Manual testing and test automation

Uy Quang Nguyen (Uyqn)
Nikola Medic (Nikolame)
`INF3121 - Project Assignment 2`

May 3, 2016

Contents

Abstract

This project assignment was different than project assignment 1 when it came to setting up the work environment. I didn't need to switch operative system. The software that was required, "Selenium IDE" was supported on OSX, so it was easier + less hassle to do than PA 1.

1 Introduction

For this project, I'm working on the same computer as PA1 (MacBook Pro), but with my original operative system, due to Selenium IDE supporting it.

This assignment was done on / with

- MacBook Pro (Retina, 15-inch, Late 2013, OSX 10.11.13 El Capitan) [1]
- i7 - 4750HQ @ 2.00ghz & 2880x1800 (resolution)
- 100 MBPS Download & Upload
- Mozilla Firefox, Version 45.0.2 (newest version as of 18.04.16) [2]
- Mozilla Firefox add-on: Selenium IDE 2.9.1 (newest version as of 18.04.16)[3]

Setting up my work environment was simple. A fresh install of OSX El Capitan, followed by installing Mozilla Firefox, and installing the additional add-on Selenium IDE. No other software was installed during this project assignment, because I didn't want any other software to interfere.

1.1 Background

The courses I've had on University of Oslo that might be relevant to this assignment:

- INF1050 - "Systemutvikling"[4], 2014 Spring
- INF1500 - "Introduksjon til design, bruk, interaksjon"[5], 2015 Autumn

These are the courses that I've had during my 2 years on UiO that I think are relevant. In INF1050 and INF1500, we learned about different design techniques, while in INF1500 we even learned about testing (prototype, testing methods), so hopefully I'll get to use some of that knowledge here.

1.2 Motive

The objective of this task is to make both manual and automated test for a website that is presented to us, testing specific items that are described. We will be using black-box / experience-based test design techniques. Fortunately, this is what I used last project assignment, so I have some to basic experience with that. But what is the purpose of creating manual and automated tests, and why bother making them both?

- Manual Testing
  Manual testing is the process in which we run tests manually. This is done by comparing expected results and the actual results, together with pre and post conditions. It's basically the same as the end user using the program, but in this case, we'll be determining if the software is doing what it should.
- Automated Testing
  Automated is the process in which automated testing tools run the tests with predefined actions that are repeated. The results are compared with the preset expectations and outcomes. Automated tests make regression testing and repeating tests easy.

By combining these, we not only get the effectiveness of automatic testing, but also the thorough testing of Manual Testing, which is more likely to find real user issues.

[1] https://support.apple.com/kb/SP690?locale=en$_US$
[2] https://www.mozilla.org/en-US/firefox/new/
[3] https://addons.mozilla.org/en-US/firefox/addon/selenium-ide/
[4] https://www.uio.no/studier/emner/matnat/ifi/INF1050/
[5] https://www.uio.no/studier/emner/matnat/ifi/INF1500/

2 Requirement 1 – Manual tests, test design, bugs (10 points)

In your PDF delivery, design the manual tests you will perform by using one or more test design technique. You can choose different techniques for different items tested.

For this requirement, I've written manual tests for all of the items, which should cover all of the cases. I've below made a table that shows which cases I've tested, and what item covers it, together with what technique I've used and the explanation for that approach.

Table 1: List of use cases and approach.

