first component however using the jQuery mobile user interface, data is requested from the user. When the user has entered a location, the function will request data from the REST API. This data is then validated and converted to be displayed on the homepage, replacing the original solution. This component also calls another function which calculates which icon to be displayed on the screen using the time and date of the REST API data.

The third component uses jQuery to consume the REST API data and displays multiple values onto the webpage in a 4-grid layout. This data produces a 3-, 6-, 9- and 12-hour weather forecast and displays the correct icon dependent on the name of the weather from the REST data. This approach was taken to allow the user to read forecast information for the future 12 hours.

A. ANNOTATED CODE

Below you can see the annotated code for the application.

```
<!--
- jQuery UI,, jQuery and Javascript JSON REST Impleme
ntation -->
<?session_start()>
```

```
<html>

<head>
<title>Weather App</title>
<meta name="viewport" content="width=device-
width, initial-scale=1">
<!-- Import jQuery mobile and weather icon css --
>
<link rel="stylesheet" href="http://code.jquery.com/m
obile/1.4.5/jquery.mobile-1.4.5.min.css" />
<link rel="stylesheet" href="http://localhost/css/wea
ther-icons.css" />
<!-- Import jQuery javascript -->
<script src="http://code.jquery.com/jquery-
1.11.1.min.js"></script>
<script src="http://code.jquery.com/mobile/1.4.5/jque
ry.mobile-1.4.5.min.js"></script>
<!-- Import googlefonts -->
<link rel="preconnect" href="https://fonts.gstatic.co
m">
<link href="https://fonts.googleapis.com/css2?family=
Poppins:wght@600&display=swap" rel="stylesheet">
<!-- Import custom stylesheet -->
<link rel="stylesheet" href="/css/styles.css"/>
</head>

<!-- Onload activate component -->
<body onLoad="javascript:userWeather()">

<!-- Single page -->
<div data-role="page" id="page1">

<!-- Navigation -->
<div data-role="header">
<h1>Weather App</h1>
<a href='#leftpanel' data-
icon="cloud" class='ui-btn-left'>Weather</a>

<!--
- Check if user is logged in from session variables -
->

<? if (@$_SESSION["username"]!= "")
{
    echo "<a href='logout.php' data-
icon='user' class='ui-btn-right'>Logout</a>";
```

```

    }
    else {

        echo "<a href='login.php' data-
icon='user' class='ui-btn-right'>Login</a>";

    }
    ?>

</div> <!-- end of header -->

<!-- Main content page -->
<div role="main" class="ui-content">
    <!--
- if user logged in display welcome and their name --
>

    <? if (@$_SESSION["username"]!="")
    {
        echo "<h1>Welcome ".$_SESSION["userna
me"]."</h1>";
    }
    <!--
- Display city data from session variables -->
    <div class="cityTitle" id="CityTitle">Cit
y:

        <div id="sessionData">
        <? if (@$_SESSION["username"]!="")
        {
            echo $_SESSION["city"];
        };?>
        </div>

    </div>
    <!--
- JSON REST container element for user weather & inpu
t weather -->
    <div id="Weather"></div>

    <!-- Container for forecast data -->
    <div class="ui-grid-c my-breakpoint">

        <div class="ui-block-
a" id="Forecast3hr"></div>

        <div class="ui-block-
b" id="Forecast6hr"></div>

        <div class="ui-block-
c" id="Forecast9hr"></div>

```

```

        <div class="ui-block-
d" id="Forecast12hr"></div>
    </div>
    <!--
- If user isn't logged in display login page -->
    <?
    }else
    {

        echo "<h1>Welcome Guest</h1>";
        echo "<a href='login.php'>Login</a>";

    }
    ?>
</div>
<!--end of content -->

<!--
- jQuery mobile left panel navigation menu -->
    <div data-role="panel" id="leftpanel">

        <form action="javascript:inputWeather()"
method="post" data-ajax="false">
            <h2>Enter city name: </h2>
            <input type="text" name="InputCity" id="i
nputCity" placeholder="<? echo $_SESSION['city']; ?>"
>

            <input type="Submit" name="Submit" value=
"Check Weather">
        </form>

        <form action="javascript:forecastWeather(
)"
            method="post" data-ajax="false">
            <input type="Submit" value="Forecast"
name="Forecast">

        </form>
    </div>
<!-- Footer -->
    <div data-role="footer">
        <h4>Weather App</h4>
    </div><!-- end of footer -->

</div><!-- end of page1 -->

