CS5363 Blockchain Technologies and Applications: Hw8 – Report
Decentralized App (DApp)

Name: 高瑞宏
Student Id: 109062584

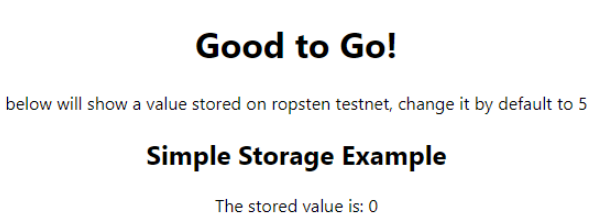
Task: A DApp on Web

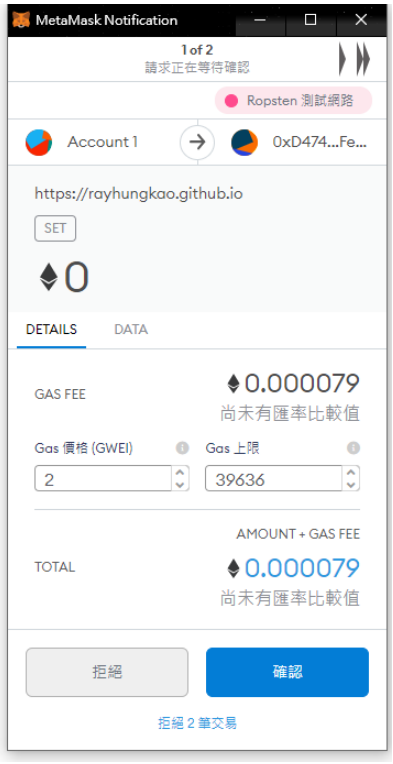
Introduction

- Product/Service Description (Describe the application scenario, including the UI & functions)

"Simple Storage" is a very simple dapp for you to easily send a transaction by using Metamask to change the value stored on ropsten testnet.

As you can see: [Homepage | Simple Storage]

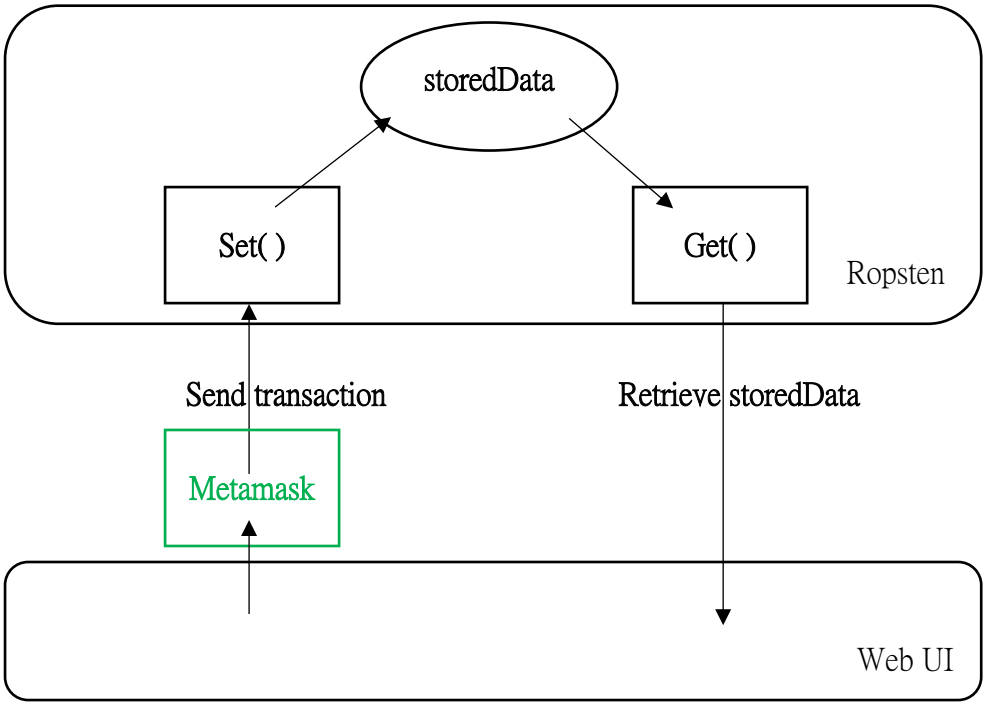




Design your contracts & your web app

- Functional Description (Describe what your contracts & your web app can do)
- Design Diagram & Flow Chart (Explain the operation process in your contracts & your web app)
- ...

My smart contract has two functions, one is “set”, and the other is “get”.



My web app can automatically display the current value stored on Ropsten testnet by “get” function, and for users to send a transaction to change the stored value by using “set” function.

Test your contracts & your web app

- Test Flow Chart (Explain how can you ensure your contracts & your web app is correct)
- Test cases (Ensure your test cases provide enough testing coverage)
- ...

For smart contract testing, I use “Truffle” to simulate a transaction after my web app is deployed correctly.

Testcase is for a user to give an input through my web app, trigger “set” function and then compare the input with the stored data on the blockchain by using “get” function.

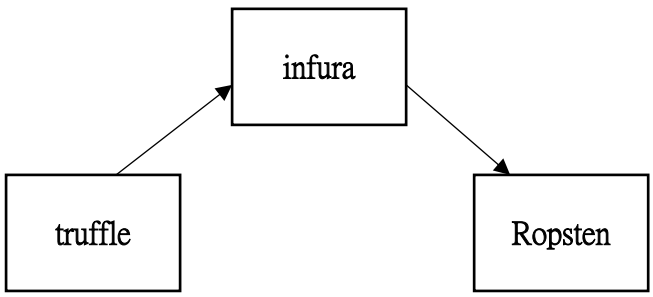
```
test > JS simplestorage.js > ...
1   const SimpleStorage = artifacts.require("./SimpleStorage.sol");
2
3   contract("SimpleStorage", accounts => {
4     it("...should store the value 89.", async () => {
5       const simpleStorageInstance = await SimpleStorage.deployed();
6
7       // Set value of 89
8       await simpleStorageInstance.set(89, { from: accounts[0] });
9
10      // Get stored value
11      const storedData = await simpleStorageInstance.get.call();
12
13      assert.equal(storedData, 89, "The value 89 was not stored.");
14    });
15  });
16
```

```
Contract: SimpleStorage
  √ ...should store the value 89. (396ms)
```

Release your contracts & your web app

- Release Flow Chart (Explain how to release your contracts & your web app on the Ropsten Testnet & a public URL)
- The addresses of your contracts & their URLs on Etherscan
- The public URL of your DApp
- ...

For backend, I use “Infura” to release my smart contracts to Ropsten testnet to avoid tedious infrastructure settings.
The relationship is:



Truffle configuration is:

For frontend, I use Github Pages to deploy my React app very fast.
Tutorial ref: https://github.com/gitname/react-gh-pages

- ✓ Address of contract: 0xd474ca4e5ab236d7855f3a86f0c3b49a85c1fed0
- ✓ URL on Etherscan to check: https://ropsten.etherscan.io/address/0xd474ca4e5ab236d7855f3a86f0c3b49a85c1fed0
- ✓ Frontend UI: https://rayhungkao.github.io/simple-dapp/

Other Discussion

- Any notable design concern you find…

Smart contract development, testing, compilation, and migration are simple for developers. Many tools help a lot.

But when it terms to a Dapp, we lack a systematic approach to combine frontend. We rarely discuss how to implement a web UI to interact with blockchain, although we can somehow use Metamask instead.

Complicated web development is fatal, hard to test.