Visual Code Review

Group 5

Daniel Andersson Tenninge

Gustaf Carleson

Johan Björk

Patrik McKiernan

1. Functional Test Cases

1.1. Login as developer

Function being tested:
Logging in to the system as a developer.

Use Case:
Use Case 2.

Input:
Username and Password valid for the system.

Expected output:
The user is logged in as a developer and redirected to the Review listing page.

Instructions:
1. Write the username in the username textbox.
2. Write the password in the password textbox.
3. Press the login button.

1.2. Login as administrator

Function being tested:
Logging in to the system as an administrator.

Use Case:
Use Case 4.

Input:
Username and password for a valid administrator account.

Expected output:
The user is logged in as an administrator and is redirected to the Administration page.

Instructions:
1. Write username in username textbox
2. Write password in password textbox
3. Press login

1.3. Remember me

Function being tested:
If the system can remember login data.

Use Case:
Use Case 2,4.

Input:
Username and password for a valid account.

Expected output:
The user can automatically access any login protected page, without entering username and password again.

Instructions:
1. Write username in username textbox.
2. Write password in password textbox.
3. Click the “Remember me” checkbox.
4. Press login.
5. Close your browser window
6. Open and try to access the website.

1.4. Wrong password

Function being tested:
If the system can report invalid password.

Use Case:
Use Case 2,4.

Input:
Username and password that do not correspond to a valid account.

Expected output:
The user is redirected back to the login page, with an indicator that the wrong password was entered.

Instructions:
1. Write username in username textbox
2. Write password in password textbox
3. Press login

1.5. Logout

Function being tested:
That you are able to logout

Use Case:
Use Case 2,4.

Input:
None.

Expected output:
The user is redirected back to the login page, having to enter a password and username to get back to the site.

Instructions:
1. Login (Follow test “Login as Developer”)
2. Press Logout

1.6. Add Developer

Function being tested:
Adding a developer to the system.

Use Case:
Use Case 5

Input:
N/A

Expected output:
A new developer is added to the systems database.

Instructions:
1. Click the “Add developer” button on the “Administrate developers” page.

1.7. Remove Developer

Function being tested:
Remove a developer to the system.

Use Case:
Use Case 5

Input:
N/A

Expected output:
A developer is removed from the systems database.

Instructions:
2. Click the “Remove developer” button on the “Administrate developers” page.

1.8. Save new changes

Function being tested:

Save new changes made to a selected developer.

Use Case:
Use Case 5

Input:
Name and email for the developer and any selected rights for that developer.

Expected output:
The information is about the developer is updated and is shown when she is selected from the list of developers.

Instructions:
3. Enter the “Administrate developers” page.
4. Select a developer.
5. Make changes to the information displayed.
6. Click the “Save” button.

1.9. Cancel changes

Function being tested:
Cancellation of changes made to a developer.

Use Case:
N/A

Input:
N/A

Expected output:
The information displayed about the developer is set back to what is stored, thereby removing any changes made.

Instructions:
- Enter the “Administrate developers” page.
- Select a developer.
- Make changes to information.
- Click the “Cancel button”.

1.10. Reset password

Function being tested:
Resetting a developer’s password.

Use Case:
N/A

Input:

N/A

Expected output:
A new randomized password is set and is then email to the selected developer.

Instructions:
7. Select a developer.
8. Click on the “Reset password” button on the “Administrate developers” page.

1.11. Test save with no selected developer

Function being tested:
Trying to click the save button with no selected developer.

Use Case:
N/A

Input:
N/A

Expected output:
An alert dialog shall be displayed informing the user that no developer is selected and therefore the save function is not applicable.

Instructions:
9. Enter the “Administrate developer” page.
10. Click on the “Save” button.

1.12. Switch to branch

Function being tested:
Switching from administrating developers to administrating branches.

Use Case:
N/A

Input:
N/A

Expected output:
The administrator is redirected from the “Administrate developers” page to the “Administrate branches” page.

Instructions:
11. Enter the “Administrate developers” page.
12. Click on the link titled “Branches”.

1.13. Select developer

Function being tested:
Selecting a particular developer from the list of developers.

Use Case:
N/A

Input:
N/A

Expected output:
A developer is selected and information about the developer is displayed in the fields. The information displayed includes, the developers name and email and also the rights of the developer.

