# John Hancock Project Coeus Frontend Assembly Specification

# Project Overview

## System Description

Currently, various JH business units within the US Division have their own risk assessment process. Some business units are more advanced than others. The process involves assessing the risks of many functions and reporting on results. This siloed approach leads to a lack of consistency in regards to data input, templates, frequency of assessments and reporting at the US Division Operations level. Furthermore, there is no tool or means of capturing an overall inventory of controls.

This project will build a tool to capture and assess risks and controls data at the function level across various Business Units within US Division Operations

This architecture covers both backend and frontend implementations.

This assembly will implement Web Admin app using HTML5 and AngularJS. It will be responsible for integrating backend REST API. **This assembly is only responsible for implementing the Login page, Common Element and Add Assessment page.**

## Competition Task Overview

A complete list of deliverables can be found in the TopCoder Assembly competition Tutorial at:

<http://apps.topcoder.com/wiki/display/tc/Assembly+Competition+Tutorials>

*Note: Extensive implementation notes are provided at method documentation on UML Class Diagrams. Please follow them for implementation.*

*Note: Please read the whole Application Design Specification first. All the details not mentioned in this specification are provided in that document.*

### Front End Classes

This assembly is responsible for implementing AngularJS services/controllers defined in Frontend Class Diagram. It will also be responsible for integrating backend REST API with frontend pages. **This assembly is only responsible for implementing the Login page, Common Element(See 1.2.2) and Add Assessment page(the pages under "Add a New Assessment" menu).**

## *Common Element*

Prototype: the header of most pages

Controller: masterCtrl

masterCtrl will control the data in the header:

* The user name and role can be retrieved from $rootScope.user.
* If user clicks "Logout" button, MasterCtrl.logout will be called.
* $scope.draftCount/awaitingApprovalCount/approvedCount/rejectedCount can be used for the counts for assessments.

## *Login page*

Prototype: Login page

Controller: loginCtrl

* If user clicks "login" button, loginCtrl.login will be called.
* "Forgot Password" feature is out of scope.

## *Add Assessment page*

Prototype: Add Assessment page

Controller: addAssessmentCtrl

* When user clicks "Save" or "Save as draft" button, $scope.createAsDraft will be called.
* When user clicks "Submit for Review" button, $scope.submit will be called.
* When user clicks "Export" button, $scope.export will be called.
* $scope. getOverallRiskRatingReport will be used to get overall risk rating report.
  + [jqplot](http://www.jqplot.com/) will be used to draw charts. [Here](http://www.jqplot.com/examples/barTest.php) is the example for how to draw bar charts using jqplot. The pseudo-code is as below (take LikelihoodOfOccurrenceReport for example):

|  |
| --- |
| $(document).ready(function() {  var s0 = [OverallRiskRatingReport.LikelihoodOfOccurrenceReport.CompositeBars[0].CategoryBars[0],  OverallRiskRatingReport.LikelihoodOfOccurrenceReport.CompositeBars[0].CategoryBars[1],  OverallRiskRatingReport.LikelihoodOfOccurrenceReport.CompositeBars[0].CategoryBars[2],  ...  ];  var s1 = [OverallRiskRatingReport.LikelihoodOfOccurrenceReport.CompositeBars[1].CategoryBars[0],  OverallRiskRatingReport.LikelihoodOfOccurrenceReport.CompositeBars[1].CategoryBars[1],  OverallRiskRatingReport.LikelihoodOfOccurrenceReport.CompositeBars[1].CategoryBars[2],  ...  ];  var s2 = [...];  ...  var ticks = ['Annual', 'Quarterly', 'Monthly', 'Weekly', 'Daily']; // OverallRiskRatingReport.LikelihoodOfOccurrenceReport.CompositeBars.Distinct(x => x.RiskType).ToList()    plot2 = $.jqplot('chart2', [s0, s1, s2...], {  seriesDefaults: {  renderer:$.jqplot.BarRenderer,  pointLabels: { show: true }  },  axes: {  xaxis: {  renderer: $.jqplot.CategoryAxisRenderer,  ticks: ticks  }  }  });  ...  }); |

* + For LikelihoodOfOccurrenceReport:
    - CompositeBar.RiskType will be LikelihoodOfOccurrence(Annual, Quarterly, Monthly, Weekly, Daily).
    - CategoryBar.Category.Name will be each Category.
    - RiskReport.OverallRiskScore/MaxRiskType will be used for "Overall: All Categories", "Likelihood of Occurance: Daily" in prototype.
    - CategoryBar.Value will be the height for each bar.
    - If "All Categories" is selected in dropdown list, all category bars will be shown. If any category is selected, other category bars should be hide. No interaction with server is needed.
  + RiskExposureReport and KeyControlsMaturityReport are very similar with LikelihoodOfOccurrenceReport
  + For ResidualRiskReport:
    - ResidualRiskReport.ResidualRiskScore/MaxRiskType will be used for "Overall Residual Risk: 000", "Likelihood of Occurance: Daily" in prototype.
    - CategoryBar.Value will be the height for each bar.
* LookupService should be used to get all KPISLAs to populate "KPI's / SLA's" tab. KPISLA.Name is KPI, KPISLA.SLAs are used to populate the dropdown list of corresponding SLA.
* LookupService should be used to get all ProcessRisks to populate the other categories. ProcessRisk.Name is process, ProcessRisk.Risk is risk.

## Technology overview

* Windows Server 2008+ / Azure with .NET Framework 4.5 installed
* SQL Server 2012
* Microsoft IIS 8.5
* C# 5.0
* .NET Framework 4.5
* ASP.NET Web API 2.2
* HTML5
* Unity 3.5 <http://unity.codeplex.com/>
* Log4net 1.2.13 <http://logging.apache.org/log4net/>
* AngularJS 1.2.28 <https://angularjs.org>
* Microsoft OpenXML SDK 2.0 <http://msdn.microsoft.com/en-us/office/ee358824.aspx>
* ClosedXML 0.76.0 <https://closedxml.codeplex.com>
* jqplot 1.0.9: <http://www.jqplot.com/>
* Flee 0.9.26.0: <http://flee.codeplex.com/>

## Existing Documents

* Class Diagrams
* Sequence Diagrams
* Application Design Specification
* Assembly Specification
* REST API Specification
* ERD