과제

First Sprint

장유선

2023.09.08

1. 문제 정의

설계한 내용을 TDD 방식으로 구현

로그인

|  |
| --- |
| 관리자 계정: admin / admin  사용자 계정: user01 / password  위 2개 계정 이외에는 로그인 실패  로그인 시 관리자는 admin, 사용자 계정은 user를 반환 |

POI 편집

|  |
| --- |
| 단일 POI GET 요청 시 아래의 데이터 리스트 반환  ID: 1  NAME: 테스트용  이외의 ID가 입력 되는 경우는 None 반환 |

POI 추가 시

|  |
| --- |
| Name이 None이 아닌 경우 ID는 2를 반환  Name이 None인 경우 ID는 None 반환 |

POI 편집 시

|  |
| --- |
| ID가 1이고, NAME이 None이 아닌 경우 True 반환  이외의 경우에는 False 반환 |

POI 삭제 시

|  |
| --- |
| ID가 1인 경우 True 반환  이외의 경우에는 False 반환 |

2. POI.py

class POI:

    def \_\_init\_\_(self):

        self.poi\_list = {

            1: "테스트용",

        }

    def login(self, username, password):

        if username == "admin" and password == "admin":

            return "admin"

        elif username == "user01" and password == "password":

            return "user"

        else:

            return None

    def get\_poi(self, poi\_id):

        return self.poi\_list.get(poi\_id)

    def add\_poi(self, name):

        if name is not None:

            new\_id = 2

            return new\_id

        else:

            return None

    def edit\_poi(self, poi\_id, name):

        if poi\_id == 1 and name is not None:

            return True

        else:

            return False

    def delete\_poi(self, poi\_id):

        if poi\_id == 1:

            return True

        else:

            return False

3. test\_POI.py

import unittest

from unittest import TestCase

from POI import \*

class TestPOI(TestCase):

    def setUp(self):

        self.poi = POI()

    def test\_login\_admin(self):

        self.assertEqual(self.poi.login("admin", "admin"), "admin")

    def test\_login\_user(self):

        self.assertEqual(self.poi.login("user01", "password"), "user")

    def test\_login\_fail(self):

        self.assertIsNone(self.poi.login("me", "me"))

    def test\_get\_poi\_existing(self):

        self.assertEqual(self.poi.get\_poi(1), "테스트용")

    def test\_get\_poi\_non\_existing(self):

        self.assertIsNone(self.poi.get\_poi(2))

    def test\_add\_poi\_with\_name(self):

        self.assertEqual(self.poi.add\_poi("마라탕집"), 2)

    def test\_add\_poi\_without\_name(self):

        self.assertIsNone(self.poi.add\_poi(None))

    def test\_edit\_poi\_valid(self):

        self.assertTrue(self.poi.edit\_poi(1, "POI 수정"))

    def test\_edit\_poi\_invalid(self):

        self.assertFalse(self.poi.edit\_poi(2, "POI 수정"))

    def test\_delete\_poi\_valid(self):

        self.assertTrue(self.poi.delete\_poi(1))

    def test\_delete\_poi\_invalid(self):

        self.assertFalse(self.poi.delete\_poi(2))

if \_\_name\_\_ == "\_\_main\_\_":

    unittest.main()

3. 결과

Total number of tests expected to run: 11

Total number of tests run: 11

Total number of tests passed: 11

Total number of tests failed: 0

Total number of tests failed with errors: 0

Total number of tests skipped: 0

Finished running tests!

4. 결과 화면