Use Case	**Test-Technique used**	**Explanation of approach**
Sign-in	Decision Table	I've chosen to use a decision table for this use case. Fast to construct, and the most important thing is that it was easily understood, showing us the logical conditions and requirements. It was a perfect match for this use case, considering that there wasn't many test combinations and conditions, making our table compact and accurate.
Select Products	State Transition	For this use case, I've chosen state transition instead of decision table. In my opinion decision table is a good fit when you need to have the conditions in check. When selecting product, I believe that the most important thing is to see if the software moves from state to state like it's supposed to, and using a state diagram easily shows us the inputs and events that triggers the state transitions.
Change num of items	State Transition	I've used state transition once more for this use case. Like "Select Products", showing the state of the shopping cart in a diagram is easy, and gives us a good understanding of what the different input does to the carts state.
Checkout	Error Guessing	For Checkout & Finalize order, Ill be using error guessing as my test technique. This is because the checkout & finalize procedure requires the user to fill out the most amount of fields, and based on my experience, these fields tend to lead to alot of bugs & such as accepting invalid inputs, or limiting post code to be exactly 4 numbers (while some countries use 5), or checking out with an empty cart.
Finalize order	Error Guessing (together with checkout)	Merged with the above, as they were very similar in nature.
Log out	State Transition	For the last, "Log Out", I've used state transition, showing what input will lead to a logout state from logged in. Simple as that.

2.1 Manual tests

2.1.1 Use Case: Sign-In

As explained above, I've made a decision table for this use case. I've taken the example given in the assignment and added some actions and conditions to it. Together with this table, a table showing the manual tests for the relevant items are below. The manual tests table is created from this table, testing that the conditions leads to how the actions are treated.

Table 2: Decision Table

CONDITIONS	New user?	Y	N	N	N	N	N	N
	Valid Username?	-	N	-	Y	Y	N	Y
	Valid Password?	-	-	N	N	Y	N	Y
	Logged in?	-	-	-	-	-	N	Y
ACTIONS	Create a new account	Y	X	X	X	X	X	X
	Login unsuccessfully	X	Y	Y	Y	X	Y	X
	Login successfully	Y	X	X	X	Y	X	Y
	Reset password	X	X	X	X	X	X	Y

PS. The definition of "Login unsuccessfully" is when the user receives an error message.

Table 3: List of manual tests for Sign-In

Test Name	Description	Steps	Pre-Condition / Post-Condition	Expected Results	Actual Results
"Test that a new user can create an account"	The purpose of this test is to check if a new user can create an account. We assume that all fields are filled out correctly.	1. User navigates to avactis website. 2. User clicks "Sign-in" or "my account" 3. User clicks "Register" 4. User inputs all required fields correctly. 5. User clicks "Register"	PRE-CONDITION 1. None POST-CONDITION 1. System stores account information 2. User is authenticated.	The e-commerce site should create an account based on the information given by the user, and grant the user access to the website after registration.	The e-commerce site creates an account based on the information given by the user, and grants the user access to the website after registration
"Test that a registered user can login"	The purpose of the test is to check if a registered user can login given that the correct information is given to the system.	1. User navigates to avactis website. 2. User clicks "Sign-in" 3. User fills out password & e-mail correctly. 4. User clicks "Sign-in"	PRE-CONDITION 1.Valid Username 2.Valid Password 3.User has an account POST-CONDITION 1. System authenticates the user	If a registered user inputs a valid username & password combination, he should be granted access to his/her account	The registered user is granted access to his/her account
"Test that a registered user can reset his password"	The purpose of the test is to check if a registered user can reset his password given that he is already authenticated and logged in.	1. User clicks "My account" 2. User clicks "Change password" 3. User fills out current password correctly. 4. User inputs new desired password twice. 5. User clicks "Continue"	PRE-CONDITION 1. User has an account 2. User is already logged in POST-CONDITION 1. Password is changed to new desired password.	Assuming that the user enters his current password correctly, the system should change his password to the new desired one.	The user has his/her password changed to the new desired one.
"Test that wrong password will grant error messsage"	The purpose of the test is to make sure that an error message displays when an user enters wrong password.	1. User navigates to avactis website. 2. User clicks "Sign-in" 3. User fills out password with invalid information. 4. User clicks sign in	PRE-CONDITION 1. None POST-CONDITION 1. Error message is displayed	If an invalid combination of password and username is given, the system should display an error message.	The system displays an error message, preventing the user to log in.
"Test that invalid username will grant error message"	The purpose of the test is to make sure that an error message displays when an user enters invalid username	1. User navigates to Avactis website. 2. User clicks "Sign-in" 3. User fills out username field with invalid information 4. User clicks sign in	PRE-CONDITION 1. None POST-CONDITION 1. Error message is displayed	If an invalid combination of password and username is given, the system should display an error message.	The system displays an error message, preventing the user to log in.

