GANPAT UNIVERSITY

U. V. Patel College of Engineering

Computer Engineering Department

Sub: Enterprise Mobile Application Development

Enrollment – 15012111007 **PRACTICAL – 1**

AIM: Create a calculator's Hybrid App, using HTML, CSS and JavaScript.

Index.html

```
      <!DOCTYPE HTML>
<html>
      <head>
            <meta charset="UTF-8">
            <title>Pract_1</title>
<meta name="viewport"
      content="width=device-width, initial-scale=1, maximum-scale=1, user-
scalable=no">
<!--
                  <link rel="shortcut icon" href="images/favicon.png">
                  <link rel="apple-touch-icon" href="images/apple-touch-
icon.png">
            -->
            <link rel="stylesheet" href="css/main.css">
            <link href="jqueryMobile/jquery.mobile-1.3.1.css" rel="stylesheet">
            <script>window.$ = window.jQuery = WLJQ;</script>
            <script src="jqueryMobile/jquery.mobile-1.3.1.js"></script>
<script type="text/javascript" src="dojox/mobile/deviceTheme.js"></script>
<script type="text/javascript"
      data-dojo-config="isDebug: false, async: true, parseOnLoad: true,
mblHideAddressBar: false"
      src="dojo/dojo.js"></script>
<meta name="apple-mobile-web-app-capable" content="yes">
</head>

      <body style="display: none;">
            <div data-role="page" id="page" data-theme="e">
<!-- Header section -->
             <div data-role="header" id="header" data-position="fixed">
                  <h3>Calculator</h3>
            </div>
<!-- Content section -->
<div data-role="content">
<div id="calculator">
      <!-- Screen and clear key -->
      <div class="top">
            <span class="clear">C</span>
            <div class="screen"></div>
      </div>
      <div class="keys">
            <!-- operators and other keys -->
            <span>7</span>
            <span>8</span>
            <span>9</span>
            <span class="operator">+</span>
            <span>4</span>
            <span>5</span>
            <span>6</span>
            <span class="operator">-</span>
            <span>1</span>
            <span>2</span>
            <span>3</span>
            <span class="operator">/</span>
            <span>0</span>
            <span>.</span>
            <span class="eval">=</span>
            <span class="operator">x</span>
      </div>
</div>
</div>
            <!-- Footer section -->
            <div data-role="footer" data-position="fixed" id="footer" data-
theme="f" style="position: fixed">
                  <h3>Made by Laxman</h3>
            </div>


                  </div>
            <script src="js/initOptions.js"></script>
            <script src="js/main.js"></script>
            <script src="js/messages.js"></script>
      </body>
</html>
```

main.cs

```
      * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;

      /* Better text styling */
      font: bold 14px Arial, sans-serif;
}

/* Finally adding some IE9 fallbacks for gradients to finish things up */

/* A nice BG gradient */
html {
      height: 100%;
      background: white;
      background: radial-gradient(circle, #fff 20%, #ccc);
      background-size: cover;
}

#calculator {


      margin: 30px auto;
      padding: 10px 10px 9px;


      background: #9dd2ea;
      background: linear-gradient(#9dd2ea, #8bceec);
      border-radius: 3px;
      box-shadow: 0px 4px #009de4, 0px 10px 15px rgba(0, 0, 0, 0.2);
}

/* Top portion */
.keys span.clear {
      float: bottom;
}
```

```css
/* Inset shadow on the screen to create indent */
.top .screen {
	width: 185px;
	height: 70px;

	float: right;
	margin: 0 15px 15px 0;
	padding: 0 10px;

	background: rgba(0, 0, 0, 0.2);
	border-radius: 3px;
	box-shadow: inset 0px 4px rgba(0, 0, 0, 0.2);

	/* Typography */
	font-size: 27px;
	line-height: 40px;
	color: white;
	text-shadow: 1px 1px 2px rgba(0, 0, 0, 0.2);
	text-align: right;
	letter-spacing: 1px;
}

/* Clear floats */
.keys, .top {overflow: hidden;}

/* Applying same to the keys */
.keys span, .top span.clear {
	float: left;
	position: relative;
	top: 0;

	cursor: pointer;

	width: 60px;
	height: 68px;

	background: white;
	border-radius: 3px;


	margin: 3px 3px 3px 3px;

	color: #888;
	line-height: 36px;
	text-align: center;

	/* prevent selection of text inside keys */
	user-select: none;

	/* Smoothing out hover and active states using css3 transitions */
	transition: all 0.2s ease;
}

/* Remove right margins from operator keys */
/* style different type of keys (operators/evaluate/clear) differently */
.keys span.operator {
	background: #FFF0F5;
	margin-right: 0;
}

.keys span.eval {
	background: #f1ff92;
	box-shadow: 0px 4px #9da853;
	color: #888e5f;
}
```

```
.top span.clear {
      background: #ff9fa8;
      box-shadow: 0px 4px #ff7c87;
      color: white;
}

/* Some hover effects */
.keys span:hover {
      background: #9c89f6;
      box-shadow: 0px 4px #6b54d3;
      color: white;
}

.keys span.eval:hover {
      background: #abb850;
      box-shadow: 0px 4px #717a33;
      color: #ffffff;
}

.top span.clear:hover {
      background: #f68991;
      box-shadow: 0px 4px #d3545d;
      color: white;
}

/* Simulating "pressed" effect on active state of the keys by removing the box-
shadow and moving the keys down a bit */
.keys span:active {
      box-shadow: 0px 0px #6b54d3;
      top: 4px;
}

.keys span.eval:active {
      box-shadow: 0px 0px #717a33;
      top: 4px;
}

.top span.clear:active {
      top: 4px;
      box-shadow: 0px 0px #d3545d;
}
```

