Proposal for Larry Shiller

From Zach Burks
661-302-8985
spader89d@yahoo.com

Time for proposal - 2.5 hr

GammonCoin

http://gammoncoin.com/

Overview

GammonCoin is a cryptocurrency startup that is connecting real world gamers/activities to the Ethereum blockchain with an ERC20 token and web interface. The point of GammonCoin and the project, is to allow for users who have attended real world events, to be able to track that attendance and their scores during competition, in an online database hosted on GammonCoin.com. This database will store the total number of points each user has earned over multiple events and allow a user to claim these point as ERC20 tokens. The user will sign up and provide an Ethereum Address that will then be sent GammonCoin from the ERC20 smart contract. Users will also be able to pay for items in their store with this coin. During payment, coins are automatically forwarded from the contract into a corresponding exchange address for conversion into fiat

1. **ERC20 Token + Managing Smart contracts:** Standard ERC20 smart contract implementation. Then Managing Smart contract with functionality to move the funds given to it and other standard admin functions.
2. **Payment Functionality within Smart Contracts:** Transfer of saved/earned rewards to the user. Security testing to ensure 100% safety.

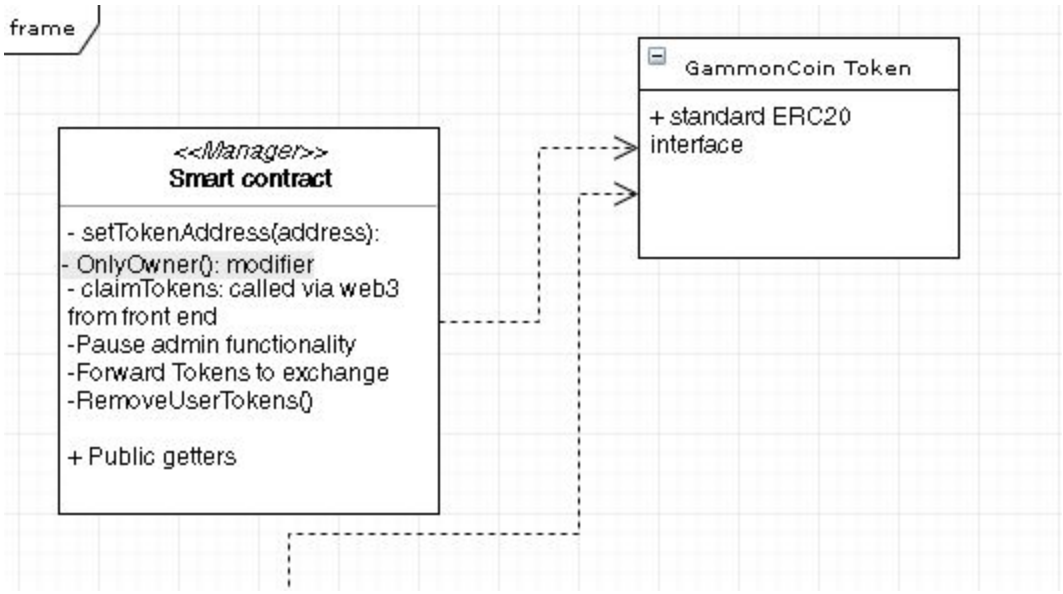
3. **Web3.js/Front end interactions for listening to, sending, and receiving the ERC20:** Front end listening to the contract for events, to provide details about payment from user via the shop to the contract with tokens as payment. Initiating transactions for the payment of goods on shop, at checkout. Confirming that the ERC20 has been received by the contract and the user can get receipt

Specifications

Below is the specific use cases of each contract, as well as the important functionality that needs to be made, and then tested.

Milestones

1. ERC20 + Managing Smart contract



The ERC20 contract needs to be able to be called from the Managing contract. Therefore managing contract needs the following functionality:

setTokenAddress() - setter function to allow only owner to set the cotract address for the erc20

setExchangeAddress() - set the withdrawal address for tokens to be sent for liquidation

OnlyOwner() - modifer

Pauseable - normal admin functionality

claimTokens() - this is called via the web3 front end. User will submit his ethereum address to front end, front end will check the total number of tokens owed to that user, then pass in the address provided from the user as a check into the smart contract (ie: require(msg.sender == correctAddress)). And also confirm the amount of tokens allowed to be sent. Then transfer ownership of tokens to the user's address

forwardTokens() - Take deposited tokens from user as payment for shop items, and forward the same amount over to an exchange wallet.

