Instructions for Second Coding Assignment 1DL610 HT23

1 General Instructions

Please upload your work on the git repository you created for assignment 1. This assignment is worth 10 points, points for substasks are as indicated. In case there are only $i < 4$ people in a group, please do the first i tasks. Book an appointment with the assigned TA before the final exam to present your work. **Please note it will be difficult to accommodate extensions, since the TAs will not be available later.**

2 Tasks

Task 1

Regression testing. (5 points)

- Create a regression test suite which consists of 5 tests for each function selected from the set of unit tests you have already created for assignment 1. Ensure that the suite created by each person overlaps with another person in the group on at most 2 tests per function. Eg 1,2,3,4,5, 4,5,6,7,8, 1,2,7,8,9 can be chosen from a set of tests 1,2,3,4,5,6,7,8,9,10 for a function. In addition, create 3 more tests which test the new functionality you are about to implement.
- Run the test suite and enable recording of the results in a file, say logfile1.
- After doing the implementation below, run the test suite again and record the results in logfile 2.
- Do a diff between logfile1 and logfile2. Report the difference in a text file.

Implementation: Strengthening the product system. (2 points)
At any point, the user can query and remove individual items from the cart. On removal, display remaining items in the cart. Ensure that the product is put back into the inventory.

Smoke testing. (3 points)
Create a test suite of 10 test cases which does end to end testing of the whole

system (i.e. checking if login, logout, payment, product selection etc work correctly). User inputs which are input at different stages of the program must be provided by each test case. Ensure that all control flow paths through your particular implementation are covered. Enable test logging into a logfile. Report the results in a text file.

Task 2

Regression testing. (5 points)
Details as given for Task 1.

Implementation:Strengthening the payment system. (2 points)
At the point of payment, ask the user whether he wants to use his wallet or a card. If "card" is selected, display the list of his cards and ask which card the user wants to use.

Smoke testing. (3 points)
Details as given for Task 1.

Task 3

Regression testing. (5 points)
Details as given for Task 1.

Implementation: Strengthening the user profile. (2 points)
Expand the user file with the following fields: Address, phone number, email address, credit card details (card number, date of expiry, name on card, CVV for each card). Add a module that allows the user to change these details in the file. Expand the user registration to allow entry of these additional details.

Smoke testing (3 points)
Details as given for Task 1.

Task 4

Coverage. (2 points)
Measure the coverage of the code by the unit tests your group created in assignment 1 for the function searchAndBuyProduct using pytest-cov (https://pytest-cov.readthedocs.io/en/latest/) Measure statement coverage (what fraction of lines was executed). Measure branch coverage (what fraction of branches was taken). Create a document to report your findings.

Top-down integration. (3 points)
Do top-down integration testing of searchAndBuyProduct with login, replacing all other functions with stubs you created for assignment 1. Write test cases which check all possible control flow paths within these two functions. As mentioned in the lecture, use main as your driver.

Bottom-up integration. (5 points)
Do bottom-up integration testing of checkoutAndPayment function by creat-

ing a cluster of all functions which are used by it along with itself. Use a driver to provide $login\_info$. Write 10 different test cases to cover as much of the control flow as possible.

3 Suggestions regarding Control Flow Graph

You will need to draw the CFG of the program to come up with tests. You could do it by hand, or use some existing software (such as pycfg, as explained here: https://medium.com/@vinoothna.kinnera/creating-control-flow-graphs-using-pycfg-ba84311ca59) or staticfg (https://pypi.org/project/staticfg/).