[TopCoder]

User Community Management 1.0 Requirements Specification

1. Scope

1.1 Overview

The TopCoder registration process to become a member of TopCoder is going to be redesigned. The main goal is to allow for a minimal set of required info that the user needs to enter to sign up (handle, email, and password).

The existing TC web registration process will ask a new user to enter a lot of data when he/she tries to register to TC web-site. The registration process takes a lot of time, so many users can simply break the registration process and leave non-registered

Thus there is a need of an easy to use and user friendly web-application for supporting registration on the new users and management of registered user profiles at TC web-site.

The main goal of this application is to simplify registration of the new users on TC web-site by minimizing the count of mandatory data fields for new account registration, and to improve usability of user profile management for the registered users. Any additional profile information will be requested from the user by the system as it is needed. For example, if a user registers to compete in an assembly contest the site would prompt them for any required info that they have not yet entered.

This component is responsible for implementing user referral and card/badges struts actions and updates user and user dao implementations

1.2 Logic Requirements

Each of the following actions will be designed as a Struts action. It is up to the designer to make best use of Struts capabilities, including interceptors and base action classes. The only client of this application will be the JSPs, so there is no API or structural requirements. The designer is free to refactor any interceptor, action, or controller as long as requirements are met.

1.2.1 View Referrals Action

This struts action should display the users who are referral by the given user.

Implementation notes:

It should implement the logic defined in the process method of
`com.topcoder.web.tc.controller.request.util\ViewReferrals.java.`

```
It should get the logged in user instance using the following statement.

User u = DAOUtil.getFactory().getUserDAO().find(new
Long(getUser().getId()));

Create referral request db query request

Request r = new Request();
r.setContentHandle("referral_list");
r.setProperty("uid", String.valueOf(getUser().getId()));

user referral will be obtained using

getDataAccess().getData(r).get("referral_list"));
```

It will mainly do the following steps:

- Get the user (guest or authenticated) from session or cookie.
- Check if user has permission to access the request URI corresponding to the struts action.
- Proceed to add preferences if user has the permission, otherwise, redirect to an authorization error page.

Note that the authorization error page will be configured globally in struts configuration file, and the created BasicAuthentication will be stored in session if it's not present.

1.2.2 GenerateReferralLinkAction

This action is new which will be used to generate referral link for the user. There is no existing code which will be reused for this purpose.

The generated referral code should stored in user entity. Referral link should be generated according to the user handle. Use TC Document Id generator component for generating Referral link code.

1.2.3 CardAndBadgesAction

This struts action should implement the ability to display information of TC cards and badges, able to download them and able to preview the user statistic badge.

General card badges here are displayed with static files. we don't need any strut actions for those. However to preview the user card strut action is needed.

Implementation notes:

display information of TC cards and badges and able to download them will be static and implemented using frontend jsp pages.

To preview member statistics badge, It should implement the logic defined in the process method of `com.topcoder.web.tc.controller.request.card\Preview.java.`

```
Prepare db query to get codes statistics.

  Request r = new Request();
  r.setContentHandle("member_profile");
  r.setProperty("cr", String.valueOf(getUser().getId()));

Execute the db query to preview data

ResultSetContainer coderInfo =  (ResultSetContainer)
      getDataAccess(DBMS.DW_DATASOURCE_NAME,true).getData(r).get("Coder_
      Data");

coderInfo contains the all code statistics information.
```

1.2.4 Updates to UserDAO and User entity

UserDAO should updated to generate and retrieve the user referral link code. A field to hold referral code should be incorporated into user entity.

1.2.5 *Logging*

The application will log activity and exceptions using the Logging Wrapper in struts actions, and the Logging Wrapper should be configured to use Log4j.

It will log errors at Error level, and method entry/exit information at DEBUG level.

Specifically, logging will be performed as follows, if logging is turned on.

- Method entrance and exit will be logged with DEBUG level.
  - Entrance format: [Entering method {*className.methodName}*]
  - Exit format: [Exiting method {*className.methodName}]*. Only do this if there are no exceptions.
- Method request and response parameters will be logged with DEBUG level
  - Format for request parameters: [Input parameters[{request_parameter_name_1}:{ request_parameter_value_1}, {request_parameter_name_2}:{ request_parameter_value_2}, etc.)]]
  - Format for the response: [Output parameter {response_value}] . Only do this if there are no exceptions and the return value is not void.
  - If a request or response parameter is complex, use its toString() method. If that is not implemented, then print its value using the same kind of name:value notation as above. If a child parameter is also complex, repeat this process recursively.
- All exceptions will be logged at ERROR level, and automatically log inner exceptions as well.
  - Format: Simply log the text of exception: [Error in method {*className.methodName}*: Details {*error details*}]

In general, the order of the logging in a method should be as follows:

1. Method entry
2. Log method entry
3. Log method input parameters
4. If error occurs, log it and skip to step7
5. Log method exit
6. If not void, log method output value
7. Method exit

The toString() method of the entity can be used for the logging.

1.2.6 *Auditing*

The existing application uses RequestTracker to track user's request, but it only tracks the user id, request url, session id and timestamp. while this module requires more data to be audited. Part1 registration process module http://www.topcoder.com/tc?module=CompContestDetails&pj=30017040 Has already implemented the audit requirements. It defines AuditDAO interface to perform the audit.

