

System testing 2.0 - 1조

test.py를 통한 Test 진행

- python의 unittest 활용
- 최종 발표 시 프로젝트 파일에 대한 testing 실시
- oauth 추가 이외 큰 변동사항 없어 system testing 1.0과 같은 코드 사용

```

test.py
C:\Users> CSS > Downloads > swsec_team1 > test.py
1  import os
2  import tempfile
3  import unittest
4
5  from flask import Flask
6  from swsec_team1 import app, db
7  from swsec_team1.Models import *
8  from num2words import num2words
9  from datetime import datetime
10 from werkzeug.security import check_password_hash, generate_password_hash
11 from flask_sqlalchemy import SQLAlchemy
12 from sqlalchemy import create_engine, MetaData
13 from sqlalchemy.orm import sessionmaker
14
15
16 def test_populate_database():
17     # assert User.query.get(1).role.name != None
18     # assert db.session.query(User).filter(id=1).first() != None
19     assert Menu.query.filter_by(id=1).first().name == 'AAAA'
20
21
22 class GregTestCase(unittest.TestCase):
23     def create_users(self):
24         # raise ValueError(User.query.all())
25         new_user = User(username='Xxxx Xxxx', email='xxx@xxx.xxx', password='xxxx', access=0)
26         new_user.password = generate_password_hash(new_user.password)
27         db.session.add(new_user)
28         new_user = User(username='Zzzz Zzzz', email='zzz@zzz.zzz', password='zzzz', access=1)
29         new_user.password = generate_password_hash(new_user.password)
30         db.session.add(new_user)
31         db.session.commit()
32
33     def create_menus(self):
34         menus = ['AAAA', 'BBBB', 'CCCC']
35         for i, x in enumerate(menus):
36             nb = Menu()
37             nb.name = x
38             nb.price = i + 1
39             nb.ingredients = 'xxx'
40             nb.uploaded_date = datetime.now()
41             nb.user = User(username='test', email='test@test.com', password=generate_password_hash('test'),
42                             db.session.add(nb)
43             db.session.commit()
44
45     def setUp(self):
46         # self.db_gd, app.config['DATABASE'] = tempfile.mkstemp()
47         app.config['TESTING'] = True
48         app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:/// ' + app.config['DATABASE']
49         basedir = os.path.abspath(os.path.dirname(__file__))
50         app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:/// ' + \
51             os.path.join(basedir, 'test.db')
52         self.app = app.test_client()
53         # db = SQLAlchemy(app)
54         global engine
55         engine = create_engine(app.config['SQLALCHEMY_DATABASE_URI'])
56         Session = sessionmaker(bind=engine)
57         db.session = Session()
58         db.create_all()
59         # self.app.testing = True
60         self.create_users()
61         self.create_menus()
62
63     # with app.app_context():
64     #     db.create_all()
65     #     self.create_roles()
66     #     self.create_users()
67     #     self.create_buildings()
68
69     def tearDown(self):
70         # with app.app_context():
71         #     with app.request_context():
72         #         db.drop_all()
73         #         os.close(self.db_gd)
74         basedir = os.path.abspath(os.path.dirname(__file__))
75         os.unlink(os.path.join(basedir, 'test.db'))
76
77     # Login/Logout
78     def login(self, username, password):
79         return self.app.post('/auth/login', data=dict(
80             username=username,
81             password=password,
82             follow_redirects=True)
83
84     def logout(self):
85         return self.app.get('/auth/logout', follow_redirects=True)
86
87     def register_visitor(self, name, email):
88         return self.app.get('/greg/register_visitor', data=dict(
89             name=name,
90             email=email,
91             follow_redirects=True)
92

```

```

92
93     # Appointments
94     '''def test_scheduling(self,user):
95         #rv = self.login('xxx','admin')
96         pass'''
97
98
99     ...
100
101     def test_login_logout(self):
102         rv = self.login('xxx@gmail.com','admin')
103         assert 'Welcome' in rv.data
104         rv = self.logout()
105         assert 'sign in' in rv.data
106         rv = self.login('xxx@gmail.com','admin')
107         assert 'invalid' in rv.data'''
108
109 if __name__ == '__main__':
110     unittest.main()

```

결과

The screenshot shows the PyCharm IDE interface. The main editor displays a Python test file named `test.py` with the following content:

```

1  import os
2  import tempfile
3  import unittest
4
5  from flask import Flask
6  from swsec_team1 import app, db
7  from swsec_team1.Models import *
8  from flask_breadcrumbs import breadcrumbs
9  from datetime import datetime
10 from werkzeug.security import check_password_hash, generate_password_hash
11 from flask_sqlalchemy import SQLAlchemy
12 from sqlalchemy import create_engine, MetaData
13 from sqlalchemy.orm import sessionmaker
14
15 def test_populate_database():
16     # assert User.query.get(1).role.name != None
17     # assert db.session.query(User).filter(id=1).first() != None
18     # assert Menu.query.filter_by(id=1).first().name == 'AAA'
19
20
21 class GregTestCase(unittest.TestCase):
22     def setUp(self):
23         # False ValueError(User.query.get(1).role.name != None)
24         new_user = User(username='xxx', password='xxxx', access=0)
25         new_user.password = generate_password_hash(new_user.password)
26         db.session.add(new_user)
27
28
29 if __name__ == '__main__':
30     unittest.main()

```

The Run window at the bottom shows the execution results:

```

Run: test.py
-----
Ran 0 tests in 0.000s
OK
Process finished with exit code 0

```

The status bar at the bottom indicates the environment: Material Oceanic, 22:39, LF, UTF-8, 4 spaces, Python 3.9.