Free Market P2P Energy Trading

Design Document

Team #41

Arun Sondhi: Embedded Engineer

Alec Dorenkamp: Software Engineer

Noah Eigenfeld: Software Engineer

Brendon Geils: Founder/Software Engineer

Jack Myers: Hardware Engineer

Joe Staudacher: Hardware Engineer

Client: Sodima Solutions

Adviser: Goce Trajcevski

sdmay18-41@iastate.edu

sdmay18-41.sd.ece.iastate.edu

Revised: 12/4/17

Table of Contents

List of figures/tables/symbols/definitions

1 Introduction

In this section, we discuss the problem our project is based around and our proposed solution. In section 2, we explain potential solutions to our problem, and outline our specific solution design. In section 3, we discuss methods for testing each major component of our design.

1.1 Problem and Project Statement

The primary aim of our project is to incentivize renewable energy generation from individuals and small businesses by facilitating peer to peer trading of surplus energy. By creating a free market environment for energy trading, individuals will think more about how they produce and consume energy, and will be inclined to generate energy of their own. With this new understanding and market accessibility, energy prices will fluctuate to be at parity with their true value, not just what the utility company dictates. A more detailed description of our implementation of this free market solution is found later in this document.

Our secondary goal is aiding in the decentralization of the power generation market. The interconnectedness of grids has already contributed to the reduction of blackouts, as one individual power plant or utility company is not solely responsible for all energy generation. At the time of writing, a Chicago grid can pull from a Toronto plant if they approach their capacity curve. Our system would aid this interconnectedness a degree further, as the power loss and cost to transfer energy a mile up the road would be less than the power loss and cost to move that energy from Toronto to Chicago. A more decentralized grid would help the overall grid to be robust to fluctuations, as the sources of energy would be widespread and independent.

The connection between decentralizing energy generation and our solution is clear. We hope that an open market will allow individuals to feasibly operate their own renewable energy sources like solar panels or wind turbines. If this goal is realized, energy will not be produced only at large plants. Because of this, consumers will be able to rely on a more diverse array of energy sources than those currently available.

A positive side effect of incentivizing renewable energy generation is that it will also incentivize slowing down climate change. Renewable energy usage is seen as one of the key ways to target this problem, but a large portion of worldwide energy does not come from this “clean” energy generation. To change this, the generation of renewable energy needs to be more accessible to individuals and businesses, rather than only those who have the resources and capabilities of a large energy company.

Our project has two major components: the development of an inexpensive and user friendly smart power meter and software to facilitate peer to peer trading of surplus energy. When these two stages of the project are complete, individuals using our hardware and software will be able to buy or sell surplus energy at significantly better rates than could be obtained going through a utility company.

A goal for the project that is still being defined is developing a marketplace where these peer to peer interactions could be facilitated. We hope to develop a way to automate this process so users would enter their desired parameters for energy transactions (maximum amount, times, etc.) and

the system would be automated such that these transactions could be financially optimized with minimal work from the user.

1.2 Operational Environment

The operational environment for our solution will be a relevant factor in the final implementation. Our smart power meter has to be able to withstand all of the conditions that existing power meters currently withstand. It will have to be able to survive in various weather conditions that naturally come with being a product that lives outside of the home.

On the software side, the "operational environment" will be the economic and political climate in which our solution is being used. There are many legal factors that could come into play with this kind of trading, like use of the utility company's infrastructure or trading between different cities, states, countries. These are all factors that would need to be explored in further detail if this project were to be expanded beyond a simple test environment with two nearby homes or small businesses.

As the project continues (potentially with future senior design groups), more focus can be put into refining the robustness of the hardware and making sure our software implementation integrates effectively with the economy and politics of the location where our solution is being used.

1.3 Intended Users and Uses

The users can be split into two groups, which we will refer to as "producers" and "consumers." The producers are the users who will supply excess energy that they produce into the system. Producers look to maximize the profit that they can create from producing energy, and our tradable energy market will enable them to do just that. In order to best serve these users, we are minimizing the transaction costs and maximizing the ease with which they can find buyers for their energy. The consumers are the users who will be consuming the excess energy that producers create. We can best serve them by minimizing the transaction costs and making it as easy as possible for them to find producers whose energy they can consume.

The intended use can therefore be described as incentivising personal generation of energy by creating an accessible market for both producers and consumers. Ideally, intended users would eventually include anyone with an electrical service connected to their home or business but in the current utility climate, the intent is simply to attract as many users that already have direct generation installations as possible. More producers in our user base will allow us to obtain more consumers by making the market more competitive and attractive. In due course, the flow of consumers to the marketplace will encourage more people to become producers and install their own direct generation setups. This circular growth will eventually lead to us reaching our intent of widespread incentivisation of direct generation.

1.4 Assumptions and Limitations

Because our goal is to make an impact on large scale issues like the energy market and climate change, it is important that we set certain restrictions about what aims we can realistically hope to achieve with our project. The following list gives the most important of these assumptions and limitations regarding the scope of our project.

Assumptions:

- Enacting the distribution of power after our transaction is completed is outside of the scope of our project, including the new power equipment that could be required for this distribution
- For a full implementation of our project, an agreement will have to be completed with the utility company owning the power infrastructure so they will allow these transactions to take place
- Failing to reach an agreement with a regular utility, an agreement will have to be completed with a developer of a subdivision, whom would typically own the power infrastructure of said subdivision
- The level of testing that we will complete will be within an individual municipality, so interstate/international trading laws will not be applicable

Limitations:

- The cost of the IoT smart power meter must not exceed that of the average power meter used in Ames, IA (the area of testing)
- The purchase of all hardware components and software licenses must be approved by our client and must not exceed the amount of funds they have allocated for the project
- The cost for user operation of our smart power meter must be minimal so as to make the implementation worthwhile for the customer

1.5 Expected End Product and Deliverables

The basic deliverables for this project can be boiled down into the following three items.

IOT Smart Meter - Estimated delivery: March 15

An “internet of things” capable smart meter will need to be installed at a user’s property to read the flow of energy into their home/building. This smart meter will be connected to the internet, and will interact with the energy marketplace to enact transactions. The smart meter will be able to verify transactions of power over set periods of time, as determined by the agreement between the buyer and seller in the transaction. The goal for the end of this semester is to have functioning basic communication between the smart meter and the web application. By the end of the year, we will develop a working prototype that is able to verify the completion of a transaction, connect to the web application, and display vital information to the user via a user interface directly on the physical meter.

Energy Marketplace Implementation - Estimated delivery: April 1

Power transactions will be made and recorded using a MongoDB database which is manipulated by a marketplace controller on the backend of the web application. This marketplace controller will be manipulated using the web application. A stretch goal is to implement a transaction matchmaker, which would automatically create and accept transactions based on predetermined criteria from the user.

Web Application - Estimated delivery: April 15

Users will manage their power transactions through an easy-to-use web application, which will interact with the energy marketplace through API calls, receiving and changing information through a MongoDB API. This application will allow the user to monitor and learn from their energy use and production, and will also allow users to buy and sell energy through a marketplace interface. Users will also be able to download personal usage and production statistics through the web application. The web application will include login/account creation capability and viewing transaction history as well.

