Due: 11/10/2018 by midnight

PROJECT 3: WORKING WITH ETHEREUM BLOCKCHAIN

1. Problem statement:

Blockchain is receiving a lot of attention as the next big thing in the emerging technologies. In this project, we will explore an installation of the blockchain on Center for Computational Research (CCR) by testing decentralized application deployed on the and also by designing and deploying applications on the blockchain.

2. Learning outcomes:

On completion of this project you will able to:

1. Explain the application development process on a blockchain.
2. Deploy an application on a blockchain.
3. Interact with an application deployed on a blockchain thus generating transactions.
4. Understand the structure and operational details of a blockchain: transaction, block and the chain of blockchain.

3. Preparation before lab

a. Read and understand this project document.
b. Make sure you have VPN installed and know how to use it. Only UB-enrolled students are allowed to interact with our blockchain.
c. Download the package provided for setting a blockchain node on your compute machine and deploy all the components needed.
d. Explore the blockchain technology by reading Bitcoin whitepaper [1] and Ethereum whitepaper [2].
e. You can work in groups of two or by yourself. Choose the right partner who compliments your abilities so that you can learn from each other through the process.

4. Problem Description

We have deployed a blockchain and several applications on it. You will work on this blockchain, once again in phases. Please read the documents we have referenced in the pre-requisite section and have a mental picture of what a blockchain is and the role it plays. We have provided in figure 1 a simplified diagram of the blockchain process or application flow. Discuss this with your group and friends and get a good understanding of this process.

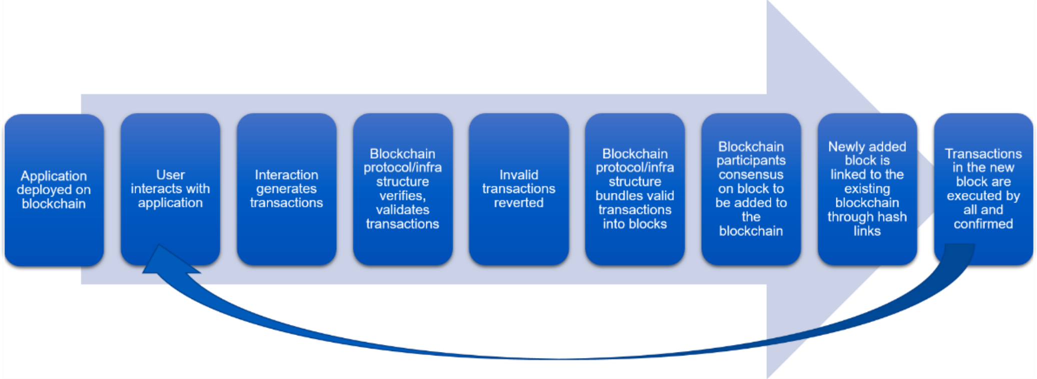

Application deployed on blockchain
User interacts with application
Interaction generates transactions
Blockchain protocol/infra structure verifies, validates transactions
Invalid transactions reverted
Blockchain protocol/infra structure bundles valid transactions into blocks
Blockchain participants consensus on block to be added to the blockchain
Newly added block is linked to the existing blockchain through hash links
Transactions in the new block are executed by all and confirmed

Figure 1: A simplified blockchain application flow ©2018 Bina Ramamurthy

Figure 1 shows the flow of a transaction from the time a user initiates through an interaction with an application deployed on the blockchain.

In this project, you will be user first, before you are a developer. Therefore, two major tasks for you are, (i) interact with the applications deployed on the chain, understand the blockchain application and (ii) design and deploy a blockchain application of your choice, (test it) and interact with it.

Phase 1: (30%) Install all the required software needed. Download and install the Ethereum blockchain node as instructed in the instructions in Appendix A and B. Now create a key pair identifying your peer node to the blockchain and an account number for coin transfer: instructions for these are given. You will be interacting with a Coin app that has been deployed already. Now you are ready to transact on the chain using the transactions for sending and receiving coins that you are given. You can send and receive coins using your account addresses. Have some fun with your friends in the class. Don't be afraid, you can interact as many times as you want. We want see significant load on the chain. This project is limited to ONLY students from this class. Please follow this rule strictly. We don't want the blockchain overloaded. We want to simulate a permissioned /private chain. Don't wait till the last minute. We may undeploy the application after sometime. Estimated time for installation – 1day, you can interact all you want. **Hard deadline: 11/26**
What to submit: Create a P3Phase1 folder: save couple of screen shots of your interaction and your account balance. Zip or tar it and submit it before deadline. Both partners have to submit the same files.