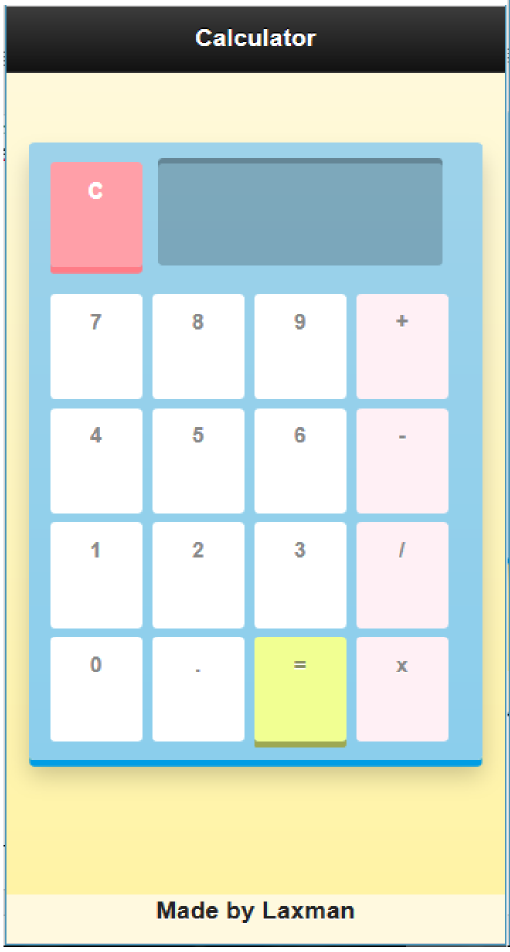
Main.js

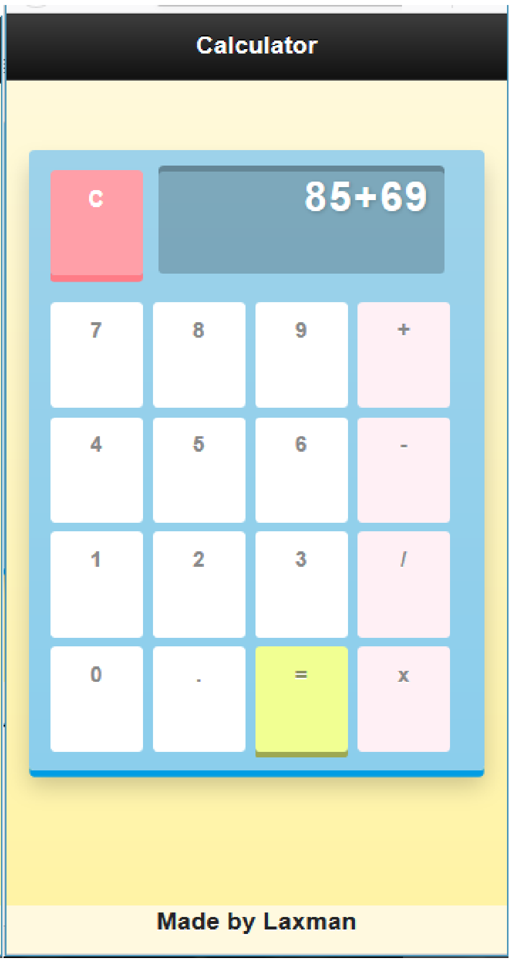
```
      function wlCommonInit(){
      require([ "layers/core-web-layer", "layers/mobile-ui-layer" ], dojoInit);
      /*
       * Use of WL.Client.connect() API before any connectivity to a MobileFirst
Server is required.
       * This API should be called only once, before any other WL.Client methods
that communicate with the MobileFirst Server.
       * Don't forget to specify and implement onSuccess and onFailure callback
functions for WL.Client.connect(), e.g:
       *
       *    WL.Client.connect({
       *                onSuccess: onConnectSuccess,
       *                onFailure: onConnectFailure
       *    });
       *
       */

      // Common initialization code goes here
```

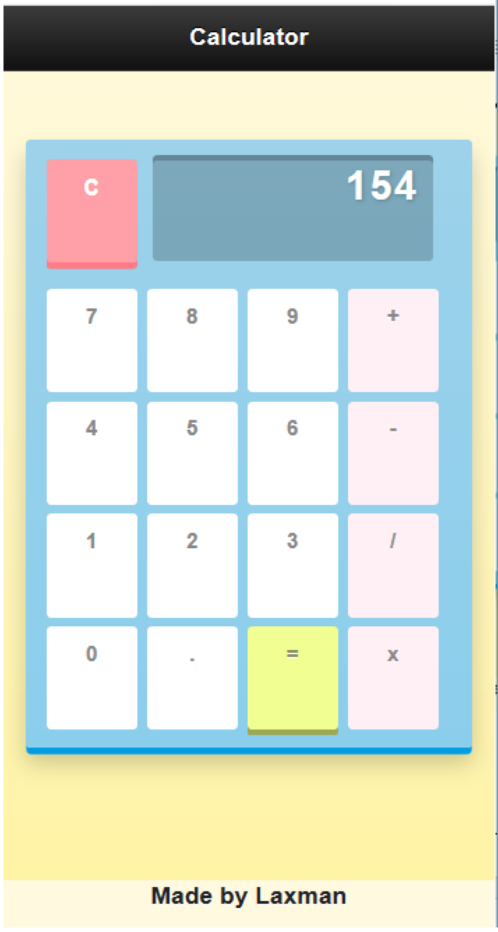
```
        }

        function dojoInit() {
              require([ "dojo/ready", "dojo/parser", "dojo/dom", "dijit/registry",
                          "dojox/mobile/Button", "dojox/mobile", "dojox/mobile/Container"
        ], function(ready) {
                    ready(function() {
                    });
              });
        }

        //Get all the keys from document
        var keys = document.querySelectorAll('#calculator span');
        var operators = ['+', '-', 'x', '÷'];
        var decimalAdded = false;

        // Add onclick event to all the keys and perform operations
        for(var i = 0; i < keys.length; i++) {
              keys[i].onclick = function(e) {
                    // Get the input and button values
                    var input = document.querySelector('.screen');
                    var inputVal = input.innerHTML;
                    var btnVal = this.innerHTML;

                    // Now, just append the key values (btnValue) to the input string and
        finally use javascript's eval function to get the result
                    // If clear key is pressed, erase everything
                    if(btnVal == 'C') {
                          input.innerHTML = '';
                          decimalAdded = false;
                    }

                    // If eval key is pressed, calculate and display the result
                    else if(btnVal == '=') {
                          var equation = inputVal;
                          var lastChar = equation[equation.length - 1];

                          // Replace all instances of x and ÷ with * and / respectively.
        This can be done easily using regex and the 'g' tag which will replace all
        instances of the matched character/substring
                          equation = equation.replace(/x/g, '*').replace(/÷/g, '/');

                          // Final thing left to do is checking the last character of the
        equation. If it's an operator or a decimal, remove it
                          if(operators.indexOf(lastChar) > -1 || lastChar == '.')
