**DHIRUBHAI AMBANI INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY**


**IT314 - SOFTWARE ENGINEERING**

---

**UNIT TESTING**

# CRIME AND HAZARD MANAGEMENT SYSTEM

---

**GROUP NO: 30**

**GROUP MEMBERS**

| | |
|---|---|
| 202001410 | PATEL AYUSH SANJAYBHAI |
| 202001447 | VAKANI HETAV ABHAYBHAI |
| 202001467 | JAY GROVER |
| 202001421 | PATEL KUNJ RAKESH |
| 202001446 | GONDALIYA VENIL CHANDUBHAI |
| 202001275 | AYUSH JAIN |
| 202001458 | KRIS PATEL |
| 202001264 | HARSH SANJAY MAKWANA |
| 202001466 | KALP KINJALBHAI PANDYA |
| 202001438 | NARODIA JEET NILESHKUMAR |
| 202001457 | HITARTH VYAS |

# Unit Testing

Django comes with its own test suite within the project for testing. One can run Unit Tests within this framework. We wrote code for these tests in a file named tests_unit.py within our django application. The django app is created such that these tests can be executed by running in command python manage.py test. A robust structure was followed while writing scripts to test the login functionality as shown below:

## Login Test Cases:

Below is code for constructor and destructor which connects to the database and adds test data for testing.

```
class LoginTestCase(TestCase):
    @classmethod
    def setUpClass(self):
        super(LoginTestCase, self).setUpClass()
        self.client =
pymongo.MongoClient("mongodb+srv://superuser:superuser%40SWE30@swe-cluster.xxvswr
z.mongodb.net/?retryWrites=true&w=majority")
        self.db = self.client['swe_test_db']
        self.collection = self.db['users']
        new_user = {'UserName': 'testuser', 'Password': 'testpass', 'Email':
'test@gmail.com', 'FirstName': 'test', 'LastName': 'user', 'DOB': '2002-07-20'}
        self.collection.insert_one(new_user)

    @classmethod
    def tearDownClass(self):
        super(LoginTestCase, self).tearDownClass()
        self.collection.delete_many({'UserName': 'testuser'})
```

## Test Case 1: When corrects credentials are entered

```
    # test login success
    def test_login(self):
        response = self.client.post('/myApp/login/', {'UserName': 'testuser',
'Password': 'testpass'})
        self.assertEqual(response.status_code, 302)
        self.assertEqual(response.url, '/myApp')
        self.client.get('/myApp/logout/')
```

Here, we gave correct credentials to the login page, so after that it redirects to another url for that we equate the response status code with 302 and also check if the redirected url is the default home page(/myApp) or not.

## Test Case 2: When incorrect credentials are entered

```
    # test login failure
    def test_login_failure(self):
        response = self.client.post('/myApp/login/', {'UserName': 'testuser',
'Password': 'wrongpass'})
        self.assertTemplateUsed(response, 'myApp/login.html')
```

We entered invalid credentials, so the page should not redirect to another url and should be on same page so we check that we are on the same HTML template or not.

## Test Case 3: When no credentials are entered

```
    # test no input
    def test_login_no_input(self):
        response = self.client.post('/myApp/login/', {'UserName': '', 'Password':
''})
        self.assertTemplateUsed(response, 'myApp/login.html')
```

### Test Case 4: When only username is entered

```
    # test only username input
    def test_login_only_username(self):
        response = self.client.post('/myApp/login/', {'UserName': 'testuser',
'Password': ''})
        self.assertTemplateUsed(response, 'myApp/login.html')
```

### Test Case 5: When only password is entered

```
    # test only password input
    def test_login_only_password(self):
        response = self.client.post('/myApp/login/', {'UserName': '', 'Password':
'testpass'})
        self.assertTemplateUsed(response, 'myApp/login.html')
```

For Test Cases 3,4,5 we check if we are on the same HTML template or not.This is because we have not given correct credentials so it should not redirect to another url.

