COSC 310 - 001

Software Engineering

2021 Winter Term 2

Automated Unit Testing Documentation

Thomas Buchholz

Ryan Grant

Vinu Ihalagamage

Tanner Wright

Dima Zhuravel

Table of Contents

1.0 Automated Unit Testing bot.test.py

1.1 testNonExistingFile

```
def testNonExistingFile(self):
    with self.assertRaises(SystemExit) as ex:
        Bot("file.json")
        self.assertEqual(ex.exception, "Error")
```

This test case is used to ensure that if the file does not exist then the expected error is catched and handled.

1.2 testEmptyBotConstructor

```
def testEmptyBotConstructor(self):
    with self.assertRaises(TypeError) as ex:
        Bot()
        self.assertEqual(ex.exception, "Error")
```

This test case is testing to see if the bot has an empty constructor, if so an error is to be expected by the system.

1.3 testEmptyUsername

```
def testEmptyUsername(self):
    self.assertEqual(Bot("data.json").getUserName(), -1)
```

Testing to see if the username the user provides is empty than a -1 should be expected

1.4 testNonEmptyUsername

```
def testNonEmptyUsername(self):
    bot = Bot("data.json")
    bot.setUserName("NewUser")
    self.assertEqual(bot.getUserName(), "NewUser")
```

We are testing that the user has entered a non-empty username and expecting that the NewUser variable and the bot.getUserName method match

1.5 testNonEmptySetUsernameOutput

```
def testNonEmptySetUsernameOutput(self):
    bot = Bot("data.json")

    username = "NewUser"

    self.assertEqual(bot.setUserName(username), f"> Bot: Hello {username}.\nI am glad to have you here today, How are you feeling?\n\n")
```

System will display a message with the username of NewUser

1.6 testEmptyArgumentSetUsernameOutput

```
def testEmptyArgumentSetUsernameOutput(self):
    bot = Bot("data.json")
    with self.assertRaises(TypeError) as ex:
        bot.setUserName()
        self.assertEqual(ex.exception, "Error")
```

Testing if the user has entered an empty argument as input. If so an error is to be expected from the system.

1.7 testSpaceInputSetUsernameOutput

```
def testSpaceInputSetUsernameOutput(self):
    bot = Bot("data.json")
    username = "      "
    self.assertEqual(bot.setUserName(username), -1)
```

This test case ensures that if the user provides only spaces, then a -1 should be expected from the system.

1.8 testEmptyInputSetUsernameOutput

```
def testEmptyInputSetUsernameOutput(self):
    bot = Bot("data.json")
    self.assertEqual(bot.setUserName(""), -1)
```

This test case checks if the user has provided an empty input. If so, than a -1 should be expected from the system

1.9 testGetSentimentPolarityScoreOfNegativeResponse

```
def testGetSentimentPolarityScoreOfNegativeResponse(self):
    bot = Bot("data.json")
    self.assertNotEqual(bot.getSentimentPolarityScore(["tired", "sick", "anxiety"]), 1)
```

Test case which tests a sentiment polarity score of negative user responses which checks using the sentimentPolatiryScore. If they do not match then it will not return a 1.

1.10 testGetSentimentPolarityScoreOfPositiveResponse

```
def testGetSentimentPolarityScoreOfPositiveResponse(self):
    bot = Bot("data.json")
    self.assertEqual((bot.getSentimentPolarityScore(["lovely", "good", "well"])).get('pos'), 1)
```

Test case which tests a sentiment polarity score of positive user responses which checks using the sentimentPolatiryScore. If it is, it will return a 1 and then it will be checked using the assertEqual method if they match.

1.11 testGetSentimentPolarityScoreOfNeutralResponse

```
def testGetSentimentPolarityScoreOfNeutralResponse(self):
    bot = Bot("data.json")
    self.assertEqual((bot.getSentimentPolarityScore(["disinterested", "inactive"])).get('neu'), 1)
```

Test case which tests a sentiment polarity score of neutral user responses which checks using the sentimentPolatiryScore. If it is, it will return a 1 and then it will be checked using the assertEqual method if they match.

1.12 testGetSentimentPolarityScoreOfMixedResponse

```
def testGetSentimentPolarityScoreOfMixedResponse(self):
    bot = Bot("data.json")
    self.assertNotEqual((bot.getSentimentPolarityScore(["disinterested", "good", "sick"])).get('compound'), 1)
```

Test case which tests a sentiment polarity score of mixed user responses which checks using the sentimentPolatiryScore. If it is, it will return anything but 0 and then it will be checked using the assertEqual method

1.13 testGetWordNetSynsetResultWithNonEmptyResponse

```
def testGetWordNetSynsetResultWithNonEmptyResponse(self):
    bot = Bot("data.json")
    self.assertEqual(bot.getWordNetSynsetResult("exhausted")[1], "exhaust")
```

Test case for ensuring that the method will return a list of values such that the element at index 1 matches our expected output.

1.14 testGetWordNetSynsetResultWithEmptyResponse

```
def testGetWordNetSynsetResultWithEmptyResponse(self):
    bot = Bot("data.json")
    self.assertEqual(bot.getWordNetSynsetResult(""), -1)
```

Test case for getting the results from NetSynset with an empty response. If so, expect a result of -1.

1.15 testGetPosTagWithQuestionResponse

```
def testGetPosTagWithQuestionResponse(self):
    bot = Bot("data.json")
    self.assertEqual(bot.getPosTag({ 'pos': ['VB', 'WP', 'WRB', 'WDT'] }, ["what", "should", "i", "do", "?"]), ['VB', 'WP'])
```

System tests getting the PosTag with a question response.

1.16 testGetPosTagWithModalResponse

```
def testGetPosTagWithModalResponse(self):
    bot = Bot("data.json")
    self.assertEqual(bot.getPosTag({ 'pos': ['VB', 'WP', 'WRB', 'MD'] }, ["could", "you", "help", "me", "?"]), ['VB', 'MD'])
```

Test case for getting the PosTag with modal response.

2.0 Automated Unit Testing fileReader.test.py

2.1 testExisitingData

```
def testExistingData(self):
    fileReader = FileReader("data.json")
    self.assertEqual(len(fileReader.getFileContent()), 2)
```

Testing if the data does exist then reading the data and ensuring it has the correct amount of information in the file.

2.2 testExistingDataConditions

```
def testExistingDataConditions(self):
    fileReader = FileReader("data.json")
    self.assertEqual(len(fileReader.getFileContent()['conditions']), 5)
```

Testing each of the conditions from the data.json file and getting the content of the conditions and ensuring it has the correct amount of conditions.

2.3 testExistingDataNodes

```
def testExistingDataNodes(self):
    fileReader = FileReader("data.json")
    self.assertEqual(len(fileReader.getFileContent()['nodes']), 50)
```

Testing the existing data nodes from data.json file and ensuring it has the correct amount of nodes.