User Project Data Store 1.1 Requirements Specification

1. Scope

1.1 Overview

The component provides API to access the TopCoder user and project database in order to integrate with the Online Review application.

In Version 1.1 the ExternalProject interface is extended to include data already being stored in the project database but missing in the retrieved objects. The ProjectRetrieval implementation must retrieve the extended information. This new version must not break backwards compatibility; interfaces are only to be extended and J2EE compliance is to be kept.

1.2 Logic Requirements

1.2.1 User Database

Component will provide read only access to the TopCoder user database. Retrieval should be based on either external identifier or user handle. The following information will be retrieved: external identifier, user handle, first name, last name, emails, rating and reliability. Rating and reliability are optional and will be per phase (dev/design). A batch mode should be provided as well.

1.2.2 Project Database

Component will provide read only access to the TopCoder project database. Retrieval should be based on either external identifier, or project name/version. The following information will be retrieved: external identifier, project name, version, catalog type (Java/.NET), catalog reference identifier, forum reference identifier. A batch mode should be provided as well.

In version 1.1, the following information will also be retrieved: Functional Description, Long Description, Short Description and Technologies Involved.

1.2.3 Database Schema

The relevant database schema portion needed to specify the SQL statements for version 1.1 is available in the Appendix A.
The database field mapping the Technologies Involved data is technology_name from the technology_types table.

1.2.4 J2EE compliant

Version 1.0.2 is already compliant to be used inside an EJB context. This must not be broken.

1.2.4.1 It must get the connection in every method and never store it in an instance variable.

1.2.4.2 It must never call commit() or rollback() on the connection.

1.2.4.3 It must comply with EJB development restrictions (http://java.sun.com/blueprints/qanda/ejb_tier/restrictions.html).

1.2.5 Backward compatibility

In terms of external use, version 1.1 must not break backwards compatibility. Modifications to the interfaces are only new methods added. Existing methods must behave exactly as they do for version 1.0.2.

1.3 Required Algorithms

No specific algorithms are required.

1.4 Example of the Software Usage

- After user is logged into the application, the handle of the logged user will be displayed in the header. The session contains the user's external identifier. The component can be used to retrieve the user's handle as well as rating, so that the handle can be rendered in the color corresponding to the user's rating.

- When registering to projects, users have the ability to view project details including the short and the long description.

1.5 Future Component Direction

None.

