CSE 403
Software Engineering
Spring 2023

#14: Testing
<table>
<thead>
<tr>
<th>Week 5</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>04/24</td>
<td>L: Build Systems</td>
<td></td>
</tr>
<tr>
<td>04/25</td>
<td>T:</td>
<td>DUE:</td>
</tr>
<tr>
<td>04/26</td>
<td>L: Testing</td>
<td>Testing &amp; CI/CD (TCC)</td>
</tr>
<tr>
<td>04/27</td>
<td>P:</td>
<td></td>
</tr>
<tr>
<td>04/28</td>
<td>L: CI/CD</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Week 6</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>05/01</td>
<td>L: Test Coverage</td>
<td></td>
</tr>
<tr>
<td>05/02</td>
<td>T:</td>
<td>DUE:</td>
</tr>
<tr>
<td>05/03</td>
<td>L: Mutation Testing</td>
<td>Alpha_Release_(R1)</td>
</tr>
<tr>
<td>05/04</td>
<td>P:</td>
<td></td>
</tr>
<tr>
<td>05/05</td>
<td>LX: Code Defenders</td>
<td></td>
</tr>
</tbody>
</table>
Today

- Software testing 101
- Test-Driven Development (TDD)
- Live demo
Software testing: unit testing example

Testing: is there a bug?

```java
import java.util.Arrays;

public class Math {
    public static double avg(double[] nums) {
        int n = nums.length;
        double sum = 0;
        int i = 0;
        while (i < n) {
            sum = sum + nums[i];
            i = i + 1;
        }
        double avg = sum / n;
        return avg;
    }
}

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.assertEquals;

public class MathTest {
    @Test
    public void testAvg() {
        double nums = Arrays.asList(1.0, 2.0, 3.0);
        double actual = Math.avg(nums);
        double expected = 2.0;
        assertEquals(expected, actual, EPS);
    }
}
```
Software testing: unit testing example

```java
1 double avg(double[] nums) {
2     int n = nums.length;
3     double sum = 0;
4     int i = 0;
5     while (i<n) {
6         sum = sum + nums[i];
7         i = i + 1;
8     }
9 }
10
11 double avg = sum * n;
12 return avg;
13 }
```

Testing: is there a bug?

```java
@Test
public void testAvg() {
    double nums =
        new double[]{1.0, 2.0, 3.0});
    double actual = Math.avg(nums);
    double expected = 2.0;
    assertEquals(expected, actual, EPS);
}
```

testAvg failed: 2.0 != 18.0
Software testing vs. software debugging

Testing: is there a bug?

```java
@Test
public void testAvg() {
    double[] nums = {1.0, 2.0, 3.0};
    double actual = Math.avg(nums);
    double expected = 2.0;
    assertEquals(expected, actual, EPS);
}
```

testAvg failed: 2.0 != 18.0

Debugging: where is the bug?
how to fix the bug?
Software testing vs. software debugging
Unit testing, integration testing, system testing

Unit testing
- Does each unit work as specified?

Integration testing
- Do the units work when put together?

System testing
- Does the system work as a whole?
Unit testing, integration testing, system testing

Unit testing
  ● Does each unit work as specified?

Integration testing
  ● Do the units work when put together?

System testing
  ● Does the system work as a whole?

Key focus in 403: unit testing
Unit testing

- A **unit** is the *smallest testable part* of the software system (e.g., a method in a Java class).
- **Goal**: Verify that each software unit performs **as specified**.
- **Focus**:
  - Individual units (not the interactions between units).
  - Usually input/output relationships.
Example: Unit Testing

Average of the absolute values of an array of doubles

```java
public double avgAbs(double ... numbers) {
    // We expect the array to be non-null and non-empty
    if (numbers == null || numbers.length == 0) {
        throw new IllegalArgumentException("Array numbers must not be null or empty!");
    }

    double sum = 0;
    for (int i=0; i<numbers.length; ++i) {
        double d = numbers[i];
        if (d < 0) {
            sum -= d;
        } else {
            sum += d;
        }
    }

    return sum/numbers.length;
}
```