2.1.2 Use Case: Select Products

For this requirement, we've written manual tests in a table similar to the earlier use case, but this time, we've used a different testing technique. We've used the state transition test technique. Our diagram below is inspired by the one given in the example. The state diagram is created with the tool supplied on the website Creately[6].

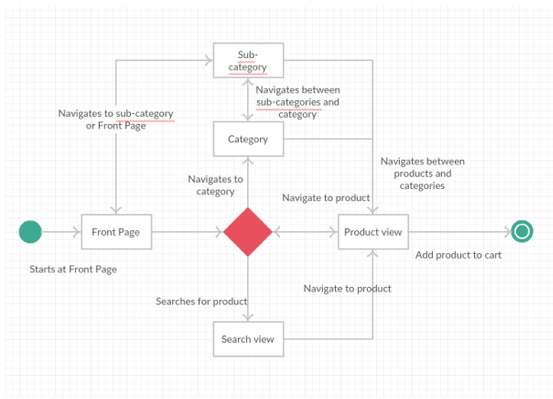


Figure 1: State Transition Diagram of select products

In the diagram, we don't only check if the user can add items to his cart, but also the navigation itself. Considering that the user most likely will navigate through the categories or even search for the product, I think it's essential to test it, even if not listed in the requirements.

Table 4: List of manual tests for Selecting Product

Test Name	Description	Steps	Pre-Condition / Post-Condition	Expected Results	ActualResults
"Test that a user can navigate to categories from frontpage and back."	The purpose of this test is to see if a user can navigate between the front pages and categories while shopping. We assume that the user has a internet connection at all times.	1. Navigates to category or subcategory 2. User navigates back to frontpage	PRE-CONDITION 1. User is on frontpage of site. POST-CONDITION 1. User can continue navigating. 2. No error message is shown.	There should be no problems for the user to navigate between the categories and the frontpage while looking for a product.	There are no problems navigating the categories and the front page.
"Test that the search function on the website is working"	The purpose of this test is to see if search function on the website is working properly.	1. User clicks on magnifying glass 2. User fills out search field 3. User clicks "Search"	PRE-CONDITION 1. User is on the website. POST-CONDITION 1. Matching products are returned and shown.	If a user searches for a product, and there exists matching products, the results should be returned and shown, giving the user the opportunity to go to the product pages.	Matching products are returned when searched for.
"Test that the user can add products to the shopping cart"	The purpose of this test is to check if an user can add products to the shopping cart.	1. User clicks on a product 2. User clicks "ADD TO CART"	PRE-CONDITION 1. User is on the website POST-CONDITION 1. Product added to cart.	The user should be able to add items to his cart without errors.	The item is added to the users shopping cart.
"Test that the user can continue shopping after adding products to cart"	The purpose of this test is to check if adding a product to the cart prevents a user from navigating to another product.	1. User clicks on a product. 2. User clicks "ADD TO CART"	PRE-CONDITION 1. User is on the website 2. User has already added something to his/her cart. POST-CONDITION 1. Product added to cart	Adding a product to the shopping cart should not prevent the user from navigating the site, and adding another product after the first should be no issue.	The next item is added to the users shopping cart.

[6]https://creately.com

2.1.3 Use Case: Change number of items

For this use case, we've also used State Transition test-technique, as we feel it makes the controlling and checking of quantities much easier to understand.