Phase 2: (35%) In this phase you will create a blockchain decentralized application of your own. Think of a good use case. Think of some cool but simple application that you can showcase for your future job interviews. Design and implement a smart contract solution using Solidity language. Read the Solidity documentation [3]. Test it on Remix Integrated Development Environment IDE [4]. Very important that the smart contract have a unique name on the chain: your will have to follow this naming convention: {first four letters of team members' last name} followed by "P32.sol". For example, Stephen Hawking and Carl Sagan are team mates, then the project, phase 2 would name the smart contract HawkSagaP32.sol. The project 3 phase 2 folder will have the .sol as well as screen shots of this sol working on Remix. **Hard Deadline: 12/2 Midnight**. **What to submit**? Create a P3Phase2 folder: Save the smart contract source code in Solidity and couple of screen shots of it working on Remix.

Phase 3: (25%) Next step is to implement deploy the exact use case you developed in Phase on to the CCR blockchain. This may require a simple HTML and JS front end. Save all the source in a folder P3Phase3 folder and submit. Make sure you provide a README as to how to assemble the code and deploy it on the blockchain. **Hard Deadline**: **12/9**.

Good design: (10%) Discuss the choice of technologies and UI details in your report. Comment you code appropriately. Provide a detailed design representation like figure 1, customized to your design and technologies you used. Directory structure, documentation and readme. Provide a readme file that provides details of how to run deploy and run your code. Feel free to include screen shots of a working application.

5. Project Implementation Details and Steps:

1. **Building systems for the blockchain**:
   This involves designing the solution for a problem and represent it using a block diagram or UML tool. May be state diagram or finite state machine and/or a class diagram.
2. **Working with UB CCR Blockchain:**

Try accessing this as soon as possible. Do not wait till the last minute. See Appendix A and Appendix B. Follow the instructions. No programming needed! Isn't that nice? You will interact with a Coin transacting application.

3. **Building a decentralized application***:*
   You will design and write the solution in Solidity language. This is a language similar in style to any of the popular high level language like Java and JS. Please refer to Solidity documentation for reviewing the language syntax and special data structures introduced by the language for code execution the blockchain using smart contracts.
4. **Study, understand and design a blockchain system**
   Review the functions and the working of the blockchain system described in Figure 1. Design the architecture, data structures, and functions needed to implement the system: Make the web interface simple and easy to use. Plan the layout well.
5. **Design, implement the blockchain system in phases:**
   We want to see incremental development steps. This also lets to plan your time so that you do not wait till the last minute to mash something together to satisfy the requirements. Also please DO NOT implement all phases at one time and submit one package. We want to observe the incremental development. We will take off points if we do not see the evidence for incremental development. This is a requirement.
6. **Deploy the integrated system and test it for phase 3:**
   The various components listed above should have been deployed and tested individually. In this step you will run the entire integrated system. You can create a good test data that will completely test all the components of your system.

6. Project Deliverables:

- A standard directory structure for the three versions of the software developed.
- A readme files documenting all the details to run the application and any special features. Don't forget to include screen shots of the application running.
- You may also submit other media such as a video of your application working (optional).

7. Submission Details:

Create a compressed deployable distribution of your project. Use the steps given in the document for project 1. Submit it online on the CSE server timberlake. You should include all the details about the technology requirements to deploy and use your software and also a readme file providing clear instructions as to how to deploy and use your software.

8. First steps:

Write up an abstract of 100 words with a suitable title. Submit it on timberlake, the CSE server before 11/19/2018 midnight.

9. References:

1. S. Nakamoto, Bitcoin: A peer-to-peer working Electronic Cash System, https://bitcoin.org/bitcoin.pdf
2. Ethereum Whitepaper: https://github.com/ethereum/wiki/wiki/White-Paper
3. Solidity Language: https://solidity.readthedocs.io/en/v0.4.24/
4. Remix Web IDE: **https://remix.ethereum.org/**

10. Good luck, start your work right now.

Appendix Setup Details, A, B and C.

Set up details:

The following are the prerequisites for this project. Make sure to install them correctly before proceeding.

