Time Tracker Project 3.2 Requirements Specification

1. Scope

1.1 Overview

The Time Tracker Project custom component is part of the Time Tracker application. It provides an abstraction of projects. This component handles the persistence and other business logic required by the application.

The design for this specification exists, but requires modification. The text in RED is new requirements. You are to make the additions to the existing design.

1.2 Logic Requirements

1.2.1 Company Account

A company owns everything in the context of the application, such as users, clients, and projects. The following sections will describe how the company entity relates to the existing Time Tracker Project entities in details.

1.2.2 Project

1.2.2.1 Overview

Projects will now be associated with a company ID. This component extends TimeTrackerBean and will model the following project information:

- Company ID – the company ID associated with the project
- Client ID – the client that owns the project
- Project ID – the unique project ID number
- Name – the name of the project
- Description – a brief description of the project
- Start Date – the estimated start date of the project
- End Date – the estimated end date of the project
- Terms – The payment terms for Invoices
- Sales Tax – The Sales tax for Invoices

1.2.2.2 Contact Information

The Project also contains contact information, the Contact component is to be used to retrieve and persist the contact information. The contact information, which the Project will use, is as follows:

- Contact Name
- Phone
- Email

1.2.2.3 Address Information

The Project also contains address information, the Contact component is to be used to retrieve and persist the project information. The address information, which the Project will use, is as follows:

- Address Line 1

 ©TopCoder, Inc. 2007

- Address Line 2
- City
- State
- Zip code
- Country

1.2.2.4 Search Filters

This component will provide search functionalities based on a logical (AND, OR, NOT) combination of search filters. The following is a summary of the required filters:

- Return all projects with a given company ID
- Return all projects with a given client ID
- Return all projects with name that contains a given string
- Return all projects with description that contains a given string
- Return all projects with start date within a given inclusive date range (may be open-ended)
- Return all projects with end date within a given inclusive date range (may be open-ended)
- Return all projects with a given project manager user ID
- Return all projects with a given project worker user ID
- Return all projects with a given expense entry ID
- Return all projects with a given fixedbilling entry ID
- Return all projects with a given time entry ID
- Return all projects created within a given inclusive date range (may be open-ended)
- Return all projects modified within a given inclusive date range (may be open-ended)
- Return all projects created by a given username
- Return all projects modified by a given username

1.2.2.5 Database Schema

The project information will be stored in the following tables (refer to TimeTrackerClient_Project_ERD.jpg):

- project
- project_time
- project_expense
- project_fixedbilling
- project_manager

1.2.2.6 Required Operations

- Create a new project
- Retrieve an existing project by ID
- Update an existing project information
- Delete an existing project
- Enumerate all existing projects
- Batch versions of the CRUD operations
- Search projects by filters
- Add time entry IDs to an existing project (company ID must match)
- Remove time entry IDs from an existing project
- Get all time entry IDs associated with an existing client
- Add expense entry IDs to an existing project (company ID must match)
- Remove expense entry IDs from an existing project
- Get all expense entry IDs associated with an existing client

- Add fixedbilling entry IDs to an existing project (company ID must match)
- Remove fixedbilling entry IDs from an existing project
- Get all fixedbilling entry IDs associated with an existing client
- Add project managers to an existing project (company ID must match)
- Remove project managers from an existing project
- Get all project managers associated with an existing project Add project managers to an existing project (company ID must match)
- Remove project workers from an existing project
- Get all project workers associated with an existing project

1.2.2.7 Audit Requirements

The component is required to allow for a boolean on any method, which allows for the modification of the data, to determine is the action is to be audited. The Time Tracker Audit component will encapsulate the actual auditing of the data. Note that the audit should be in the transaction and rolled back if the transaction fails.

The application area for this will be TT_PROJECT.

1.2.3 Project Worker

1.2.3.1 Overview

A project worker is a user assigned to work on a particular project. A project can have any number of workers, and a user can be working on multiple projects at the same time. This component TimeTrackerBean and models the following project worker information:

- Project ID – the project ID associated with the worker
- User ID – the user ID of the worker
- Start Date – the estimated date of starting work on the project
- End Date – the estimated date of ending work on the project
- Pay Rate – the hourly pay rate of the worker for the project

1.2.3.2 Search Filters

This component will provide search functionalities based on a logical (AND, OR, NOT) combination of search filters. The following is a summary of the required filters:

- Return all workers with a given project ID
- Return all workers with start date within a given inclusive date range (may be open-ended)
- Return all workers with end date within a given inclusive date range (may be open-ended)
- Return all workers with pay rate within a given inclusive amount range (may be open-ended)
- Return all workers created within a given inclusive date range (may be open-ended)
- Return all workers modified within a given inclusive date range (may be open-ended)
- Return all workers created by a given username
- Return all workers modified by a given username

1.2.3.3 Database Schema

The project information will be stored in the following tables (refer to TimeTrackerClient_Project_ERD.jpg):

- project_worker

1.2.3.4 Required Operations

- Create a new project worker
- Retrieve an existing worker by user and project ID
- Update an existing worker information
- Delete an existing worker
- Enumerate all existing workers
- Batch versions of the CRUD operations
- Search workers by filters

1.2.3.5 Audit Requirements

The component is required to allow for a boolean on any method, which allows for the modification of the data, to determine is the action is to be audited. The Time Tracker Audit component will encapsulate the actual auditing of the data. Note that the audit should be in the transaction and rolled back if the transaction fails.

The application area for this will be TT_PROJECT_WORKER.