```

```

</body>

<script>

    // Component 1

    function userWeather () {
        // request session data city value
        var City = document.getElementById("sessionData").textContent;
        //instantiate XML Request
        let xmlhttp = new XMLHttpRequest();
        // Send API url and City variable to request data
        xmlhttp.open('GET', 'http://api.openweathermap.org/data/2.5/weather?q='+City+'&appid=aaec84b5cbf5f9e8c3ee5c2f074db965');

        // set to JSON
        xmlhttp.responseType = 'json';
        // Send the request
        xmlhttp.send();

        // If xml request returned data, activate this function
        xmlhttp.onload = function() {

            // get weather element in html to display content
            weatherEle = document.getElementById("Weather");

            // set response values to an object
            let myObj = xmlhttp.response;

            // search through the object to get name and description
            for (i = 0; i < myObj["weather"].length; i++){

                weather = myObj["weather"][i]["main"];
                weatherDesc = myObj["weather"][i]["description"];
            }

            // get temperature value from JSON and convert to celcius
            temperature = parseFloat(myObj["main"]["temperature"]) - 273.15;
            // set 0.00 decimal places
            temperature = temperature.toFixed(2);
            // temperature feels like value from JSON and convert to celcius
            tempfeels = parseFloat(myObj["main"]["feels_like"]) - 273.15;
            // set 0.00 decimal places
            tempfeels = tempfeels.toFixed(2);
            // Min & Max temperature, convert to celcius and set decimal places.
            tempMin = parseFloat(myObj["main"]["temp_min"]) - 273.15;
            tempMin = tempMin.toFixed(2);
            tempMax = parseFloat(myObj["main"]["temp_max"]) - 273.15;
            tempMax = tempMax.toFixed(2);

            // get pressure reading
            pressure = myObj["main"]["pressure"];
            // get humidity reading
            humidity = myObj["main"]["humidity"];
            //get windspeed
            windSpeed = myObj["wind"]["speed"];
            // get cloud data
            clouds = myObj["clouds"]["all"];
            // get last update of REST API
            lastUpdate = myObj["dt"];

            // use lastUpdate value to create a timestamp variable
            var timestamp = lastUpdate ; // replace your timestamp

            // create a date object using Date function on timestamp, multiplied by 1000 because of UNIX time conversion
            var date = new Date(timestamp * 1000);

```

```

        // Format date
        var formattedDate = ('0' + date.getDate()
    ).slice(-
    2) + '/' + ('0' + (date.getMonth() + 1)).slice(-
    2) + '/' + date.getFullYear() + ' ' + ('0' + date.get
    Hours()).slice(-
    2) + ':' + ('0' + date.getMinutes()).slice(-2);

    // get current hour
    hour = date.getHours();
    // get current month + 1
    month = date.getMonth();
    month = month + 1;

    // activate background image function - t
    his function takes the current time, and month to dis
    play different images
    dateBackground(hour,month);

    // activate weather icon function to calc
    ulate the correct icon based on weather and time
    weathericon = weatherIcon(hour, weather)

    // Display weather data onto application
    weatherEle.innerHTML += weathericon + "
    <br>" +
    "Current weather: " + weather +
    "<br>" + "Description: " + weatherDesc +
    "<br>" + "Temperature: " + temperature + "°" + "<br>"
    + "Feels like: " + tempfeels + "°" + "<br>" + tempMi
    n + "/" + tempMax + "°" + "<br>min/max" + "<br>"
    + "Pressure: " + pressure + " hPa<br>" +
    "Humidity: " + humidity + "%" + "<br>" + "Wind speed:
    " + windSpeed + "mph" + "<br>"
    + "Cloudiness: " + clouds + "%" + "<br>"
    + "Updated: " + formattedDate;

    }

    }
    // End function

    // Component 2 user input weather
    function inputWeather()
    {
        // get user input and city session da
    ta

```

```

        input = document.getElementById("inpu
    tCity").value;
        city = document.getElementById("sessi
    onData")

        // set the city session value as inpu
    t data, if there is no input the application will use
    the city data

        city.innerHTML = input;
        // request API data
        let xmlhttp = new XMLHttpRequest();

        xmlhttp.open('GET', 'http://api.openwe
    athermap.org/data/2.5/weather?q='+input+'&appid=aaec8
    4b5cbf5f9e8c3ee5c2f074db965');

        xmlhttp.responseType = 'json';

        xmlhttp.send();

        xmlhttp.onload = function () {

            // get all display elements
            weatherEle = document.getElementById(
    "Weather")

            forecastEle3hr = document.getElementB
    yId("Forecast3hr");
            forecastEle6hr = document.getElementB
    yId("Forecast6hr");
            forecastEle9hr = document.getElementB
    yId("Forecast9hr");
            forecastEle12hr = document.getElement
    ById("Forecast12hr");

            // Display weather element and hi
    de forecast elements

            weatherEle.style.display = "block";
            forecastEle3hr.style.display = "none"

            ;

            forecastEle6hr.style.display="none";
            forecastEle9hr.style.display="none";
            forecastEle12hr.style.display="none";

            let newObj = xmlhttp.response;
            // Consume JSON values

