Team No: 5
Members:

- Joshua Oertel j583o901@ku.edu
- Thomas Brooks t090b057@ku.edu
- Chong Tan chong.tan@ku.edu
- Lu Yin lu.yin@ku.edu

Project Name: Seathereum

Synopsis:

Use the Ethereum blockchain to build a decentralized app (DApp) that behaves like a Tamagotchi collectible game with various features designed to promote interactivity.

Description:

With the popularity of DApps such as Cryptokitties, there has been an increase in curiosity for the capabilities and implications of DApps. We have decided to undertake the development of an application that incorporates this innovative technology. The potential for DApp products is not fully recognized by the general public and the market is minimal, yet full decentralization can provide many benefits over alternatives that operate within a centralized network. These benefits will be explored throughout the development of our DApp.

Our DApp can be categorized as entertainment. It will essentially be a virtual collectible game that implements the ERC-721 standard which details how to create non-fungible, unique tokens on the Ethereum blockchain. We have chosen to use the Ethereum blockchain over alternatives because its Virtual Machine is Turing-complete granting full control over how the application behaves when it is deployed.

We're developing a DApp which aims to demonstrate the potential of blockchain technology and that offers a simpler introduction for individuals who are less familiar with the concepts surrounding blockchain. The DApp will explore how, unlike similar applications, focusing on interactivity can leverage a more enjoyable experience for the consumer. We hope to gain insight into the process and concern for the overall effectiveness of utilizing blockchain technology in a public, consumer application.

Milestones:

Initial Project Deployment - October 26th, 2018

Initial Interaction Between Frontend and Network Completed - November 9th, 2018

Smart Contract Token Attributes Initialized - November 30th, 2018.

Smart Contracts Completed- March 1st, 2019

Finish 1st Game for our Website - March 8th, 2019

Website Completed - April 12th, 2019

Budget:

Part of the advantage of utilizing the blockchain for much of the storage is that costs will be minimized substantially. The blockchain will host most of the data towards our tokenable Tamagotchi-esque creatures, The web application will be responsible for serving the web assets such as images to the consumer. The blockchain will only record the necessary information about transactions that has occured between users of a DApp. We have favored using technologies and frameworks that advocate open-source and decentralization because of the minimal cost. Examples include:

- The Truffle Suite and the many technologies it borrows from https://truffleframework.com/
- Github: https://github.com/joshwashywash/Seathereum
- Node.js and various web development packages
- React https://reactjs.org/
- Bootstrap https://getbootstrap.com/

Other resources may be incorporated into the project to overcome any unforeseen obstacles.

One cost that we may encounter is a VPS or proper server (Prices may vary between plans) to host our application specific environment. Some services like Heroku may be able to sufficiently serve our application, however, at this time, we are unsure of the specifics. This is because we may choose to go with a private blockchain just to get the idea established for our project, rather than make it public.

Work Plan:

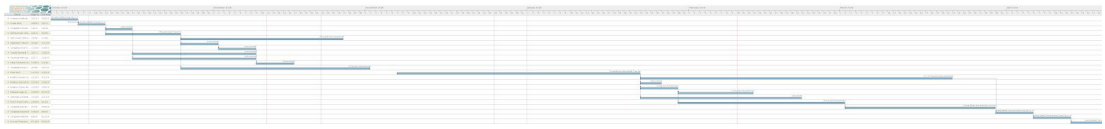
We have planned to divide the work according to team members' interest. The chart below indicates major components of the project and which teammate has expressed interest in that task. Each aspect of the project will be open to everyone,

however, teammates that have an affinity for a specific task will naturally focus more of their attention towards the design, development, and implementation of this particular task.

Task	Joshua Oertel	Thomas Brooks	Chong Tan	Lu Yin
Solidity and Ethereum Development	x	x	x	
Backend Work	x		x	
Frontend Site Work	x	x		x
Documentation and Testing	x	x		

Gantt Chart:
A link to a higher resolution image of the gantt chart is provided below:

https://drive.google.com/file/d/1vJwz-D2fBpPda5nKUL4PoEwlj2MM2B6U/view?usp=sharing