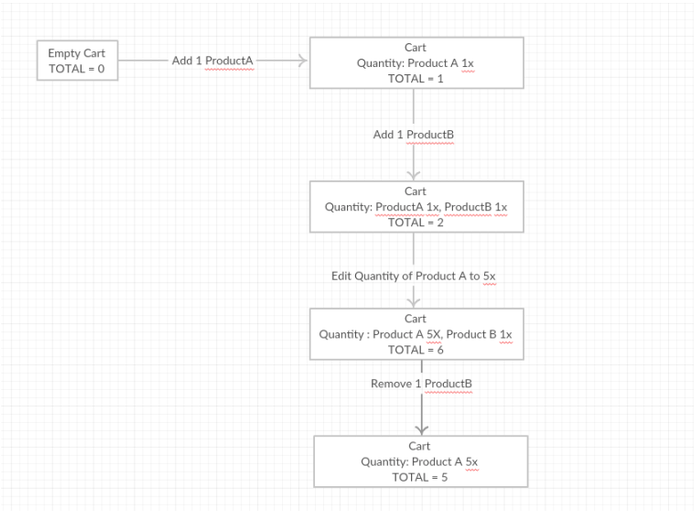


Figure 2: State Transition Diagram of editing items in cart

Table 5: List of manual tests for editing shopping cart

Test Name	Description	Steps	Pre-Condition / Post-Condition	Expected Results	ActualResults
"Test that editing the quantity of an item in shopcart is possible"	The purpose of this test is to check if the user can edit the amount of an item in the shop cart by using the drop down menu when viewing the cart.	1. User clicks on "My Cart" 2. User clicks dropdown menu below "QTY" on desired product. 3. User chooses new quantity amount	PRE-CONDITION 1. Shopping cart is not empty 2. User has navigated to website POST-CONDITION 1. Quantity of chosen item changed.	The quantity of the item should be changed to the new desired quantity given by the user.	The quantity of the item is changed.
"Test that removing an item from the shopcart is possible"	The purpose of this test is to check removing an item from the shopping cart is possible.	1. User clicks on "My Cart" 2. User clicks (X) on the right of the product the user wishes to remove.	PRE-CONDITION 1. Shopping cart is not empty 2. User has navigated to website POST-CONDITION 1. Item is removed from shopping cart	When an user clicks on the X to the right of the product, the product should be removed from the shopping cart.	The item is removed from the shopcart.

2.1.4 Use Case: Checkout & Finalize

As described in the table on page 2, We've decided to use another test technique for this use case, also an experience based testing, error guessing. Down below is a small list of possible errors and our manual tests.

- Empty fields during checkout process
  Leaving empty fields during the checkout process may lead to some problems. For example, if required empty fields are left empty, and it still gets accepted, or if empty fields cause the system to crash.

- Required field
  Testing that all required fields need to be filled out is also essential.

- Special characters
  The system might deny special characters such as æ,ø,å or any other characters that are not in the English alphabet. This will be a problem if for example a Chinese person living in one of the accepted countries orders products, and his/her name gets denied.

- Invalid input
  All fields should restrict invalid inputs based on what information is asked for. For example, characters from the English alphabet should not be accepted in the phone number field.

- Empty Shopping Cart
  Shopping cart that are empty should not be allowed to be checked out. This can quickly happen if the developer has forgotten to check if it is empty before checkout, so we'll test that.

- Input field character limits
  Testing the input field limits is important. Different persons have different names, from 1 letter to 100+. It's essential that the system accepts all possible names, while also not crashing due to a large amount of characters.

