**Lab 4 IS4302**

**Truffle Suite**

**Introduction**

In this lab, we’ll cover the following:

1. Setting up a Truffle Environment
2. Writing Contracts in Truffle
3. Deploying contracts/ Migrating contracts
4. Testing Contracts

**Setting Up Truffle**

To download truffle, you first need Node : <https://nodejs.org/en/download/>

Please download and install Node JS

Once Node JS is installed, please open your terminal and run: ***npm install truffle -g***

This will install the Truffle environment

Apart from Truffle, we also need **Ganache**

Ganache is a development blockchain running the Ethereum Virtual Machine which can simulate the Ethereum blockchain. You can think of it as the “backend” when you deployed your smart contracts on Remix IDE.

To download Ganache, click here: <https://trufflesuite.com/ganache/>

Follow the instructions to download and install ganache

The last item we need is a local IDE (recall that Remix is an online IDE). The recommendation is to use **Visual Studio Code** which can be downloaded here: <https://code.visualstudio.com/download>

Go ahead and install all three

**Writing Contracts in Truffle**

Once all the tools are downloaded appropriately, you should be all set to create your environment. Open your terminal or command prompt and create a folder to store your project.

Use the command **truffle init** to create a folder structure with the basic files needed in your project.

Text

Description automatically generated

If initialisation is successful, you should see the message above.

Now if you examine the contents of this folder, you should see some files have been created for you. For **Windows** type **dir**. For **Mac** type **ls**. Then hit enter

Text

Description automatically generated

You will see 3 folders:

1. Contracts
2. Migrations
3. Test

You will also see truffle-config.js, a file that configures the truffle environment.

|  |  |
| --- | --- |
| Folder | What it does |
| contracts | Stores your .sol files |
| migrations | Stores instructions on how your contracts should be compiled and then committed on the blockchain |
| test | Stores your unit test files |

Let’s start real slow. We are going to use the Dice.sol and DiceMarket.sol contracts for this lab (not the ones using DiceToken).

To open your environment, first open Visual Studio Code

A screenshot of a computer

Description automatically generated

Go to File -> Open and select **the folder containing the 3 folders + 1 script**. Open that folder.

Graphical user interface, application

Description automatically generated

This is what the left hand side menu should show. Feel free to open the contracts folder and copy over your .sol files. A screenshot of a computer

Description automatically generated with medium confidence

If prompted to install an extension that allows you to code on Solidity, please do so. It will give you hints about errors to help make your code bug free.

You can always create new files, write your contracts and save them using VS Code.

**Deploying Contracts**

Contracts written like this are not useful unless we are able to deploy them to our development blockchain and test them.

In order to deploy blockchains, you need to use the **Migrations** folder.

Graphical user interface, text, application, chat or text message

Description automatically generated

You will see that the migrations folder already has a javascript file. We need to add another file and describe how we want the blockchain to work.

1. Create a file called **“2\_deploy\_contracts.js”**
2. Inside this file, follow the structure shown below  
     
   Text

   Description automatically generated  
     
   First, by declaring Dice and DiceBattle, we are requiring these contracts to exist in our environment  
     
   Then we are using deployer to deploy Dice first. Once Dice has been deployed, we ask it to deploy DiceBattle and pass Dice’s address to DiceBattle’s constructor.
3. Open truffle-config.js and scroll down to networks. **Uncomment out the section shown below and change the port to 7545**Text

   Description automatically generated

Awesome, now we’ve made all the changes needed to deploy our contracts. But, we still don’t have a blockchain to deploy these to.

Open the Ganache app. It should look something like this:

Graphical user interface

Description automatically generated

Click quickstart Ethereum. That should end up with something like this:

Table

Description automatically generated

Nice!

Test that everything is working now. Go to your terminal and type:

*truffle compile*

then

*truffle migrate*

*Text

Description automatically generated*

Truffle Compile

*Text

Description automatically generated*

Truffle Migrate

You will see that your ganache shows some accounts have less than 100 ETH. This is the contract deployment process.

**Testing Contracts**

Go to the test folder and create a file called ‘test\_dice.js’.

You can opt to download Truffle Assertions to test your events here: <https://www.npmjs.com/package/truffle-assertions>

To start, let’s first require the contracts to be deployed and our assertion frameworks to be correctly initialised:

Text

Description automatically generated

Now, let us create variables to represent these contracts:

Text

Description automatically generated

Now, let us establish our testing :

Text

Description automatically generated

This essentially waits for the 2 contracts to be deployed before any testing can occur.

Test 1: Test the ability to get dice

Text

Description automatically generated

Test 2: Test transfer ownership

Text

Description automatically generated

Test 3: Dice Battle working

Text

Description automatically generated

Nice! We have written unit tests!

**Actual Testing**

Once your test file is done, go to your terminal and type ‘truffle test’. Hit enter and see how your test cases perform!

Text

Description automatically generated

**NOTE: Unit tests must test small aspects of code to isolate bugs or errors. They should not test large chunks because if they fail, you are not sure where the bug is.**

**Homework Exercises:**

**Ex 1:**

Create a truffle environment and use Truffle and ganache to create unit tests for the DiceMarket contract from Lab 3.

Test cases must include:

1. Test the creation of the dice
2. Test that if ether is not supplied to the Dice contract’s add function, an error is returned
3. Test that the Dice can be transferred to the DiceMarket contract
4. Test that a Die cannot be listed if the price is less than value + commission
5. Test that a Dice can be listed
6. Test that the owner can unlist a die
7. Test that another party can buy the die

Feel free to expand the test cases beyond this.