```

```

        for (i = 0; i < newObj["weather"].length;
i++){
            weather = newObj["weather"][i]["main"
];
            weatherDesc = newObj["weather"][i]["d
escription"];
        }
            temperature = parseFloat(newObj["mai
n"]["temp"]) - 273.15;
            temperature = temperature.toFixed(2);
            tempfeels = parseFloat(newObj["main"]["fe
els_like"]) - 273.15;
            tempfeels = tempfeels.toFixed(2);
            tempMin = parseFloat(newObj["main"]["temp
_min"]) - 273.15;
            tempMin = tempMin.toFixed(2);
            tempMax = parseFloat(newObj["main"]["temp
_max"]) - 273.15;
            tempMax = tempMax.toFixed(2);

            pressure = newObj["main"]["pressure"];

            humidity = newObj["main"]["humidity"];

            windSpeed = newObj["wind"]["speed"];

            clouds = newObj["clouds"]["all"];

            lastUpdate = newObj["dt"];

            // convert timestamp to formatted
data
            var timestamp = lastUpdate ; // replace y
our timestamp
            var date = new Date(timestamp * 1000);
            var formattedDate = ('0' + date.getDate()
).slice(-
2) + '/' + ('0' + (date.getMonth() + 1)).slice(-
2) + '/' + date.getFullYear() + ' ' + ('0' + date.get
Hours()).slice(-
2) + ':' + ('0' + date.getMinutes()).slice(-2);

            // get current time
            hour = date.getHours();

            // calculate correct icon using time and
current weather.

```

```

        weathericon = weatherIcon(hour, weather);

        // display input weather.
        weatherEle.innerHTML = weathericon + "<br
>" + weather + "<br>" + "Description: " + weatherDes
c + "<br>" + "Temperature: " + temperature + "°" + "<
br>" + "Feels like: " + tempfeels + "°" + "<br>" + te
mpMin + "/" + tempMax + "°" + "<br>min/max" + "<br>"
+ "Pressure: " + pressure + " hPa<br>" +
"Humidity: " + humidity + "%" + "<br>" + "Wind speed:
" + windSpeed + "mph" + "<br>"
+ "Cloudiness: " + clouds + "%" + "<br>"
+ "Updated: " + formattedDate;

    }

    // end of function

    // component 3 forecast data
    function forecastWeather()
    {
        // request city from session data
        city = document.getElementById("sessi
onData").textContent;

        // make XML Request
        let xmlhttp = new XMLHttpRequest();
        // Set xml Request url and input
        xmlhttp.open('GET', 'https://api.openw
eathermap.org/data/2.5/forecast?q=' + city + '&appid=aa
ec84b5cbf5f9e8c3ee5c2f074db965', true);

        // set xml request type = JSON
        xmlhttp.responseType = 'json';

        // Send the Get Request
        xmlhttp.send();

        // If data loaded activate function
        xmlhttp.onload = function () {
            // set object variable to JSON re
sponse

            let myObj = xmlhttp.response;

