CSE 403

Software Engineering

Spring 2023

#14: Testing

Logistics

WEEK 5		
04/24	L: Build Systems	
04/25	T:	DUE: DnA!!!
04/26	L: Testing	Testing & CI/CD (TCC)
04/27	P:	
04/28	L: CI/CD	
WEEK 6		
05/01	L: Test Coverage	
05/02	T:	DUE: TCC!!!
05/03	L: Mutation Testing	Alpha Release (R1)
05/04	P:	
05/05	LX: Code Defenders	

Today

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

Software testing: unit testing example

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

Testing: is there a bug?

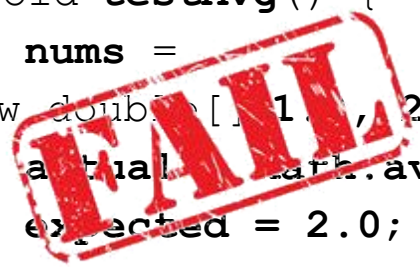
```
@Test
public void testAvg() {
  double nums =
      new double[]{1.0, 2.0, 3.0});
  double actual = Math.avg(nums);
  double expected = 2.0;
  assertEquals(expected,actual,EPS);
}
```

Software testing: unit testing example

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

Testing: is there a bug?

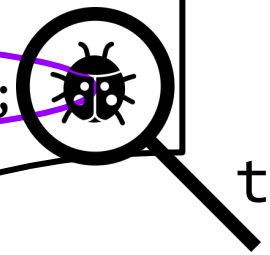
```
@Test
public void testAvg() {
  double nums =
      new double[]{1.0, 2.0, 3.0});
  double actual = ark.avg(nums);
  double expected = 2.0;
  assertEquals(expected,actual,EPS);
}
```




```
testAvg failed: 2.0 != 18.0
```

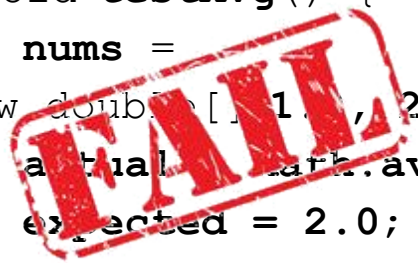
Software testing vs. software debugging

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



Testing: is there a bug?

```
@Test
public void testAvg() {
  double nums =
      new double[]{1.0, 2.0, 3.0});
  double actual = Stats.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

```
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

```
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

```
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

```
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!)

```
nigini@librarian-xps:~/WORKSPACE/TEACHING/
..E
======================================
ERROR: test_crazy (test_bad_math.TestAbsAv
--------------------------------------
Traceback (most recent call last):
  File "/home/nigini/WORKSPACE/TEACHING/te
    abs_avg(['a',-2.0,3])
  File "/home/nigini/WORKSPACE/TEACHING/te
    accumulator += abs(number)
                   ^^^^^^^^^^
TypeError: bad operand type for abs(): 'st