Instructions:
1. Enter the “Administrate developers” page.
2. Select a developer from the list of developers by clicking on a name.

1.14. Add branch

Function being tested:
Adding a branch and setting policies to it.

Use Case:
Use Case 4.

Input:
Path to where the branch is located, number of reviewers required and their rights.

Expected output:
The new branch gets added to the list of available branches.

Instructions:
1. Press the “Add branch” button.
2. Enter the path to the branch in the path textbox.
3. Enter number of reviewers required in the # of reviewers’ textbox.
4. Select the rights required by the reviewers.
5. Press the save button.

1.15. Remove branch

Function being tested:
Removing all the review policies from a branch.

Use Case:
Use Case 4.

Input:
The branch selected for removal.

Expected output:
The selected branch is removed from the list of branches being controlled by review policies.

Instructions:
1. Select a branch from the list over available branches.
2. Press the remove branch button.

1.16. Save new changes

Function being tested:
Saving new changes to an already added branch.

Use Case:
Use Case 4.

Input:
Path to where the branch is located, number of reviewers required and their rights.

Expected output:
The information about the branch is updated with the new one.

Instructions:
1. Select a branch from the list over available branches.
2. Change the information in the appropriate text boxes.
3. Press the save button.

1.17. Cancel changes

Function being tested:
Cancel an ongoing change in the information about a branch.

Use Case:
Use Case 4.

Input:
None.

Expected output:
All the text boxes gets updated with the old information about the selected branch and none of the new one is stored.

Instructions:

1. Select a branch in the list over available branches.
2. Make some changes in the text boxes.
3. Press the cancel button.

1.18. Test save with no selected branch

Function being tested:
Trying to change settings without having selected a branch.

Use Case:
N/A

Input:
Path to where the branch is located, number of reviewers required and their rights.

Expected output:
A dialog box will appear stating the no branch is selected. None of the new information is stored.

Instructions:

1. Make sure the no branch is selected.
2. Enter some information in the text boxes.
3. Press the save button.

1.19. Switch to developer

Function being tested:
Switching from the branch settings page to the developer settings page.

Use Case:
N/A

Input:
None.

Expected output:
The user is presented with the developer settings page.

Instructions:

1. Press the developer link.

1.20. Select branch

Function being tested:
Selecting a branch.

Use Case:
Use Case 4.

Input:
None.

Expected output:
The branch gets selected and its information is shown in the text boxes.

Instructions:
1. Select a branch in the list over available branches.

1.21. Search

Function being tested:
That you are able to search the changesets

Use Case:
Use Case 2.

Input:
A search string

Expected output:
The user is presented with a RevList with only matching items

Instructions:
1. Follow testcase 1.1 to login
2. Enter a search string into the searchfield and press enter

1.22. Select changeset

Function being tested:
That you can select changesets

Use Case:
Use Case 2.

Input:
Select a changeset

Expected output:
The user is redirected to the Review Single page for the selected changeset.

Instructions:
1. Follow testcase 1.1 to login
2. Select any changeset

1.23. Change page

Function being tested:
That you can display all pages of the changesets.

Use Case:
Use Case 2.

Input:
Selected pagenumber

Expected output:
The user is presented with a Rev Listing with different items.

Instructions:
1. Follow testcase 1.1 to login
2. Select a pagenumber

1.24. Select a file in a commit

Function being tested:
To display the source code of a file in a commit.

Use Case:
Use case 2.

Input:
The selected file to display.

Expected output:
A new form showing the selected file content and a field where you can leave a comment.

Instructions:
1. In a browser window, log in as a developer.
2. Open a commit up for review.
3. Select a file by clicking at the name of the file in the list to open it.

1.25. Denial of commit acceptance

Function being tested:
The denial of acceptance of the entire commit if at least one file in it has been rejected.

Use Case:
Use case 2.

Input:

The commit or change set that you want to accept.

Expected output:
A message informing the user that part of or the entire commit has been rejected.

Instructions:
1. In a browser window, log in as a developer.
2. Open a commit and reject at least one file in it.
3. Try to accept the entire commit.

1.26. Accept a commit without supplying a comment

Function being tested:
The acceptation of a commit even if the user has chosen not to comment on the action taken.

Use Case:
Use case 2.

Input:
The commit/change set that you want to accept.