```

```

        // Set weather element
        weatherEle = document.getElementById(
            "Weather");

        // set forecast container element
        forecastEle3hr = document.getElementById("Forecast3hr");
        forecastEle6hr = document.getElementById("Forecast6hr");
        forecastEle9hr = document.getElementById("Forecast9hr");
        forecastEle12hr = document.getElementById("Forecast12hr");

        // Hide weather element, display forecast elements on page
        weatherEle.style.display = "none";
        forecastEle3hr.style.display = "block";
        forecastEle6hr.style.display = "block";
        forecastEle9hr.style.display = "block";
        forecastEle12hr.style.display = "block";

        // consume JSON
        for (i = 0; i < myObj["list"].length; i++){
            if (i == 0)
            {
                // Grab timestamp of update for each forecast
                lastUpdate = myObj["list"][i]["dt"];
                lastUpdate6hr = myObj["list"][i+1]["dt"];
                lastUpdate9hr = myObj["list"][i+2]["dt"];
                lastUpdate12hr = myObj["list"][i+3]["dt"];

                // Grab and convert temperature values from API
                temperature = myObj["list"][i]["main"]["temp"];

                // Convert value to celcius
                temperature = parseFloat(temperature)-273.15;

                // Convert value to decimal value
                temperature = temperature.toFixed(2);

                // 6 hour temperature forecast
                temperature6hr = myObj["list"][i+1]["main"]["temp"];
                temperature6hr = parseFloat(temperature6hr)-273.15;
                temperature6hr = temperature6hr.toFixed(2);

                // 9 hour temperature forecast
                temperature9hr = myObj["list"][i+2]["main"]["temp"];
                temperature9hr = parseFloat(temperature9hr)-273.15;
                temperature9hr = temperature9hr.toFixed(2);

                // 12 hour temperature forecast
                temperature12hr = myObj["list"][i+3]["main"]["temp"];
                temperature12hr = parseFloat(temperature12hr)-273.15;
                temperature12hr = temperature12hr.toFixed(2);

                // Feels like temperature for forecasts
                //Get value, convert to float, and 0.00 decimal value
                feelsLikeTemp = myObj["list"][i]["main"]["feels_like"];
                feelsLikeTemp = parseFloat(feelsLikeTemp) - 273.15;
                feelsLikeTemp = feelsLikeTemp.toFixed(2);

                // 6 Hr forecast for feels like temperature

```

```

        feelsLikeTemp6hr = myObj["list"][i+1]["main"]["feels_like"];
        feelsLikeTemp6hr = parseFloat(feelsLikeTemp6hr) - 273.15;
        feelsLikeTemp6hr = feelsLikeTemp6hr.toFixed(2);
        // 9 Hr forecast for feels like temperature
        feelsLikeTemp9hr = myObj["list"][i+2]["main"]["feels_like"];
        feelsLikeTemp9hr = parseFloat(feelsLikeTemp9hr) - 273.15;
        feelsLikeTemp9hr = feelsLikeTemp9hr.toFixed(2);
        // 12 Hr forecast for feels like temperature
        feelsLikeTemp12hr = myObj["list"][i+3]["main"]["feels_like"];
        feelsLikeTemp12hr = parseFloat(feelsLikeTemp12hr) - 273.15;
        feelsLikeTemp12hr = feelsLikeTemp12hr.toFixed(2);

        // Get minimum temperature values, convert to celcius and set 0.00 decimal places
        tempMin = myObj["list"][i]["main"]["temp"];
        tempMin = parseFloat(tempMin) - 273.15;
        tempMin = tempMin.toFixed(2);
        // 6Hr minimum temperature
        tempMin6hr = myObj["list"][i+1]["main"]["temp"];
        tempMin6hr = parseFloat(tempMin6hr) - 273.15;
        tempMin6hr = tempMin6hr.toFixed(2);
        // 9 Hr minimum temperature
        tempMin9hr = myObj["list"][i+2]["main"]["temp"];
        tempMin9hr = parseFloat(tempMin9hr) - 273.15;

        tempMin9hr = tempMin9hr.toFixed(2);
        // 12 Hr minimum temperature
        tempMin12hr = myObj["list"][i+3]["main"]["temp"];
        tempMin12hr = parseFloat(tempMin12hr) - 273.15;
        tempMin12hr = tempMin12hr.toFixed(2);

        // get maximum temperature values, convert to celcius and set 0.00 decimal places
        tempMax = myObj["list"][i]["main"]["temp"];
        tempMax = parseFloat(tempMax) - 273.15;
        tempMax = tempMax.toFixed(2);
        // 6Hr maximum temperatures
        tempMax6hr = myObj["list"][i+1]["main"]["temp"];
        tempMax6hr = parseFloat(tempMax6hr) - 273.15;
        tempMax6hr = tempMax6hr.toFixed(2);
        // 9 Hr maximum temperatures
        tempMax9hr = myObj["list"][i+2]["main"]["temp"];
        tempMax9hr = parseFloat(tempMax9hr) - 273.15;
        tempMax9hr = tempMax9hr.toFixed(2);
        // 12 Hr maximum temperature
        tempMax12hr = myObj["list"][i+3]["main"]["temp"];
        tempMax12hr = parseFloat(tempMax12hr) - 273.15;
        tempMax12hr = tempMax12hr.toFixed(2);

        // Get forecast pressure, 3hr, 6hr, 9hr & 12hr
        pressure = myObj["list"][i]["main"]["pressure"];