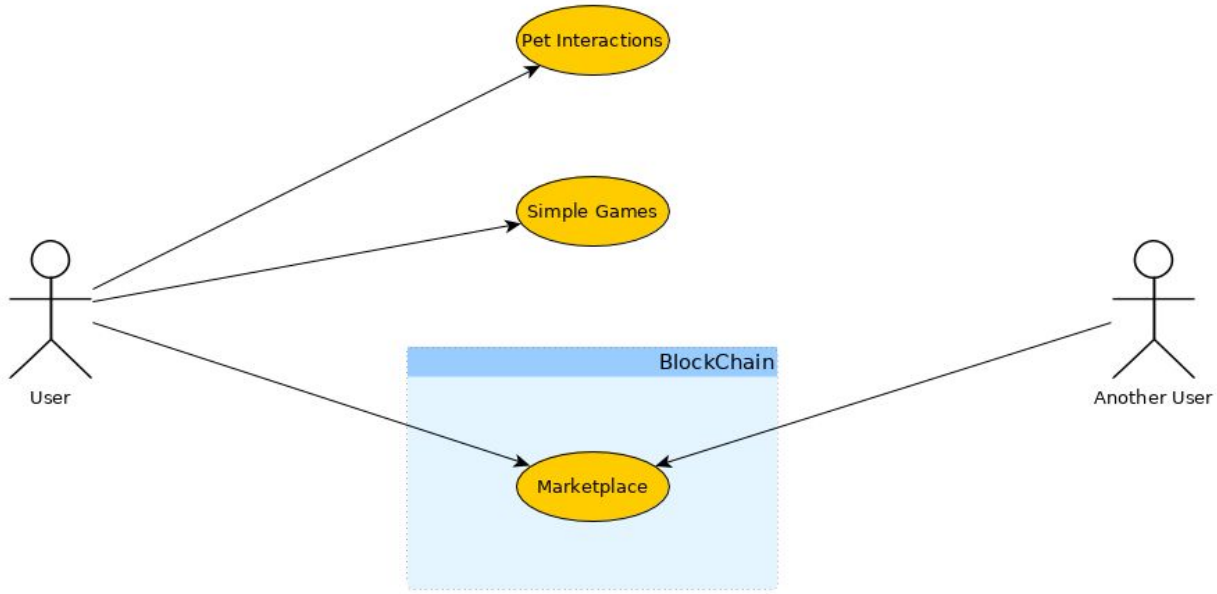
582-Project Feb 10, 2019

Tasks 2

Name	Begin date	End date
Choose a Software License	10/1/18	10/5/18
Divide Work	10/6/18	10/11/18
Complete Frontend Template	10/12/18	10/16/18
Define Smart Contract Attributes	10/12/18	10/25/18
Test Smart Contract Connection to Backend	10/26/18	11/26/18
Implement Initial Production Environment	10/26/18	11/1/18
Complete Initial Interaction of Frontend and Network	11/2/18	11/8/18
Create Backend To Hold Initial Frontend	10/17/18	11/8/18
Continue Working on Frontend	10/17/18	11/8/18
Initial Frontend Code Completed	11/9/18	11/16/18
Complete Initial Smart Contract Token Attributes	10/26/18	12/1/18
Free Work	12/7/18	1/22/19
Buildout Games to be Put in Frontend	1/23/19	3/21/19
Buildout Format for Displaying Tokens	1/23/19	1/26/19
Buildout Token Attributes in Contract	1/23/19	1/29/19
Prepare Logic to Handle Contract in Backend	1/30/19	2/12/19
Arbitrate UI Elements for each Page in the Frontend	1/23/19	2/21/19
Finish Smart Contract	1/30/19	3/1/19
Complete Backend Interaction Between Contract and Website	3/2/19	3/29/19
Complete Backend	3/30/19	4/5/19
Complete Website	4/6/19	4/12/19
Ensure Production Matches Development Environment	4/13/19	4/19/19

Final Project Design:

From the client perspective, the DApp will work in the following manner: users will purchase a creature either from another player or by a direct offer from the developers. After acquiring a creature, the user has a variety of different actions that can be performed. The actions can be split into two categories. Some actions involve basic interaction with their creature in the style of a virtual pet game. These activities do not adjust or manipulate the creature nor require any blockchain interaction. These actions offer a more casual and relaxing experience to the user during the downtime of trying to make a profit or engaging in the second category. The second category involves actions which require blockchain communication. As mentioned, players will have the option to sell and trade the creatures they acquire, and will be able to influence the prices and creature market. Certain creatures will be more favorable than others due to certain traits and features not found on other creatures. These rarities and attributes will drive the market and facilitate user to user interaction through the blockchain contracts that will be developed in tandem with the application.



