Score Watch - Competency Management

SRS Document

Disclaimer

This Software Requirements Specification document is a guideline. The document details all the high level requirements. The document should be used as a guideline by the students to design the Solution Architecture for the project. The document also describes the broad scope of the project and high level logical object model. But while developing the solution if the developer has a valid point to add more details being within the scope specified then it can be accommodated after consultation.

Score Watch

Introduction

The purpose of this document is to define scope and requirements of a Performance on competencies system - Score Watch for a leading business conglomerate, OxyGen. They have a major challenge in managing Job Competency of its large workforce through their existing spreadsheet driven manual system. The proposed system will automate the Job Competency management process across the enterprise.

This document should be used by the development team to architect the solution the project.

Management Summary

OxyGen, a leading business conglomerate, is experiencing challenge in managing Job Competencies of its large workforce. Currently they have spreadsheet driven manual system, where Competency of thousands of employees are maintained manually. This lacks well defined mechanisms to appraise employees, in order to address the challenge, they needed a web based system to automate job competency management across the enterprise. The proposed system will:

1. Allow easy online maintenance of Job Positions and Competencies required to execute the role.
2. Allow Self, Reporting Manager and Peer to rate the Job Competencies.
3. Create Reports based on Score Cards.
4. Search for competencies with proficiency levels.

Score Watch will be a web-based system. It will be designed & developed to run on IBM WebSphere Application Server and IBM DB2 Universal Database in a 2-tier architecture.

Key Assumptions

1. The planned release will maintain only skills and not their levels (e.g. expert, advance, etc.) to reduce the development time.

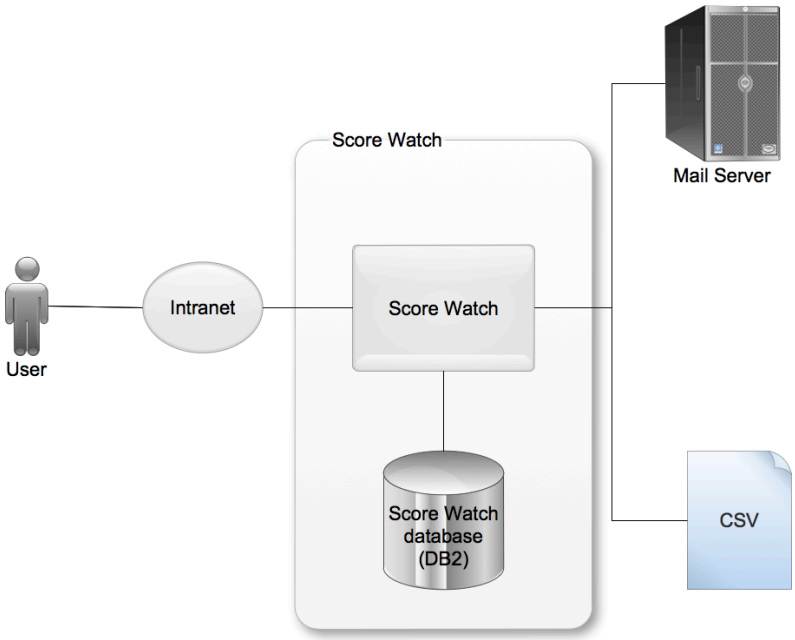
2. The developer of application should read about the Job Competencies and various Appraisal processes followed by the organizations

High Level Architecture

Score Watch's high level architecture is illustrated through the context diagram shown in figure 1 below. It will have following categories of users:

1. Employees and their Reporting Managers

2. HR User



Score Watch Context Diagram

Score Watch	The proposed web-based application, Score Watch will be accessed by the OxyGen employees to manage job competencies, rating, score finalization and reporting.
Score Watch Database	This will hold all the Score Watch data including Job Positions, Position wise Competencies, employee Competencies, ratings by supervisor, peers.
Mail Server	It is responsible for delivering notification e-mails to Score Watch users.
CSV	Master data such as Departments, Employees, Reporting relationship are uploaded directly into the system via CSV as one time setup

Functional Requirements

The high level functional requirements for the Score Watch system are outlined in the Use Case diagram described in this section.

