SWIM v2

Testing Document

Authors:

Affetti Lorenzo

Canidio Andrea

January 24th, 2013

Summary

1. Introduction

1.1. Provided Material

In order to test our platform we provide the following documents:

- Requirements and Analysis Specification Document (RASD);
- Design Document (DD);
- The source code.

2. Test Cases

We write here the possible tests for the most important functionalities of the system.

We do not provide test cases for guest's functionalities because all his functionalities can be exploited also by a user (except from signing up).

1.2. Log in and Sign up

Goal	User registration.
Environment	The *Sign up Page*.
Input	Random but valid profile information (at least name, surname, e-mail address, password, one ability).
Expected output	The system confirms the sign up.
Obtained output	The same as above. Information is correctly stored into the database. A new user is created.
Final output	The system loads the *Home Page*.
Possible errors	• Empty mandatory fields: the system prevents the sign up phase to proceed (no data previously typed lost); • Not valid e-mail address: the system prevents the sign up phase to proceed (no data previously typed lost); • Mismatching passwords: the system prevents the sign up phase to proceed (no data previously typed lost); • Too short password: the system prevents the sign up phase to proceed (no data previously typed lost); • Another user has already signed up with the e-mail address provided: the system prevents the sign up phase to proceed. The guest is redirected to an *Error Page*.

Goal	Generic log in.
Environment	Any of the guest's pages which contains the top menu.
Input	E-mail address and password already contained into the database.
Expected output	The system confirms the log in.
Obtained output	The same as above.
Final output	User's/administrator's *Home Page*.
Possible errors	• The user/administrator enters not correct information: the system shows an *Error Page* in which it is possible to perform any action. • Some mandatory field is empty: the system doesn't let the user/administrator log in.

1.3. User's Functionalities

Goal	Send a friendship request.
Environment	The *Profile Page* of the user we want to send the friendship request to.
Input	The two users involved.
Expected output	The system confirms the action performed. The system "notifies" the other user.
Obtained output	The database stores the new pending friendship request putting the two users in association.
Final output	The system reloads the *Profile Page* and notifies that the friendship request has been stored.
Possible errors	There are no possible errors.

Goal	Reply to a friendship request.
Environment	The profile of a user that sent a friendship request (to the logged user) or the *Pending Friendship Requests Page*.
Input	The recipient of the friendship request.
Expected output	The system confirms/removes the friendship.
Obtained output	The same as above. The database stores/removes the new friendship correctly.
Final output	The system reloads the page where we started from and informs the user that the friendship has been accepted/removed.
Possible errors	There are no possible errors.

Goal	Post a new help request.
Environment	The *New Help Request Page*.
Input	Random but valid data about the help request (at least the subject, the ability needed, the date and the hour).
Expected output	The system confirms the post.
Obtained output	The same as above. The database stores the new help request in association with the user that posted it.
Final output	The system loads a page containing the new help request.
Possible errors	• Empty mandatory fields: the systems doesn't allow the user to post the help request.

Goal	Reply to a help request.
Environment	Any *Help Request Page*.
Input	The user.
Expected output	The system confirms our reply.
Obtained output	The same as above. The system stores the reply in association with the user that replies.
Final output	The help request is reloaded showing our reply.
Possible errors	There are no possible errors.

Goal	Select a best reply.
Environment	A *Help Request Page* which contains a help request which somebody has already replied to.
Input	The reply.
Expected output	The system confirms our action.
Obtained output	The same as above. The system changes the reply into a best reply.
Final output	The system reloads the page showing the help request together with all its replies and the new best reply.
Possible errors	There are no possible errors.

Goal	Give a feedback.
Environment	A *Help Request Page* which contains a help request and the user (that posted that help request) has already chosen a best reply.
Input	The best reply.
Expected output	The system confirms our feedback.
Obtained output	The same as above. The system stores the feedback given in association with the best reply considered.
Final output	The help request is reloaded showing our feedback.
Possible errors	There are no possible errors.

Goal	Send a message.
Environment	The *New message Page* or any *Conversation Page*.
Input	The message and the two interacting users or the conversation between them.
Expected output	The system confirms the message sent and "notifies" the other user involved.
Obtained output	The same as above. The systems stores the new message in association with the conversation (if it already exists) or it creates a new conversation (using the two user's information).
Final output	The system load the page of the conversation showing all messages between the two users.
Possible errors	• If the user is sending a message (starting from the *New Message Page*), he has no friends and tries to send a message to nobody, then the system shows an *Error Page*.

Goal	Send an ability request.
Environment	The *Modify Profile Page*.
Input	An ability which has not been already stored.
Expected output	The system confirms our request and "notifies" the administrator.
Obtained output	The same as above. The system stores the request.
Final output	The system reloads the *Modify Profile Page*. The new ability requested is in our abilities, but we are not be able to use it (to post new help requests) until the administrator accepts our request.
Possible errors	• The ability name we inserted is already into the database: the system loads an *Error Page* containing an error log.

Goal	Search for a user.
Environment	Any page which contains the search bar or the *Advanced Search Page*.
Input	A random username and/or a random city and/or an ability which is already contained in the database (the city and the ability are available only in the case of an Advanced Search).
Expected output	A result comparable with the input given.
Obtained output	The same as above.
Final output	The system loads a page containing the search result.
Possible errors	• No result found: the system shows the *Result Page*, but with the log "No result found".

1.4. Administrator's Functionalities

Goal	Add an ability.
Environment	The *Add New Ability Page*.
Input	The new ability.
Expected output	The system confirms our ability.
Obtained output	The same as above. The ability is correctly stored.
Final output	The system reloads the *Home Page* together with a confirmation message.
Possible errors	• The ability name we inserted is already into the database: the system loads an *Error Page* containing an error log.

Goal	Accept/decline an ability request.
Environment	The *Pending Abilities Page*.
Input	The ability we are trying to accept/decline.
Expected output	The system correctly confirms/removes the ability given.
Obtained output	The same as above. In case the administrator accepts, the ability is correctly stored into the database. In case the administrator declines, the ability is removed from the database and removed from the set of abilities of the user that sent the request.
Final output	The page is reloaded containing a confirmation message of our action.
Possible errors	There are no possible errors.