2 Specifications and Analysis

We have researched various ways to go about solving our problem, the most significant of which are detailed below.

2.1 Implementation Options

Software

1. Ethereum Approach

   Using the open-source ethereum (blockchain) platform to develop smart contracts for buying and selling energy. Using ethereum we can utilize their stable Solidity language to implement our smart contracts. The advantage here is there are many projects built with this stack allowing for more resources and support.

2. Hyperledger Approach

   Using the open-source hyperledger (blockchain) platform, which is newer compared to the more established ethereum approach. Hyperledger is supported by larger organizations such as IBM. We believe this will allow the technology to stabilize long term with the backing of a large company compared to the burn-out many open-source projects that lack an organization have seen.

3. Traditional Marketplace Approach

   Using a traditional marketplace instead of a blockchain implementation will allow us to provide cheaper transaction rates to our customers. This is because paying for each blockchain transaction is a fairly expensive overhead ($0.20-0.40 in the last 6 months) to selling energy. Implementing a traditional marketplace will also be more simple and easier to customize.

Hardware

1. Raspberry Pi Approach

   This approach would use GPIO to continuously ready energy input and output. The data will then be used for buying and selling. Raspberry Pi has built in IoT capabilities which provide the needed support for transactions.

   In the prototype form, the Raspberry Pi will non-intrusively connect to the existing power buses, serving as a smart meter to connect to the Mongo database in addition to the existing traditional meter. Current and power usage will be tracked and data will be transmitted to the server.

2. Microcontroller/custom PCB Approach

   This approach is functionally identical to the Raspberry Pi, but built with a different set of components. Both will record and send data for buying and selling. This approach would require more work, but it may allow us to optimize power consumption and only use the hardware that is vitally necessary for the functionality of the meter. The options that we have explored on this front include sticking with standard Internet connectivity, local connections such as Bluetooth or Zigbee, or cellular connectivity. Internet connectivity is the implementation strategy with which our team is the most familiar, but the other options discussed could provide benefits in ease of configuration for the user. The pros and cons of each of these solutions will be explored in more detail as we continue our designs and prototyping into the second semester of senior design.

2.2 Proposed Design

For our project to be successful, there are a series of requirements that must be met. These are analyzed in detail below.

Functional requirements:

1. *An IoT Smart Meter device*
2. *Web app for management of transactions*
3. *API for communication between the smart meter and the web application*

Non-functional Requirements:

1. *Ease of setup*: Any user must be able to easily install and configure our hardware/software.
2. *Portability*: The web application must be usable on various platforms.
3. *Robustness*: The hardware must be able to withstand environmental conditions and be able to respond to signal loss and power outages. The software must be tested to handle edge cases and avoid fatal errors.
4. *Scalability*: The hardware and software must designed in such a way that it could handle a large network homes that would be required for a full implementation of our design.
5. *Code quality/documentation*: In order to fully achieve the aforementioned requirements, we must write code that is understandable and well-documented.

IoT Smart Meter

Based on research about products that are readily available for a reasonable cost, we determined that is was in our best interest to begin our development of the smart meter on a module that already has basic functionality like a Raspberry Pi. We will focus our efforts in this preliminary testing on the network and communication capabilities that will require us to learn the Raspberry PI's general setup and software libraries. Once we are able to get this version of the hardware fully functional, we will assess this form of the solution and determine if it is acceptable. Depending on the results of this assessment, we may move on to trying to create the hardware with a custom PCB if time permits. Having two versions of the hardware implementation will be the most effective way for us to compare these solutions and determine which is the best one to choose for the long term, taking into account the cost and effort that went into creating both. For either implementation strategy, the basic block diagram describing the major components and functionality of the IoT smart meter is shown in Figure 1.

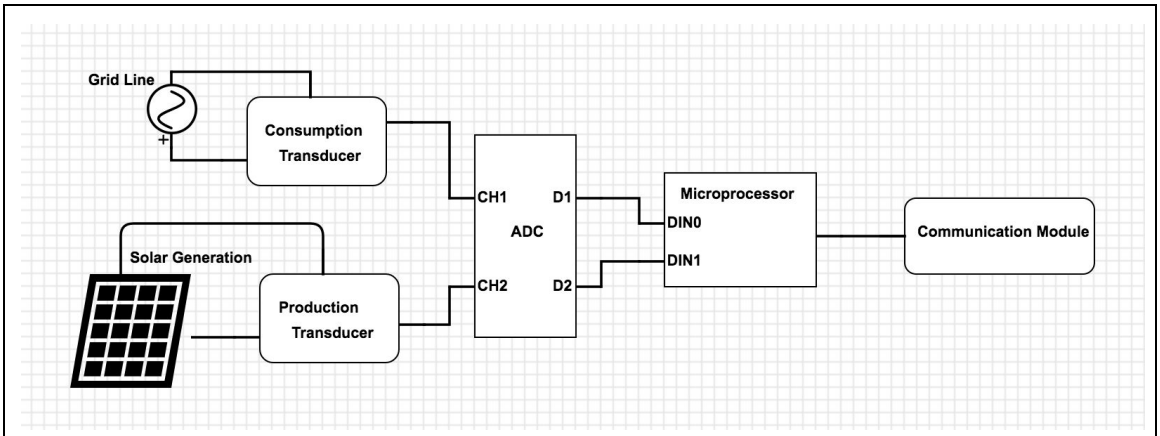

Figure 1: A top-level block diagram of the smart meter basic functionality

Web Application and API

All relevant user data, including account information, energy usage statistics, and transaction details, will be stored in a MongoDB database on AWS. Our web application will interact with our MongoDB database via an API. These API method calls, along with some additional backend functions, will allow the user of the web application to view information on their energy usage and production (from multiple smart meters if necessary), and will allow users to search for, buy, and sell energy. A component diagram of the web application can be seen below in Figure 2.