Table 6: List of manual tests for Checkout & Finalize

Test Name	Description	Steps	Pre-Condition / Post-Condition	Expected Results	ActualResults
"Test that filling out all required fields will grant successful checkout"	The purpose of this test is to see if an user leaves an optional field empty crashes the system. We assume that all required fields are filled out properly.	1. User clicks on "My Cart" 2. User clicks "CHECKOUT" 3. User only fills out mandatory fields. 4. User clicks "Continue Checkout" 5. Choose shipping method and finalize order.	PRE-CONDITION 1. Shopping cart is not empty 2. User has navigated to website POST-CONDITION 1. Order is finalized and placed.	Fields that are not mandatory should be optional to fill out, so the system should proceed to the next phase even if optional fields are empty.	Finalization of order is possible even if there are empty fields. The requirement is that the mandatory fields are filled out.
"Test to see if the system accepts special characters"	The purpose of this test is to see if special characters are accepted in fields such as name and address.	1. User clicks on "My Cart" 2. User clicks "CHECKOUT" 3. User fills out name and address with special letters such as æ, ø, å. 4. User fills out rest of mandatory fields. 5. User clicks "Continue Checkout" 6. Choose shipping method and finalize order.	PRE-CONDITION 1. Shopping cart is not empty 2. User has navigated to website POST-CONDITION 1. Order is finalized and placed.	Checkout should move into the next phase even if fields strings contain special characters.	Finalization of order is possible even if there are special characters in the input fields.
"Test that invalid inputs are not accepted during checkout"	The purpose of this test is to see if the system rejects and prevents a checkout if there are invalid inputs" Invalid inputs are for example letters in the phone number field.	1. User clicks on "My Cart" 2. User clicks "CHECKOUT" 3. User fills phone field with letters. 4. User fills out rest of mandatory fields. 5. User clicks "Continue Checkout" 6. Choose shipping method and finalize order.	PRE-CONDITION 1. Shopping cart is not empty. 2. User has navigated to website. POST-CONDITION 1. Error message is displayed	Invalid inputs should prevent the checkout process, and present the user with an error message.	Invalid input does not prevent the system from finalizing order. It is possible to write anything in phone field and still finalize order.
"Test that checking out with empty cart is not possible"	The purpose of this test is to see if the system rejects and prevents a checkout if there are no items in the shopping cart.	1. User clicks on "My Cart" 2. User clicks "CHECKOUT"	PRE-CONDITION 1. Shopping Cart is empty 2. User has navigated to website POST-CONDITION 1. Checkout is prevented	The user should not be allowed to check out if the shopping cart is empty.	User is not given the opportunity to check out if the shopping cart is empty.
"Test that the name and address fields can accept long inputs"	The purpose of this test is to see if the system is capable of handling extremely long names.	1. User clicks on "My Cart" 2. User clicks "CHECKOUT" 3. User fills out name and address with extremely long strings. 4. User fills out rest of mandatory fields. 5. User clicks "Continue Checkout" 6. Choose shipping method and finalize order.	PRE-CONDITION 1. Shopping cart is not empty 2. User has navigated to website POST-CONDITION 1. Order is finalized and placed.	The user should be allowed to finalize order even if the name and address fields are filled out with extremely long strings.	Finalization of order is possible even if the name and address fields are filled out with extremely long strings.

2.1.5 Use Case: Log Out

For the final use case, we decided to a simple state transition diagram.

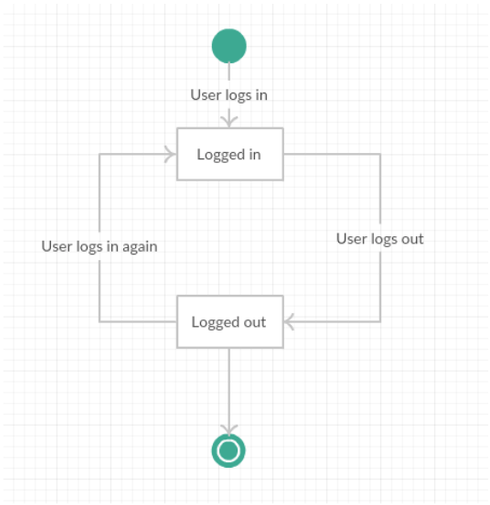


Figure 3: State Transition Diagram of logging in and out

Table 7: Manual test for logging out

Test Name	Description	Steps	Pre-Condition / Post-Condition	Expected Results	Actual Results
"Test that the user can logout successfully"	The purpose of this test is to see if an user can log out from his/her account without problems.	1. User clicks on "Sign Out"	PRE-CONDITION 1. User is logged in 2. User has navigated to webpage. POST-CONDITION 1. User is signed off	Clicking sign off should sign the user off the account.	The user is signed off his/her account.

