INF3121 Assignment 2, Document 3
Explanation of Manual Tests That Could Not Be Automated

Even Langfeldt Friberg* Husein Mehmedagic†

18th May 2015

1 Test Case: CreateUserAccount.file

The value of `id=wpName2` is *husven_ test3*, but since we already created this user when we performed the manual test, this username is no longer unique. One could of course change this value to something unique, but the automated test would still fail because the value of `id=wpCatchaWord` was *aryanbusy* when we first interpreted the captcha, and this is completely unlikely to be the answer to a new captcha given.

One could probably make an automated test that copied the source of the captcha image, made use of an online decaptcha service, copied the computed captcha word and pasted it into `id=wpCaptchaWord`, but we did not find such a service available freely.

2 Test Case: Recent_Changes.file

The tester couldn't automated step (8)-(10), which in the manual test case is negative testing to assure that giving invalid input (i.e. the word `fifteen` instead of the positive integer `15`) to textfield (*Number of edits to show in recent changes, page histories (...)*) should give an error. There is an error given, but it's in the form of a JavaScript (?) coloring the textbox red and a popup message. The error message does not hinder the user to save.

In the same manner we could not automate step (11)-(12) as giving invalid input (i.e. `-10`) to textfield (*Number of edits to show in recent (...)*) did not give an error what so ever, but accepted `-10` as input. Obviously `-10` was actually interpreted as `0`, which was confirmed if one loaded the *Recent changes* page after giving this input. This could be categorized as a low-importance bug in Wikipedia.

We could not automate steps (23)-(31) that is about showing the latest approved (stable) or absolutely latest (might be non-approved) article page. Coincidentially we found an article on Iceland (as of 2015-05-11) that existed in two latest editions (approved and non-approved) that differed in their stating

*evenlf@student.matnat.uio.no.
†huseinm@student.matnat.uio.no.

of the 2013 HDI numbers for the country. This is subject to change fast, and I found it makes no sense to automatize this specific check. An article can change status from non-approved to approved as administrators and trusted users approves changes.

3 Test Case: Editing.file

We can se that it's possible to test for the positive and negative testing since we get error message when we try to put an invalid value in fields 8) - 12) it appears that only positive digits of specified value are legal. Otherwise it would be possible to enter any value that wouldn't give sense for the user. **Popups when entering invalid value e.g letters or other characters dont give error but a popup that says only digits possible. This is a bit tricky to test in Selenium IDE this probably has something to do with the Javascript and the popups don't show up in Safari browser as popups but as an error message like the one's when we enter an invalid digit.** But any way it's possible to test that the correct value is entered either if it's a caracter or negative value it's not possible to make a change and wiki forces one to enter a correct value so this part is possible to test in Selenium IDE. Further we can se in the automated test using the command "assertHtmlSource" and set target to look for regex expressions to look for the specified elements work very good. This gives us an option to controll that columns and rows are reduced to the chosen size that we prefer. The size is chosen such that we can both se it with our own eyes and that the regex expression can controll and check that the size we have chosen is the changed to chosen size. The "Edit pages on double click" under "General options" is possible to se that works when we run the automated test. The tests performed here and under other menu options are chosen based on what is importent for one normaluser and wich options seem reasonable to test. One could argument wich radio buttons and clicks are more important but in this case it's more subjective what is more important for some normal user.

4 Test Case: Watchlist.file

We can se that it's possible to test for the positive and negative testing since we get error message when we try to put an invalid value in fields 2) - 7) it appears that only positive digits of specified value are legal. Otherwise it would be possible to enter any value that wouldn't give sense for the user. The reason of choosing "Hide my edits from the watchlist" in advanced options is because we can make Selenium IDE to go in to the HTML and CSS source file and specifie what to look for because the user username is not a variable that changes like the variables in Watchlist. **Options that changes with time and are not possible to automate Changes are variables that change and some times it can say "Below are the last 37 changes in the last 720 hours" another day it may say Below are the last 60 changes in the last 720 hours. SO it's difficult to test a variable. You can also here use command "assertHtmlSource" and regex expression but the proble is that it would find it first time and all the times there are no**

changes. But after a while it wouldn't work since the variable would change and the automated test wouldn't have any value anymore in this case. Here is the same case as in the "Editing" when it comes to positive and negative testing. One important thing that we saw possible to test was "Hide my edits from the watchlist" under "Advanced options". Reason for testing this is that it's possible to automate it usig "assertHtmlSource" command in Selinium IDE. We can test that changes made by our user are visible to begin with and then when we do our changes we can go to "Watchlist" menu and use "assertNotHtmlSource" command in Selenium IDE to controll that edits made by our username are not visible anymore. This test we could se both by the manual test and get confirmed by the automated test using "assertNotHtmlSource" command in Selenium IDE and the regex expression in target field. We can also se that when we turned "Edit area font style" to "Serif font" under "Editor" options that we could use command "assertHtmlSource" and use regex expresion in taget field to confirm that the "Edit area font style" was changed to "Serif font" with Selenium IDE.

5 Test Case: Notifications.file

This one seems to be difficult to automate due to the changes and email receiving to confirm that the changes are are made witch makes it difficult to automate. Reason for this is that actions relies of other people interacting with "Talk" and "notifications" which may be difficult to interpret with Selenium IDE and even with manual test. The problem here is that when we run the other tests like Watchlist we need to go to Talk option and then the message indicator in the toolbar despair and it gives us an error because the test relies that the message is not red. **So its possible to test but it may give som issues when running the whole test suite in specific order. If dropping to test "Watchlist" before testing Notifications it may work just fine with the same technics as before and specify the command and target using "assertHtmlSource" to begin with and then use "assertNotHtmlSource" when there has been a notification and not been read**. As mentioned before this one may fail and is not possible to automate with out failing many times. It works under special circumstances but we always ned a automated test that works no mather the circumstances. **So the conclusion is this one fails to be automated to work 100%. An automated test is suposed to work with out needing to think of circumstances, and just being able to run them and perform the tasks.**