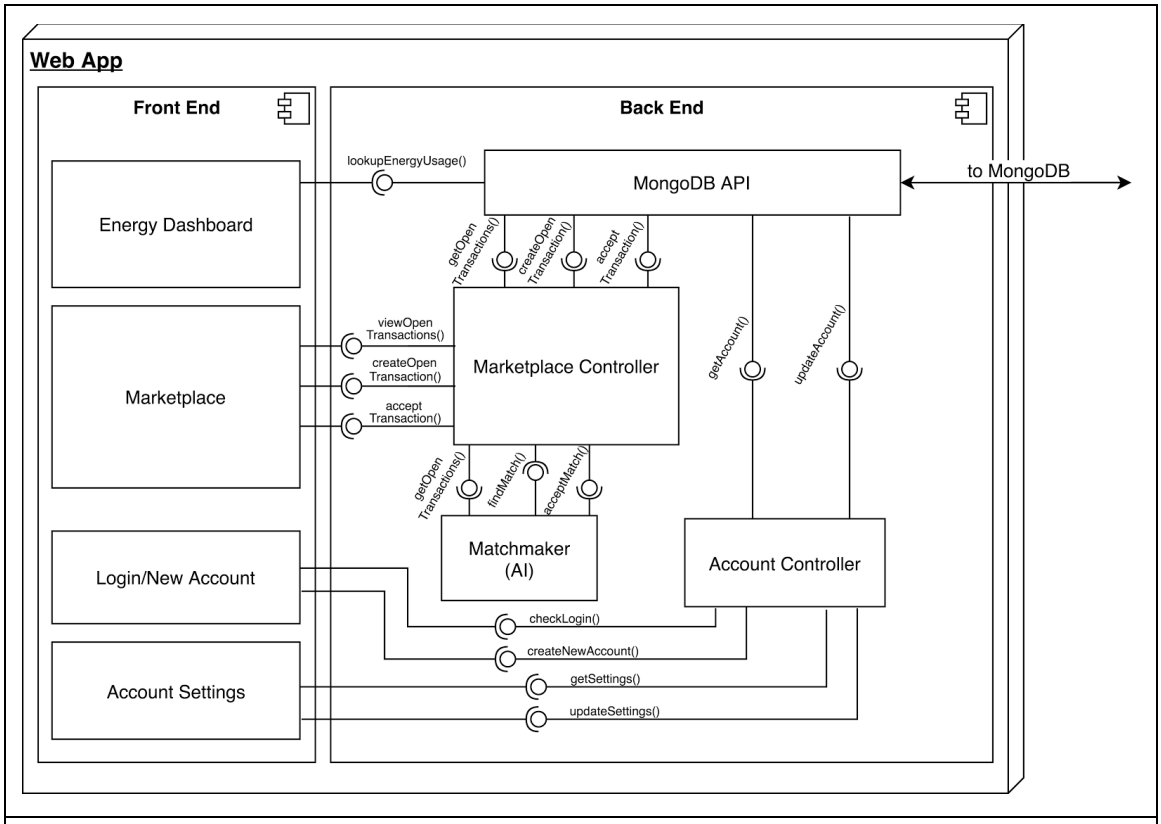

Figure 2: A component diagram of the web application

Overall Design

Details on the overall design and development process of our project can be seen in Figures 3 and 4 below. Figure 3 shows an overview of our proposed system. Users will install smart meters on their property where the meters can read the current. The smart meters will communicate with the web application, and all of the user information will be stored in a MongoDB database. Users can view their information and make energy transactions by accessing the web application through any browser.

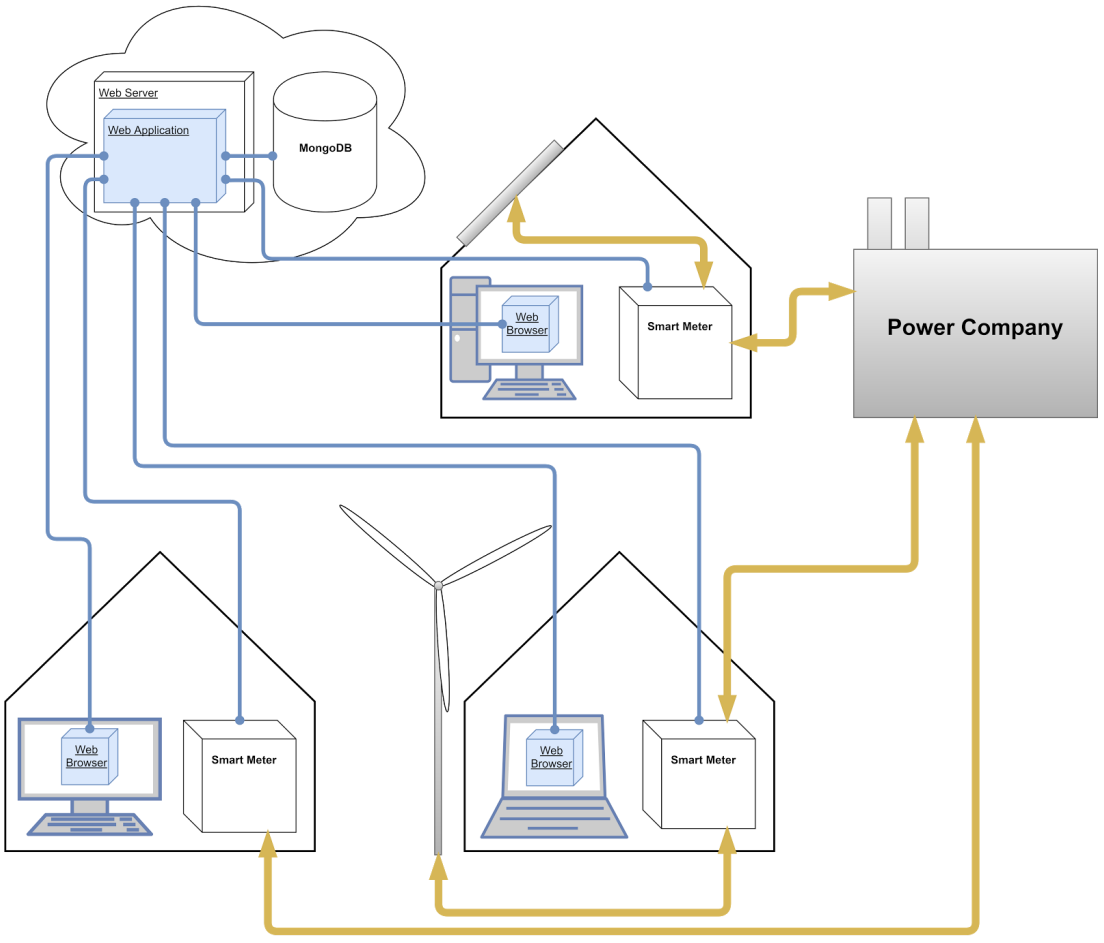

Figure 3: An overview of our proposed system

Figure 4 shows our proposed DevOps deployment. We have a production environment/instantiation of the application that all users will see and use, individual environments for each of the software developers on the team to work in, and a staging environment for testing changes before pushing out to the production environment. The user information will be stored in MongoDB databases in Amazon Web Services (AWS), and the web application is deployed on Heroku.