2.2 Error finding and reporting

We found a minor issue on the Avactis website. When registering an user, if the email field is not filled out (empty field), the process will end in an internal server error.

Date: 27.04.16
Project: Avactis E-commerce Website

Programmer: Unknown **Tester**: Uy Quang Nguyen

Program/Module: Register new user **Build/Revision/Release**: Unknown

Software Environment: OSX 10.11.3, El Capitan **Hardware Environment**: MacBook Pro 15”

Status of the incident: Not treated

Number of Occurrences: 3 **Severity**:Low **Impact**: Low **Priority**: Low

Detailed Description: Registering a new account without filling out the e-mail field will result in an internal server error. See picture attached below for error message.

Expected result / Actual result: Registering a new account without filling out the e-mail field should result in an message warning the user about it, and then letting the user fill it out.

Change history: Unknown

References: Unknown (including the identity of the test case specification that revealed the problem

Assigned To: Unknown

Incident Resolution: Unknown

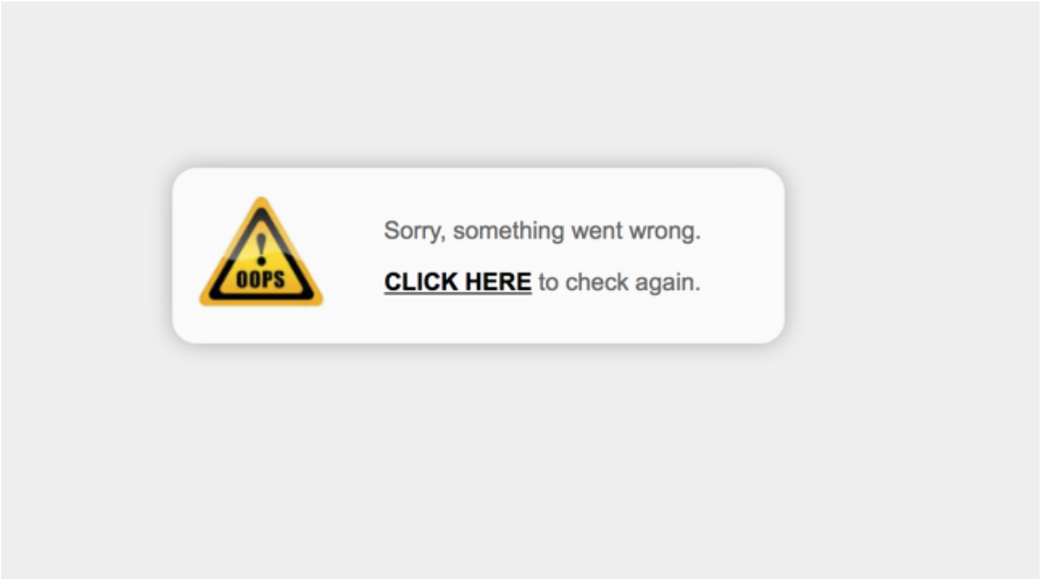


Figure 4: Picture of the internal server error

3 Requirement 2 - Automated tests and grouping (10 points)

For the list with test cases you have obtained at the previous requirement, create the corresponding automated test with the Selenium IDE plugin.
We had a kind of different take on this. On Piazza, Uzma wrote that it's okay to not automate all tests if they had low priority[7], but in an email sent to Raluca, we were told to automate all of them [8]. The reason we asked was because we had some manual tests that we, in our opinion though was important, but not relevant to the use cases given (such as checking if the user can navigate between categories). The result was that we got many test cases (maybe too many)?.

