

(a) Describe your experience and problems encountered with the three steps. How you solve the problems if you did.

1.) Setting up a development environment

a.) The first three things I had to do were download Ganache, Metamask, and Truffle. Ganache gave me pre-funded accounts for local blockchain testing, which I then used to import an account onto Metamask to allow me to make transactions on the blockchain. This wouldn't work with real Ethereum, so this made it very easy to test my logic and see if it works, and it taught me the steps to developing real-world blockchain applications. Metamask connected my Ganache account to the blockchain through my front-end application of a grocery list. Truffle was used to create the contracts for the blockchain, which the Ganache account reached through Metamask. To send these transactions, I had to create a front-end application through Metamask's API, as I previously stated, and connect my Ganache account to the contracts.

b.) For my smart contract, I wanted to create a decentralized grocery list where users can add and remove items as they wish. My idea for the use of this would be for a relatively big family that may make two trips to the grocery store a week and may not know exactly what everyone wants or needs. This is still local and not useful in the real world. Anyways, to make this grocery list, I needed to make a contract that could write items to the list and remove them. To complete this, I created a struct that would be the basic building block of any grocery, which contained a unique ID, the name of the grocery, and whether I had gotten the item yet or not. Next, I had to create two events, which were writing the grocery to the list and removing it. The next thing to do was to create the functions to initiate transactions through the blockchain. These functions were the same as the events in terms of creating items and removing them. To initiate any of these transactions, users would use the front-end application which is shown further down in the document. Below is the picture of the smart contract in solidity:

```
contract GroceryList {
    uint public groceryCount = 0;

    struct Grocery {
        uint id;
        string content;
        bool completed;
    }

    mapping(uint => Grocery) public grocery;

    event GroceryCreated(
        uint id,
        string content,
        bool completed
    );

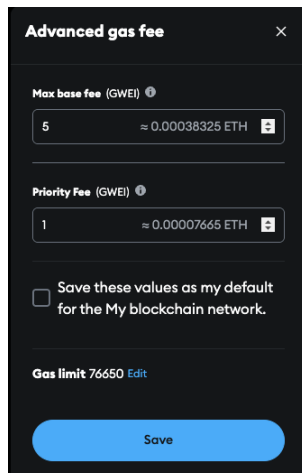
    event GroceryCompleted(
        uint id,
        bool completed
    );

    constructor() public {
        createGrocery("Example Item: Milk");
    }

    function createGrocery(string memory _content) public {
        groceryCount++;
        grocery[groceryCount] = Grocery(groceryCount, _content, false);
        emit GroceryCreated(groceryCount, _content, false);
    }

    function toggleCompleted(uint _id) public {
        Grocery memory _grocery = grocery[_id];
        _grocery.completed = !_grocery.completed;
        grocery[_id] = _grocery;
        emit GroceryCompleted(_id, _grocery.completed);
    }
}
```

- c.) To query and update assets on the blockchain, I used the front-end application through an app.js file to call the smart contract to write new grocery items and remove them through the contract functions. Once these functions were called, the metamask extension popped up and would request ETH to make the transaction. Once accepted, it would go through Ganache and take from the available balance and then log it in the transaction tab. I experienced a lot of problems along the way, and here they are:
- i.) Originally, I tried to remove/add items from the grocery list with MetaMask, but it said I did not have enough Ethereum to make the transaction. This was confusing because Ganache gave me a lot of fake Ethereum for local development and tests, and it still would not allow for the transaction. Not knowing exactly what this meant, I continued to try to lower the max base fee and the priority fee. I ended up putting both of these values to zero because I thought it did not matter, and would just simplify the process. However, I then ran into an RPC error because there was no transaction occurring. Thus, I figured out that I could just use low values and raise the gas limit through this Advanced gas fee, which resulted in success.



- ii.) Another problem I encountered was realizing that whenever I changed or edited my contracts, I had to migrate the changes with Truffle, like below:

Unset

```
~truffle migrate --reset
```

So, long story short, whenever you update your contracts, the only way to update what renders is to migrate the changes and reset the front end. After doing this, the changes will take effect.

- iii.) After a while of not being active on the web page my application would time out. You would think this would be a simple fix, just like running npm run dev again and wallah. But no. You have to restart your ganache app, make sure that your account is connected to the local host through metamask, and you have to migrate your contracts again. This took me a

lot of time to figure out simply because I had no idea what was wrong when it was working perfectly fine the previous day.