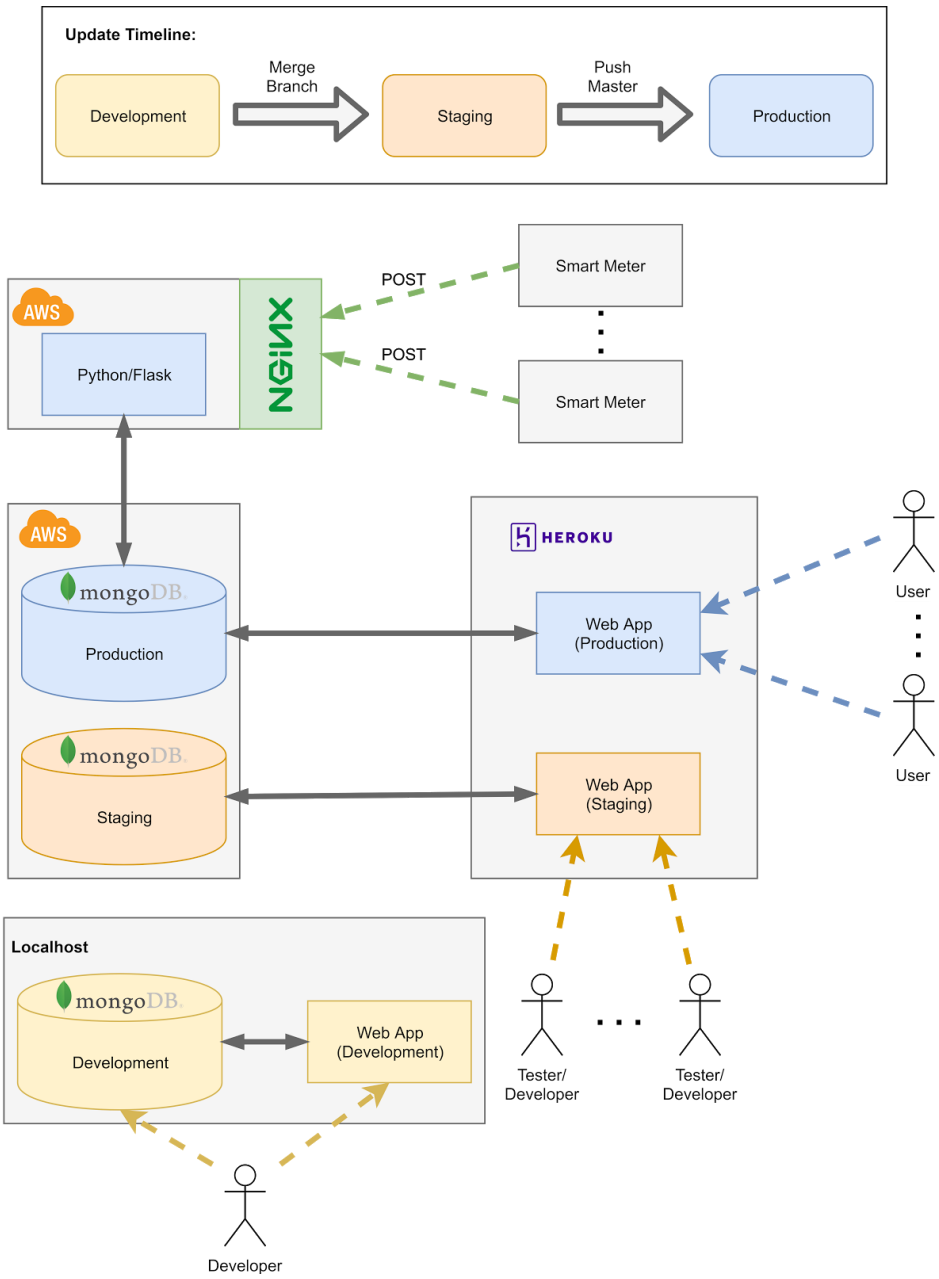

Figure 4: A DevOps diagram illustrating the design and development process of the project

2.3 Design Analysis

In section 2.1, we discussed some of the various implementation options that we explored and the pros and cons of each. In this section, we will discuss the specific implementation strategies that we have chosen to use at this point and some of the challenges that we expect to face with each.

For our software design, we are implementing a web application using the MERN stack, an open source end-to-end dynamic application framework. This MERN stack includes a MongoDB database for data storage and manipulation. Express is a back end web application framework that runs on top of NodeJS. React is a JavaScript library that allows us to build a dynamic user interface. Finally, NodeJS is an open source server framework. Although this stack is very robust and results in clean looking applications, there are some risks associated with it. The most notable of these is the relatively small amount of documentation for this software stack, since it is a fairly new stack. This means that debugging issues will take more effort than if we were using less modern, but more tested, technologies. The app is deployed using Heroku, which makes running our NodeJS app fairly straightforward.

For the hardware design we have decided to use a Raspberry Pi in conjunction with a custom current sensor to create the IoT Smart Meter device. The Raspberry Pi was chosen over a standard Arduino because the Raspberry Pi is a more powerful device and the fact that members of our software team will not necessarily need as deep of a knowledge of embedded system programming as they would with an Arduino. As work goes on, we may decide that we want to create another prototype with an Arduino or PCB in order to reduce price and power consumption for the end user. For the moment, we see Raspberry Pi as the best option because it should be easier to debug and troubleshoot while we are in the early stages of prototyping. The Raspberry Pi will also have a built-in Ethernet port which will make the networking that is necessary for our device much easier.

For the current sensor, we are using the MASTECH MS3302 AC current clamps (with a voltage and current range within our required operating conditions) to a signal processing circuit and then feeding that information to the Raspberry Pi for networking. We decided on this method of implementation because of its relative simplicity and the fact that it will allow us to compartmentalize the work for different groups of team members. The issues we foresee with this method is that it may not be streamlined enough for a final prototype. Cobbling the various different elements together could result in a clunky prototype but at this stage, but for our initial prototype, our main aim is functionality.

3 Testing and Implementation

Successful and efficient testing is necessary to implementing our solution correctly. How we have and will test our solution, including our current progress, is detailed below.

3.1 Interface Specifications

There are a few interfaces in our design. The first is between the current sensor and the GPIO of a Raspberry Pi. The Raspberry Pi will be keeping track of the energy consumption rate of an entity via the current sensor. In order to test this interface, the sensor's output data will be verified followed by unit and functional testing of the current sensor and Raspberry Pi together. It is imperative that this data is accurate, so thorough testing will be necessary. The Raspberry Pi will then send information to the MongoDB using a Nginx server. This interface can be tested by viewing the data from the sensor and comparing it to the values in the MongoDB database after the information is sent. Finally, the web application interfaces with the information stored in the database. The web application will both send and receive data from the database, so this will need to be unit and

functional tested as well to ensure that the user has all the functionality that is desired, as mentioned in section 2.2, Web Application and API.

3.2 Hardware and Software

We will implement several levels of testing including unit, integration, and functional testing.

Hardware Tests

The hardware element of the project will be the smart meter. Within the smart meter, it will consist of two main parts: the current sensor and the Raspberry Pi. Both of these parts will be used in the testing phase because they are critical to the success of the project. The best way to ensure that both of these parts function the way they need to is to test them.

The current sensor is a device that will be attached to a home or business in addition to, or in place of, their utility meter so it can measure the power being used by the building. This can be tested by using the lab equipment within the university for the initial and intermediate stages. For the final stage of testing, a test circuit of a few homes could be used.

The Pi will be used as a gateway between the current sensor and the blockchain software, sending and receiving the necessary information for each element. It will be tested in conjunction with the current sensor by feeding known values to and from current sensor and checking to verify if the information provided by the Raspberry Pi is correct.

Software Tests

There are three software components that need to be tested. These include the MongoDB database and functions to add and change data in it, the user web application, and full integration between these parts and the smart meter. The database implementation will be the first major component to test. It is the backbone of the energy trading system that we are establishing, and will need to be rigorously tested with test payloads and changes to ensure that the transactions are stored and work as intended. When we can ensure the basic functionality of the database,, the development and testing of the web application can commence.