RemoveTokens() - ability for manager to remove tokens

ERC20 contract -

Name = GammonCoin

Symbol = ?

Amount = ? (total amount of Gammoncoins to be made)

mintable() = true/false (can you print more gammonCoins or is it one time? Ie: make 100mil and thats it, or make as many as you want, on command)

Estimated Time for production: 50 hours

ERC20 - 3 hours - 10 hours

Manager Contract - 20 - 40 hours

Calendar Date for Completion: August 12th

2. Payment functionality within Manager Smart Contract

Majority of this phase is developing tests to ensure that the smart contract has zero bugs.

Transfer needs to be able to receive an address from web server, amount to be sent, and msg.sender to check that the user requesting the tokens is entitled to those tokens.

PayandForward() - needs to be able to receive an amount from the web server, a time requirement, and then check that the tokens received are the correct amount and within the right time frame, then execute forward() so that the tokens are sent to an exchange immediately.

Make test and then run test to ensure functions are safe.

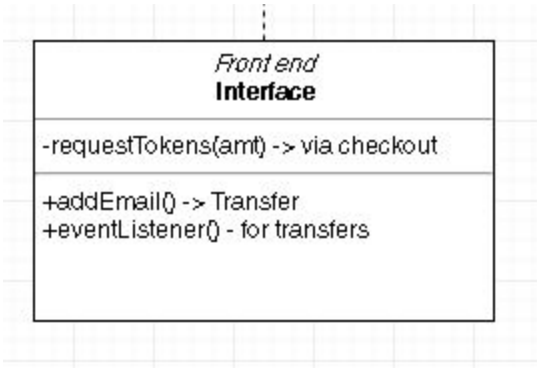
Estimated Time for production: 25 hours

Transfer functions and forwarding- 3 hours-10 hours

Testing functionality - 5-15 hours

Calendar Date for Completion: August 18th

3. Web3.js Frontend (note: I will provide exportable web3.js functions that you can then call from whatever JS you use for your frontend. That way it is modular and you can do whatever you need to do on your front end and just be able to fetch



the data)

AddUser'sAddress() - User will supply the web server with their personal Ethereum address. Then once the address is known, web3 will initiate a transaction for the user to claim their tokens from the managing smart contract. Issue: The user will be allotted a special amount of tokens that they can claim, you don't want them to claim more than they need, or less either. So in order for the smart contract to check that the tokens are supposed to be claimed, there needs to be an internal balance tracked. Meaning

someone has to manually add the user's information to the smart contract Note: adding data **cannot be automated**

Solution: Manually add a users data when they send you their ethereum address. Meaning, bob says "my address is XYZ" and then someone will go into the admin panel, and create an ethereum transaction that sends the data "XYZ, and the amount of tokens for bob" to the blockchain. Then bob can call the blockchain and get his coins.

Admin addUserData() - Admin panel function, takes users address and amount of tokens to be given to the user. Then initiates a web3 transaction to send the data to the managing contract

PayforItems() - User will send funds to the contract, from an amount provided to them via the front end. The front end needs to listen for a transaction on the contract, then check and return that the tokens sent was the correct amount. Also, all of this needs to happen within the span of a timer that is running to prevent the price from changing too dramatically. Needs to have checks for incomplete orders, for example, user needs to pay 100 tokens, sends 95. Needs to check and deal with those scenarios.

Estimated Time for production: 20 hours

AddUser'sAddress() - 1 - 5 hours

addUserData() - 1 - 5 hours

listenForPayment() - 5 - 10 hours

Calendar Date for Completion: August 25th

Deliverables

- Smart contracts
    - ERC20
    - Managing Contract
- Tests
    - Truffle tests for smart contract implementations
- Web3.js frontend files (these will be .js files that export the data required and take the data needed as arguments)
    - web3.js
    - addUsersData.js
    - ListenForPayment.js

Breakdown

Max hours worked for completion:	100 hours
Minimum hours worked for completion:	38 hours
Estimated Costs:	$2600 - $7000
Delivery Date:	August 21st 2018