TopCoder Security Groups Frontend Part 2 Requirements Specification

1.Scope

1.1 Overview

TopCoder is a company which administers contests in computer programming and provides software development services to other companies using a component based development approach.

The development process is managed using a series of tools, known as the Topcoder competition platform. The tools empower customers and client employees to directly take advantage of the competition platform to develop quickly and efficiently their software projects.

From time to time Topcoder updates different applications composing the competition platform in order to meet client needs or to make the development process more efficient.

This project addresses one of these enhancement requests, namely management of client overall account in the entire platform. The client account will include all client assets as well as arbitrary groups of users. For example, a client has many projects that are created in Cockpit and each of those projects has multiple contests. Each customer, project and contest also has a set of users that need to be aware of them.

This front-end module will provide all the necessary pages and corresponding front-end actions for all the new user interfaces.

This component implements JSPs and Struts 2 Actions related to ARS 2.7 – 2.9, 2.17 – 2.19.

1.1.1 *Version*

1.0

1.2 Logic Requirements

This component implements JSPs and Struts 2 Actions related to ARS 2.7 – 2.9, 2.17 – 2.19.

Note that this component uses the BaseAction of the Frontend Part 1 component.

1.2.1 *AcceptRejectGroupInvitationAction*

This action accepts or rejects group invitation.

It does the following:

Get invitation:GroupInvitation by invitationId using the groupInvitationService;

If accepted is true, set invitation.status with InvitationStatus.APPROVAL_PENDING;

Else set invitation.status with InvitationStatus.REJECTED;

Set invitation.acceptedOrRejectedOn with current time;

Update invitation using the groupInvitationService;

1.2.2 *ViewInvitationStatusAction*

This action searches invitations to display.

It does the following:

Filter invitations of just the current user as follow:

```
// get current user id
long userId = getCurrentUserId();
criteria.setOwnedUserId(userId);
```

Search invitations using the input fields and the groupInvitationService, result is put to the invitations output field;

1.2.3 *ViewPendingApprovalUserAction*

This action is used to view pending approval users. It does the following:

```
// add “pending approval” condition to the criteria
criteria.setStatus(InvitationStatus.APPROVAL_PENDING);
// filter invitations
// get current user id
long userId = getCurrentUserId();
criteria.setMasterUserId(userId);
// search invitations
invitations = groupInvitationService.search(criteria, clientId, pageSize, page);

// get user for each invitation, front end page needs to render user data
users = new ArrayList<UserDTO>();
for each invitation in invitations {
   users.add(userService.get(invitation.getGroupMember().getUserId()));
}
```

1.2.4 *ApproveRejectPendingUserAction*

This action approves or rejects user to a group.

It does the following:

```
for each invitationId of invitationIds {
   Get invitation:GroupInvitation by invitationId using the groupInvitationService;
   If approved is true, set invitation.status with InvitationStatus.APPROVAL_ACCEPTED;
   Else set invitation.status with InvitationStatus.APPROVAL_REJECTED;
   Set invitation.approvalProcessedOn with current time;
   Update invitation using the groupInvitationService;

   If approved is true {
      // update group member
      GroupMember groupMember = invitation.groupMember;
      groupMember.setActive(true);
      groupMember.activatedOn(new Date());
```

```
            groupMemberService.update(groupMember);
        }
    }
```

1.3 *Validation*

Validation is performed on server side using Struts 2 XML configuration based validators.

For complicated validation that can not be easily performed using the above mentioned validators, the action may override the validate method to perform custom validation.

Validation errors should be rendered back to the JSP pages.

Designers should refer to ARS corresponding sections for validation details.

1.3.1 *Prototype*

Prototype will be provided for this component. The pages related to ARS 2.7 – 2.9, 2.17 – 2.19 are in scope.

These pages should be converted to JSP pages, and properly bound to the above actions.

Note that in the above actions, result view names of the execution methods are not discussed, these are up to designers, also, sample Spring and Struts 2 configuration should be provided, so that page navigation works properly.

1.3.2 *Application Management*

See ADS 1.3 for application management details.

1.4 **Required Algorithms**

None.

1.5 **Example of the Software Usage**

This component implements JSPs and Struts 2 Actions related to ARS 2.7 – 2.9, 2.17 – 2.19. This component will be used for handling browser requests specific to group management, and for rendering results to be shown to the end user.

1.6 **Enhancement policy**

N/A - See details here:
http://apps.topcoder.com/forums/?module=Thread&threadID=728272&start=0

1.7 **Future Component Direction**

None

2.Interface Requirements

2.1.1 *Graphical User Interface Requirements*

None

2.1.2 *External Interfaces*

None

2.1.3 *Environment Requirements*

Development language: Java 1.6

Compile target: Java 1.6

2.1.4 *Package Structure*

com.topcoder.security.groups.actions

3. Software Requirements

3.1 Administration Requirements

3.1.1 *What elements of the application need to be configurable?*

See the actions in TCUML, fields marked with “config” should be configurable, they are injected by Spring IoC container.

3.2 Technical Constraints

3.2.1 *Are there particular frameworks or standards that are required?*

Spring 2.5.6

Struts 2.1.8

3.2.2 *Component Dependencies:*

Frontend Part 1 1.0

Its BaseAction is used in this component.

3.2.3 *Third Party Component, Library, or Product Dependencies:*

Spring 2.5.6: http://www.springsource.org/

Struts 2.1.8: http://struts.apache.org/

3.2.4 *QA Environment:*

JBoss 4

3.3 Design Constraints

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

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 the TopCoder UML Tool.