Biomedical Data Transfer Using Blockchain and Smart Contracts

Rensselaer Institute for Data Exploration and Applications (IDEA)

Fall 2019

Student: Roman Silen (silenr@rpi.edu)
Advisor: Oshani Seneviratne (senevo@rpi.edu)

ABSTRACT

The primary goal of this project is to develop a biomedical image sharing decentralized application using a blockchain platform such as Hyperledger or Ethereum, and the InterPlanetary File System (IPFS) for secure image storage. One of the ultimate goals of the project is to conduct a cost-effectiveness analysis for the system, such as assessing the economic tradeoff of what information should be transferred, whether the system lives up to its expectation in terms of cost-savings and patient outcomes, etc.

KEYWORDS

CID: Content Identifier
IPFS: InterPlanetary File System
NPM: Node Package Manager

INTRODUCTION

This project was conducted as an Undergraduate Research Project (URP) for Fall 2019. As a second-year student with no experience in blockchain technology, nearly every bit of this process was new to me. The goal for me was to gain as much of an understanding about blockchain as possible for the purpose of eventually construct a decentralized application for storing and sharing files using IPFS.

We decided to use Ethereum as our blockchain platform for our decentralized application. The idea is that any user can upload a file to IPFS through the application's interface, and a smart contract containing its hash and creation date is generated once the file is successfully uploaded. The user then receives a hash for accessing the file through IPFS.

When a file is uploaded to IPFS, a hash is generated based off of the content of the file. Network nodes then each store some content of the file as well as some indexing information in order to determine which node is storing what. When this hash is called, it searches the network for the nodes storing the content of the hash. This hash, known as a content identifier (CID), is unique to the content that it came from. Storing the content on multiple nodes makes sharing files much easier, as downloading files from multiple nodes at once rather than from one computer at a time is much more efficient and much less expensive.

The smart contract generated by the application holds important information about the file. Due to this sensitive data, it is imperative that the contents of the smart contract are not altered. Ethereum uses SHA-256 to generate its hashes, and this makes it computationally impossible to calculate the input from the hash itself. Thus once a smart contract is deployed, it cannot be changed.

This combination of the immutability of Ethereum and the accessibility of IPFS makes them the ideal tools for storing and distributing sensitive biomedical data. In the following sections, we will explore the step-by-step implementation of decentralized applications.

PROJECTS

System Specifications

- Windows 10 (64-bit)
- Oracle VM VirtualBox
- Ubuntu (64-bit)
- Ganache v2.1.1
- NPM v6.12.0
- Node v12.11.1

Creating a Blockchain

The original plan was to create a blockchain with smart contracts using the IBM Blockchain Platform extension on Visual Studio Code[1]. However, the IBM Blockchain Platform does not work with Windows Subsystem for Linux, so I had to find an alternate solution.

This was a short project[7] involving the use of Go and Postman to create a basic blockchain. The purpose of this project was to gain a general understanding of how blockchain technology works. Each block in this blockchain represents each time a person checks their pulse rate (in BPM). The struct `Block` contained the following fields: `Index` (position of data record in the blockchain), `Timestamp`, `BPM`, `Hash` (SHA-256 representation of current block), `PrevHash` (SHA-256 representation of previous block in chain). The web server was set up in Go, and Postman (see Figure 1) was used to add blocks to the chain.

I completed this project before installing VirtualBox, so this tutorial should work on any operating system. This was a good alternative to using the IBM Blockchain Platform in Visual Studio Code for the purpose of understanding the basics of blockchain technology.

User Profiles Application

This application[10] is one where any user can register their name and "status" onto the blockchain. The user's name and status creates a profile associated with their Ethereum wallet address. Once a user creates a profile, they can go back and edit their name and status through a MetaMask transaction. The smart contract used is designed to give each user control over their own information, where even the creator of the smart contract will not be able to have control over that information.

Figure 2 shows a list of all users and their statuses, and Figure 3 shows the interface for updating a user's information. I believe that this could be integrated into a file uploading application (see below) fairly easily, and for this project it makes sense to use it.

This tutorial was good for gaining insight into how Ethereum smart contracts interact with a user interface. It can also serve as a good base for creating a clean-looking application if you have little experience with UI/UX design.

File Upload Application

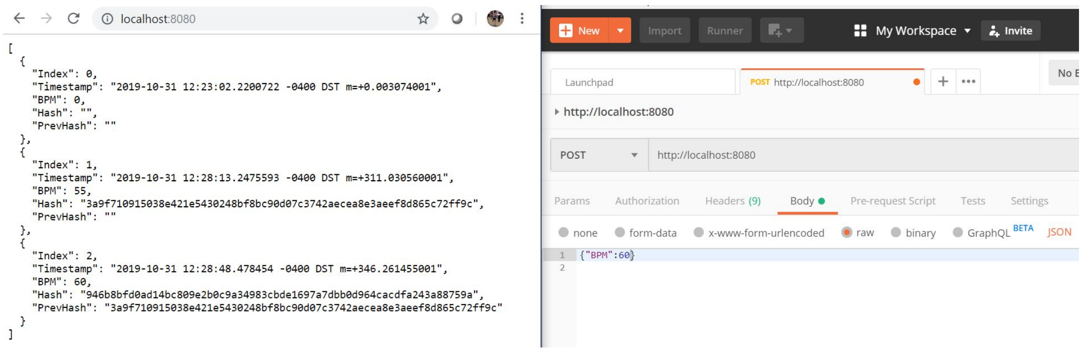
Creating an application for uploading files to IPFS was the ultimate goal by the end of the Fall 2019 semester. However, the combination of rapidly updating technology along with limited tutorials about decentralized applications made it difficult for someone with little to no experience to create such an application.