The exact details of what the web application will contain are still being deliberated, but will likely include a dashboard within which the user can monitor their energy usage and production and a marketplace to sell and buy energy. Like all web pages, its functionality, usability, and performance will be tested to ensure that everything works as intended. During development, each feature will be carefully unit tested to ensure that there are not any simple errors preventing the web page for functioning properly. Its ease of use can be tested by many users, ensuring that it is easy to use for all potential users, not just blockchain experts.

The last thing to test will be the communication between the smart device, database, and web application. While integrating these three components, basic testing will be done, but the most extensive testing will be done by creating some basic end-to-end test cases that test the performance of the project as a whole. Only once these test cases are successfully completed can we be confident in the software performance of our project.

3.3 Process

Since we have yet to complete testing on each component in our system, we will focus on the tests we plan on using once we reach that milestone. Our initial tests will consist of white and black box testing of each component. We will first unit test the current sensor followed by unit testing the Raspberry Pi related methods. We will then conduct integration tests to ensure the current sensor and Raspberry Pi are communicating correctly.

The second stage in testing will be to unit test the API methods written to interact with the MongoDB database. This will be followed by completing unit and functional tests on the web application. At this point integration testing for the entire system, from meter to database to web application, can be conducted. In general this testing will be very similar to the previous process however due to the nature of dealing with a live environment, we will simulate some aspects. For example we will simulate several users with smart meters and test the ability to purchase, transfer and verify the transaction.

3.4 Results

We have done extensive testing on a variety of project components. We were just recently able to read current from power lines using current clamps attached to our Raspberry Pi. A clean, but barebones, version of our web application is visible, and is reading mock data from the database. Some correct API functions between the application and the database have been created as well.

We have also done extensive tests on the Ethereum platform to study feasibility, which have shown that we can create a tradable energy cryptocurrency that could be used on a blockchain. However, we decided relatively late in this semester that we are going to use a traditional marketplace and forego using a blockchain implementation of the marketplace because of the high costs and implementation complexity associated with Ethereum.

4 Conclusion

Money is the primary motivator for many making energy decisions, and as of right now, there is not enough of an economic incentive to prioritize the generation of renewable energy. We hope to produce this economic benefit by providing a way to make or save money from renewable energy via a marketplace implementation of peer to peer energy transactions.

We determined that our solution would require three main components:

1. IoT smart meter: for reading energy usage and verifying transactions
2. Marketplace functionality: for facilitating secure transactions
3. Web applications: for allowing users to interface with our system

We have created a working prototype using a Raspberry Pi for data processing and communication and an external current sensing module. Along with this, we want to complete basic functionality on the web application in order to test our system on a small scale. As time progresses, we will

improve our system by learning what works and what does not, refining smaller aspects after building a solid functional foundation.

While there are groups like Grid+, LO3, and ConsenSys that have already made strides towards a similar solution, we feel that our team is starting our project at just the right time. We are able to learn from the mistakes from our predecessors by taking what they would have done differently and actually doing it differently. While we are not the first to work on this type of project, we are early enough that we are not fighting against any other groups that are dominating or monopolizing the market. With the structural background that we have established in this document, we hope to be able to develop a solution that meets the needs of the users at hand and takes strides towards increasing the worldwide consumption and generation of renewable energy.

4.1 Acknowledgement

We would like to thank those who have made contributions to the project, outside of the members of our team. First and foremost, we would like to thank our adviser, Dr. Goce Trajcevski, for his technical assistance and comprehensive guidance throughout the project. We would also like to thank Sodima Solutions for providing the funding for the required hardware for the project, as well as providing the overall idea of the project. Lastly, we would like to express our gratitude towards the faculty of Iowa State University for their support in giving us the technical background and knowledge for us to handle a project of this scale. Without the support of these individuals and organizations, our project's success would not have been possible.

4.2 References

[1] "Bitfinex." *BitFinex*. N.p., n.d. Web.

[2] "ERC20 Token Standard." *ERC20 Token Standard - The Ethereum Wiki*. N.p., n.d. Web.

[3] Geils, Brendon. "Bgeils/pwr-blockchain." *GitHub*. N.p., n.d. Web.

[4] Geils, Brendon. "Pwr.company." *Pwr.company*. Blockchain News, n.d. Web.

[5] Lumb, David. "This New York Project Fuses Energy Microgrids With Blockchain Technology." *Fast Company*. Fast Company, 06 May 2016. Web.

[6] Morgen E. Peck and David Wagman. "Blockchains Will Allow Rooftop Solar Energy Trading for Fun And Profit." *IEEE Spectrum: Technology, Engineering, and Science News*. N.p., 01 Oct. 2017. Web.

[7] Sebnem. "Token Model for Energy - Part 1: Review of the Power Ledger Token Model." *Medium*. Medium, 12 July 2017. Web.

[8] "PowerLedger Token Generation Event." PowerLedger Token Generation Event, powerledger.io/.

[9] Ji, Ling. "Global Electricity Trade Network: Structures and Implications." *Public Library of Science,* 9 August 2018. Web. 22 November 2017.

[10] Upton, Eben. "Raspberry Pi User Guide." *Wiley and Sons,* 2012. Web. 10 October 2017.

5 Appendices

Here, we provide any figures used in this document, as well as several from past iterations of this document.

5.1 Current Design Diagrams

Hardware Block Diagram

Figure 1 is duplicated below. It shows the functionality of our smart meter design. For more information, see Section 2.2: Proposed Design.

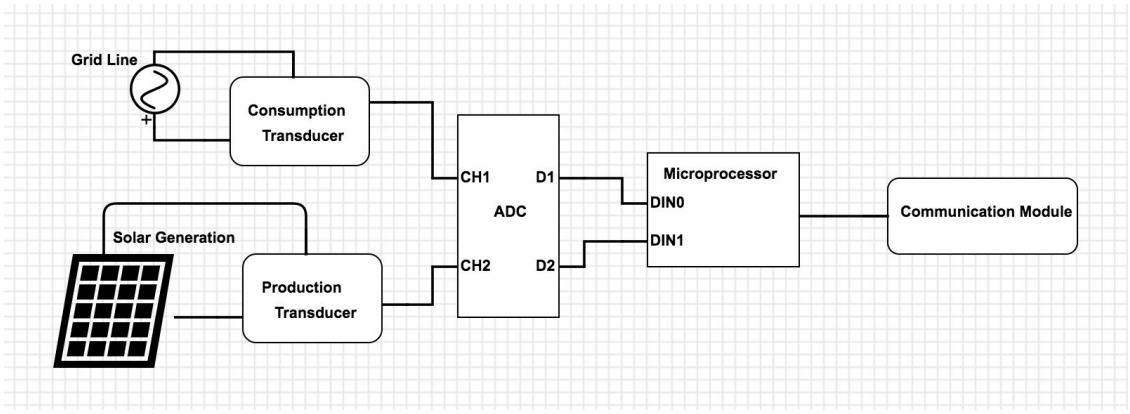

Figure 1: A top-level block diagram of the smart meter basic functionality

Updated Web App Macro-Component Diagram

Figure 2 is duplicated below. It shows the overall components in our web application design. For more information, see Section 2.2: Proposed Design. For the previous design, see Figure 6 in Section 5.2: Previous Design Diagrams.