In this struts actions, the AuditDAO will be resued the operations should be audited using the AuditDAO, and the following data should be audited:

- Operation type (like create new account, change demographic data, etc)
- Username of the person
- User IP address
- Previous value
- New value
- Date/time stamp.

1.2.7 *Error Handling*

Refer to 2.11 *Exception Handling Overview in ADS.*

1.2.8 *Validation*

Refer to 2.13 validation in ADS

1.2.9 Performance

Refer to 2.5 Performance Enhancements in ADS

1.2.10 Existing systems

Refer to 2.14 Existing systems in ADS

1.2.11 Threading

Refer to 2.2 Threading in ADS

1.2.12 Guideline

Designer is not responsible for the front-end JSP pages, but the interactions between the JSP pages and struts actions must be clearly defined.

Designer is allowed to refactor the predefined struts actions in the module architecture as long as the requirements are addressed properly.

Designer is also expected to provide the struts-config.xml and input validation XML for the struts actions.

The implementation in the provided existing application code might be different from the ARS; in this case, the ARS takes the priority.

1.3 Method argument handling

This section describes the generic convention for expected handling of arguments, and may not refer to any specific operation for this component. However, any operation defined in this component that is affected by these conventions may override part or all of this. Such an override will be explicitly mentioned in the operation descriptions above.

- If getting an entity and it is not found, a method will return null.
- If getting a list of entities, and none are found, a method will return an empty list.
- Any list handled by the component must not contain null elements.
- Unless stated, all input arguments will be required (non-null). String arguments must not be null/empty.

If any input parameter requirement is not met, an ApplicationException with error code set to 500 will be thrown. If JSON input is invalid, then error code set to 400

1.4 Avoiding name collisions

Please be aware that there will be other service classes from other components in the services package, so please use unique and specific names for any base or helper classes so that they will not conflict with other such classes in other components for this architecture.

1.5 Required Algorithms

None

1.6 Example of the Software Usage

This component provides the struts actions for the TC registration process.

1.7 Enhancement policy

In order to eliminate superfluous, useless, and/or bloated enhancements from the application, the following policy on enhancements is in effect for this competition.

All major enhancements must be explicitly approved by the architect (the approval of PM and/or co-pilot is not sufficient). All enhancements proposed in the future direction section are considered to be approved. Only if the architect approves the enhancement may it be added to a design. Any attempt to add a major enhancement to a design without this approval will result in that enhancement to not be eligible for a score of 4 in the requirements section (unless this idea happens to correspond to another submission's enhancement that was approved).

You may outline the enhancement proposal in the forum. You may also contact the architect directly to retain the privacy of your ideas. After the conclusion of the submission phase, the architect will notify the reviewers of the approval so they may score for it.

Be aware that the approval of an architect does not automatically assure a 4 in the requirements section. The architect will approve an enhancement or enhancements based on how useful and pertinent they are to the application. The reviewers, though, will decide if the enhancement or sum of enhancements is substantial. It is possible that the architect may advise the reviewers of how substantial they may be to the application, but the final decision will be in the hands of the reviewers.

When making an enhancement request via Contact Manager, please put the following in your first line:

Enhancement Request

At this time, it may also help to contact this architect directly with Member Contact since Contact Manager does not send a notification to the architect. This would most likely expedite the process.

1.8 Future Component Direction

None

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None

2.1.2 External Interfaces

Refer to the provided architecture TCUML.

2.1.3 Environment Requirements

- Development language: Java
- Compile Target: Java 1.6

2.1.4 Package

com.topcoder.web.reg.actions.miscellaneous

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- The back-end DAO and EJB should be injected
- Session key for the BasicAuthentication object
- Email subject, email body template, email from address
- AuditDAO

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

Struts 2+

3.2.2 TopCoder Software Component Dependencies:

Logging Wrapper 2.0
Base Exception 2.0
Document Generator 1.1
Email Engine 3.2
User Profile and Audit Back End 1.0
Basic Struts Actions 1.0
Profile Struts Actions 1.0

3.2.3 Third Party Component, Library, or Product Dependencies:

- Spring 2.5.6 http://s3.amazonaws.com/dist.springframework.org/release/SPR/spring-framework-2.5.6-with-dependencies.zip
- Hibernate 3.6 http://www.hibernate.org/downloads.html

3.2.4 QA Environment:

- JAVA 1.6.0_04 http://www.oracle.com/technetwork/java/javase/downloads/index.html
- Apache 2.2.4 http://archive.apache.org/dist/httpd/
- jira 4.1.2 http://www.atlassian.com/software/jira/ArchiveDownloads.jspa
- wiki 2.7 http://confluence.atlassian.com/display/DOC/Confluence+2.7+Release+Notes
- Jive Professional 4.2.5 http://www.jivesoftware.com/
- Informix 11.5 http://www.ibm.com/developerworks/downloads/im/dsexp/
- JBoss-4.0.4.GA http://sourceforge.net/projects/jboss/files/JBoss/JBoss-4.0.4.GA/
- JavaScript (jQuery 1.4.4) http://api.jquery.com/category/version/1.4.4/
- Log4j 1.2.15: http://logging.apache.org/log4j/1.2/index.html
- Unit Testing JUnit 1.4 www.**junit**.org

3.3.3 Source control

- CVS

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

XML documentation must provide sufficient information regarding component design and usage.