```

```

        pressure6hr = myObj["list"][i+1][
"main"]["pressure"];
        pressure9hr = myObj["list"][i+2][
"main"]["pressure"];
        pressure12hr = myObj["list"][i+3][
"main"]["pressure"];

        // Get humidity for forecast,
        3hr, 6hr, 9 hr & 12hr
        humidity = myObj["list"][i][
"main"]["humidity"];
        humidity6hr = myObj["list"][i+1][
"main"]["humidity"];
        humidity9hr = myObj["list"][i+2][
"main"]["humidity"];
        humidity12hr = myObj["list"][i+3][
"main"]["humidity"];

        // Get weather name & description
        forecast 3hr
        for (j=0; j < myObj["list"][i][
"weather"].length; j++)
        {
            weatherName = myObj["list"][i][
"weather"][j][
"main"];

            weatherDesc = myObj["list"][i][
"weather"][j][
"description"];
        }
        // Get weather name & description
        forecast 6hr
        for (j=0; j < myObj["list"][i+1][
"weather"].length; j++)
        {
            weatherName6hr = myObj["list"][i+1][
"weather"][j][
"main"];

            weatherDesc6hr = myObj["list"][i+1][
"weather"][j][
"description"];
        }

        // Get weather name & description
        forecast 9hr
        for (j=0; j < myObj["list"][i+1][
"weather"].length; j++)
        {
            weatherName9hr = myObj["list"][i+1][
"weather"][j][
"main"];

            weatherDesc9hr = myObj["list"][i+1][
"weather"][j][
"description"];
        }

        // Get weather name & description
        forecast 12hr
        for (j=0; j < myObj["list"][i+3][
"weather"].length; j++)
        {
            weatherName12hr = myObj["list"][i+3][
"weather"][j][
"main"];

            weatherDesc12hr = myObj["list"][i+3][
"weather"][j][
"description"];
        }

        // Get cloud data for forecast
        clouds = myObj["list"][i][
"clouds"][
"all"];
        clouds6hr = myObj["list"][i+1][
"clouds"][
"all"];
        clouds9hr = myObj["list"][i+2][
"clouds"][
"all"];
        clouds12hr = myObj["list"][i+3][
"clouds"][
"all"];

        // Get windspeed for forecast
        , 3hr,6hr,9hr,12hr
        windSpeed = myObj["list"][i][
"wind"][
"speed"];
        windSpeed6hr = myObj["list"][i+1][
"wind"][
"speed"];
        windSpeed9hr = myObj["list"][i+2][
"wind"][
"speed"];
        windSpeed12hr = myObj["list"][i+3][
"wind"][
"speed"];

```

```

    }
}
// Process values gathered to display
content

// replace the timestamp with updated
values

var timestamp = lastUpdate ;
var timestamp6hr = lastUpdate6hr;
var timestamp9hr = lastUpdate9hr;
var timestamp12hr = lastUpdate12hr;

// Convert date for forecast
var date = new Date(timestamp * 1000)
;

var date6hr = new Date(timestamp6hr *
1000);

var date9hr = new Date(timestamp9hr *
1000);

var date12hr = new Date(timestamp12hr
* 1000);

// Formate Date to display for each t
ime

var formattedDate = ('0' + date.getDa
te()).slice(-
2) + '/' + ('0' + (date.getMonth() + 1)).slice(-
2) + '/' + date.getFullYear() + ' ' + ('0' + date.get
Hours()).slice(-
2) + ':' + ('0' + date.getMinutes()).slice(-2);

var formattedDate6hr = ('0' + date6hr
.getDate()).slice(-
2) + '/' + ('0' + (date6hr.getMonth() + 1)).slice(-
2) + '/' + date6hr.getFullYear() + ' ' + ('0' + date6
hr.getHours()).slice(-
2) + ':' + ('0' + date6hr.getMinutes()).slice(-
2);

var formattedDate9hr = ('0' + date9hr
.getDate()).slice(-
2) + '/' + ('0' + (date9hr.getMonth() + 1)).slice(-
2) + '/' + date9hr.getFullYear() + ' ' + ('0' + date9
hr.getHours()).slice(-
2) + ':' + ('0' + date9hr.getMinutes()).slice(-2);

var formattedDate12hr = ('0' + date12
hr.getDate()).slice(-
2) + '/' + ('0' + (date12hr.getMonth() + 1)).slice(-
2) + '/' + date12hr.getFullYear() + ' ' + ('0' + date
12hr.getHours()).slice(-
2) + ':' + ('0' + date12hr.getMinutes()).slice(-
2);

// get time
hour3 = date.getHours();
hour6 = date6hr.getHours();
hour9 = date9hr.getHours();
hour12 = date12hr.getHours();

// using time calculate forecast Icon
from weather function.
forecast3hrIcon = weatherIcon(hour3,
weatherName );
forecast6hrIcon = weatherIcon(hour6,
weatherName6hr);
forecast9hrIcon = weatherIcon(hour9,
weatherName9hr);
forecast12hrIcon = weatherIcon(hour12
, weatherName12hr);

// Display Values for 3 hour forecas
t

forecastEle3hr.innerHTML =
forecast3hrIcon + "<br>" +
"weather: " + weatherName + "<br>" +
"Description: " + weatherDesc + "<br>"
+
"Temperature: " + temperature + "°" +
"<br>" +
"Feels like: " + feelsLikeTemp + "°"
+ "<br>" +
tempMin + "/" + tempMax + "°" + "<br>"
+
"min/max" + "<br>" +
"Pressure: " + pressure + "hPa" + "<b
r>" +
"Humidity: " + humidity + "%" + "<br>"
+