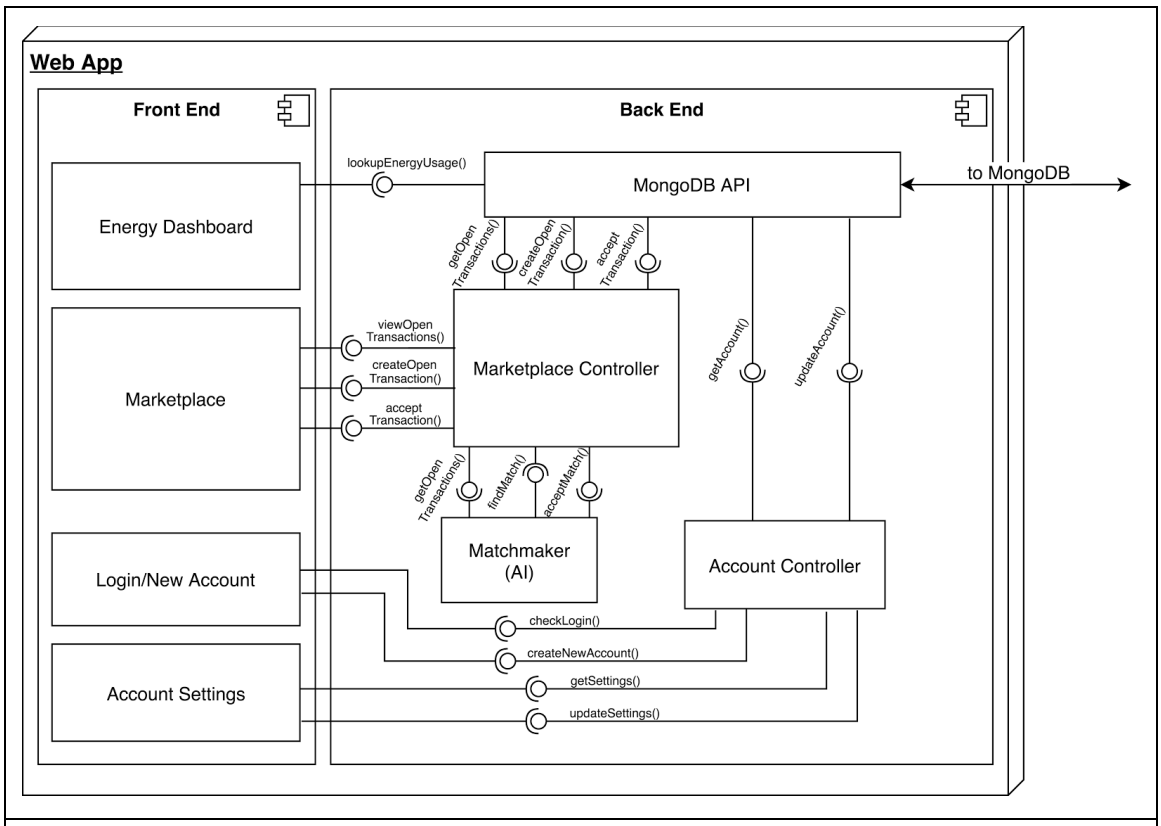

Figure 2: A component diagram of the web application

Updated System Overview Diagram

Figure 3 is duplicated below. It shows our current plan to implement our system using smart meters and a web application/MongoDB deployment for our energy marketplace. For the previous design, see Figure 5 in Section 5.2: Previous Design Diagrams.

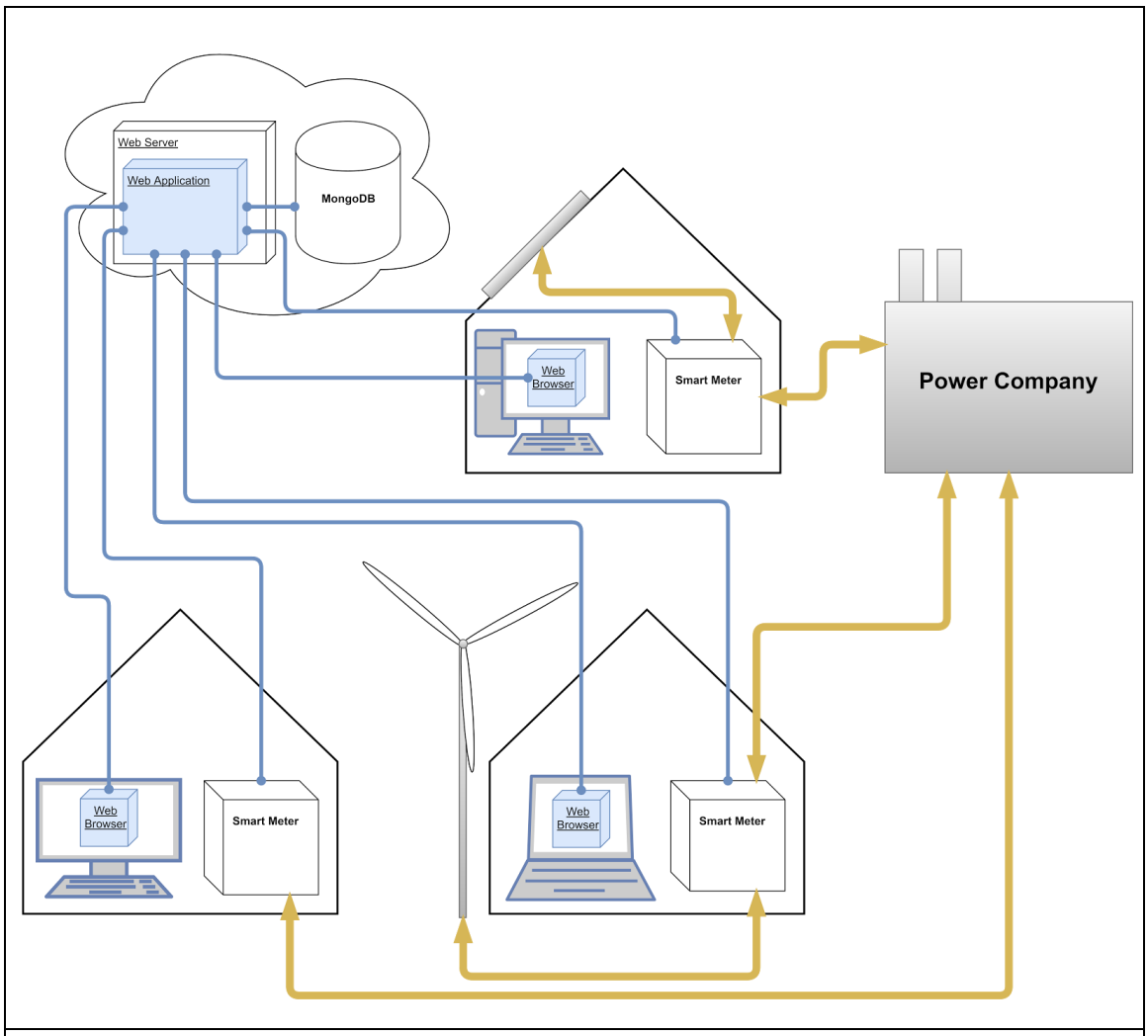

Figure 3: An overview of our proposed system

DevOps Deployment Diagram

Figure 4 is duplicated below. It shows the DevOps deployment for our web application. For more information, see Section 2.2: Proposed Design.

