Tampere University   Unit of Computing Sciences
COMP.SE.610 Software Engineering Project 1
COMP.SE.620 Software Engineering Project 2

G16

NOKIA Connecting Colleagues

Test Report

Contents

1 Introduction

1.1 Purpose and scope of document

In this project report we are going to present the final test plan on an open-source project. The project is developed using Joomla plugin, Python, HTML5 and CSS. We chose several effective tools to complete the testing and we are going to describe briefly about those tools in this report. As some of the tools are not familiar with us, we concentrated on learning them in depth. Hence, we mentioned our learning platforms also in this report. After that, we described our test cases and testing methods. In the end, we mentioned the technique of our test case documentation. Our goal is to perform all the testing approach that we mentioned in this report.

1.2 Product and environment

The product is based on the ‘Nokia Connecting Colleagues Initiative’ project where Nokia proposed an online platform for the internal stuff to connect with each other based on their mutual preferences. This application encourages the creation of social links, sharing of ideas, and participation in shared activities. The project environment consists of Joomla plugin, a python implementation and front-end application using HTML & CSS. Our targeted testing platform is the python implementation and the overall application including the database, user interface and Joomla components.

1.3 Project constraints related to testing

We proceed with unit testing approach for the Python implementation and system testing approach for the overall application. From the customer’s end there is not any specific protocols or issues regarding the testing method. Our targeted platform to conduct the testing process are as below:

- Mocha Tests – Python unit testing
- Selenium IDE – System testing

The project is free and open source licensed under GPL v2. Customer’s demand is to develop the application using suitable platforms. So, from that point of view, there is no major constraints related to testing.

1.4 Definitions, abbreviations and acronyms

UI- User interface
User/Customer- Nokia employee
SoC- System-on-Chip

2 Testing process

2.1 General approach

Initially the testing was performed module wise in the developers' environment. Each responsible developers performed manual testing in the front end, back end and data-base side. Matching algorithm testing was performed in the Joomla admin side. A sep-arate python file was created to perform the unit test. There were total 8 unit test cases developed to check form_groups () function and calculate_group_spots() function. We have used PEP8 code convention to form the unit test cases.

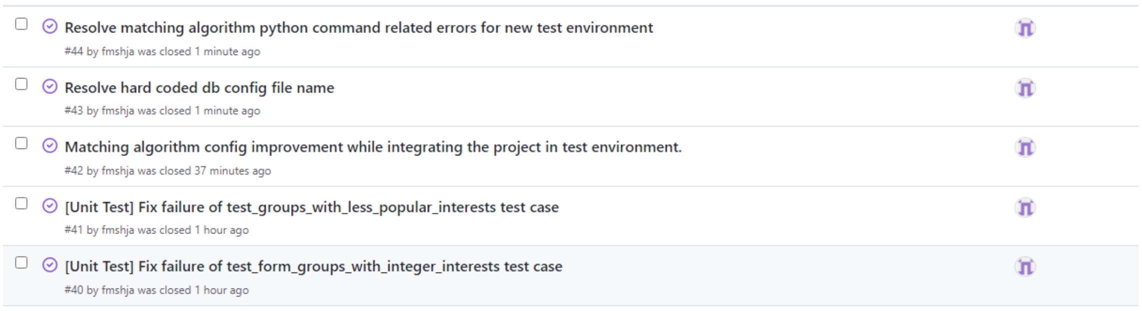
2.2 Test schedule

The test engineer started to develop the unit test from week 47 as long as the initial structure of the matching algorithm was prepared. As the unit test cases were prepared at the initial stage, two major bugs were quickly found within a week. The developer analysed and the bugs were resolved within week 48. In week 49, the test engineer started to perform integration testing by setting up the whole environment in the test environment. A configuration related error occurred before executing the matching algo-rithm in new test environment. There was also another bug found regarding the db con-nection, The developer resolved the hard coded part in the config file.