Score Watch will provide a secure user-id/password based secured login mechanism to access its services. The details of this are not outlined here. The development team is expected to create these keeping in mind the general practices followed by the web applications. Login will be a prerequisite to use Score Watch.

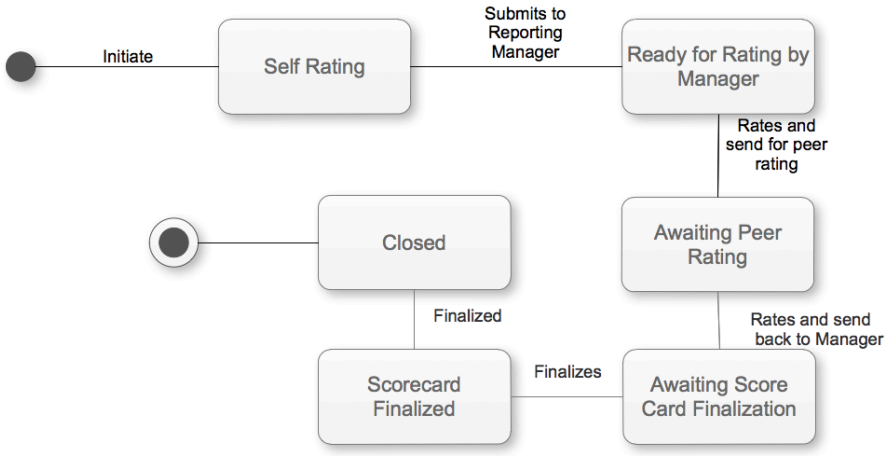
Once employee logs in, s/he can view his/her competencies in details on “My Page”. This will be the default landing page for every employee.

Score Watch will provide a search mechanism to enable managers to search for employee’s possessing specific competency. The search will be carried out by entering one or more competency (separated by comma) in the search text box and clicking on search button. Search option will be available on all pages.

There are no use cases for creating/deleting employee records. This is out of scope from the current system. They are managed directly from the DB backend.

State Transition Workflow

The following figure illustrates the Workflow State Transition rating of job competencies for an employee.



1. The workflow begins with Self Rating activity on the Job competencies for employee’s own job position, the score card moves into Self Rating state.
2. Employee submits the self rated competency score card to the Reporting Manager for rating, the score card is in the state of Ready for Rating by Manager. At this stage, the employee cannot modify anything in the score card.
3. The Reporting Manager does assessment of the employee by giving rating on a scale of 1 to 10 for each of the competency mentioned in the score card.
4. On completion of this activity the reporting manager selects 1 peer from the employee’s team and sends invitation to rate the employee whose score card is being processed. The score card moves to Awaiting Peer Rating.

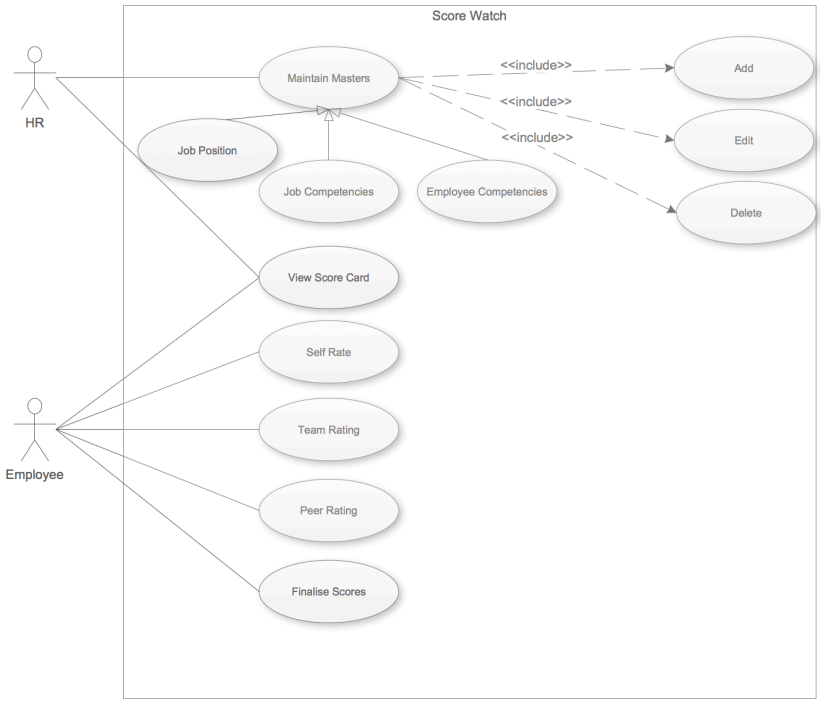
5. The Peer rates the competencies. Please note at this stage, the peer cannot see any other rating apart from the rating being entered by him or her. On submitting the peer rating, the score card moves into Awaiting Score Card Finalization.