```

```

        "Wind Speed: " + windSpeed + "mph" +
    "<br>" +
    "Cloudiness: " + clouds + "%" + "<br>" +
    " +
    formattedDate;

    // Display Values for 6 hour forecast

    forecastEle6hr.innerHTML =
    forecast6hrIcon + "<br>" +
    "Weather: " + weatherName6hr + "<br>"
    +
    "Description: " + weatherDesc6hr + "<
    br>" +
    "Temperature: " + temperature6hr + "°
    " + "<br>" +
    "Feels like: " + feelsLikeTemp6hr + "
    °" + "<br>" +
    tempMin6hr + "/" + tempMax6hr + "°" +
    "<br>" +
    "min/max" + "<br>" +
    "Pressure: " + pressure6hr + "hPa" +
    "<br>" +
    "Humidity: " + humidity6hr + "%" + "<
    br>" +
    "Wind Speed: " + windSpeed6hr + "mph"
    + "<br>" +
    "Cloudiness: " + clouds6hr + "%" + "<
    br>" +
    formattedDate6hr;

    // Display Values for 9 hour forecast

    forecastEle9hr.innerHTML =
    forecast9hrIcon + "<br>" +
    "Weather: " + weatherName9hr + "<br>"
    +
    "Description: " + weatherDesc9hr + "<
    br>" +
    "Temperature: " + temperature9hr + "°
    " + "<br>" +
    "Feels like: " + feelsLikeTemp9hr + "
    °" + "<br>" +
    tempMin9hr + "/" + tempMax9hr + "°" +
    "<br>" +
    "min/max" + "<br>" +

```

```

    "Pressure: " + pressure9hr + "hPa" +
    "<br>" +
    "Humidity: " + humidity9hr + "%" + "<
    br>" +
    "Wind Speed: " + windSpeed9hr + "mph"
    + "<br>" +
    "Cloudiness: " + clouds9hr + "%" + "<
    br>" +
    formattedDate9hr;

    // Display Values for 12 hour forecas
    t

    forecastEle12hr.innerHTML =
    forecast12hrIcon + "<br>" +
    "Weather: " + weatherName12hr + "<br>"
    +
    "Description: " + weatherDesc12hr + "
    <br>" +
    "Temperature: " + temperature12hr + "
    °" + "<br>" +
    "Feels like: " + feelsLikeTemp12hr +
    "°" + "<br>" +
    tempMin12hr + "/" + tempMax12hr + "°"
    + "<br>" +
    "min/max" + "<br>" +
    "Pressure: " + pressure12hr + "hPa" +
    "<br>" +
    "Humidity: " + humidity12hr + "%" + "
    <br>" +
    "Wind Speed: " + windSpeed12hr + "mph
    " + "<br>" +
    "Cloudiness: " + clouds12hr + "%" + "
    <br>" +
    formattedDate12hr;

    }
}
// end of component 3

```

// function to determine icon from icons folder

```

function weatherIcon(time, weather) {
    // instantiate weather icon v
    weathericon = "";