Node js installation:

1. Install node js and npm using https://nodejs.org/en/. Make sure you select the **npm package manager** feature while installing node
2. In your terminal/command prompt check if node and npm were successfully installed using the commands
   a. node -v
   b. npm -v

Truffle Installation: You need to have truffle installed to interact with the application deployed on CCR

1. In the terminal/cmd install truffle using the command: **npm install -g truffle**
2. Check if truffle is properly installed using **truffle version**

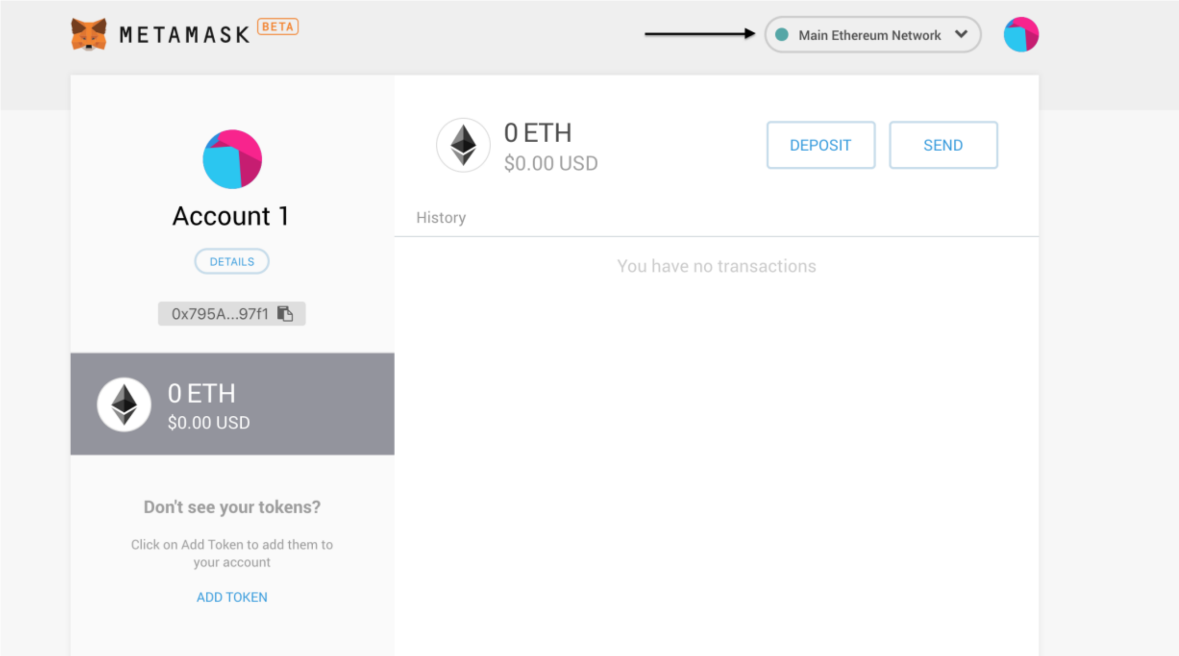
Appendix A:

How to connect to CCR?

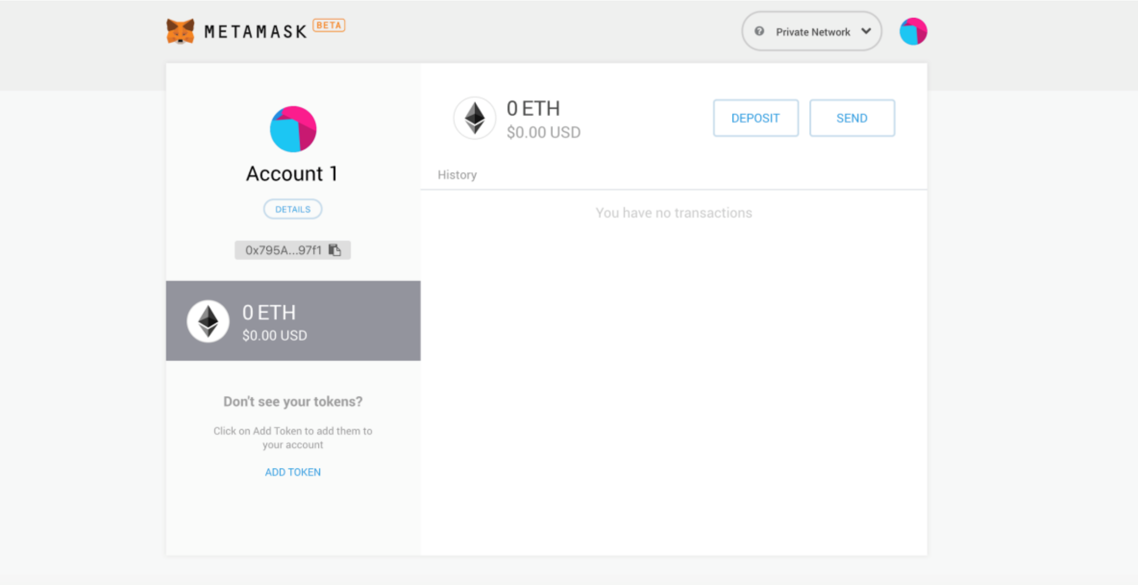
Once you have the prerequisites installed, the next step is to install metamask. Metamask is a browser plugin that allows your browser to interact with the blockchain.