6. The supervisor looks for deviations in the 3 ratings, takes a call and enters the final rating. The score card moves into Score Card Finalized State.

7. This Score card Finalized state is the final state, no more action is allowed at this stage.

Use Case Diagrams

The following figure illustrates the Use Case diagram for the system.



Use Case Diagram

Use Cases

Maintain Masters

Use Case Element	Description
Number	UC.01

Use Case Element	Description
Application	Master maintenance in terms of basic operations viz. **add**, **edit**, **delete** and **view**. All master maintenance i.e. Job Positions, Common competencies and Job Competencies, Employee Master are child use cases of this Use Case. Job Positions master will have a Job Id, Job Position, Level (lowest is 1, 2, 3, 4, 5, 6 and 7 being the highest) Common Competencies master will have Competency id, Name of Competency, Applicable level onwards(Starting level e.g. 3 or 4 ...) Job Competency master will have Competency id, Name of Competency, Job id Employee master is pre-loaded, only competencies are required to be added to each of the employees. This will be executed by adding the Job id to the employee record.
Use Case Name	Maintain Masters
Primary Actor	HR User
Secondary Actor	None
Pre-condition	None
Trigger	HR User clicks on the **Maintain Masters** menu item on the admin interface page
Basic Flow	• System presents a list of masters that can be maintained. User selects the desired master. • System displays a list *view* and links for *add*, *edit* and *delete*. 1. In case add, a new master record data entry form is presented. The master record is saved on clicking the save button, provided form clears all the data validations (if any). The list view is updated accordingly. 2. In case of edit, from the list view user is prompted to select the desired record to edit, Selected record is opened for editing. The edited master record is updated on clicking the update button, provided form clears all the data validations (if any). 3. In case of delete, from the list view user selects the check box(s) against each record. Selected records are deleted up on clicking the delete button. However, user is presented a confirmation dialog before deleting the records.
Alternate Flow	• In event of any error, it is clearly displayed and user is asked to renter data or perform operation again.
Output	System displays the details of the successful operation.

Self Rate Competencies

Use Case Element	Description
Number	UC.02

Use Case Element	Description
Application	Annual process of employee performing self rating on the competencies applicable for his/her job position. In an enterprise, review process beginning and closure are marked events supported by timely notifications, for this project, there are no start or end dates in which this activity has to be performed.
Use Case Name	Self Rating
Primary Actor	Employee
Secondary Actor	None
Pre-condition	None
Trigger	User clicks on the **Self Rating** menu item on the employee landing page
Basic Flow	• System displays the list of common and job specific competencies applicable for the employee as per the job position. • User enters the rating on a scale of 1 to 10 for each of the competencies. • User clicks on submit button to send it to Reporting Manager else clicks on Cancel to abandon the entries. • On Submit, an employee Score Card record follows the workflow state as mentioned in the workflow diagram given in earlier sections of this document.
Alternate Flow	• In event of any error, user is alerted to make correction.
Output	Error message, in event of error. Notification to Reporting Manager with Employee's Score Card Link.

Rate Team Member

Use Case Element	Description
Number	UC.03
Application	Reporting Manager rates the team member
Use Case Name	Rate Team Member
Primary Actor	Reporting Manager
Secondary Actor	None
Pre-condition	None
Trigger	User clicks on the **Rate Team Member** menu item on the Employee landing page

Use Case Element	Description
Basic Flow	• System displays the list of employees whose score card is in Ready for Manager's Rating' state. • Manager selects one employee, the score card opens up and Job competencies, Employee's self rating (not editable) and a column for reporting manager to enter the rating. • User enters the rating for each of the competencies listed. • User now selects a Peer from the list of employees reporting into the logged in manager. Upon clicking invite button the Score card's state changes as per the Workflow diagram. The peer is notified by the system via e-mail.
Alternate Flow	None
Output	E-mail notification to peer. Error messages, if any.