2.3 Test documentation

Our developed unit test generated test results with detailed failure logs while executing the test cases. Results can be found in section 4.2. Integration testing results and auto-mation test results can be found in section 4.3 and 4.4. Followings are the version con-trol bugs list:

Closed Bugs:


Resolve matching algorithm python command related errors for new test environment
#44 by fmshja was closed 1 minute ago

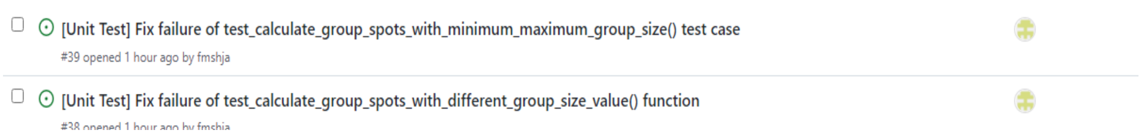
Resolve hard coded db config file name
#43 by fmshja was closed 1 minute ago

Matching algorithm config improvement while integrating the project in test environment.
#42 by fmshja was closed 37 minutes ago

[Unit Test] Fix failure of test_groups_with_less_popular_interests test case
#41 by fmshja was closed 1 hour ago

[Unit Test] Fix failure of test_form_groups_with_integer_interests test case
#40 by fmshja was closed 1 hour ago

Open Bugs:


[Unit Test] Fix failure of test_calculate_group_spots_with_minimum_maximum_group_size() test case
#39 opened 1 hour ago by fmshja

[Unit Test] Fix failure of test_calculate_group_spots_with_different_group_size_value() function
#38 opened 1 hour ago by fmshja

3 Testing Tools

We have used Python 3.9, Pytest to perform the unit test. For the system testing, we have used Selenium IDE and generated test results for end to end scenarios.

4 Test cases and results

4.1 Test results

After each module and component wise development, front developers performed the testing in the dev environment specific resolutions. Algorithm developer tested the matching algorithm with one set of data and later took the help of unit test to verify its special test cases. Database was separately tested in the dev environment and it was initially integrated with the front end and tested if the database can communicate properly with the front end. When the unit tests generated bugs, the developer resolved those issues immediately. Finally, the whole project has been integrated in the test environment including the algorithm in the admin side. There was some configuration related and hard coded bugs found while integrating the project. Then after resolving these failures, an integration test was performed and then the whole team performed some exploratory testing. The test engineer developed a system test using selenium ide after the bugs got fixed.

4.2 Unit testing

The following test cases is considered:

- **form_groups:** This function is used to form a group according to the users' interests.
  Definition of the arguments are given below:
  - ✓ min_group_size (int): The smallest allowed group size
  - ✓ group_size (int): The desired group size

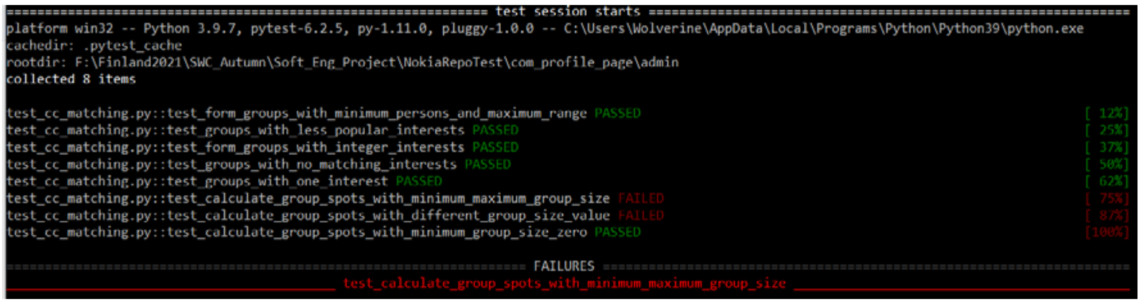
- ✓ users_to_interests (dict [UserId, set [Interest]]): A mapping from users to their interests
- ✓ interests_to_users (dict [Interest, set [UserId]]): A mapping from interests to their users
- ✓ old_groups (set [Group]): The previous list of formed groups

