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

```python
import os
import tempfile
import unittest

from flask import Flask
from swsec_team1 import app, db
from swsec_team1.Models import *
from num2words import num2words
from datetime import datetime
from werkzeug.security import check_password_hash, generate_password_hash
from flask_sqlalchemy import SQLAlchemy
from sqlalchemy import create_engine, MetaData
from sqlalchemy.orm import sessionmaker


def test_populate_database():
    # assert User.query.get(1).role.name != None
    # assert db.session.query(User).filter(id==1).first() != None
    assert Menu.query.filter_by(id=1).first().name == 'AAAA'


class GregTestCase(unittest.TestCase):
    def create_users(self):
        # raise ValueError(User.query.all())
        new_user = User(username='Xxxx Xxxx', email='xxx@xxx.xxx', password='xxxx', access=0)
        new_user.password = generate_password_hash(new_user.password)
        db.session.add(new_user)
        new_user = User(username='Zzzz Zzzz', email='zzz@zzz.zzz', password='zzzz', access=1)
        new_user.password = generate_password_hash(new_user.password)
        db.session.add(new_user)
        db.session.commit()

    def create_menus(self):
        menus = ['AAAA', 'BBBB', 'CCCC']
        for i, x in enumerate(menus):
            nb = Menu()
            nb.name = x
            nb.price = i + 1
            nb.ingredients = 'xxx'
            nb.uploaded_date = datetime.now()
            nb.user = User(username='test', email='test@test.com', password=generate_password_hash('test'),
            db.session.add(nb)
        db.session.commit()

    def setUp(self):
        # self.db_gd, app.config['DATABASE'] = tempfile.mkstemp()
        app.config['TESTING'] = True
        # app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///' + app.config['DATABASE']
        basedir = os.path.abspath(os.path.dirname(__file__))
        app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///' + \
                                                os.path.join(basedir, 'test.db')
        self.app = app.test_client()
        # db = SQLAlchemy(app)
        global engine
        engine = create_engine(app.config['SQLALCHEMY_DATABASE_URI'])
        Session = sessionmaker(bind=engine)
        db.session = Session()
        db.create_all()
        # self.app.testing = True
        self.create_users()
        self.create_menus()

    # with app.app_context():
    #   db.create_all()
    #   self.create_roles()
    #   self.create_users()
    #   self.create_buildings()

    def tearDown(self):
        # with app.app_context():
        # with app.request_context():
        # db.drop_all()
        # os.close(self.db_gd)
        basedir = os.path.abspath(os.path.dirname(__file__))
        os.unlink(os.path.join(basedir, 'test.db'))

    # Login/Logout
    def login(self, username, password):
        return self.app.post('/auth/login', data=dict(
            username=username,
            password=password),
                             follow_redirects=True)

    def logout(self):
        return self.app.get('/auth/logout', follow_redirects=True)

    def register_visitor(self, name, email):
        return self.app.get('/greg/register_visitor', data=dict(
            name=name,
            email=email),
                            follow_redirects=True)

```

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

결과