I attempted many tutorials (see "References" below), but most of them would give me an error in the last few steps. Most of the errors were related to outdated code or software version conflicts.

One of the tutorials worked on the first try[8], but when attempting it again there was an error with web3.js and the

JSON files for the project. Perhaps there was a file conflict when I tried to redo the tutorial that I did not catch, but I was not sure of how to fix this error. An image of the successful image upload from the first attempt can be seen in Figure 4. This specific tutorial was the closest I got to creating a successful file upload application.

IMAGES

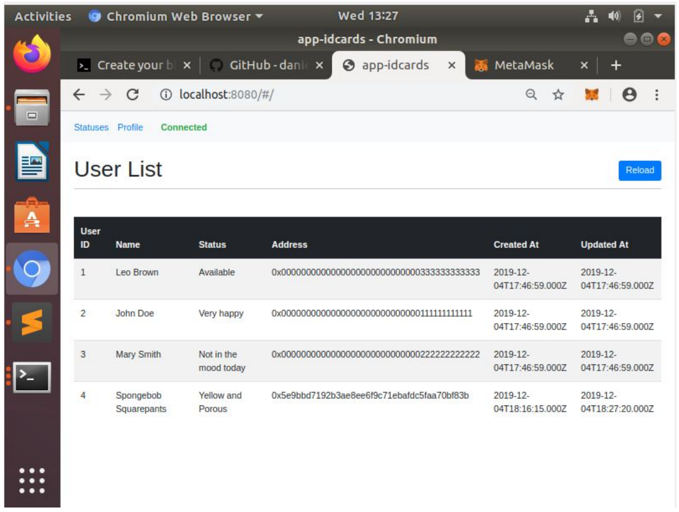


```
[
  {
    "Index": 0,
    "Timestamp": "2019-10-31 12:23:02.2200722 -0400 DST m=+0.003074001",
    "BPM": 0,
    "Hash": "",
    "PrevHash": ""
  },
  {
    "Index": 1,
    "Timestamp": "2019-10-31 12:28:13.2475593 -0400 DST m=+311.030560001",
    "BPM": 55,
    "Hash": "3a9f710915038e421e5430248bf8bc90d07c3742aecea8e3aeef8d865c72ff9c",
    "PrevHash": ""
  },
  {
    "Index": 2,
    "Timestamp": "2019-10-31 12:28:48.478454 -0400 DST m=+346.261455001",
    "BPM": 60,
    "Hash": "946b8bfd0ad14bc809e2b0c9a34983cbde1697a7dbb0d964cacdfa243a88759a",
    "PrevHash": "3a9f710915038e421e5430248bf8bc90d07c3742aecea8e3aeef8d865c72ff9c"
  }
]
```

POST http://localhost:8080

```
{"BPM":60}
```

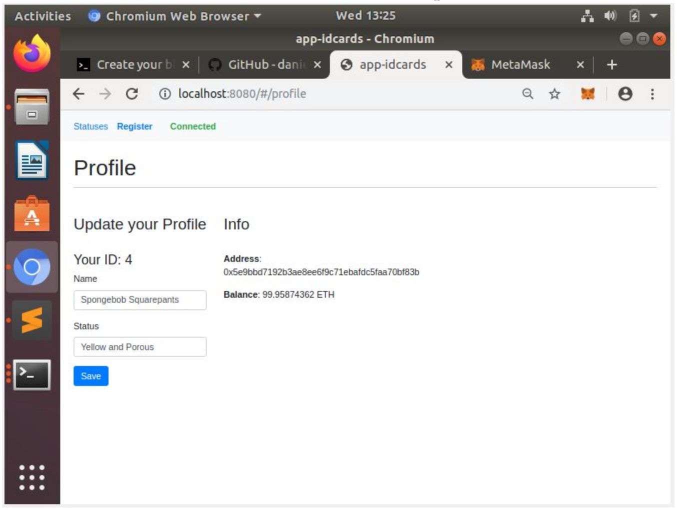
Figure 1. This screenshot demonstrates how blocks in the blockchain change as new blocks are added. The left side contains the values of each block's fields, and the right side displays the Postman interface.