- **calculate_group_spots:** This calculates how many groups spots each interest should have. The total amount of spots will be the same as `number_of_users`. **Definition of the arguments are given below:**
  - ✓ min_group_size (int): The smallest allowed group size
  - ✓ group_size (int): The desired group size
  - ✓ number_of_users (int): The total amount of users is being matched
  - ✓ interests_to_users (dict [Interest, set [UserId]]): A mapping from interests to their users

Functions won't be tested:

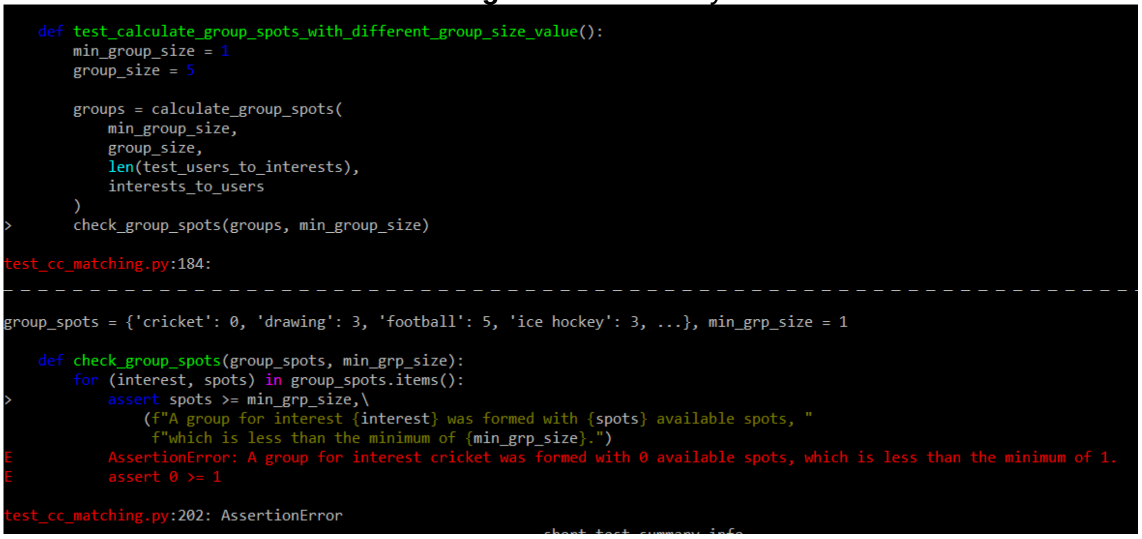
- **hopcroft_karp:** For the time bound and being established algorithm, we have kept this function apart from our unit testing. This will be tested in future if necessary.
- **dfs_augmenting_path:** This function is used as a helper function for hopcroft_karp algorithm. So, performing unit testing is not necessary here.

Among the eight unit tests, 4 test cases failed initially in total. We have created separate four issues for that and two of them were resolved. There are two open issues, which are in progress to be resolved. Following is the test results:



