Time Tracker User 2.0 Requirements Specification

1. Scope

1.1 Overview

The Time Tracker User custom component is part of the Time Tracker application. It provides an abstraction of user accounts and company accounts in the system. This component handles the persistence and other business logic required by the application.

The new version introduces the concept of “companies”, where each user will be associated with a company. User account information will now be stored directly in the database.

Note: the name of the package has changed. Version 2.0 is a complete redesign of version 1.0.

1.2 Logic Requirements

1.2.1 Company Account

1.2.1.1 Overview

A company is an entity that owns everything in the context of the Time Tracker application, such as users, clients, and projects. Each company will set up initial person to contact upon registration. This component will model the following company account information:

- Company ID – the unique company ID number
- Name – the name of the company
- Contact’s First Name – the first name of the company’s contact
- Contact’s Last Name – the last name of the company’s contact
- Contact’s Phone – the phone number of the company’s contact
- Contact’s Email – the email address of the company’s contact
- Address Line 1 – line 1 of the company’s address
- Address Line 2 – line 2 of the company’s address (optional)
- Company’s Phone – the company’s phone number
- Company’s City – the city where the company is located
- Company’s State – the State where the company is located
- Company’s Zip Code – the zip code of the company’s address
- Company’s Passcode – the secret passcode for employees to identify their companies during user registration (encrypted in persistence, must be unique)
- Creation Date – the date the company account was created
- Creation User – the username that created the company account
- Modification Date – the date the company account was modified
- Modification User – the username that modified the company account

1.2.1.2 Search Filters

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

- Return all companies with name that contains a given string
- Return all companies with contact’s first name that contains a given string
- Return all companies with contact’s last name that contains a given string
- Return all companies with a given contact’s phone number
- Return all companies with a given contact’s email

- Return all companies with address (line 1 or line 2) that contains a given string
- Return all companies with a given company's phone number
- Return all companies with a given city
- Return all companies with a given State
- Return all companies with a given zip code
- Return all companies created within a given inclusive date range (may be open-ended)
- Return all companies modified within a given inclusive date range (may be open-ended)
- Return all companies created by a given username
- Return all companies modified by a given username

1.2.1.3 Database Schema

The company account information will be stored in the following tables (refer to Time_Tracker_User.sql):

- address
- contact
- company_address
- company_contact

1.2.1.4 Required Operations

- Create a new company account
- Retrieve an existing company account by ID
- Update an existing company account information
- Delete an existing company account
- Enumerate existing company accounts
- Batch versions of the CRUD operations
- Search company accounts by filters

1.2.2 User Account

1.2.2.1 Overview

Users of Time Tracker will be able to self-register and obtain a user account. All users will be associated with an existing company. This component will model the following user account information:

- Company ID – the company ID associated with the user
- User ID – the unique user ID number
- Username – the login username
- Password – the login password (encrypted in persistence)
- First Name – the user's first name
- Last Name – the user's last name
- Phone – the user's phone number
- Email – the user's email address (must be unique)
- Address Line 1 – line 1 of the user's address
- Address Line 2 – line 2 of the user's address (optional)
- City – the city where the user lives in
- State – the State where the user lives in
- Zip Code – the zip code of the user's address
- Status – the user account status
- Creation Date – the date the user account was created

- Creation User – the username that created the user account
- Modification Date – the date the user account was modified
- Modification User – the username that modified the user account

1.2.2.2 Account Status

A user account has one of several possible pre-defined statuses:

- Active – the user account is active
- Inactive – the user account is inactive or deleted
- Locked – the user account is locked

1.2.2.3 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 users with a given username
- Return all users with a given password
- Return all users with first name that contains a given string
- Return all users with last name that contains a given string
- Return all users with a given phone number
- Return all users with a given email
- Return all users with address (line 1 or line 2) that contains a given string
- Return all users with a given city
- Return all users with a given State
- Return all users with a given zip code
- Return all users created within a given inclusive date range (may be open-ended)
- Return all users modified within a given inclusive date range (may be open-ended)
- Return all users created by a given username
- Return all users modified by a given username

1.2.2.4 User Authentication

The Authentication Factory component should be used for pluggable authentication strategies. For this version, the encrypted passwords are stored in the database. This component will implement an authenticator that automatically encrypts the given plaintext password and compare against the version stored in the database.

1.2.2.5 User Authorization

The Authorization component should be used to handle user authorization. The design will interact with the Authorization classes/interfaces to support adding/removing user roles. Users will act as “principals” in the context of Authorization. There will be pre-defined application roles.

1.2.2.6 Database Schema

The user account information will be stored in the following tables (refer to Time_Tracker_User.sql):

- user_account
- account_status
- user_address
- user_contact

1.2.2.7 Required Operations

- Create a new user account
- Retrieve an existing user account by ID
- Update an existing user account information
- Delete an existing user account
- Enumerate existing user accounts
- Batch versions of the CRUD operations
- Search user accounts by filters
- Authenticate user with a given password
- Add role to an existing user
- Remove role from an existing user
- Remove all roles (if any) from an existing user
- Enumerate all roles assigned to an existing user

1.2.3 Reject Reason

1.2.3.1 Overview

Time and expense entries of a contractor may be rejected by a project manager. The manager will select one or more reasons from a drop down list of enumerations. Each company will maintain its own set reject reasons stored in the database lookup table. The component will model the following information for reject reason:

- Company ID – the company ID associated with the reject reason
- Reason ID – the unique reject reason ID number
- Description – free-form text description of the reject reason
- Creation Date – the date the reject reason was created
- Creation User – the username that created the reject reason
- Modification Date – the date the reject reason was modified
- Modification User – the username that modified the reject reason

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

1.2.3.3 Database Schema

The reject reason information will be stored in the following tables (refer to Time_Tracker_User.sql):

- reject_reason
- comp_rej_reason

1.2.3.4 Required Operations

- Create a new reject reason
- Retrieve and existing reject reason by ID
- Update an existing reject reason information
- Delete an existing reject reason
- Enumerate all existing reject reasons
- Search reject reasons by filters

1.2.4 Reject Email

1.2.4.1 Overview

When a time or expense entry is rejected by the project manager, a notification email will be sent to the contractor. The email will be generated from a template tailored for the company. The email template will be stored in the database. The component will model the following information for reject email:

- Company ID – the company ID associated with the reject email
- Email ID – the unique reject email ID number
- Body – the email template body
- Creation Date – the date the reject email was created
- Creation User – the username that created the reject email
- Modification Date – the date the reject email was modified
- Modification User – the username that modified the reject email

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

1.2.4.3 Database Schema

The reject email information will be stored in the following tables (refer to Time_Tracker_User.sql):

- reject_email
- comp_reject_email

1.2.4.4 Required Operations

- Create a new reject email
- Retrieve an existing reject email by ID
- Update an existing reject email information
- Delete an existing reject email
- Enumerate all existing reject emails
- Search reject emails 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.

1.3 Required Algorithms

None.

1.4 Example of the Software Usage

The Time Tracker application will use this component to perform operations related to company and user authentication and authorization.

1.5 Future Component Direction

Other database systems maybe plugged in for some client environments. Multiple user stores may be used for the same client environment.

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.user
com.cronos.timetracker.company
com.cronos.timetracker.common

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?

None.

3.2.2 TopCoder Software Component Dependencies:

- Configuration Manager
- DB Connection Factory
- ID Generator
- Encryption
- Authentication Factory
- Authorization

**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.