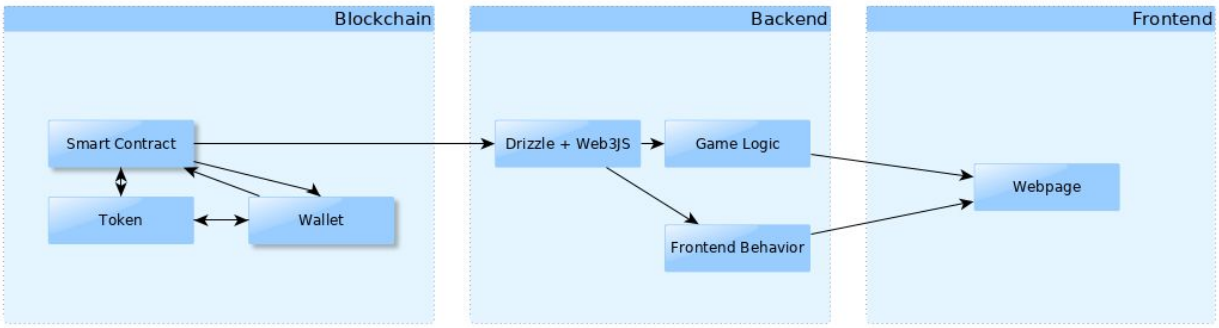
Use Case Diagram

Our application is going to be reliant on the blockchain for delivering the appropriate content to each user. As a consequence, certain prerequisites need to be fulfilled in order to properly interact with our app. The most immediate requirement for new user would be the use of a DApp viewer. Certain applications such as Mist or MetaMask provide this functionality. Just like how a browser is used to view and interpret web pages, DApp viewers function in a similar way. While there are not that many well-maintained or documented DApp viewers, those that exist come in multiple

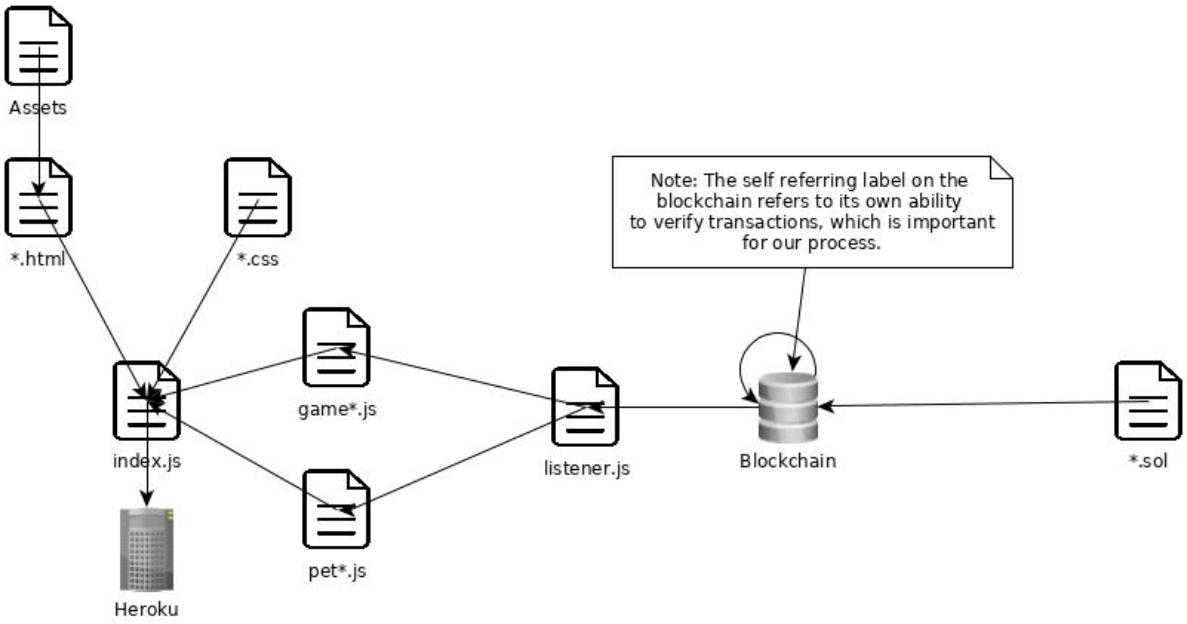
different fashions. The popular DApp viewer, MetaMask, operates as an extension to currently existing web browsers, thus providing an easy installation and integration. Certain parts of the application would be open for all users, while other parts would be only accessible to users employing a DApp viewer.

By design, the DApp can be divided into three different portions. There is a the smart contract, which interacts with the Ethereum blockchain to handle transactions and dictates the components of each token. Unlike traditional cryptocurrencies, even Ethereum itself, Seathereum will feature tokens based on the ERC-721 standard. Each token will be indivisible, unique, and unstackable in the way other coins are. The second component is the back-end of the application, which will feature mostly JavaScript utilities. This will be responsible for intercepting the transactions made by the smart contract, and provide the logic to the application. This could include things such as interpreting the components of each token, and the game logic for the interactions hosted on the website. Finally, there is the front-end of the application, which will feature the standard HTML, CSS, and JavaScript found on most web applications. This is for displaying the contents in the typical web page format. The JavaScript implementation for HTML requests to blockchain nodes has gained the most support out of any of the available APIs offered by the Ethereum Foundation for interacting with smart contracts