Rate Peer

Use Case Element	Description
Number	UC.04
Application	Peer Rating
Use Case Name	Rate Peer
Primary Actor	Employee(Peer)
Secondary Actor	None
Pre-condition	None
Trigger	User clicks on the **Rate Peer** menu item on the employee landing page
Basic Flow	• System displays the list of employees who have to be rated by a Peer. • User selects the required employee to update. The list of competencies is displayed and a column to enter the rating. • User may choose to update this list. Upon submit, the Score card moves into a new state as per workflow transition diagram mentioned in earlier sections.
Alternate Flow	None
Output	E-Mail to employee and his/her reporting manager about the update.

Finalize Score Card

Use Case Element	Description
Number	UC.05
Application	Score card of an employee is finalized after all three ratings are available.
Use Case Name	Finalize Scores
Primary Actor	Reporting Manager (Employee)
Secondary Actor	None
Pre-condition	None

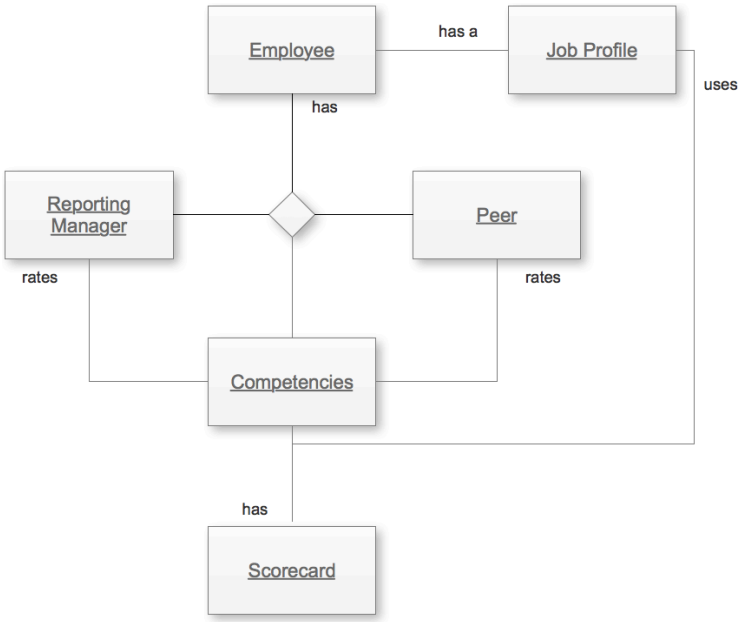
Use Case Element	Description
Trigger	Employee clicks on the **Finalize Score Card** link on Employee landing page.
Basic Flow	• System displays the list of employees whose peer rating has been received. • User can select an employee to finalize the score card. • Before displaying the score card, system computes the median score for each competency. (All three scores are sorted, and the middle score is taken as the median score) • If the min or max score has a difference >3 then that competency will be displayed in a column for manager's input of final score. • Once all the final scores are computed, the list with final scores (Median score) are displayed. The records where min and max had a difference greater than 3, is highlighted for manager's intervention. • System allows the manager to change only the highlighted median scores. • Manager submits the modified version of scores as final scores. The score card moves into the closed state of Finalized Scores as per the workflow diagram. No changes are allowed on this score card post this submission.
Alternate Flow	None
Output	E-Mail to employee about the finalized scores

View Scores

Use Case Element	Description
Number	UC.06
Application	Viewing Scores Status
Use Case Name	View Scores
Primary Actor	Reporting Manager
Secondary Actor	HR User
Pre-condition	None
Trigger	User clicks on the **View Scores** link on Landing page. This link is visible to the HR and the employee who have people reporting to them.
Basic Flow	• System displays the list of all the direct reports in case of Reporting Manager and List of departments in case of HR. • HR user selects the department to view scores • Reporting manager user views the scores of direct reports. • The list of employees with all the scores summary as part of process is displayed along with the status of each of the score card.
Alternate Flow	None
Output	None

Logical Object Model

A high level logical object model of the system is shown below. During technical design it will be transformed into a physical model covering all system entities. Such a diagram will include their relationship and its cardinality.