## Register Test Cases

First we connect to the database through constructor. In destructor we delete the test user from the database.

```
class RegisterTestCase(TestCase):
    @classmethod
    def setUpClass(self):
        super(RegisterTestCase, self).setUpClass()
        self.client =
pymongo.MongoClient("mongodb+srv://superuser:superuser%40SWE30@swe-cluster.xxvswr
z.mongodb.net/?retryWrites=true&w=majority")
        self.db = self.client['swe_test_db']
        self.collection = self.db['users']
```

```
    @classmethod
    def tearDownClass(self):
        super(RegisterTestCase, self).tearDownClass()
        self.collection.delete_many({'UserName': 'testuser'})
```

## Test Case 1: Test successful registration

```
    # test register success
    def test_register(self):
        response = self.client.post('/myApp/register/', {'UserName': 'testuser',
'Password': 'testpass', 'ConfirmPassword': 'testpass', 'Email': 'test@gmail.com',
'FirstName': 'test', 'LastName': 'user', 'DOB': '2002-07-20'})
        self.assertEqual(response.status_code, 302)
        response = self.client.get('/myApp/')
        self.assertTemplateUsed(response, 'myApp/reg_hmpg.html')
        self.client.get('/myApp/logout/')
        self.collection.delete_many({'UserName': 'testuser'})
```

We posted new registration information to the register page. We expect a 302 response code, which means the page redirected to the home page. We then check that the home page template was used. We then logout and delete the test user from the database.

## Test Case 2: Test registration failure due to same username

```
    # test register failure due to same username
    def test_register_failure(self):
        response = self.client.post('/myApp/register/', {'UserName': 'testuser',
'Password': 'testpass', 'ConfirmPassword': 'testpass', 'Email': 'test@gmail.com',
'FirstName': 'test', 'LastName': 'user', 'DOB': '2002-07-20'})
        self.assertEqual(response.status_code, 302)
        response = self.client.get('/myApp/')
        self.assertTemplateUsed(response, 'myApp/reg_hmpg.html')
        self.client.get('/myApp/logout/')
```

```
        response = self.client.post('/myApp/register/', {'UserName': 'testuser',
'Password': 'testpass', 'ConfirmPassword': 'testpass', 'Email': 'test@gmail.com',
'FirstName': 'test', 'LastName': 'user', 'DOB': '2002-07-20'})
        self.assertTemplateUsed(response, 'myApp/register.html')
        self.collection.delete_many({'UserName': 'testuser'})
```

We first register a new user. We then logou and We then try to register the same user again. We expect not to redirect to the home page and instead stay on the register page.

## Test Case 3: Test registration failure due to invalid input

```
    # test no input
    def test_register_no_input(self):
        response = self.client.post('/myApp/register/', {'UserName': '',
'Password': '', 'ConfirmPassword': '', 'Email': '', 'FirstName': '', 'LastName':
'', 'DOB': ''})
        self.assertTemplateUsed(response, 'myApp/register.html')
```

## Test Case 4: Test registration failure due to invalid input

```
    # test only username input
    def test_register_only_username(self):
        response = self.client.post('/myApp/register/', {'UserName': 'testuser',
'Password': '', 'ConfirmPassword': '', 'Email': '', 'FirstName': '', 'LastName':
'', 'DOB': ''})
        self.assertTemplateUsed(response, 'myApp/register.html')
```

## Test Case 5: Test registration failure due to invalid input

```
    # test invalid email
    def test_register_invalid_email(self):
```

```
        response = self.client.post('/myApp/register/', {'UserName': 'testuser',
'Password': 'testpass', 'ConfirmPassword': 'testpass', 'Email': 'test',
'FirstName': 'test', 'LastName': 'user', 'DOB': '2002-07-20'})
        self.assertTemplateUsed(response, 'myApp/register.html')
```

### Test Case 6: Test registration failure due to invalid input

```
    # test password mismatch
    def test_register_password_mismatch(self):
        response = self.client.post('/myApp/register/', {'UserName': 'testuser',
'Password': 'testpass', 'ConfirmPassword': 'wrongpass', 'Email':
'test@gmail.com', 'FirstName': 'test', 'LastName': 'user', 'DOB': '2002-07-20'})
        self.assertTemplateUsed(response, 'myApp/register.html')
```

For Test Cases 3,4,5,6 we test for invalid input. As we entered invalid registration information, we expect to stay on the register page and not to redirect to the home page.