--------------------------------------
Ran 3 tests in 0.000s

FAILED (errors=1)
```

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

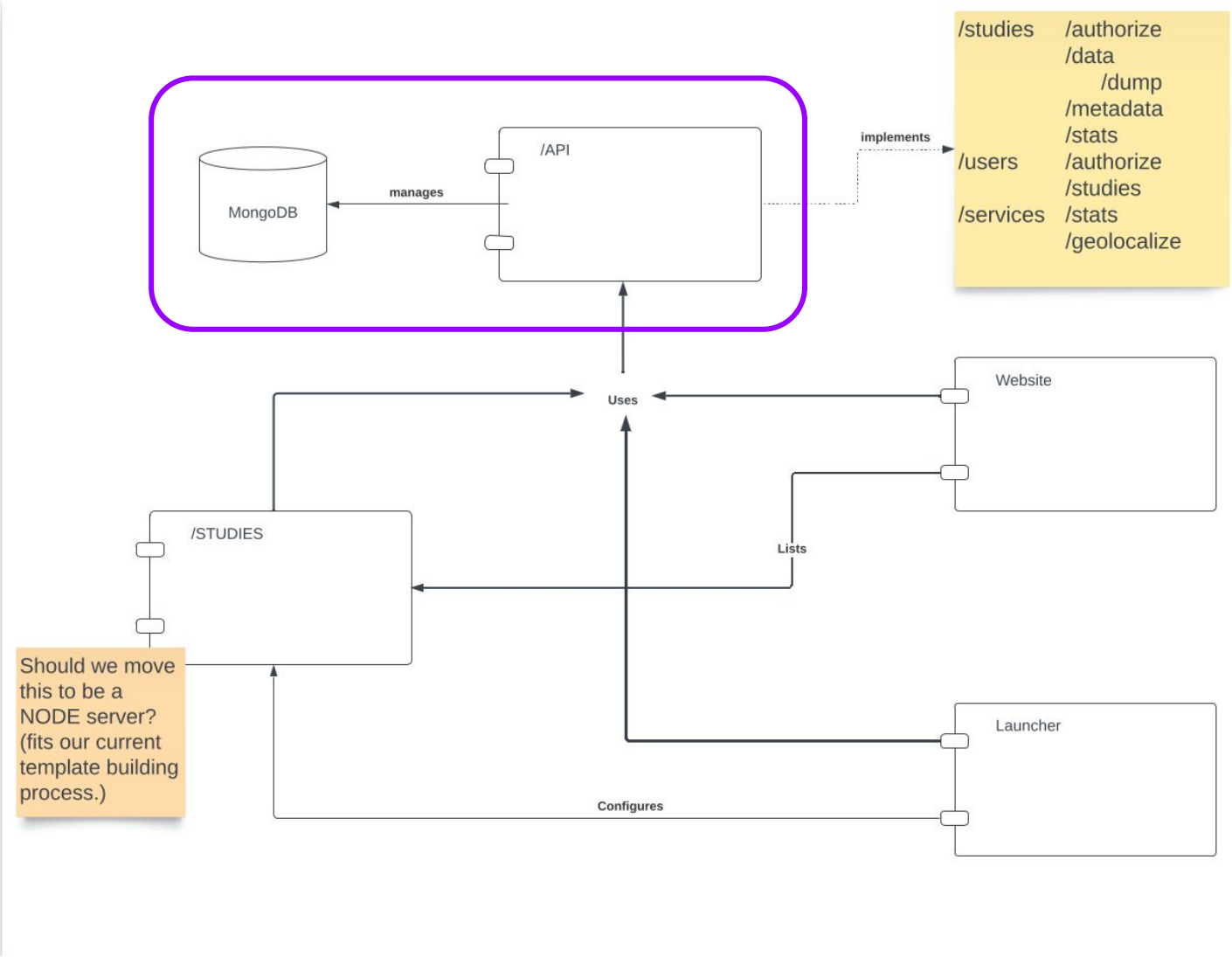
```
nigini@librarian-xps:~
 python3 -m unittest
...
----------------------

