For testing we use Mocha and chai that comes bundled in truffle

* Mocha: testing framework in javascript
* Chai : assertion library

Create a test file for the smart contract inside project/test/Marketplace.test.js:

**const** Marketplace = artifacts.require('./Marketplace.sol')

contract('Marketplace', (accounts) => { // get all accounts

**let** marketplace

**before**(**async** () => { //it runs before every test case

marketplace = **await** Marketplace.deployed()

})

describe('deployment', **async** () => { //container many test cases

it('deploys successfully', **async** () => { //test case to check address

**const** address = **await** marketplace.address

assert.notEqual(address, 0x0) //address not zero,numm,undefined,empty

assert.notEqual(address, '')

assert.notEqual(address, null)

assert.notEqual(address, undefined)

})

it('has a name', **async** () => { //test case to check name

**const** name = **await** marketplace.name()

assert.equal(name, 'Dapp University Marketplace')

})

})

})

This test does two things:

* Checks that the smart contract has an address, i.e., it was successfully deployed to the network.
* Checks that the name was set when it was deployed.

Run test:

$ truffle test

Sample file :

**const** Marketplace = artifacts.require('./Marketplace.sol')

require('chai')

.use(require('chai-as-promised'))

.should()

contract('Marketplace', ([deployer, seller, buyer]) => { // get only 3 accounts and name them

**let** marketplace

**before**(**async** () => { //runs before every test case

marketplace = **await** Marketplace.deployed()

})

describe('deployment', **async** () => { //contains many test cases

it('deploys successfully', **async** () => { //test case to check address

**const** address = **await** marketplace.address

assert.notEqual(address, 0x0) //address not zero,numm,undefined,empty

assert.notEqual(address, '')

assert.notEqual(address, null)

assert.notEqual(address, undefined)

})

it('has a name', **async** () => {

**const** name = **await** marketplace.name()

assert.equal(name, 'Dapp University Marketplace')

})

})

describe('products', **async** () => { //test case for a function

**let** result, productCount

**before**(**async** () => {

result = **await** marketplace.createProduct('iPhone X', web3.utils.toWei('1', 'Ether'), { **from**: seller }) // {from: } is used to pass metadata

productCount = **await** marketplace.productCount()

})

it('creates products', **async** () => {

// SUCCESS

assert.equal(productCount, 1)

**const** **event** = result.logs[0].args

assert.equal(**event**.id.toNumber(), productCount.toNumber(), 'id is correct')

assert.equal(**event**.name, 'iPhone X', 'name is correct')

assert.equal(**event**.price, '1000000000000000000', 'price is correct')

assert.equal(**event**.owner, seller, 'owner is correct')

assert.equal(**event**.purchased, false, 'purchased is correct')

// FAILURE: Product must have a name

**await** **await** marketplace.createProduct('', web3.utils.toWei('1', 'Ether'), { **from**: seller }).should.be.rejected;

// FAILURE: Product must have a price

**await** **await** marketplace.createProduct('iPhone X', 0, { **from**: seller }).should.be.rejected;

})

})

})

Test cases generally works in wei so set the price in wei.

Web3.utils.toWei(‘1’,’Ether’) // convert a value to wei

**const** **event** = result.logs[0].args // get the value of the event