Package Diagram



Component Diagram



The smart contract will be developed primarily with the help of the Truffle framework. This framework offers multiple components that are useful for the development of the contract. The main component, Truffle, is specifically designed to aid in the development and deployment of a contract onto a blockchain. One nice feature of Truffle is that the component also does not add any additional syntax from the contract in order to use it, as this would not be possible on the blockchain without importing its libraries, and in effect, adding more gas fees to the transactions of the users. Truffle is specifically tied to experiences around development, such as testing, migrations, and deployment. Some workflows have to be established in order to properly work with Truffle, but none are related to the actual writing of the contract. Another component of the framework is Ganache. Ganache is used for spinning up a temporary blockchain with ether on several accounts to test the transactions on this blockchain without having to pay real monetary fees. One advantage of using Ganache is that it is the only component of the framework that is reliant upon any other component. Drizzle is the final component and is for monitoring the transactions on the JavaScript portion of the program so that the DApp can properly handle changes made on the blockchain. Drizzle utilizes a state management system similar to a redux store.

Deployment will be achieved through the Heroku cloud platform. Heroku offers easy deployment and configuration from a git repo, and can follow a branch to commit changes as soon as the branch is updated and pushed. The simple configuration makes Heroku ideal for a team that has little to no experience. Should we also need additional

requirements for our application, we also have the ability to scale into paid solutions, without changing the deployment model.

The front-end will be designed using React components, which ties well into Drizzle from the Truffle framework mentioned previously. Drizzle will monitor changes on the blockchain which affect our application, and React provides a way to propagate these changes to the rest of the UI components. These components are only updated when necessary. Routing the different web-pages will be handled through the react router. The react router can render a variety of use cases, from react components, to simple html pages. This structure handles more than enough use cases for our needs, doing what a normal backend like Express or Koa would do, but for a smaller scale. Styling will be supported by the popular Bootstrap framework. Bootstrap has been chosen for its ease of use and integration allowing us to focus on more pertinent aspects of the application rather than the visuals. Bootstrap can natively handle adapting to different screen sizes to display web pages to fit more fluidly on a range of different devices. This ability adds a level of responsiveness to the UI.

Ethical Concerns:

In effect, our app will be creating a virtual market where digital goods with real monetary value will be bought, sold, and traded. With this consideration in mind, there is a strong necessity for the application to adhere to ethical norms. Fortunately, user account security is not a major concern due to the design of blockchain. Tokens are connected to each user's Ethereum wallet instead of a database. This means that the application is not responsible for storing any user information. This is the same method that other DApps have inherited. The balance of each user's wallet is also stored on the blockchain. Our app will only be responsible for matching the proper tokens to their proper owners. All transactions will be posted on the blockchain providing a self-enforcing sense of security for users' tokens.

There are, however, other concerns that blockchain apps must face that other apps do not have to deal with. In order to write to the blockchain, a monetary fee is used to compensate the nodes on the blockchain that perform the computations, and hold the data. These are passed to the wallets making the transactions, meaning our users will incur some consistent fees just for interacting with our app. These fees depend on the amount of computations and data being written to the blockchain, thus we will have to be extremely careful in ensuring we only write to the blockchain when it is absolutely necessary in order to reduce costs.

With the unique value generation of Ethereum and other cryptocurrencies, our users will determine the value of each unique token. However, we are in charge of the actual token attributes. This means that any changes that we could make to the

software when we have a user base could affect not only the value of future tokens, but existing tokens that our users might already have. This could be both in their favor, and against it. This needs to be either addressed to the user base as a possibility, or the software can not be updated outside of security issues.

Intellectual Property Issues:

With the chosen license of the BSD-3 Clause License, we will need to keep in mind that not all libraries or solutions that we could use will be accommodate for this license. For example, the GPL v3 does not allow its software to be sublicensed in any form, thus we will have to be careful what libraries we use. The LGPL v3 and 2.1 allow for some sublicensing within proprietary programs, but because our application is going to be open source, it most likely will not be a problem. Our current selection of software does allow for sub-licensing.

Change Log:

- The Work Plan was changed to reflect only having four members now.
- The Gantt chart was changed to reflect what actually occurred last semester, and to what is intended for this semester. It also reflects only having four members now.
- The budget had some technologies removed that no longer apply, as well as one potential cost removed as well.
- The Project Design removed some features that are no longer possible due to time constraints, and to reemphasize the amount of Koa we use as well, by not mentioning it. This is because it is only being used in one file, which is not as much as we anticipated, yet it is still covered by the budget in broad terms. The diagrams were also changed to reflect this as well.
- More milestones were included. Also, in order to meet the requirement of three each semester, and one was forgotten from last semester, one goal had to be expressed that was achieved unintentionally.