                                equation = equation.replace(/.$/, '');

                          if(equation)
                                input.innerHTML = eval(equation);

                          decimalAdded = false;
                    }

                    // Basic functionality of the calculator is complete. But there are
        some problems like
                    // 1. No two operators should be added consecutively.
                    // 2. The equation shouldn't start from an operator except minus
                    // 3. not more than 1 decimal should be there in a number

                    // We'll fix these issues using some simple checks

                    // indexOf works only in IE9+
                    else if(operators.indexOf(btnVal) > -1) {
                          // Operator is clicked
                          // Get the last character from the equation
                          var lastChar = inputVal[inputVal.length - 1];
                          // Only add operator if input is not empty and there is no
        operator at the last
                          if(inputVal != '' && operators.indexOf(lastChar) == -1)
                                input.innerHTML += btnVal;

                          // Allow minus if the string is empty
                          else if(inputVal == '' && btnVal == '-')
                                input.innerHTML += btnVal;

                          // Replace the last operator (if exists) with the newly pressed
        operator
                          if(operators.indexOf(lastChar) > -1 && inputVal.length > 1) {
                                // Here, '.' matches any character while $ denotes the
        end of string, so anything (will be an operator in this case) at the end of
        string will get replaced by new operator
                                input.innerHTML = inputVal.replace(/.$/, btnVal);
                          }

                          decimalAdded =false;
                    }

                    // Now only the decimal problem is left. We can solve it easily using
        a flag 'decimalAdded' which we'll set once the decimal is added and prevent more
        decimals to be added once it's set. It will be reset when an operator, eval or
        clear key is pressed.
                    else if(btnVal == '.') {
                          if(!decimalAdded) {
                                input.innerHTML += btnVal;
                                decimalAdded = true;
                          }
                    }

                    // if any other key is pressed, just append it
                    else {
                          input.innerHTML += btnVal;
                    }

                    // prevent page jumps
                    e.preventDefault();
              }
        }
```

