Unit testing of the checkout\_and\_payment function

The objective of the checkout\_and\_payment function is to get user input for either selecting a product by its number to add to their cart, display available products for purchase, checking their cart or logging out.

The checkout\_and\_payment(login\_info) function takes as input a dictionary containing the keys "username", "password" and "wallet". Thus, out of {int,float,string,list}, all are invalid types. We create one test for each of these and assert that the checkout\_and\_payment function throws a TypeError exception.

We now turn to the valid inputs. The input domain can be divided into five equivalence classes (EC):

1. user selects to logout without selecting to add a product to their cart, checking their cart, displaying available products for purchase or giving an invalid input,
2. user selects a product to add to their cart without selecting to check their cart, displaying available products for purchase or giving an invalid input,
3. user selects to check their cart without without selecting to display available products for purchase or giving an invalid input,
4. user selects to display available products for purchase without giving an invalid input, and
5. user selects an invalid input.

The coverage criteria we will use are:

For each EC, we need to test at least one input.

This leads to the following test requirements:

R1. If input is in EC1, checkout\_and\_payment should call the logout function and finish with exit code 0.

R2. If input is in EC2, checkout\_and\_payment should call the functions get\_product\_unit and add\_item, and print “x added to your cart.”, where x is the name of the selected product, if the selected product is in stock. Otherwise it should print “Sorry, x is out of stock.”.

R3. If input is in EC3, checkout\_and\_payment should call the check\_cart function, and if the user selects to check out their cart, it should also call the functions UserDataManager.load\_users and UserDataManager.save\_users.

R4. If input is in EC4, checkout\_and\_payment should call the display\_products\_available\_for\_purchase function.

R5. If input is in EC5, checkout\_and\_payment should print “Invalid input. Please try again.”

These lead to the corresponding test cases, where login\_info = {‘username’: ’user’, ‘password’: ‘P@ssword’, ‘wallet’: 100.0}:

TC1. Input = login\_info, choice =’l’

TC2. Input = login\_info, choice=’l’, ‘l’

TC3. Input = login\_info, choice=’1’, ‘l’

TC4. Input = login\_info, choice=’1’, ‘1’, ‘l’

TC5. Input = login\_info, choice=’2’, ‘l’

TC6. Input = login\_info, choice=’2’, ‘2’, ‘l’

TC7. Input = login\_info, choice=’1’, ‘2’, ‘l’

TC8. Input = login\_info, choice=’2’, ‘1’, ‘l’

TC9. Input = login\_info, choice=’c’, ‘l’, check\_cart returns True

TC10. Input = login\_info, choice=’1’, ‘c’, ‘l’, check\_cart returns True

TC11. Input = login\_info, choice=’1’, ‘1’, ‘c’, ‘l’, check\_cart returns True

TC12. Input = login\_info, choice=’c’, ‘l’, check\_cart returns False

TC13. Input = login\_info, choice=‘1’, ’c’, ‘l’, check\_cart returns False

TC14. Input = login\_info, choice=‘1’, ‘1’, ’c’, ‘l’, check\_cart returns False

TC15. Input = login\_info, choice=‘1’, ’c’, ‘1’, ‘c’ ‘l’, check\_cart=False, True,

TC16. Input = login\_info, choice=’d’, ‘l’

TC17. Input = login\_info, choice=‘d’, ‘1’, ’c’, ‘l’, check\_cart=True,

TC18. Input = login\_info, choice=‘invalid’, ‘l’,

TC19. Input = login\_info, choice=‘0’, ‘l’,

TC20. Input = login\_info, choice=‘3’, ‘l’

In order to be able to execute all test cases, the fixtures login\_info, global\_products\_stub1, global\_cart\_stub1, user\_stub1 were written. In order to be able to execute all test cases except TC2, the fixture logout\_stub1 was written. In order to be able to execute TC2, the fixture logout\_stub2 was written. In order to be able to execute TC16 and TC17, the fixture display\_products\_available\_for\_purchase\_stub1 was written. In order to be able to execute TC9, TC10, TC11, TC12, TC13, TC14, TC15 and TC17, the fixtures load\_users\_stub1 and save\_users\_stub1 were written. In order to be able to execute TC9, TC10 and TC11, the fixture check\_cart\_stub1 was written. In order to be able to execute TC12, TC13, TC14 and TC17, the fixture check\_cart\_stub2 was written. In order to be able to execute TC15, the fixture check\_cart\_stub3 was written. In order to be able to execute TC1, the fixture input\_stub1 was written. In order to be able to execute TC2, the fixture input\_stub2 was written. In order to be able to execute TC3, the fixture input\_stub3 was written. In order to be able to execute TC4, the fixture input\_stub4 was written. In order to be able to execute TC5, the fixture input\_stub5 was written. In order to be able to execute TC6, the fixture input\_stub6 was written. In order to be able to execute TC7, the fixture input\_stub7 was written. In order to be able to execute TC8, the fixture input\_stub8 was written. In order to be able to execute TC9 and TC12, the fixture input\_stub9 was written. In order to be able to execute TC10 and TC13, the fixture input\_stub10 was written. In order to be able to execute TC11 and TC14, the fixture input\_stub11 was written. In order to be able to execute TC15, the fixture input\_stub12 was written. In order to be able to execute TC16, the fixture input\_stub13 was written.

In order to be able to execute TC17, the fixture input\_stub14 was written. In order to be able to execute TC18, the fixture input\_stub15 was written. In order to be able to execute TC19, the fixture input\_stub16 was written. In order to be able to execute TC20, the fixture input\_stub17 was written.