Meta mask installation:

1. Go to https://metamask.io/ and install metamask. It will be added as an add on to your browser. **Please use chrome or firefox for better compatibility.**
2. Click on the metamask icon and click on try the new version.
3. Give a password and click on the Create button.
4. Proceed to the next step and accept the terms and condition.
5. You will be given with a secret phrase. **Store the secret phrase securely.** This secret phrase will be used to restore your account incase meta mask crashes.
6. In the next screen confirm the secret phrase.
7. Now change the network type by clicking top left corner (Main Network arrow mark) and choose Custom RPC.



8. In the New RPC url text box, enter the url **<# to be given>** and click on save.
9. You must see the top left corner bar changed to Private Network. Now close the new network creation screen. You should see something like the below image. You will see the following screen with 0 ETH.
10. Click on this icon and create a new account by providing an account name.

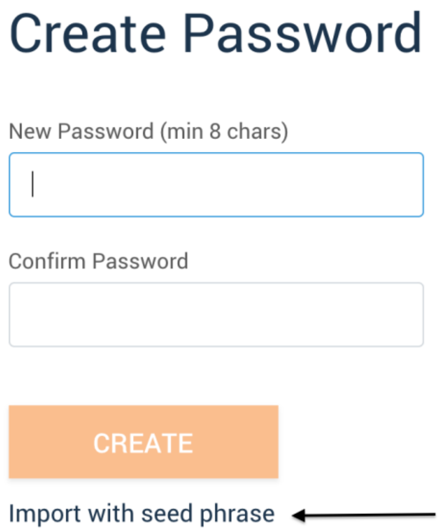


You have no transactions

Don't see your tokens?

Click on Add Token to add them to your account

11. Clicking on the details button, will display your account address. Store this account address. This is the account address to which money will be transferred.
12. Go to this google sheet and update your newly created account's address. Please do this within (**hard deadline**). After everyone has updated the sheet, we will give each team 100 ethers. After this everyone will be able to see 100 ETH in the accounts screen. (We will let you know once this step is done)
13. **Import seed phrase** : Incase metamask plugin crashes , please uninstall the plugin and reinstall again. After step 2 instead of clicking on the create button, click on the import with seed phrase and enter the seed phrase that you stored in Step 5. This will make sure your account is restored.

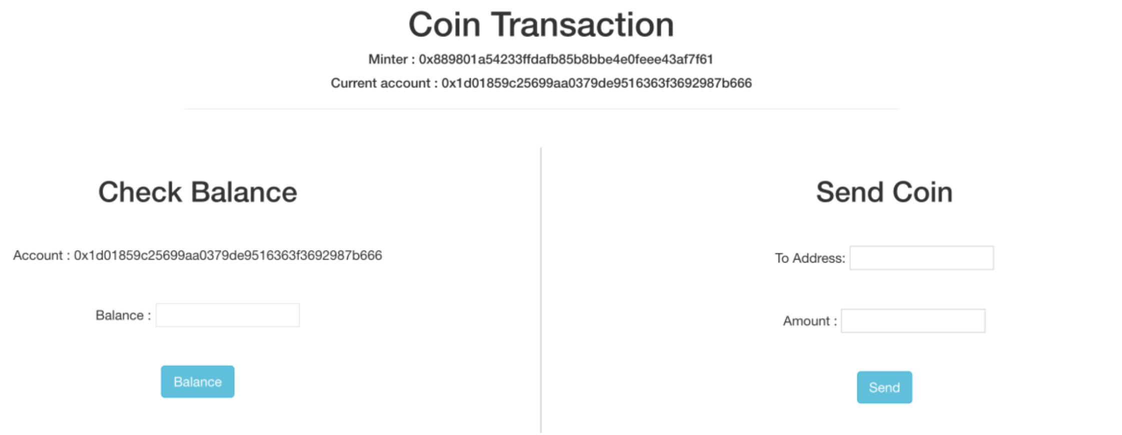


APPENDIX B:

How to work with the Coin Dapp?