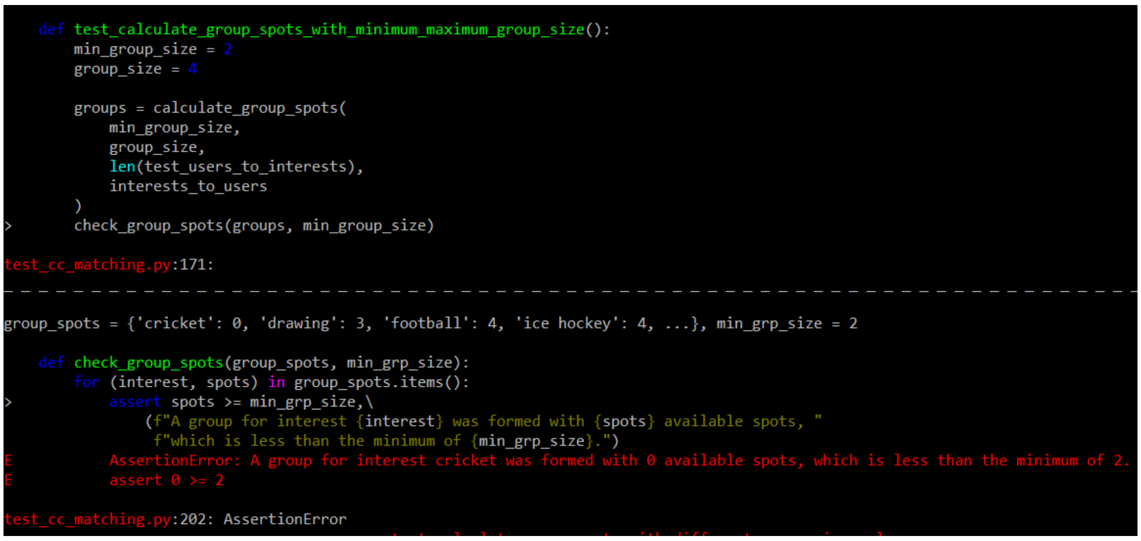
```
================================================================== test session starts ===================================================================
platform win32 -- Python 3.9.7, pytest-6.2.5, py-1.11.0, pluggy-1.0.0 -- C:\Users\Wolverine\AppData\Local\Programs\Python\Python39\python.exe
cachedir: .pytest_cache
rootdir: F:\Finland2021\SWC_Autumn\Soft_Eng_Project\NokiaRepoTest\com_profile_page\admin
collected 8 items

test_cc_matching.py::test_form_groups_with_minimum_persons_and_maximum_range PASSED                                                              [ 12%]
test_cc_matching.py::test_groups_with_less_popular_interests PASSED                                                                              [ 25%]
test_cc_matching.py::test_form_groups_with_integer_interests PASSED                                                                              [ 37%]
test_cc_matching.py::test_groups_with_no_matching_interests PASSED                                                                               [ 50%]
test_cc_matching.py::test_groups_with_one_interest PASSED                                                                                        [ 62%]
test_cc_matching.py::test_calculate_group_spots_with_minimum_maximum_group_size FAILED                                                           [ 75%]
test_cc_matching.py::test_calculate_group_spots_with_different_group_size_value FAILED                                                           [ 87%]
test_cc_matching.py::test_calculate_group_spots_with_minimum_group_size_zero PASSED                                                              [100%]

======================================================================== FAILURES ========================================================================
_____________________________________________ test_calculate_group_spots_with_minimum_maximum_group_size _____________________________________________
```

Fig1: Test Summary



```
    def test_calculate_group_spots_with_different_group_size_value():
        min_group_size = 1
        group_size = 5

        groups = calculate_group_spots(
            min_group_size,
            group_size,
            len(test_users_to_interests),
            interests_to_users
        )
>       check_group_spots(groups, min_group_size)

test_cc_matching.py:184:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

group_spots = {'cricket': 0, 'drawing': 3, 'football': 5, 'ice hockey': 3, ...}, min_grp_size = 1

    def check_group_spots(group_spots, min_grp_size):
        for (interest, spots) in group_spots.items():
>           assert spots >= min_grp_size,\
                (f"A group for interest {interest} was formed with {spots} available spots, "
                 f"which is less than the minimum of {min_grp_size}.")
E           AssertionError: A group for interest cricket was formed with 0 available spots, which is less than the minimum of 1.
E           assert 0 >= 1

test_cc_matching.py:202: AssertionError
```