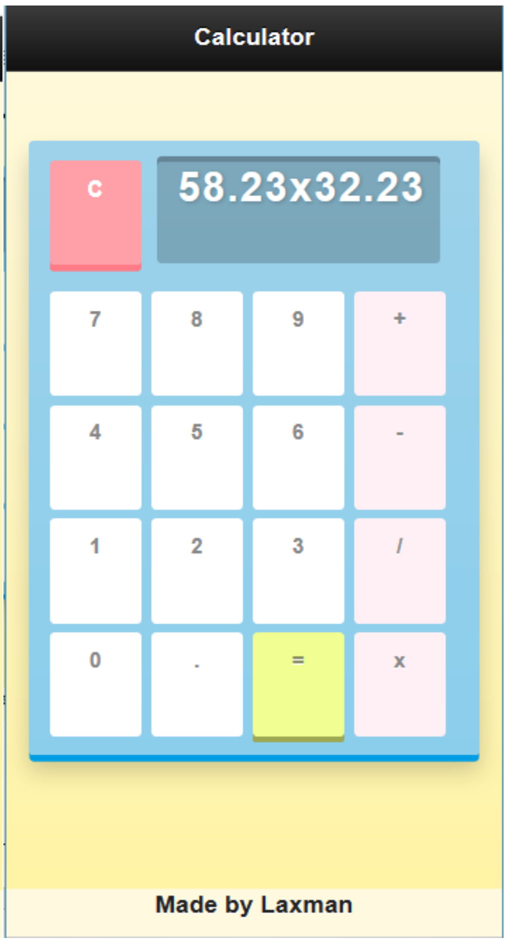
Note : more than one decimal problem solved. And there is a clear button to clear screen after every operation.

Screenshots:






Calculator
c
154
7
8
9
+
4
5
6
-
1
2
3
/
0
.
=
x
Made by Laxman


Calculator
c
58.23x32.23
7
8
9
+
4
5
6
-
1
2
3
/
0
.
=
x
Made by Laxman