The Welcome Home
Smart Rental Contract Project

Unit Test

1.0

26-APR-22

Sergio Prieto
Casey Munga

Table of Contents

1. Unit Test

1.1 Approach

The intent of this Unit Test is to perform testing on individual methods and classes of the program (StringUtility Class) using a white-box test; whereas access to methods, data structures, and signatures are provided. The traditional approach suggests to write code, compile it to eliminate syntax errors, and then write and execute test cases. Conversely, Agile approach encourage test driven developments, where tests are written before code. Unit Test are the responsibility of the software developer; although, big projects might have a dedicated team of Testers to fulfill these tasks. Fixes are transferred to the developers and failed test are reschedule upon delivery of the updated package. Testing can include code coverage and data coverage, in order to test lines of code and data correctness. Unit testing is a method where individual units of code are tested to determine where they are fit to use.

1.2 Test Cases

Test Case 1

Method being tested: isEmpty()
Short Description: Test that wallets exists
Input Data to constructor and/or method you are testing: Create SimpleQueue with no elements
Expected Results: method return value true
Actual Results: **Passed:** method returned what was expected

```
// Get a list of all accounts
  accounts = await web3.eth.getAccounts();
  inbox = await new web3.eth.Contract(abi)
    .deploy({
      data: evm.bytecode.object,
      arguments: ['Hi there!'],
    })
    .send({ from: accounts[0], gas: '1000000' });
});
…
```

Test Case 2

Method being tested: contractDeploy()
Short Description: Test that contract is deployed
Input Data to constructor and/or method you are testing: Create SimpleQueue with no elements
Expected Results: method return value true
Actual Results: **Passed:** method returned what was expected

```
it('deploys a contract', () => {
    assert.ok(inbox.options.address);
```

```
});
```

Test Case 3

Method being tested: contractDeploy()
Short Description: Test that contract is deployed
Input Data to constructor and/or method you are testing: Create SimpleQueue with no elements
Expected Results: method return value true
Actual Results: **Passed:** method returned what was expected

```
it('deploys a contract', () => {
   assert.ok(inbox.options.address);
 });
```

Test Case 4

Method being tested: contractDeploy()
Short Description: Test that contract is deployed
Input Data to constructor and/or method you are testing: Create SimpleQueue with no elements
Expected Results: method return value true
Actual Results: **Passed:** method returned what was expected

```
it('has a default transaction, async () => {
   const transaction = await inbox.methods. transaction ().call();
   assert.equal(transaction, '100!');
 });
```

2. Defects

No Defects found