Review Score Calculator Requirements Specification

1. Scope

1.1 Overview

Given a scorecard and a review, the component will be able to evaluate the review answers and calculate the overall score. Different question types will have different mechanisms to resolve the answer into scores. Simple caching strategy is provided so that calculators do not need to be created for the same scorecard.

1.2 Logic Requirements

1.2.1 Question Evaluation

Review answers should be able to resolve into scores with a pluggable mechanism. Each answer should resolve into a score from 0 to 100. The mechanism should be configurable per question type. For this release the following question types will be supported. The answers will be stored as String.

- Scale (1 – 4) (resolved to 25, 50, 75, 100)
- Scale (1 – 10) (resolved to 10, 20, …, 90, 100)
- Test Case (# of passing / # of total) (resolved to the passing percentage)
- Yes/No (resolved to 100, 0)

1.2.2 Score Calculator

Review score can be calculated given a scorecard instance and a review instance. The maximum score for a complete scorecard will be 100. If the number of answers does not match the number of questions, an error will be signaled.

1.2.3 Weighted Calculator Caching

The calculator instance should be cached per scorecard ID, name and version. The component should be thread safe.

1.3 Required Algorithms

No specific algorithms are required.

1.4 Example of the Software Usage

A scorecard/review application can use the component to calculate scores for committed reviews.

1.5 Future Component Direction

More question types can be supported in the future.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

None.

2.1.3 Environment Requirements

- Development language: Java1.4
- Compile target: Java1.4

2.1.4 Package Structure

com.topcoder.management.review.scorecalculator

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- Question evaluation mechanisms

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

None.

3.2.2 TopCoder Software Component Dependencies:

- Weighted Calculator
- Configuration Manager
- Simple Cache

**Please review the TopCoder Software component catalog for existing components that can be used in the design.

3.2.3 Third Party Component, Library, or Product Dependencies:

None.

3.2.4 QA Environment:

- Solaris 7
- RedHat Linux 7.1
- Windows 2000
- Windows 2003

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines.

3.4 Required Documentation

3.4.1 Design Documentation

- Use-Case Diagram
- Class Diagram
- Sequence Diagram
- Component Specification

3.4.2 Help / User Documentation

- Design documents must clearly define intended component usage in the ‘Documentation’ tab of Poseidon.