2. Interface Requirements

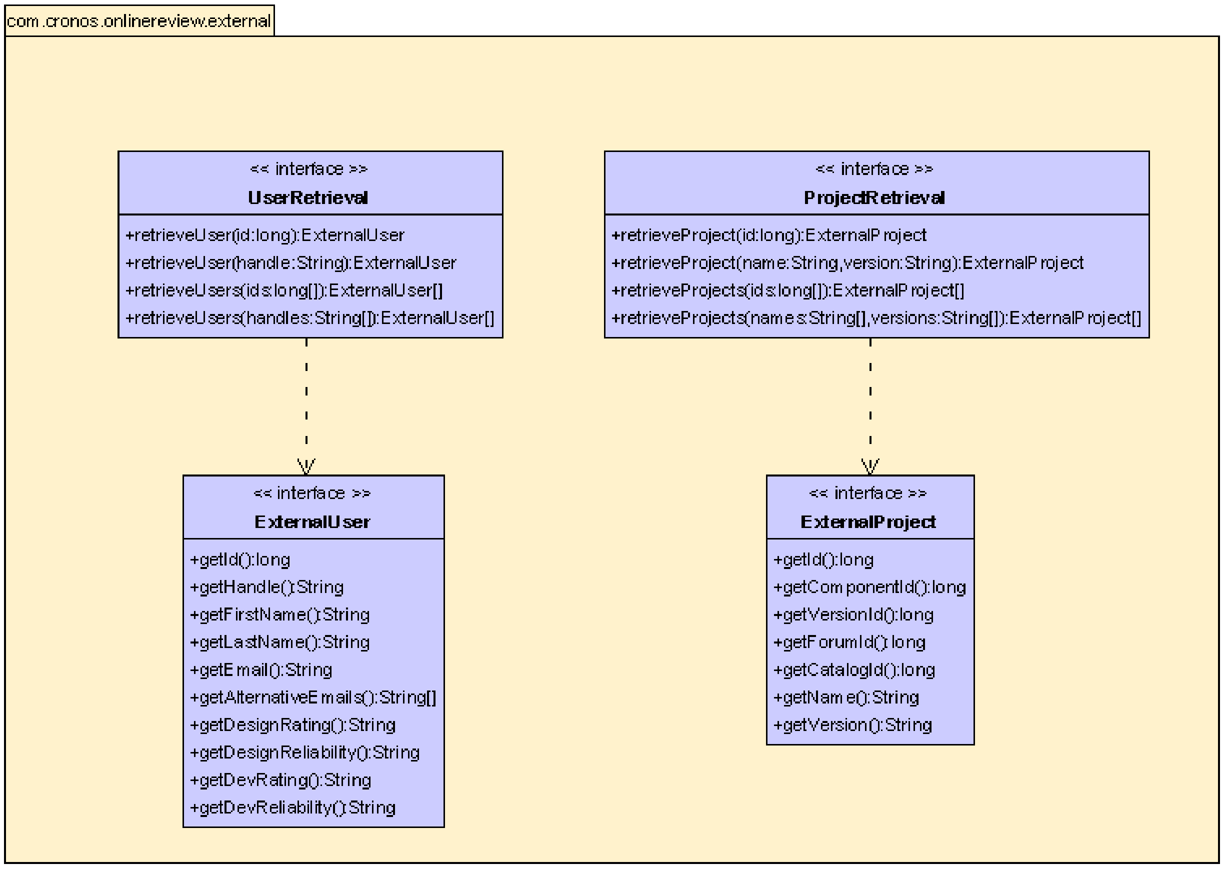
2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

Design must adhere to the interface diagram definition. Designer can choose to add more methods to the classes/interfaces, but must keep the ones defined on the diagram as a minimum. Source files can be found in the distribution.

User Project Data Store Interface Diagram


com.cronos.onlinereview.external
<< interface >>
UserRetrieval
+retrieveUser(id:long):ExternalUser
+retrieveUser(handle:String):ExternalUser
+retrieveUsers(ids:long[]):ExternalUser[]
+retrieveUsers(handles:String[]):ExternalUser[]
<< interface >>
ProjectRetrieval
+retrieveProject(id:long):ExternalProject
+retrieveProject(name:String,version:String):ExternalProject
+retrieveProjects(ids:long[]):ExternalProject[]
+retrieveProjects(names:String[],versions:String[]):ExternalProject[]
<< interface >>
ExternalUser
+getId():long
+getHandle():String
+getFirstName():String
+getLastName():String
+getEmail():String
+getAlternativeEmails():String[]
+getDesignRating():String
+getDesignReliability():String
+getDevRating():String
+getDevReliability():String
<< interface >>
ExternalProject
+getId():long
+getComponentId():long
+getVersionId():long
+getForumId():long
+getCatalogId():long
+getName():String
+getVersion():String

Created with Poseidon for UML Community Edition. Not for Commercial Use.

In version 1.1, the following methods must be added to the ExternalProject interface:

```
public String getShortDescription();
public String getLongDescription();
public String getFunctionalDescription();
public String[] getTechnologies();
```

In case there're no technologies involved in the component, getTechnologies() will return an empty array.

2.1.3 Environment Requirements

- Development language: Java1.4
- Compile target: Java1.4

2.1.4 Package Structure

com.cronos.onlinereview.external

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- Database connection

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

JDBC

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:

None.

3.2.4 QA Environment:

- Solaris 7
- RedHat Linux 7.1
- Windows 2000
- Windows 2003
- Informix 10.0

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines.

3.3.1 Database Connections

Database connections must not be cached within the component. Connections should be created for each operation and closed afterwards.