## Post Incident Test Cases

We first connect to the database through constructor and inserted a test user into the database. In destructor, we delete the test user and test incident from the database.

```
# post incident test case
class PostIncidentTestCase(TestCase):
    @classmethod
    def setUpClass(self):
        super(PostIncidentTestCase, self).setUpClass()
        self.client =
pymongo.MongoClient("mongodb+srv://superuser:superuser%40SWE30@swe-cluster.xxvswr
z.mongodb.net/?retryWrites=true&w=majority")
        self.db = self.client['swe_test_db']
        self.collection = self.db['users']
        self.collection.insert_one({'UserName': 'testuser', 'Password':
'testpass', 'Email': 'test@gmail.com', 'FirstName': 'test', 'LastName': 'user',
'DOB': '2002-07-20'})
        self.collection = self.db['incident']
```

```
    @classmethod
    def tearDownClass(self):
        super(PostIncidentTestCase, self).tearDownClass()
        self.collection = self.db['users']
        self.collection.delete_many({'UserName': 'testuser'})
        self.collection = self.db['incident']
        self.collection.delete_many({'title': 'test'})
```

## Test Case 1: Test post incident success after login assuming user is already registered

```
    # test post incident success after login assuming user is already registered
    def test_post_incident(self):
        response = self.client.post('/myApp/login/', {'UserName': 'testuser',
'Password': 'testpass'})
        self.assertEqual(response.status_code, 302)
        response = self.client.get('/myApp/postIncident/')
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'myApp/postIncident.html')
        response = self.client.post('/myApp/postIncident/', {'Title': 'test',
'Description': 'test', 'Latitude': '0', 'Longitude': '0', 'Time':
'2021-04-20T00:00','crime-or-hazard': 'crime','IncidentType': 'Cybercrime'})
        self.assertEqual(response.status_code, 302)
        self.assertEqual(response.url, '/myApp')
        self.client.get('/myApp/logout/')
        self.collection.delete_many({'title': 'test'})
```

We first login with the test user and then try to post an incident. We expect to redirect to the home page.

## Test Case 2: Test post incident failure because of not logged in

```
    # test post incident failure because of not logged in
    def test_post_incident_failure(self):
```

```
        response = self.client.post('/myApp/postIncident/', {'Title': 'test',
'Description': 'test', 'Latitude': '0', 'Longitude': '0', 'Time':
'2021-04-20T00:00','crime-or-hazard': 'crime','IncidentType': 'Cybercrime'})
        self.assertEqual(response.status_code, 302)
        self.assertRedirects(response, '/myApp/login/')
```

We try to post an incident without logging in. We expect to redirect to the login page.

### Test Case 3: Test post incident failure because of invalid input

```
    # test post incident failure because of invalid input
    def test_post_incident_invalid_input(self):
        response = self.client.post('/myApp/login/', {'UserName': 'testuser',
'Password': 'testpass'})
        self.assertEqual(response.status_code, 302)
        response = self.client.get('/myApp/postIncident/')
        self.assertEqual(response.status_code, 200)
        response = self.client.post('/myApp/postIncident/', {'Title': '',
'Description': '', 'Latitude': float(), 'Longitude': float(), 'Time':
'','crime-or-hazard': '','IncidentType': ''})
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'myApp/postIncident.html')
        self.client.get('/myApp/logout/')
```

We try to post an incident with invalid input which are None values.So, we expect to stay on the post incident page.

## Output after running all test cases simultaneously

```
PS C:\Users\ayush\Desktop\Coding\new\IT314_project_30\Code\project> python manage.py test
Found 14 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
..............
----------------------------------------------------------------------
Ran 14 tests in 3.361s

OK
Destroying test database for alias 'default'...
PS C:\Users\ayush\Desktop\Coding\new\IT314_project_30\Code\project>
```