```

```

//string Weather Icons night
clear = "<i class='wi wi-night-
sunny icon'></i>"
    nightcloudy = "<i class='wi wi-night-alt-
cloudy icon'></i>"
    nightcloudygusts = "<i class='wi wi-
night-alt-cloudy-gusts icon'></i>"
    nightcloudywindy = "<i class='wi wi-
night-alt-windy icon'></i>"
    nightfog = "<i class='wi wi-night-
fog icon'></i>"
    nightthail = "<i class='wi wi-night-alt-
hail icon'></i>"
    nightlightning = "<i class='wi wi-night-
alt-lightning icon'></i>"
    nightrain = "<i class='wi wi-night-alt-
rain icon'></i>"
    nightrainmix = "<i class='wi wi-night-
alt-rain-mix icon'></i>"
    nightrainwind = "<i class='wi wi-night-
alt-rain-wind icon'></i>"
    nightshowers = "<i class='wi wi-night-
alt-showers icon'></i>"
    nightsleet = "<i class='wi wi-night-alt-
sleet icon'></i>"
    nightsleetstorm = "<i class='wi wi-night-
alt-sleet-storm icon'></i>"
    nightsnow = "<i class='wi wi-night-alt-
snow icon'></i>"
    nightsnowthunderstorm = "<i class='wi wi-
night-alt-thunderstorm icon'></i>"
    nightsnowwind = "<i class='wi wi-night-
alt-snow-wind icon'></i>"
    nightsprinkle = "<i class='wi wi-night-
alt-sprinkle icon'></i>"
    nightstormshowers = "<i class='wi wi-
night-alt-storm-showers icon'></i>"
    nightthunderstorm = "<i class='wi wi-
night-alt-thunderstorm icon'></i>"

// string Weather Icons daytime
sunny = "<i class='wi wi-day-
sunny icon'></i>"
    cloudy = "<i class='wi wi-day-
cloudy icon'></i>"

```

```

cloudygusts = "<i class='wi wi-day-
cloudy-gusts icon'></i>"
    cloudywindy = "<i class='wi wi-day-
cloudy-windy icon'></i>"
    fog = "<i class='wi wi-day-
fog icon'></i>"
    hail = "<i class='wi wi-day-
hail icon'></i>"
    lightning = "<i class='wi wi-day-
lightning icon'></i>"
    rain = "<i class='wi wi-day-
rain icon'></i>"
    rainmix = "<i class='wi wi-day-rain-
mix icon'></i>"
    rainwind = "<i class='wi wi-day-rain-
wind icon'></i>"
    showers = "<i class='wi wi-day-
showers icon'></i>"
    sleet = "<i class='wi wi-day-
sleet icon'></i>"
    sleetstorm = "<i class='wi wi-day-sleet-
storm icon'></i>"
    snow = "<i class='wi wi-day-
snow icon'></i>"
    snowthunderstorm = "<i class='wi wi-day-
thunderstorm icon'></i>"
    snowwind = "<i class='wi wi-day-snow-
wind icon'></i>"
    sprinkle = "<i class='wi wi-day-
sprinkle icon'></i>"
    stormshowers = "<i class='wi wi-day-
storm-showers icon'></i>"
    overcast = "<i class='wi wi-day-
overcast icon'></i>"
    thunderstorm = "<i class='wi wi-day-
thunderstorm icon'></i>"
    windy = "<i class='wi wi-day-
windy icon'></i>"

```

```

// parse time
time = parseInt(time);
// check time is more than 6pm, night tim
e
if (time > 16) {
    // check weather

```

```

    if (weather == "Clear") {
        weathericon = clear;
    }
    if (weather == "Clouds") {
        weathericon = nightcloudy;
    }
    if (weather == "Thunderstorm") {
        weathericon = nightthunderstorm;
    }
    if (weather == "Drizzle") {
        weathericon = nightsprinkle;
    }
    if (weather == "Rain") {
        weathericon = nightrain;
    }
    if (weather == "Snow") {
        weathericon = nightsnow;
    }
    if (weather == "Fog") {
        weathericon = nightfog;
    }
}
// Night morning
else if (time <= 6 )
{
    // check weather data
    if (weather == "Clear") {
        weathericon = clear;
    }
    if (weather == "Clouds") {
        weathericon = nightcloudy;
    }
    if (weather == "Thunderstorm") {
        weathericon = nightthunderstorm;
    }
    if (weather == "Drizzle") {
        weathericon = nightsprinkle;
    }
    if (weather == "Rain") {
        weathericon = nightrain;
    }
    if (weather == "Snow") {
        weathericon = nightsnow;
    }
    if (weather == "Fog") {
        weathericon = nightfog;
    }
}

// else day time
else {
    // check weather
    if (weather == "Clear"){
        weathericon = sunny;
    }
    if (weather == "Clouds"){
        weathericon = cloudy;
    }
    if (weather == "Thunderstorm"){
        weathericon = thunderstorm;
    }
    if (weather == "Drizzle"){
        weathericon = sprinkle;
    }
    if (weather == "Rain"){
        weathericon = rain;
    }
    if (weather == "Snow"){
        weathericon = snow;
    }
}

// return the string
return weathericon;
}

// function to calculate night or day, and se
ason to display different background images
function dateBackground(time, month) {
    // parse the values entered to integer values
    month = parseInt(month);
    time = parseInt(time);
    //winter
    if (month > 11 || month <= 2)
    {
        // if night time
        if (time > 15 || time < 6)
        {
            $(''.ui-
content').css('background-
image', 'url(../backgrounds/winternight)');