Logical Object Model

1. Employee performs a role as mentioned in the Job Position. Employee's role changes with time in the organization due to his/her movement between various department or role elevation i.e. promotion.
2. Employee has competencies.
3. Employee reports to his/her manager (who is also an employee).
4. In order to perform a job role, specific competencies are required.
5. To know how well the employee is performing as per the competencies, the process of rating is followed, where the employee does self rating and then forwards it to the reporting manager to rate and have it rated by a peer as well.
6. The final scores are decided by the Reporting Manager based on inputs provided by the system.

Database Design Guidelines

This involves the transformation of the use cases, state diagrams, and logical object model into detailed and optimized physical database table designs.

Typically persistent classes will map to table(s) with their attributes as columns of the table. In some cases a high level object may map in to a master-child table. Invoice is one such example where it maps in to "invoice_header" and "invoice_line_item" table.

Associations between two persistent objects are realized as foreign keys to the associated objects. A foreign key is a column in one table that contains the primary key value of the associated object.

Similarly, a standard technique in relational modeling is to use an intersection entity to represent many-to-many associations. Following is a broad checklist for physical database database design:

1. Database must be properly normalized except those instances where de-normalization help improves performance. This option must be used with special care.
2. All persistent classes that use the database for persistency must map to database structures.
3. Many-to-many relationships must have an intersecting table.
4. Primary keys should be defined for each table, unless there is a performance reason not to define a primary key.
5. Indexes should be defined to optimize access.
6. Data and referential integrity constraints should be defined.

Testing Approach

Quality of the software can be achieved with basic hygiene and consistency followed during design and development of User Interface(UI), Navigation, Validations as per the business process requirement.

To ensure the project delivers acceptable quality to the customer, its important to create a checklist of the conventions to be followed across. Common checks as below are for your reference during design and development:

Common Checks	Validation Type
Page Title is valid for the feature being provided on the page	UI
Order of the Data Entry Fields is logical as per the functionality being provided by the feature	UI
Order of the Display only Fields makes viewing and understanding easy for the user	UI
Spellings and Correctness of Label for the Data Entry and Display fields	UI
The labels are not wrapping onto another row thereby adding a blank row on the page	UI
The fields with drop down are displayed in single row instead of drop down coming on the next row	UI
Data Entry field basic validations are working i.e Text field /Numbers / Dates allow data for their type only	Functional
The dates are following a standard format dd/mmm/yyyy on all forms	UI
The color scheme of all forms i.e headers labels , alerts, entry fields are uniform throughout the application	UI
The action buttons for a New Data Entry Form are uniform for all forms that is allowing data entry	UI
The action buttons are performing the desired action e.g. "submit" is creating a new record if there are no errors and recording all the input fields, whereas 'cancel' is not creating a new record in the database	Functional
The links provided on the forms are opening correctly.	Functional
The data feed mechanism for Read and Write files is generating a log with count of entries.	Navigation

For testing purpose, dummy employee records can be created on the database directly from the DB backend. Similarly, employee role change etc can be simulated by updating the employee record directly from the DB backend.

Suggested Technical Reading

The project is aimed at making the student understand concepts of Design and Development using IBM Rational tools, Web Sphere Application Server and DB2 Database. The following reading reference is easy to understand and should be read to get a clear understanding of capabilities of the tools and how you would leverage them to execute a project.

Technical Reference	URL to access
RAD - Tackling challenges of software development with Rational Application Developer for WebSphere Software	http://www.ibm.com/developerworks/rational/library/08/0926_ackerman-mahate/index.html
IBM Education Assistant - Rational Application Developer 7.5	http://publib.boulder.ibm.com/infocenter/ieduasst/rtnv1r0/index.jsp?topic=/com.ibm.iea.rad_v7/rad/rad75.html
RSA-Overview of Rational Software Architect for WebSphere Software Version 7.5	http://www.ibm.com/developerworks/rational/library/08/0926_arnold/index.html
Using the new features of UML Modeler in IBM Rational Software Architect Version 7.5	http://www.ibm.com/developerworks/rational/library/08/0926_diu/index.html
Rational Technical Library	http://www.ibm.com/developerworks/rational/library/