User List

User ID	Name	Status	Address	Created At	Updated At
1	Leo Brown	Available	0x0000000000000000000000000000003333333333	2019-12-04T17:46:59.000Z	2019-12-04T17:46:59.000Z
2	John Doe	Very happy	0x0000000000000000000000000000001111111111	2019-12-04T17:46:59.000Z	2019-12-04T17:46:59.000Z
3	Mary Smith	Not in the mood today	0x0000000000000000000000000000002222222222	2019-12-04T17:46:59.000Z	2019-12-04T17:46:59.000Z
4	Spongebob Squarepants	Yellow and Porous	0x5e9bbd7192b3ae8ee6f9c71ebafdc5faa70bf83b	2019-12-04T18:16:15.000Z	2019-12-04T18:27:20.000Z

Figure 2. This screenshot shows information about each user. You can see that the updated status and address of "Spongebob Squarepants" is listed.



Profile

Update your Profile

Your ID: 4

Name: Spongebob Squarepants

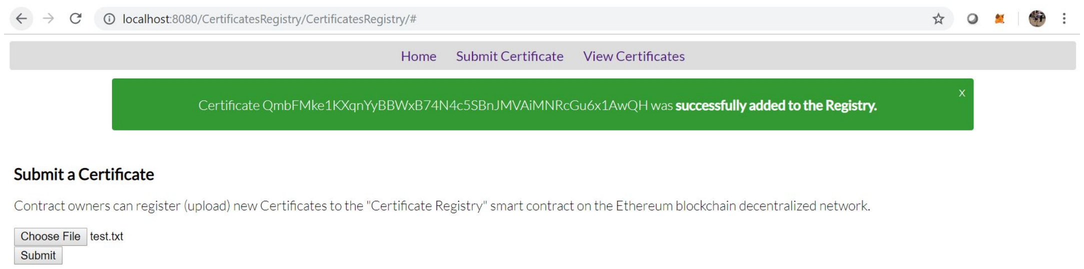
Status: Yellow and Porous

Info

Address: 0x5e9bbd7192b3ae8ee6f9c71ebafdc5faa70bf83b

Balance: 99.95874362 ETH

Figure 3. This screenshot shows the interface for which a user can update their information. As seen in Figure 2, once "Save" is hit, then Spongebob Squarepants' status will be updated to "Yellow and Porous".


localhost:8080/CertificatesRegistry/CertificatesRegistry/#

Certificate QmbFMke1KXqnYyBBWxB74N4c5SBnJMVAiMNRcGu6x1AwQH was **successfully added to the Registry.**

Submit a Certificate

Contract owners can register (upload) new Certificates to the "Certificate Registry" smart contract on the Ethereum blockchain decentralized network.

Figure 4. This screenshot is how the interface of the File Upload Application appears once the user has successfully uploaded a file to IPFS through the application.

CONCLUSIONS

Each project that I attempted to build was originally done on Windows, and I only started using VirtualBox during the last three weeks of the URP. Given more time using Ubuntu in VirtualBox, I may have been able to debug some of the issues encountered.

Having experience in building interfaces is definitely helpful for trying to build an application from scratch as well as debugging certain functions of these applications. If you have some experience in designing a web application, I would highly recommend trying to build the file uploading application from scratch after completing the user profiles application tutorial.

In the "References" section, I have listed each of the tutorials that I attempted and whether or not each tutorial was a success. The links are listed in the order that they were accessed, so my recommendation would be to follow the successful tutorials in the order that I went through them. I also recommend taking a look at some of the failed tutorials.

REFERENCES

1. HyperLedger Fabric and IBM Blockchain Platform (failed): *https://github.com/LennartFr/2019-current-blockchain-apps*
2. Dapp University Ethereum Smart Contracts (failed): *https://www.dappuniversity.com/videos/pTZVoqBUjvI*
3. Dapp University IPFS File Uploads (failed): *https://www.dappuniversity.com/videos/pTZVoqBUjvI*
4. Ethereum + IPFS + React.js DApp (failed): *https://itnext.io/build-a-simple-ethereum-interplanetary-file-system-ipfs-react-js-dapp-23ff4914ce4e*
5. Basic Solidity Tutorials (success): *https://www.dappuniversity.com/articles/solidity-tutorial*
6. Solidity Micropayments (success): *https://solidity.readthedocs.io/en/v0.5.12/solidity-by-example.html#creating-the-signature*
7. Go/Postman Blockchain (success): *https://medium.com/@mycoralhealth/code-your-own-blockchain-in-less-than-200-lines-of-go-e296282bcffc*
8. File Upload Application using Infura (success only on first try): *https://dev.to/nikolayangelov/workshop—create-decentralized-application-2b6m*
9. IPFS Image DApp (failed): *https://github.com/iwaldman/ipfs-image-dapp*
10. User Profile Application (success): *https://www.danielefavi.com/blog/create-your-blockchain-dapp-with-ethereum-and-vuejs/*

ABOUT THE STUDENT AUTHOR

I am a second-year student pursuing a dual degree in Computer Science and Mathematics with a minor in Quantitative Modeling in Economics. Before the fall semester, I had taken Data Structures, and during the fall semester I was taking Foundations of Computer Science and Computer Organization. Before this URP, I had some experience in using HTML/CSS and JavaScript syntax but not enough to confidently build a web application from scratch. My career interests for the future are in the area of using big data and machine learning in quantitative finance.