Expected output:
The status of the commit will be updated in the list of commits up for review.

Instructions:
1. In a browser window, log in as a developer.
2. Select a commit up for review.
3. Accept the commit without leaving a comment.

1.27. Accept a commit by accepting all files and leaving a comment

Function being tested:
The acceptation of a commit when the user has accepted all files separately and wishes to leave a comment on the entire commit before accepting it.

Use Case:
Use case 2.

Input:
The commit/change set that you want to accept.

Expected output:
The status of the commit will be updated in the list of commits up for review.

Instructions:

1. In a browser window, log in as a developer.
2. Open a commit and accept the files in it, one by one.
3. In the window containing the files in the commit, write a comment in the comment window.
4. Try to accept the entire commit.

1.28. Reject a commit without supplying a comment

Function being tested:
The rejection of a commit when you have not supplied a comment.

Use Case:
Use case 2.

Input:
The commit/change set that you want to reject and the comment.

Expected output:
The status of the commit will be updated in the list of commits up for review and the comment saved in the database.

Instructions:

1. In a browser window, log in as a developer.
2. Select a commit up for review.
3. Reject the commit without leaving a comment.

1.29. Reject a commit and supply a comment

Function being tested:
The rejection of a commit when you choose to supply a comment.

Use Case:
Use case 2.

Input:
The commit/change set that you want to reject and the comment.

Expected output:
The status of the commit will be updated in the list of commits up for review and the comment saved in the database.

Instructions:

1. In a browser window, log in as a developer.
2. Select a commit up for review.
3. Write a comment in the comment field.

4. Reject the commit.

1.30. Accept a single file without supplying a comment

Function being tested:
The acceptation of a commit when the user doesn't supply a comment.

Use Case:
Use case 2.

Input:
The file name of the source code you want to accept and the change set it belongs to.

Expected output:
The status of the file in the commit up for review will be updated.

Instructions:
1. In a browser window, log in as a developer.
2. Select a commit up for review.
3. Select a file belonging to that commit.
4. Accept the file.

1.31. Accept a single file and leave a comment

Function being tested:
The acceptation of a single file belonging to a commit when the user has chosen to leave a comment.

Use Case:
Use case 2.

Input:
The file name of the source code you want to accept, the change set it belongs to and the comment.

Expected output:
The status of the file in the commit up for review will be updated and the comment saved in the database.

Instructions:
1. In a browser window, log in as a developer.
2. Select a commit up for review.
3. Select a file belonging to that commit.
4. Leave a comment in the comment field.
5. Accept the file.

1.32. Reject a single file without supplying a comment

Function being tested:
The rejection of a single file when the user has not supplied a comment.

Use Case:
Use case 2.

Input:
The file name of the source code you want to accept and the change set it belongs to.

Expected output:
The status of the file in the commit up for review will be updated and the comment saved in the database.

Instructions:
1. In a browser window, log in as a developer.
2. Select a commit up for review.
3. Reject the commit without leaving a comment.

1.33. Reject a single file and leave a comment

Function being tested:
The rejection of a single file belonging to a commit when the user has chosen to leave a comment.

Use Case:
Use case 2.

Input:
The file name of the source code you want to reject, the change set it belongs to and the comment.

Expected output:
The status of the file in the commit up for review will be updated and the comment saved in the database.

Instructions:
6. In a browser window, log in as a developer.
7. Select a commit up for review.
8. Select a file belonging to that commit.
9. Leave a comment in the comment field.
10. Reject the file.

1.34. Commit to branch without policy

Function being tested:
That the code will be committed to the SVN.

Use Case:
Use Case 1.

Input:
A commit created by a third-party SVN client.

Expected output:
The user receives the normal message indicating success from its SVN client.

Instructions:
1. Commit to the company “Visual Code Review” protected SVN server, following normal procedures for this action.
2. Make sure it is stored safely in the SVN repository.

1.35. Commit to branch with policy

Function being tested:
That the code will not be commited directly to the SVN repository, and that it will be added into the database.

Use Case:
Use Case 1.

Input:
A commit created by an thirdparty SVN client.

Expected output:
The user receives a message from Visual Code Review indicating that it is due for code review.

Instructions:
1. Commit to the company “Visual Code Review” protected SVN server, to a branch that has a policy, following normal procedures for this action.