3.3.2 Component Scalability

The component needs to be scalable. Running multiple instances in the same JVM or in multiple JVM's concurrently should not cause any problem.
Version 1.1 implementation of ProjectRetrieval and UserRetrieval should not use thread synchronization primitives to synchronize access. (see EJB restrictions).

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.

4. Appendices

4.1 Appendix A: Component database structure

4.1.1 comp_versions Table

```
create table 'informix'.comp_versions (
   comp_vers_id DECIMAL(12,0) not null,
   component_id DECIMAL(12,0),
   version DECIMAL(12,0) not null,
   version_text CHAR(20) not null,
   create_time DATETIME YEAR TO FRACTION not null,
   phase_id DECIMAL(12,0) not null,
   phase_time DATETIME YEAR TO FRACTION not null,
   price DECIMAL(10,2) not null,
   comments lvarchar,
   modify_date DATETIME YEAR TO FRACTION default CURRENT YEAR TO FRACTION not null,
   suspended_ind DECIMAL(1,0) default 0
)
extent size 500 next size 124
lock mode row;

create unique cluster index 'informix'.comp_versions_i2 on 'informix'.comp_versions
          (
          component_id,
          version
          );

alter table 'informix'.comp_versions add constraint primary key
          (comp_vers_id)
          constraint pk_comp_versions;

alter table 'informix'.comp_versions add constraint foreign key
          (component_id)
          references 'informix'.comp_catalog
          (component_id)
          constraint fk_comp_versions;

alter table 'informix'.comp_versions add constraint foreign key
          (phase_id)
          references 'informix'.phase
          (phase_id)
          constraint fk_comp_phase;
```

4.1.2 comp_catalog Table

```
create table 'informix'.comp_catalog (
   component_id DECIMAL(12,0) not null,
   current_version DECIMAL(12,0) not null,
   short_desc lvarchar,
   component_name VARCHAR(254) not null,
   description lvarchar(10000),
   function_desc lvarchar,
   create_time DATETIME YEAR TO FRACTION not null,
   status_id DECIMAL(12,0) not null,
   root_category_id DECIMAL(12,0),
   modify_date DATETIME YEAR TO FRACTION default CURRENT YEAR TO FRACTION not null
)
extent size 1000 next size 500
lock mode row;

alter table 'informix'.comp_catalog add constraint primary key
          (component_id)
          constraint pk_comp_catalog;

alter table 'informix'.comp_catalog add constraint foreign key
          (status_id)
          references 'informix'.status
          (status_id)
          constraint fk_comp_status;
```

4.2 Appendix B: Technologies database structure

4.2.1 comp_technology Table

```
create table 'informix'.comp_technology (
   comp_tech_id DECIMAL(12,0) not null,
   comp_vers_id DECIMAL(12,0),
   technology_type_id DECIMAL(12,0)
)
extent size 250 next size 124
lock mode row;

create unique cluster index 'informix'.comp_tech_i1 on 'informix'.comp_technology
          (
          comp_vers_id,
          technology_type_id
          );

alter table 'informix'.comp_technology add constraint primary key
          (comp_tech_id)
          constraint pk_comp_technology;

alter table 'informix'.comp_technology add constraint foreign key
          (comp_vers_id)
          references 'informix'.comp_versions
          (comp_vers_id)
          constraint fk_comp_tech1;

alter table 'informix'.comp_technology add constraint foreign key
          (technology_type_id)
          references 'informix'.technology_types
          (technology_type_id)
          constraint fk_comp_tech2;
```

4.2.2 technology_types Table

```
create table 'informix'.technology_types (
   technology_type_id DECIMAL(12,0) not null,
   technology_name VARCHAR(100) not null,
   description VARCHAR(254) not null,
   status_id DECIMAL(12,0) not null
)
extent size 16 next size 16
lock mode row;

alter table 'informix'.technology_types add constraint primary key
          (technology_type_id)
          constraint pk_technology_type;

alter table 'informix'.technology_types add constraint foreign key
          (status_id)
          references 'informix'.status
          (status_id)
          constraint fk_tech_status;
```