2.) Show the screenshots of your experience, including your query results if you succeed in getting query results.

a.) Here is a picture of the migrations that are done to deploy the contracts I created. As you can see it costs ETH which is taken from the Ganache account and then is deployed to the blockchain:

```
1_initial_migration.js
=====

Replacing 'Migrations'
-----
> transaction hash: 0x7417f3acf0e1fd1284081cc6572b5266b1c22e4db8679ad958e8f8bc876a72eb
> Blocks: 0       Seconds: 0
> contract address: 0xe6DEC065f40f101FfF38C0dc7494740Af1be9A34
> account: 0xa3Ec48644aDB1A3eeBF17B651118b4A5688c8C7F
> balance: 99.99588606
> gas used: 205697
> gas price: 20 gwei
> value sent: 0 ETH
> total cost: 0.00411394 ETH

> Saving migration to chain.
> Saving artifacts
-----
> Total cost: 0.00411394 ETH

2_deploy_contracts.js
=====

Replacing 'GroceryList'
-----
> transaction hash: 0x37307f6d9034cb66742b3212742ad31d3e8063d6bc4e9d43a2cd9dce9428cf0a
> Blocks: 0       Seconds: 0
> contract address: 0xFEE17E97DB45Ac874ADa833FebF98858F4e20DE0
> account: 0xa3Ec48644aDB1A3eeBF17B651118b4A5688c8C7F
> balance: 99.98538862
> gas used: 479197
> gas price: 20 gwei
> value sent: 0 ETH
> total cost: 0.00958394 ETH

> Saving migration to chain.
> Saving artifacts
-----
> Total cost: 0.00958394 ETH

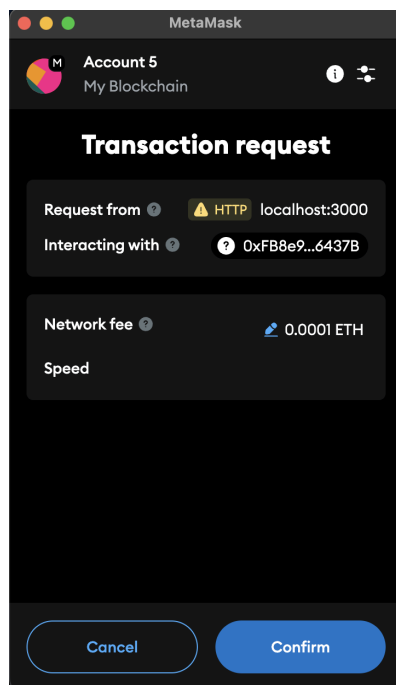
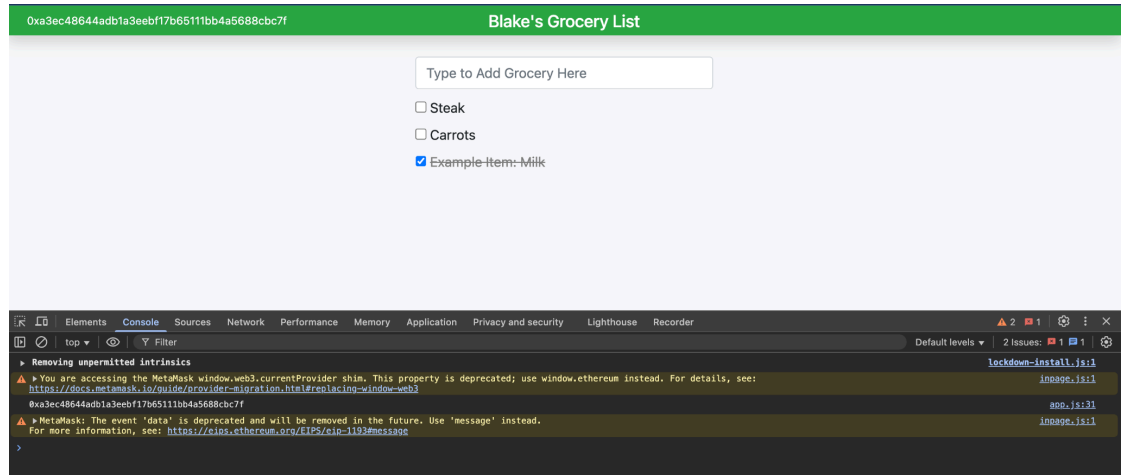
Summary
=====
> Total deployments: 2
> Final cost: 0.01369788 ETH
```

b.) Once the contracts are deployed to the blockchain, I was able to run the front-end application using the command in the terminal:

