Time Tracker Project 2.0 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 and the clients that the projects are assigned to. This component handles the persistence and other business logic required by the application.

Version 2.0 will leverage the version 1.1 design by modifying existing and adding new classes and methods to meet the new requirements.

Note: existing table names have changed, but their meanings have not.

1.2 Logic Requirements

1.2.1 Company Account

The new version of the Time Tracker application introduces a new entity called company.
A company owns everything in the context of the application, such as users, clients, and projects.
The new design will retrofit the existing classes and methods to accommodate this new entity.
The following sections will describe how the company entity relates to the existing Time Tracker Project entities in details.

1.2.2 Client

1.2.2.1 Overview

Clients will now be associated with a company ID. This component will model the following client information:

- Company ID – the company ID associated with the client
- Client ID – the unique client ID number
- Name – the name of the client (must be unique within a company)
- Creation Date – the date the client was created
- Creation User – the username that created the client
- Modification Date – the date the client was modified
- Modification User – the username that modified the client

1.2.2.2 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 clients with a given company ID
- Return all clients with name that contains a given string
- Return all clients created within a given inclusive date range (may be open-ended)
- Return all clients modified within a given inclusive date range (may be open-ended)
- Return all clients created by a given username
- Return all clients modified by a given username

1.2.2.3 Database Schema

The client information will be stored in the following tables (refer to Time_Tracker_Project.sql):

- client
- client_project

1.2.2.4 Required Operations

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

1.2.3 Project

1.2.3.1 Overview

Projects will now be associated with a company ID. This component 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
- Creation Date – the date the project was created
- Creation User – the username that created the project
- Modification Date – the date the project was modified
- Modification User – the username that modified the project

1.2.3.2 Project Manager

A project manager is the user in charge of a particular project. A project must have a project manager. A user can be the project manager of multiple projects at the same time.

1.2.3.3 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 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

1.2.3.4 Database Schema

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

- project
- project_time
- project_expense
- project_manager

1.2.3.5 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
- Get/Set project manager for an existing project (company ID must match)
- Add project workers to an existing project (company ID must match)
- Remove project workers from an existing project
- Get all project workers associated with an existing project

1.2.4 Project Worker

1.2.4.1 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 will model 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
- Creation Date – the date the worker was created
- Creation User – the username that created the worker
- Modification Date – the date the worker was modified
- Modification User – the username that modified the worker

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 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

1.2.4.3 Database Schema

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

- project_worker

1.2.4.4 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

1.2.5 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.

1.3 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.cronos.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

**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:

- Solaris 7
- RedHat Linux 7.1
- 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.