----------------------
Ran 3 tests in 0.000s

OK
```

2: Code the logic that will beat the tests!

Example: LITW API Tests



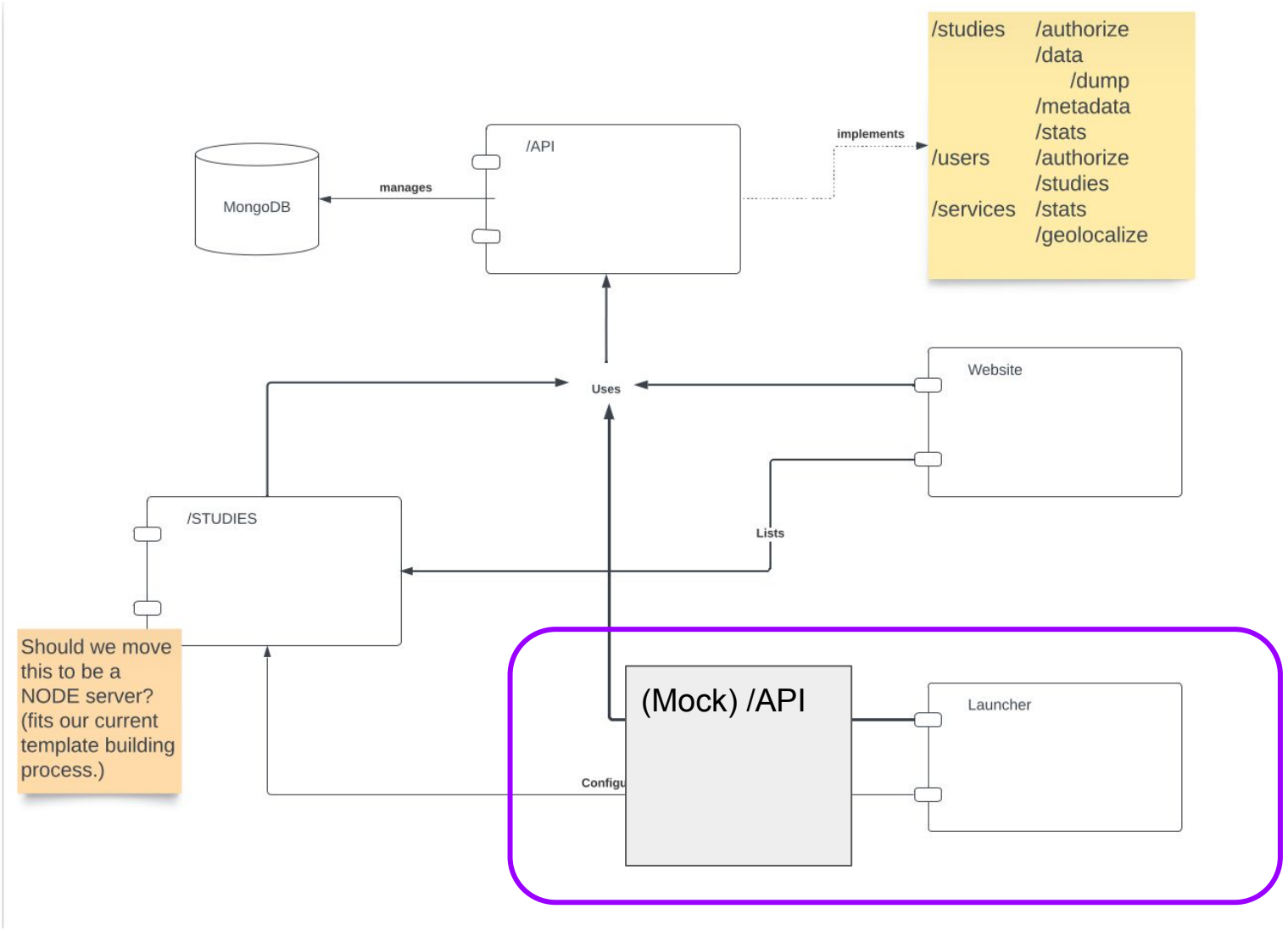
```
class TestAPI(unittest.TestCase):
    def setUp(self):
        settings.LITWS_TEST = True
        settings.MONGODB_NAME = settings.MONGODB_NAME_TEST
        client = MongoClient(settings.MONGODB_URL)
        db = client[settings.MONGODB_NAME]
        collection = db[litw_mongo.COLLECTION_STUDY]
        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.ava
        litw_data_access = data_access.access_points[data_access.avai
        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(se
        self.assertEqual(422, response.status_code)
        response = client.post(URL_AUTH, headers=URL_AUTH_HEADER,
                               content=URL_AUTH_PARAMS.format(se
        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



Today

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

Questions, please!