Fig2: Test Detail1



```
    def test_calculate_group_spots_with_minimum_maximum_group_size():
        min_group_size = 2
        group_size = 4

        groups = calculate_group_spots(
            min_group_size,
            group_size,
            len(test_users_to_interests),
            interests_to_users
        )
>       check_group_spots(groups, min_group_size)

test_cc_matching.py:171:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

group_spots = {'cricket': 0, 'drawing': 3, 'football': 4, 'ice hockey': 4, ...}, min_grp_size = 2

    def check_group_spots(group_spots, min_grp_size):
        for (interest, spots) in group_spots.items():
>           assert spots >= min_grp_size,\
                (f"A group for interest {interest} was formed with {spots} available spots, "
                 f"which is less than the minimum of {min_grp_size}.")
E           AssertionError: A group for interest cricket was formed with 0 available spots, which is less than the minimum of 2.
E           assert 0 >= 2

test_cc_matching.py:202: AssertionError
```

Fig3: Test Detail 2

4.3 Integration testing

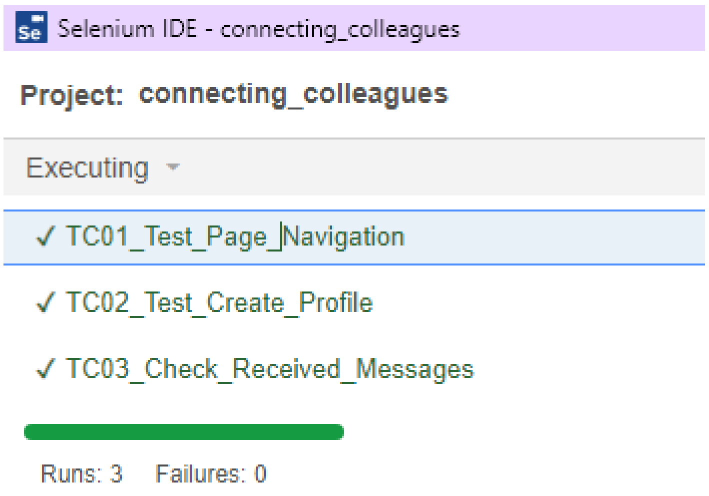
Following test cases are performed during integration tests:

#SL	Test Objective	Expected Behavior	Old Status	Severity	Priority	Actual Behavior	After Bug Fix Status
1	Navigate to all the pages	User should be able to navigate all pages	**Passed**	N/A	N/A	User is able to navigate all pages	**Passed**
2	Login with valid credentials	Successful login should be performed	**Passed**	N/A	N/A	Successful login is performed	**Passed**
3	Create new profile	User should be to create profile successfully	**Passed**	N/A	N/A	User is able to create profile successfully	**Passed**
4	Check multiple interest can be added	Multiple interests should be added	**Passed**	N/A	N/A	Multiple interests can be added	**Passed**
5	Check minimum interest selection	There should be minimum interests selection limit	**Passed**	N/A	N/A	There is minimum interests selection limit	**Passed**
6	Check users and users interest in database	Users and interests should be displayed in MySQL database	**Passed**	N/A	N/A	Users and interests are displayed in MySQL database	**Passed**
7	Check logout	Logout from the application should be performed properly	**Passed**	N/A	N/A	Logout from the application is performed properly	**Passed**
8	Check db connection	Config file should work properly	**Failed**	Critical	High	Hard coded item was found	**Passed**
9	Check if Start Algorithm button works	Match found or not found info should be displayed	**Failed**	Critical	High	Python command was not found displayed	**Passed**

10	Check re-ceived mes-sages	Message should be received after matching	**Un-tested**	N/A	N/A	Message is dis-played	**Passed**

4.4 System testing

We have automated three test cases for the system tests. Following is the test results:


Selenium IDE - connecting_colleagues

Project: connecting_colleagues

Executing

✓ TC01_Test_Page_Navigation

✓ TC02_Test_Create_Profile

✓ TC03_Check_Received_Messages

Runs: 3 Failures: 0

Fig: System Test Result