Unset

npm run dev

c.) Next, the link will be provided in the terminal, which you can copy and paste into your browser to see your frontend application. The picture provided is from after the original state, in which the only item is “Example Item: Milk”. However, I made some transactions through the blockchain to “remove”/cross out the example item and add a few other items. Any time you act on the contract, add or remove grocery items, the user would be prompted to pay for the transaction fee through Metamask. That picture is shown below the front-end.



d.) While doing this, Ganache was recording all the transactions through its interface, which is shown here:

TX HASH 0x70a03312fb97b557dfea251cab2e8a267f6cb4e5918a651f47990ebd13bfe969				CONTRACT CALL	
FROM ADDRESS 0xa3Ec48644aDB1A3ee8F17B65111Bb4A5688cBC7F	TO CONTRACT ADDRESS 0xFB8e95fe33cc2e34d6cc1F4178c290a69006437B	GAS USED 76662	VALUE 0		
TX HASH 0xe001dfdd21f394486da7e54a7f9056c5f56910df8c9f1cbcdf817f9ce2215095				CONTRACT CALL	
FROM ADDRESS 0xa3Ec48644aDB1A3ee8F17B65111Bb4A5688cBC7F	TO CONTRACT ADDRESS 0xFB8e95fe33cc2e34d6cc1F4178c290a69006437B	GAS USED 76686	VALUE 0		
TX HASH 0xea70946a6e38a6fd7f1f6de7489fb5d35dee30f0266f70f24c3ed7601d6684be				CONTRACT CALL	
FROM ADDRESS 0xa3Ec48644aDB1A3ee8F17B65111Bb4A5688cBC7F	TO CONTRACT ADDRESS 0xFB8e95fe33cc2e34d6cc1F4178c290a69006437B	GAS USED 52888	VALUE 0		
TX HASH 0xf77ec9e97b0d48bba48ed1e853c7e8e0076efc0dbd6285a927af7a7d6fa9b9e				CONTRACT CALL	
FROM ADDRESS 0xa3Ec48644aDB1A3ee8F17B65111Bb4A5688cBC7F	TO CONTRACT ADDRESS 0x69F3Bb11038ec383dbDd30945282575C3c84D8db	GAS USED 28575	VALUE 0		
TX HASH 0xbe1cb7f4d16cbbf633f1e8ffaa77410cc9e53d65e7018ee57fb69821ef9e0191				CONTRACT CREATION	
FROM ADDRESS	CREATED CONTRACT ADDRESS	GAS USED	VALUE		

3.) Analyze and discuss your results and your learning experience.

- From this experience, I learned how to create, deploy, and add smart contracts to the blockchain, create a front-end application to connect the smart contracts to user actions, and create a local developer-side application for blockchain in general.
- My learning process was honestly boosted by a YouTube video I found online at this link: <https://youtu.be/coQ5dq8wM2o?si=2K8xD3Ht0erDIC70>. I was able to follow it to set up my MetaMask wallet, connect it to Ganache, and how to use Truffle to deploy the contracts.
- Overall, I think I learned the most about how to connect to the blockchain and how it works in the real world. Even though my application might not make too much sense in the real world, I can see how many people believe in the decentralization of Bitcoin and its usefulness. Also, I understand how smart contracts are the fundamental aspect of some cryptocurrencies like Ethereum to help enable these decentralized applications.

## References

*Home | Solidity Programming Language*. (n.d.). Solidity Programming Language.

<https://soliditylang.org/>

*Home - Truffle suite*. (n.d.). <https://archive.trufflesuite.com/>

*Ganache - Truffle Suite*. (n.d.). <https://archive.trufflesuite.com/ganache/>

Contributors, M. O. J. T. a. B. (n.d.). *Get started with Bootstrap*.

<https://getbootstrap.com/docs/5.3/getting-started/introduction/>

freeCodeCamp.org. (2019b, June 28). *Build your first blockchain app using Ethereum smart contracts and solidity* [Video]. YouTube.

<https://www.youtube.com/watch?v=coQ5dg8wM2o>