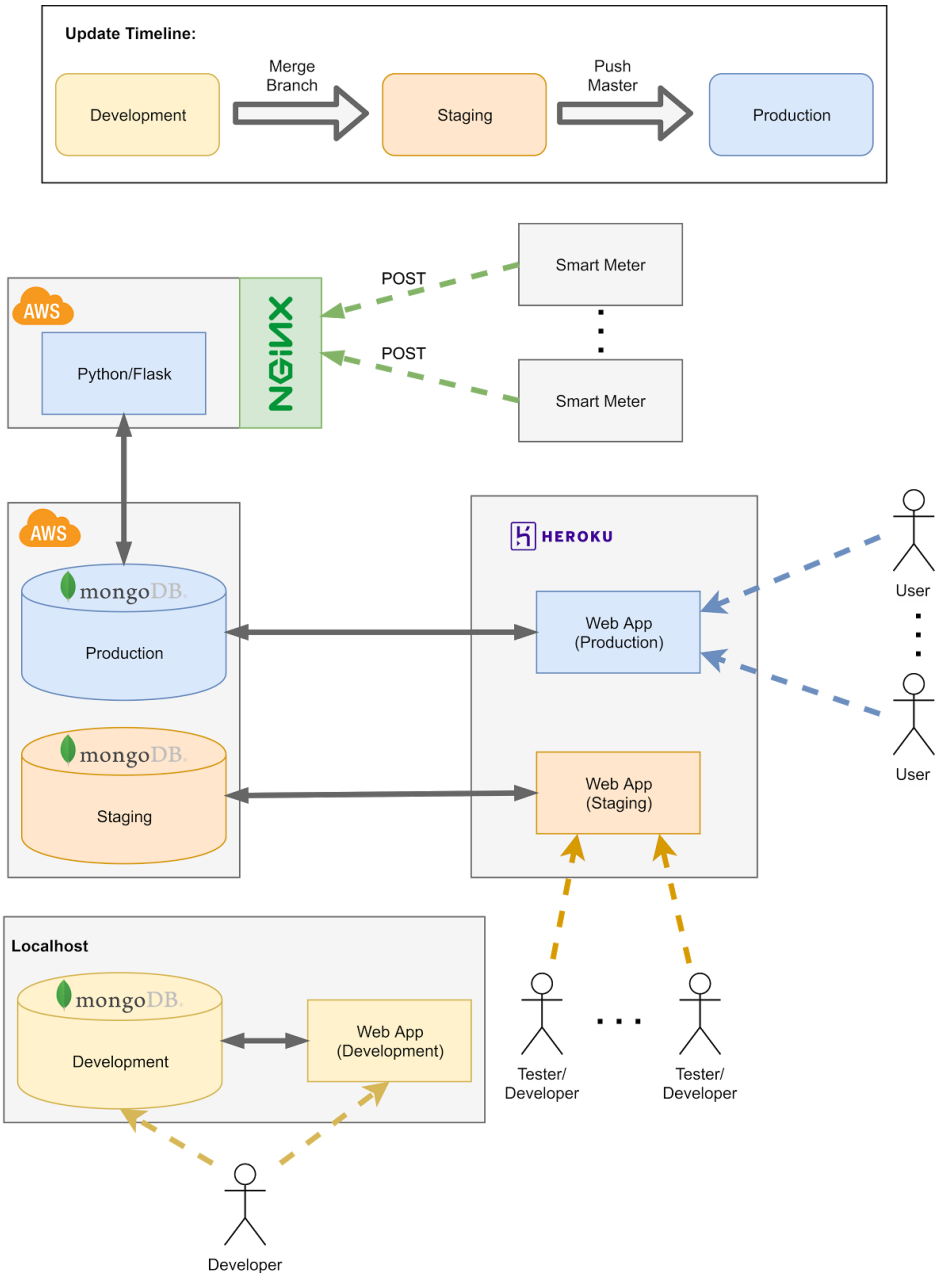

Figure 4: A DevOps diagram illustrating the design and development process of the project

5.2 Previous Design Diagrams

Original System Overview Diagram

Figure 5 is our original system design, and includes implementation of blockchain through an Ethereum deployment. For an updated system diagram, see Figure 3 in Section 5.1: Current Design Diagrams.