1.2.4 Project Manager

1.2.4.1 Overview

A project manager is a user assigned to manage on a particular project. A project can have any number of managers, and a user can be managing multiple projects at the same time. This component extends TimeTrackerBean and models the following project worker information:

- Project ID – the project ID associated with the manager
- User ID – the user ID of the manager
- Start Date – the estimated date of starting work on the project
- End Date – the estimated date of ending work on the project

1.2.4.2 Search Filters

This component will provide search functionalities based on a logical (AND, OR, NOT) combination of search filters. The following is a summary of the required filters:

- Return all managers with a given project ID
- Return all managers with start date within a given inclusive date range (may be open-ended)
- Return all managers with end date within a given inclusive date range (may be open-ended)
- Return all managers created within a given inclusive date range (may be open-ended)
- Return all managers modified within a given inclusive date range (may be open-ended)
- Return all managers created by a given username
- Return all managers modified by a given username

1.2.4.3 Database Schema

The project information will be stored in the following tables (refer to TimeTrackerClient_Project_ERD.jpg):

- project_manager

1.2.4.4 Required Operations

- Create a new project manager
- Retrieve an existing manager by user and project ID
- Update an existing manager information
- Delete an existing manager
- Enumerate all existing managers
- Batch versions of the CRUD operations
- Search managersby filters

1.2.4.5 Audit Requirements

The component is required to allow for a boolean on any method, which allows for the modification of the data, to determine is the action is to be audited. The Time Tracker Audit component will encapsulate the actual auditing of the data. Note that the audit should be in the transaction and rolled back if the transaction fails.

The application area for this will be TT_PROJECT_MANAGER.

1.2.5 Pluggable Persistence

All entities defined in previous sections will be backed by a database. The design will follow the DAO pattern to store, retrieve, and search data from the database. All ID numbers will be generated automatically using the ID Generator component when a new entity is created. All creation and modification dates will be taken as the current datetime.

For this version, the Informix database system will be used as persistence storage for this component and the Time Tracker application. Other database systems should be pluggable into the framework.

1.2.6 JavaBeans Conventions

For all the entities described in previous sections, the JavaBeans conventions will be followed (http://java.sun.com/products/javabeans/docs/spec.html):

- The class is serializable
- The class has a no-argument constructor
- The class properties are accessed through `get`, `set`, `is` methods. i.e. All properties will have get<PropertyName>() and set<PropertyName>(). Boolean properties will have the additional is<PropertyName>().

Note: event handling methods are not required.

1.2.7 Transaction Management

As a result of the fine grain components used in this design there needs to be a transaction management strategy, which allows a single transaction to exist that encompasses all components called for a single use case. Since this component will be deployed into an Enterprise Java Bean container, JBoss 4.0.x, a Stateless Session Bean will be used to manage the transaction. The container will start a transaction when a method is invoked if one is not already running. The method will then join the new or existing transaction. Transaction Management will be Container Managed.

1.2.7.1 User API for component

The user API for this component will exist in a Delegate object. This delegate will provide the contract for the component and interface with the EJB. The Delegate is not an EJB rather it will be a POJO. It will look up the EJB and call the related method, retrieve the results and return the results to the consumer. There will be no additional logic in the delegate.

1.2.7.2 Stateless Session Bean

The methods on the Stateless Session bean will have a transaction level of REQUIRED in the deployment descriptor. This will allow for either a new transaction to be created or for the method to join the existing transaction. For this release we will use a Local Bean and not a Remote Bean. There are a few obstacles, which will need to be addressed:

- No File IO from with in the EJB so ConfigurationManager cannot use a file. Values can however be stored in the Deployment Descriptor.
- All parameters passed to/from the Session bean must be Serializable, however the Filter Object in Search Builder 1.3.1 is not Serializable. This is being addressed and will be fixed.
- The Session Bean should not have any class level variables to store things like the DAO. If it does have a class level variable it must be transient, therefore after activation it will have a value of null. Any of the approaches outlined below are acceptable:
    - Have a class level dao attribute and only access it via a getDAO() method which checks for null and sets the dao attribute if it is null.
    - Like the first method have a class level attribute but on creation or activation load the DAO to the class dao attribute. You must then ensure that under all scenarios that the attribute will be not null.
    - Use a singleton to act as a DAO cache
    - There may be others, and you are not limited to one of these.
- No threads can be created with in the EJB.
- Review the Sun J2EE specification for any other limitations.

All Business logic for the componet will reside in the Stateless Session Bean. There will be no logic in the delegate or in the DAO. There is one exception to this, in that the Audit functionality will exist in the DAO.

1.2.7.3 DAO

The DAO's must retrieve the connection that it uses from the configured TXDatasource in JBoss. The configuration of the DataSource should be externalized so that is can be configured at deployment time.

All audit functionality will exist in the DAO.

1.3 Required Algorithms

None.

1.4 Example of the Software Usage

The Time Tracker application will use this component to perform operations related to client and project management.

1.5 Future Component Direction

Other database systems maybe plugged in for some client environments.

2. Interface Requirements

2.1.1 Graphical User Interface Requirement

None.

2.1.2 External Interfaces

None.

2.1.3 Environment Requirements

- Development language: Java 1.4
- Compile target: Java 1.4, Java 1.5

2.1.4 Package Structure

com.topcoder.timetracker.project

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

None.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

- JavaBeans (http://java.sun.com/products/javabeans/docs/spec.html)

3.2.2 TopCoder Software Component Dependencies:

- Configuration Manager
- DB Connection Factory
- ID Generator
- Search Builder
- Time Tracker Common
- Time Tracker Audit
- Time Tracker Contact

**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:

Informix Database.

3.2.4 QA Environment:

- JBoss 4.0

- Windows 2000
- Windows Server 2003
- Informix

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines. Modifications to these guidelines for this component should be detailed below.

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.