What tests should we write for this method?
Example: Unit Testing

Average of the absolute values of an array of doubles

```java
public double avgAbs(double ... numbers) {

    // We expect the array to be non-null and non-empty
    if (numbers == null || numbers.length == 0) {
        throw new IllegalArgumentException("Array numbers must not be null or empty!");
    }

    double sum = 0;
    for (int i=0; i<numbers.length; ++i) {
        double d = numbers[i];
        if (d < 0) {
            sum -= d;
        } else {
            sum += d;
        }
    }

    return sum/numbers.length;
}
```

What tests should we write for this method?
Example: TDD (Test Driven-Development)

Average of the absolute values of an array of doubles

```java
public double avgAbs(double ... numbers) {
    //I am intentionally empty right now!!!
}

@Test
public void testAvgAbs() {
    assertEquals(2.0, avgAbs({1.0, 2.0, 3.0}));
    assertEquals(2.0, avgAbs({1.0, -2.0, 3.0}));
    assertEquals(2.0, avgAbs({2.0}));
    // ...
}
```

What CODE should we write to PASS these tests?
Example: TDD (Test Driven-Development)

Average of the absolute values of an array of doubles

```java
public double avgAbs(double ... numbers) {
    //I am intentionally empty right now!!!
}

@Test
public void testAvgAbs() {
    assertEquals(2.0, avgAbs({1.0, 2.0, 3.0}));
    assertEquals(2.0, avgAbs({1.0, -2.0, 3.0}));
    assertEquals(2.0, avgAbs({2.0}));
    // ...
}
```

What CODE should we write to PASS these tests?

+ What TEST should we write to CAPTURE the specification?
Live example: TDD
(where things only happen when there is a broken test!)

1: Should that something needs to be coded with a test!

2: Code the logic that will beat the tests!
Example: LITW API Tests

Should we move this to be a NODE server? (fits our current template building process.)
class TestAPI(unittest.TestCase):
    def setUp(self):
        settings.LITWS_TEST = True
        settings.MONGODB_NAME = settings.MONGODB_NAME_TEST
        client = MongoClient(settings.MONGODB_URL)
        db = client[settings.MONGODB_NAME]
        collection = db[litw_mongo.COLLECTION_STUDY]
        collection.drop()
        collection = db[litw_mongo.COLLECTION_DATA]
        collection.drop()
        data_access = litw_mongo.DataAccessFactory(settings)
        litw_study_access = data_access.access_points[data_access.access_points][data_access.access_points]
        litw_data_access = data_access.access_points[data_access.access_points]
        test_study_id = litw_study_access.add_study('TEST_STUDY')
        self.assertIsNotNone(test_study_id)
        self.test_study = litw_study_access.get_study(test_study_id)
        self.time_test_start = src.litw.api.util.get_utc_timestamp()
        self.mongo_data = litw_data_access

Meanwhile, in the real world… [SetUp?]
def testAuth(self):
    with TestClient(app) as client:
        response = client.post(URL_AUTH)
        self.assertEqual(422, response.status_code)
        response = client.post(URL_AUTH, headers=URL_AUTH_HEADER)
        self.assertEqual(422, response.status_code)
        response = client.post(URL_AUTH, headers=URL_AUTH_HEADER, content=URL_AUTH_PARAMS.format(s)
        self.assertEqual(422, response.status_code)
        response = client.post(URL_AUTH, headers=URL_AUTH_HEADER, content=URL_AUTH_PARAMS.format(s
        self.assertEqual(200, response.status_code)
        request_token = response.json()['access_token']
        self.assertRegex(request_token, litw_data.JWT_TOKEN_REGEX)
        token_payload = decode_token(request_token)
        self.assertTrue(isinstance(token_payload['exp'], int))

Meanwhile, in the real world… [Mock or Real?]
Example: Testing Components: LITW MockAPI

Should we move this to be a NODE server? (fits our current template building process.)
Today

- Software testing 101
- Test-Driven Development (TDD)
- Live demo

Questions, please!