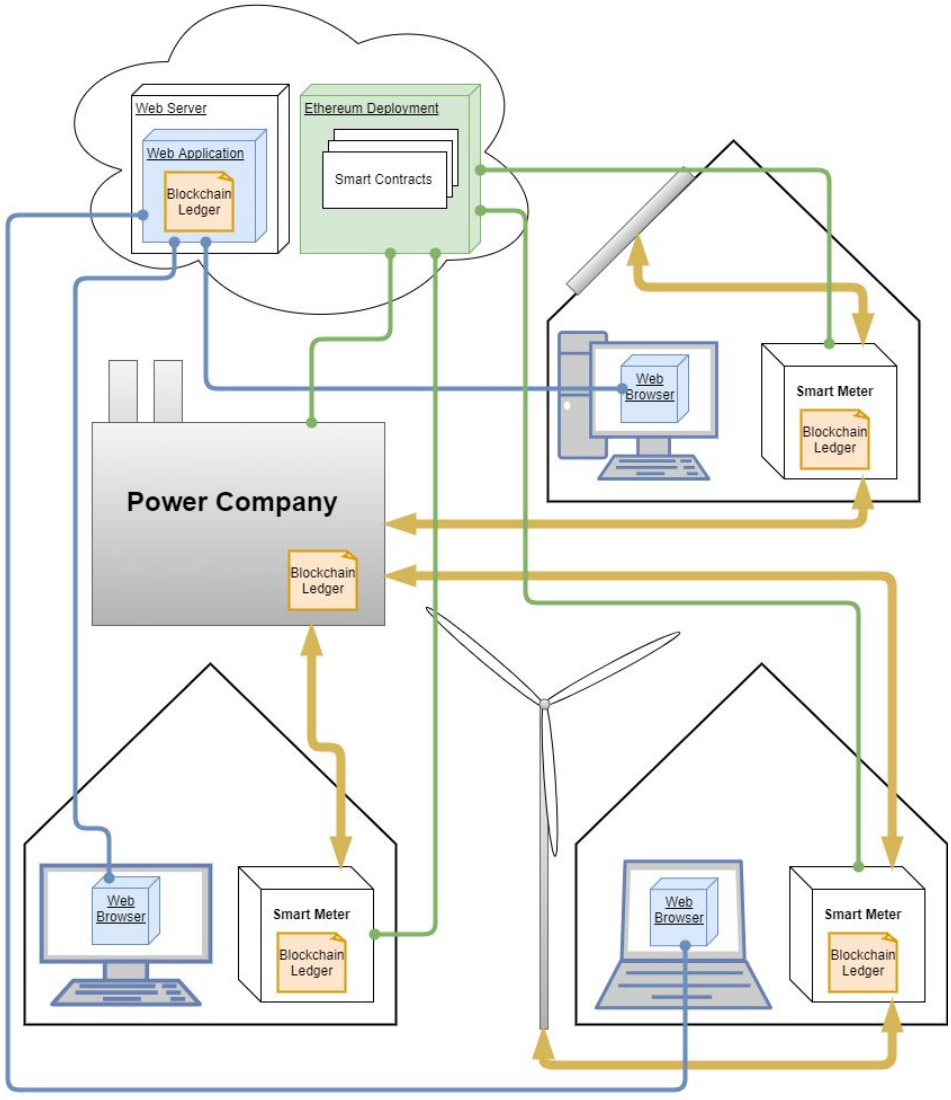

Figure 5: Original system overview diagram

Original Web App Macro-Component Diagram

Figure 6 is the original macro-component diagram for our web application, as it would have worked with an Ethereum blockchain. For an updated macro-component diagram, see Figure 2 in Section 5.1: Current Design Diagrams.

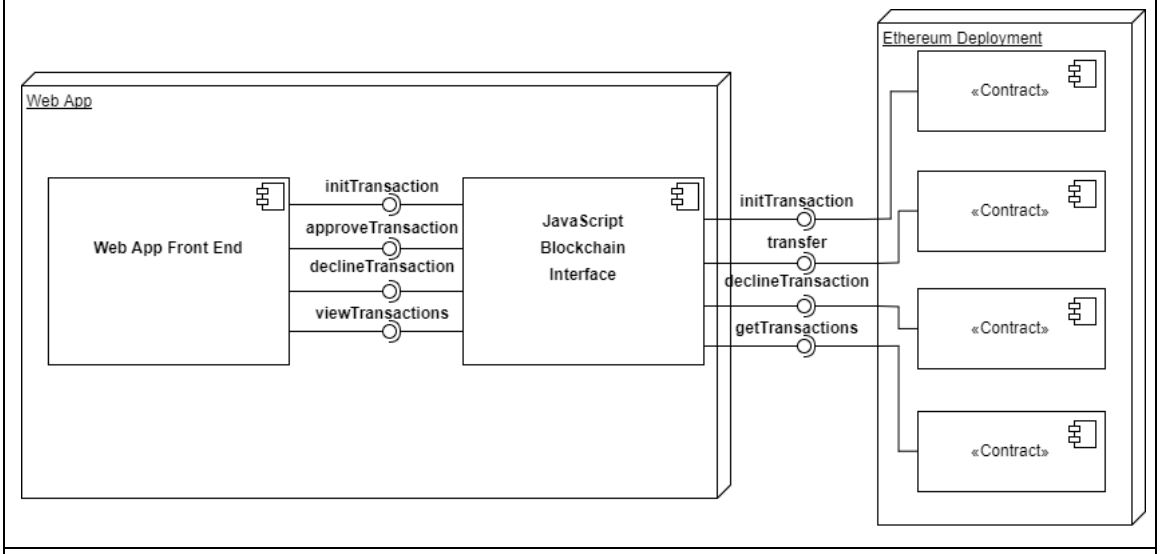

Figure 6: Original macro-component diagram for the web application and how it interacted with the Ethereum deployment

Original Web App Functionality Map

Figure 7 is a map/mockup of our web app's functionality, as it was originally planned with the blockchain implementation.

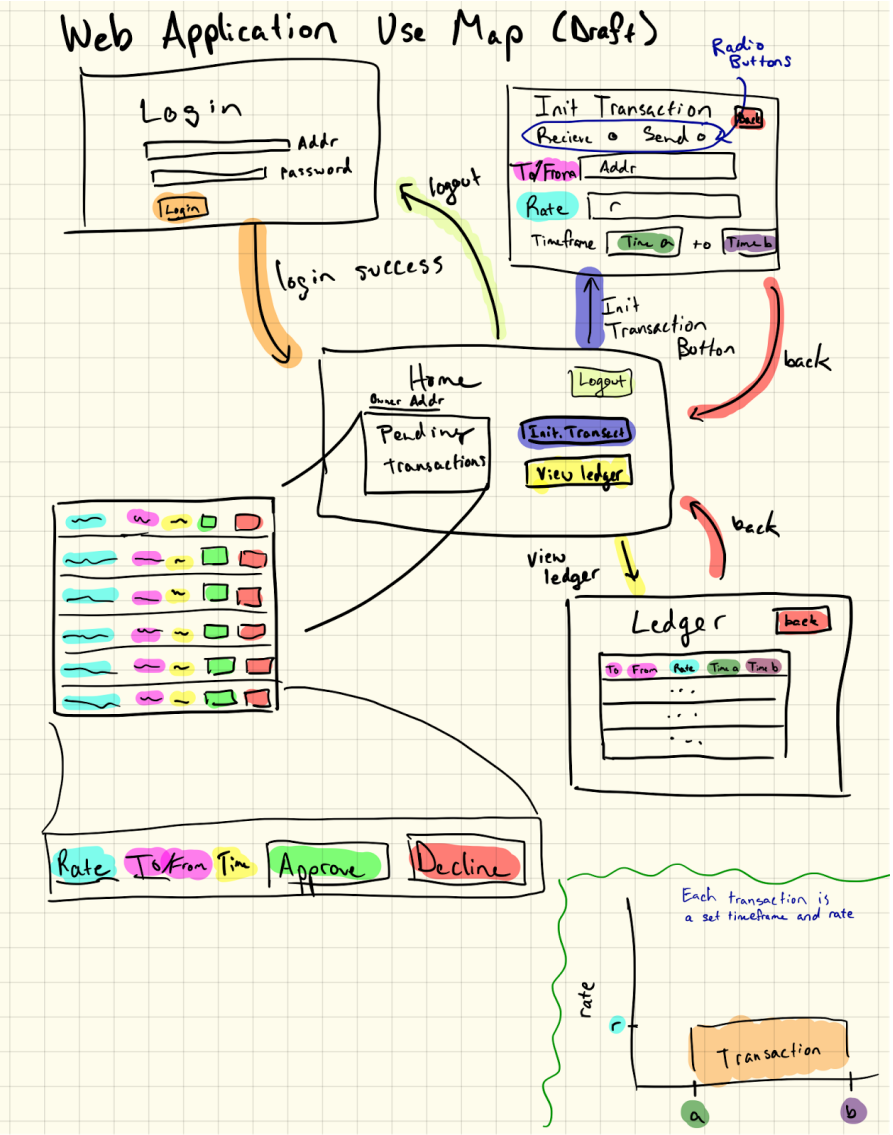

Figure 7: Original map of the functionality of the web application, as it would have worked with a blockchain design