3.1 Explanation of order of tests

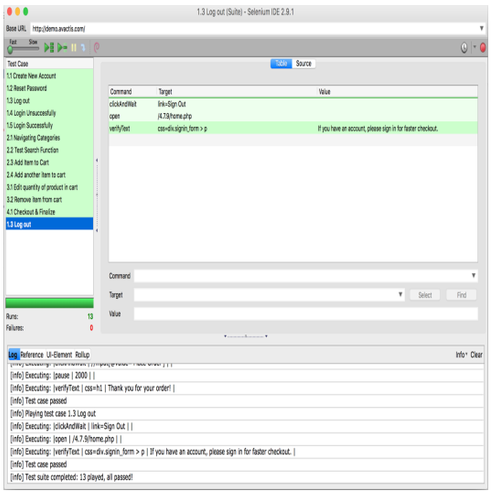
Explain the way you have ordered the tests in the automated tests script. Is the order important? Why?
The way we have ordered the tests are incredibly important. If we take a look at the multiple manual test tables above, we can clearly see that they have pre conditions such as requiring the user to be logged in during some of the steps. Doing the steps out of order will cause some, if not all of them to fail.

- 1.1 Create new account
  It is extremely important that the create new account test case is the first one to be run. It creates the password, and email variable that are used later in 1.4 and 1.5. 1.2 cant be run either without running 1.1 first, because resetting your password requires you to be logged in, which 1.1 does automatically when the account is created.
- 1.2 Reset Password
  Important that this happens after account is created, since resetting password requires user to have an account and be logged in. It also creates a new password variable which is used in 1.5.
- 1.3 Log out
  Needs to happen after 1.1, since 1.1 signs you in after creating account. "Sign out" button wont be visible if not logged in. Log out is done here to test that the new password is working, and that the old one is obsolete.
- 1.4 Login Unsuccessfully
  Needs to happen after 1.1, 1.2 and 1.3, as it uses the old password variable created in 1.1, and not the new one, and it requires you to be logged out.
- 1.5 Login Successfully
  Needs to happen after 1.1, 1.2 and 1.3, as it uses the new password variable created in 1.2, and not the old one, and it requires you to be logged out.
- 1.6 Checkout unsuccessfully
  Needs to happen before adding items to cart. It fits perfectly here, since when user logs in manually after creating account, the shopping cart should be empty, making this test pass.
- 2.1 Navigating Categories
  Doesn't really matter where this test is placed in the queue.
- 2.2 Test Search Function
  Doesn't really matter where this test is placed in the queue.
- 2.3 Add Item to Cart
  Has to be done after 1.1, and before 2.4 -¿ end. This is because 2.4 relies on there being a first item, to check that there is 2 product in cart.
- 2.4 Add another item to cart
  Has to be done after 2.4, as it checks that there are two items in cart.
- 3.1 Edit Quantity of product in cart
  Has to be done after 2.3 as it relies on the shopping cart, so the shopping cart cant be empty.
- 3.2 Remove item from cart
  Removes the item added from 2.4, so has to be done after 2.3 + 2.4.
- 4.1 Checkout & Finalize
  Important that it is done at last, because it relies on there being items in the shopping cart, and that the user is logged on for checkout.
- 1.3 Log out
  Logout has to be done either after reset password or at the end. Doing it in the middle will mess checkout and finalize up.

So basically, as seen here, the order is VERY important. There are pause commands added in some of the test cases, because sometimes the system requires time to update shopping cart etc.

[7] https://piazza.com/class/imojamlw8972iu?cid=43
[8] https://gyazo.com/170a7af03e076b600137118099d5b2b9.png

Provide screenshot(s) with the execution result of your automated test suite. Upload your automated tests to GitHub and provide the link.



