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.

1.2 Logic Requirements

1.2.1 Time Tracker Clients

1.2.1.1 Overview

A client is the entity representing the paying customers. Each client may own several projects at the same time. For the initial version, an internal ID will be generated for each client and the client name will be maintained.

The database schema to store clients is as follows:

```
CREATE TABLE Clients (
    ClientsID          integer NOT NULL,
    Name               varchar(64) NOT NULL,
    CreationDate       datetime NOT NULL,
    Creationuser       varchar(64) NOT NULL,
    ModificationDate   datetime NOT NULL,
    ModificationUser   varchar(64) NOT NULL,
    PRIMARY KEY (ClientsID)
);
```

1.2.1.2 API Requirements

The following operations for clients have been identified:

- Create a new client
- Update existing client attributes
- Delete an existing client
- Add a project to an existing client
- Remove a project from an existing client
- Enumerate the projects owned by the client

1.2.2 Time Tracker Projects

1.2.2.1 Overview

A project is a collection of tasks performed by individuals towards the completion of some end-products or to provide services. Each project consists of a project manager and a list of workers assigned to it.

The following project attributes will be maintained:

- 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
- Project Manager – the user in charge of the project
- Workers – the list of workers assigned to the project

The database schema to store projects is as follows:

```
CREATE TABLE Projects (
    ProjectsID          integer NOT NULL,
    Name               varchar(64) NOT NULL,
    Description         varchar(64) NOT NULL,
    StartDate          datetime NOT NULL,
    EndDate             datetime NOT NULL,
    CreationDate          datetime NOT NULL,
    CreationUser          varchar(64) NOT NULL,
    ModificationDate      datetime NOT NULL,
    ModificationUser      varchar(64) NOT NULL,
    PRIMARY KEY (ProjectsID)
);
```

1.2.2.2 API Requirements

The following operations for projects have been identified:

- Create a new project
- Update existing project information
- Delete an existing project
- Assign a client as a project owner
- Query the client who owns a project
- Assign a project manager to a project
- Query the project manager of a project
- Add a worker to a project
- Remove a worker from a project
- Enumerate the workers assigned to a project
- Add a new time entry to a project
- Remove an existing time entry from a project
- Enumerate time entries belonging to a project
- Add a new expense entry to a project
- Remove an existing expense entry from a project
- Enumerate expense entries belonging to a project

1.2.3 Project Managers

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.

The database schema to store project managers is as follows:

```
CREATE TABLE ProjectManagers (
    ProjectsID          integer NOT NULL,
    UsersID            integer NOT NULL,
    CreationDate          datetime NOT NULL,
    CreationUser          varchar(64) NOT NULL,
```



```
    ModificationDate     datetime NOT NULL,
    ModificationUser     varchar(64) NOT NULL,
    PRIMARY KEY (ProjectsID, UsersID)
);
```

1.2.4 Project Workers

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.

The following worker attributes will be maintained:

- 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

The database schema to store project workers is as follows:

```
CREATE TABLE ProjectWorkers (
    ProjectsID          integer NOT NULL,
    UsersID             integer NOT NULL,
    StartDate           datetime NOT NULL,
    EndDate             datetime NOT NULL,
    PayRate             money NOT NULL,
    CreationDate        datetime NOT NULL,
    CreationUser        varchar(64) NOT NULL,
    ModificationDate    datetime NOT NULL,
    ModificationUser    varchar(64) NOT NULL,
    PRIMARY KEY (ProjectsID, UsersID)
);
```

1.2.4.2 API Requirements

The following operations for project workers have been identified:

- Create a new worker
- Update existing worker information
- Delete an existing worker

1.2.5 Pluggable Persistence

All entities defined above will be backed by a database. The design will provide the necessary API to store and retrieve data from the database.

For the initial 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 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 1.3, Java 1.4

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?

None.

3.2.2 TopCoder Software Component Dependencies:

- Configuration Manager
- DB Connection Factory

**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.