```

```

    }
    // day time
    else {
        $('.ui-content').css('background-
image', 'url(../backgrounds/winterday)');
    }
}
//spring
else if (month >= 3 && month <= 5 )
{
    // if night time
    if (time > 15 || time < 6)
    {
        $('.ui-content').css('background-
image', 'url(../backgrounds/springnight)');
    }
    // day time
    else {
        $('.ui-content').css('background-
image', 'url(../backgrounds/springday)');
    }
}
// summer
else if (month >= 6 && month <= 9)
{
    // if night time
    if (time > 15 || time < 6)
    {
        $('.ui-
content').css('background-
image', 'url(../backgrounds/summernight)');
    }
    // day time
    else {
        $('.ui-content').css('background-
image', 'url(../backgrounds/summerday)');
    }
}
}

```

```

    // else autumn
    else {
        // night time
        if (time > 15 || time < 6)
        {
            $('.ui-
content').css('background-
image', 'url(../backgrounds/autumnnight)');
        }
    }
    // daytime
    else {
        $('.ui-content').css('background-
image', 'url(../backgrounds/autumnday)');
    }
}
// end function
return;
}
// end of file

</script>

</html>

```

IV. CONCLUSION

In conclusion, jQuery mobile provided an excellent base for a single page application, however its compatibility with CSS styling proved to become difficult especially when using a framework for a web browser-based application. Using a more robust JavaScript frameworks such as Vue.js, Angular or React would provide a much more refined experience for handling user interfaces [4] [5] [6].

V. REFERENCE LIST

- [1] 'Consuming a RESTful Web Service with jQuery'. <https://spring.io/guides/gs/consuming-rest-jquery/> (accessed Jan. 28, 2021).
- [2] 'Pages - jQuery Mobile Demos'. <https://demos.jquerymobile.com/1.3.1/widgets/pages/> (accessed Jan. 28, 2021).
- [3] 'Weather API - OpenWeatherMap'. <https://openweathermap.org/api> (accessed Jan. 28, 2021).
- [4] 'Vue.js'. <https://vuejs.org/> (accessed Jan. 17, 2021).
- [5] 'Angular'. <https://angular.io/> (accessed Jan. 28, 2021).
- [6] 'React - A JavaScript library for building user interfaces'. <https://reactjs.org/> (accessed Jan. 28, 2021).

Mobile-enabled website utilising REST services

GRADEMARK REPORT

FINAL GRADE

75/100

GENERAL COMMENTS

Instructor

WEBSITE

Range of code demonstrated - Good use of the AJAX call for validation checking. Some good work on the weather API - there are a few areas where a loop would have been more efficient, but this is a minor point. Some good processing of the API data to integrate it into your site. Be careful to validate on cities that don't exist - it continues to show the data from the previous valid city. I also like the use for the different icon. The website itself feels a bit sparse in terms of pages - you could have added some content to show more of the single-page application functionality of JQuery mobile for example. You could have done a bit more on your own JSON file eg exporting it from your database and using `json_encode()` - of course, you wouldn't do this with the users table! but you could have shown how you would make your own information available to a third party. Nice use of dynamic background images
46/60

Overall integration of website - You should disable the Weather button at the top for non-logged in users - it allows me to open the panel but the search doesn't work - initially confusing, but fine when I worked out I had to log in first! It would have been nice to do a bit more with the front-end interface than the standard default colours (I think you said you were colour blind so I haven't been too harsh on this - but you could get others to check your colour choices to get around this. Some pages are not styled (login/logout replies) 7/10

REPORT

Analysis of technical development - A bit brief - you obviously know what you're doing, but having looked at your code, you're not fully getting the complexities across. You'd be better using the full-width page format and including code snippets in the body of your methodology text to illustrate your point. Don't

forget to analyse what you've done and comment on what you might do differently in the future. 9/15

Annotated code listing - The two column report style isn't really suitable for showing code as the columns are very narrow - stick to full-width for reports of this kind. Apart from that the comments are suitable and help the user to understand the code well. 13/15

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16