(a) Screenshot of the execution of the test suite

The test suite executed, and all tests were successful. This might change based on the computer and network speed. During testing, we experienced that the tests would fail sometimes, due to the Avactis site bugging out (not logging you in automatically after an account was created, or not updating cart and requiring website refresh). The tests were done with an internet connection @ 100mbps down and up, which means that it might fail on anything below because the pause commands are timed after 100mbps. We added the 1.6 test (checkout unsuccessful) on the last day of the assignment, so it's not included in the picture, due to the avactis website crashing on the last day. Please have a look at the test cases we have uploaded, as we have commented several of them within selenium on why and what we were doing.

https://github.com/UyNG/INF3121/tree/master/PA2

4 Requirement 3 - Transitioning manual to automated tests (5 points)

4.1 Relations between manual and automated tests

Write and comment the way your manual tests and automated tests interrelate (it can be in form of a table, traceability matrix, etc): are all manual tests covered by automated tests? For this task, we'll be using a similar table to the one given as example on piazza [9]. We'll be changing it a little, since creating a table with column for every test case will be too large.

Table 8: Which manual tests are covered by which automated tests

Manual Test	**Automated Test**	**Use Case**
Test that a new user can create an account	1.1 Create New Account	"Sign-in"
Test that a registered user can login	1.5 Login Successfully	"Sign-in"
Test that a registered user can reset his password	1.2 Reset Password	"Sign-in"
Test that wrong password will grant error message	1.4 Login Unsuccessfully	"Sign-in"
Test that invalid username will grant error message	1.4 Login Unsuccessfully	"Sign-in"
Test that a user can navigate to categories from frontpage and back	2.1 Navigating Categories	"Select Products"
Test that the search function on the website is working	2.2 Test Search Function	"Select Products"
Test that the user can add products to the shopping cart	2.3 Add Item to Cart	"Select Products"
Test that the user can continue shopping after adding products to cart	2.4 Add another item to cart	"Select Products"
Test that editing the quantity of an item in shopcart is possible	3.1 Edit quantity of product in cart	"Change num of items"
Test that removing an item from shopcart is possible	3.2 Remove item from cart	"Change num of items"
Test that filling out all required fields will grant successfull checkout	4.1 Checkout & Finalize	"Checkout & Finalize"
Test to see if the system accepts special character	NONE	"Checkout & Finalize"
Test that invalid input are not accepted during checkout	NONE	"Checkout & Finalize"
Test that checking out with empty cart is not possible	1.6 Checkout Unsuccessfully	"Checkout & Finalize"
Test that the name and address fields can accept long inputs	NONE	"Checkout & Finalize"
Test that the user can logout successfully	1.3 Log Out	"Log out"

[9] https://i.gyazo.com/22549d33be1a3c16703fce471da8f262.png

Is the coverage partial or total?
Unfortunately the coverage is partial. We did not manage to cover all of the Checkout & Finalize tests. On the positive side of it, we did at least manage to check that all of the items in the use case given was working. (Everything from registering an account to checking out is possible).

For each manual test you could not automate, provide a brief explanation: why test automation was not possible in this particular case.
There were 3 tests that we could not automate. They all had the same reason for why we couldn't do it. The 3 tests that we could not automate were

"Test to see if the system accepts special character"
"Test that invalid input are not accepted during checkout"
"Test that the name and address fields can accept long inputs"

We couldn't automate these 3 tests. This is because the input fields during checkout different target value for every single checkout, so everytime we located an input field, its value would be different for the next time we checked out, so selenium wouldn't know where to input our predefined strings and values. The only reason that we managed to checkout with test 4.1 was because we didn't need to edit any of the input fields when running it, because they were already filled out by our predefined information when we created the account.

4.2 Afterword

Incredibly fun project assignment. Doing the manual test design was a little on the boring side, but using selenium was an incredibly effective method (in our opinion) to learn about automated tests. We'd just like to point out that we'd probably manage to automate the failed tests that we did not do (checkout tests) with the help that was posted on piazza, but since the task required us to write about tests that we could not automate, we thought that we didn't really need to stress about it.