Install and Run the Application:

1. Download the coin dapp application from the link provided <#**Link to download**>
2. Unzip the zip file downloaded from the link.
3. Fire up the terminal/cmd prompt, navigate into coin dapp folder and run
   a. truffle compile – This will make sure all the relative paths are set properly
   b. npm install – This will add all the dependencies for the application to run
   c. After successful installation, you are now ready to run the application
   d. npm run dev – this will give you a url, where your application is hosted
   e. Go to your browser and access the url. Now you are good to interact and play with the application. You should see a start up screen like this



Coin Transaction

Minter : 0x889801a54233ffdafb85b8bbe4e0feee43af7f61

Current account : 0x1d01859c25699aa0379de9516363f3692987b666

Check Balance

Account : 0x1d01859c25699aa0379de9516363f3692987b666

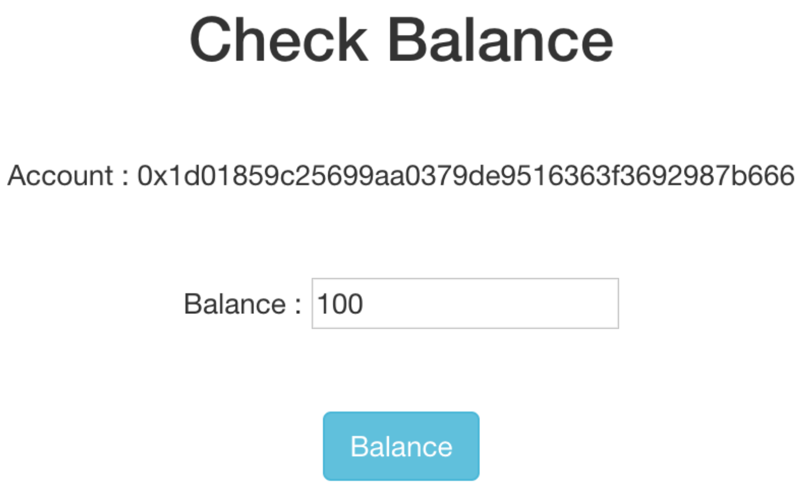
Balance :

Send Coin

To Address:

Amount :

   f. You can check your balance by clicking on the balance button. You should be able to see 100 coins as balance initially. Check the coin balance after each and every transaction you make.

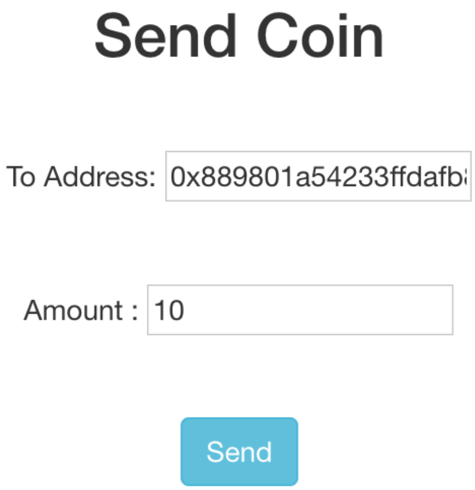


Check Balance

Account : 0x1d01859c25699aa0379de9516363f3692987b666

Balance : 100

g. Get your friends account addresses and play around by sending coins to them. You can enter the address in the “To Address” text box and the amount of coins to be transferred in the “Amount” text box (say 10 coins).

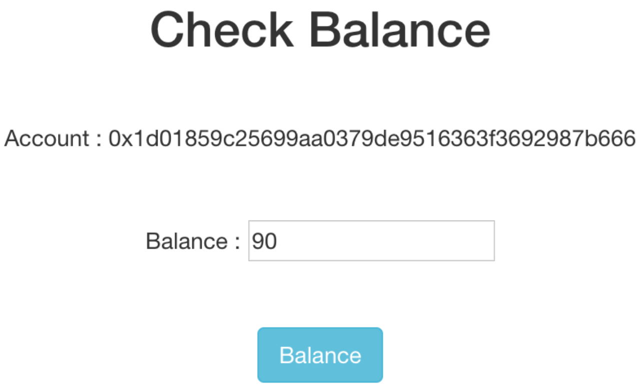


Send Coin

To Address: 0x889801a54233ffdafb

Amount : 10

h. Now again get the balance, you should see the new balance.



Check Balance

Account : 0x1d01859c25699aa0379de9516363f3692987b666

Balance : 90

APPENDIX C:

How to deploy your own application on CCR?

1. Download the base template from the link provided <# **Link to download>**
2. After you have tested your contract locally paste the contract code in the contract folder inside the downloaded code
3. Run **truffle compile** for compiling
4. Run **truffle migrate --reset --network live** for migrate the artifacts into the CCR network
5. Now you can interact with your contract via a frontend (use coin application for creating the frontend)