Blockchain for Sustainable Agriculture: Design Document

CS2: AgCheck: Date: 12/12/2019

Issuing organization: AgCheck

Authors: Joshua Fisher and Aaron Galati

Change history: 11/23/2019 - Created, 12/12/2019 - Revised, 4/20/2020 - Revised

Contents

I Introduction 2

I-A Purpose 2

I-B Scope 2

I-C Context 2

I-D Summary 2

II Glossary 3

III Body 3

III-A Identified stakeholders and design concerns 3

III-B Blockchain 3

III-C Database 5

III-D Server 5

III-E Front End 6

III-F Drone 6

IV References 7

Revisions: Revised May 2020

Section	Original	New
Drone	no drone security	drones send data to raspberry pi which is hashed
Front End	no searching	searching is a feature
Blockchain	VIDs are unsecure	secure generation of VIDs

I. Introduction

A. Purpose

The purpose of this document is to define each element that will make up the AgCheck system. It will also set the teams scope during the project's duration to communicate what the team is agreeing to complete during development.

B. Scope

AgCheck will be a simple way for anyone to see trusted and verified sustainable agriculture methods from various farms using drone images. This will be accomplished through five different, major components. These are a blockchain, a server, two databases, a web frontend, and drones to take aerial imagery. Various information for the project will be stored on a blockchain platform called VeChain including farm data and picture hashes. One of the databases, created in MySQL, will be used to also hold picture information as well as website login information in two separate tables. The other database will be through Amazon Web Services which will host the server and drone images. The web frontend will be where all important information is displayed from various categories of sustainability such as no-till, crop rotation, and cover crop. Finally, at the center of it all will be the server to handle all requests to both databases, the blockchain, drones and the website.

C. Context

AgCheck stems from the notion that people want to know exactly where their food comes from. When deciding on a blockchain platform VeChain certainly helps in that regard. VeChain handles the tracking of all of Walmart's supply chain in China. AgCheck is to turn it from tracking products to tracking sustainable agriculture methods that certain farms are using. Sustainable agriculture methods are becoming important now more than ever. As stated in a Politico article, farmers are starting to realize that sustainable agriculture methods are an important part of sequestering carbon to fight climate change [1]. AgCheck will, instead of helping people learn where their products are coming from, teach them how those products are grown sustainably.

D. Summary

AgCheck is a project that will include a blockchain platform, a server, a web frontend, databases and drones to help show verified sustainable agriculture methods. Drones will be used to take images on various farms to showcase these methods. The blockchain platform will hold hashes of the drone images and the location information. The website will showcase the images and methods. Two different databases will hold the information needed for the project that can't be put on the blockchain. Finally, the server will tie together all of the other parts of the project to make for a seamless integration.

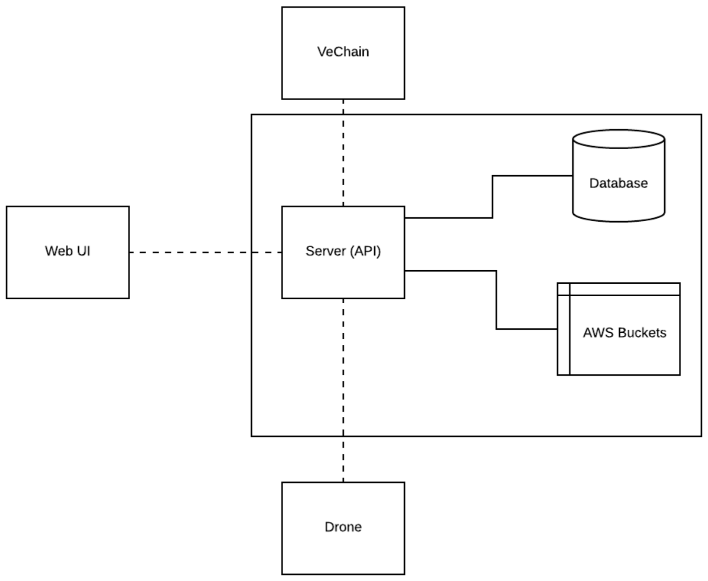

Fig. 1. Design of AgCheck

II. Glossary

AWS – Amazon Web Services, a platform for on-demand cloud computing

Buckets – A service offered through AWS, holds an object and for this project, those objects are pictures

Cover Crop - An off season crop that is used to promote fertility and biodiversity in the soil.

Crop Rotation - Varying successive crops on the same ground to avoid depleting the soil and to control weeds, pests, and diseases

Drone – A quadcopter that can be remotely controlled to take photographs

Hash - End result of a hash function that will convert a value to something completely different

MySQL – A database management system

NodeJs – A Javascript runtime that runs code outside of the browser

No Till/Reduce Till - A farming practice that minimizes tilling to reduce erosion and minimize the disturbance of the soil.

Smart contract – A blockchain feature that facilitates and governs a transaction between two parties through contracts. These don't require actions from third parties, and are trackable and irreversible.

III. Body

A. *Identified stakeholders and design concerns*

The stakeholders for this project are the students working on it, the client AllTheFarms, VeChain, the Lane Community College Drone program (LCC), and the various farms that the team is partnering with for testing. The students are responsible for developing the middleware system and server. LCC is responsible for designing the drone to meet the requirements to integrate with the server. AllTheFarms is responsible for overseeing development, contacting farms for testing, and setting requirements for the system so that it accomplishes what farms and certifiers need. The two design concerns are that have been considered during the design process are security and usability because these are the two problems that the project is aiming to solve. The system needs to be secure from drone to blockchain, so there should be no human interaction with the data along the way. Data being displayed on the website needs to be verified for authenticity. The user interface design must also be user friendly so that non-technical users can find and input the information that they need to.

B. *Blockchain*

To ensure that all data is immutable from drone to the web frontend, various data including farm information, picture hashes, date, and location data will be sent to a blockchain platform VeChain. Special IDs will be generated using unique image data to go along with each image to ensure that there is no tampering of images on our end. To begin to interact with VeChain, a time-limited token must be created using pre-given app ids and app keys. The following request is one such way to generate a token:

```
POST /v1/tokens HTTP/1.1
-H content-type: application/json;charset=utf-8
-D
{
    "appid": "15cf8af3ebeca443d377bd5c2658cae9",
    "nonce": "3806143519628774",
    "signature":
    "1c442f56664317d81c32e41010660f54ba1e71223fa6d766ebbe45de3906998b",
    "timestamp": "1552185608"
}
```

Fig. 2. Example request to receive a token from the blockchain

The response will include a token and expiration time. That token can then be used to create and make space for our VIDs. The VIDs will be what holds our hashed data. The following is a way to generate those VIDs and how many we want:

```
POST /v1/vid/generate HTTP/1.1
-H language: en_us
-H content-type: application/json;charset=utf-8
-H x-api-token:
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9
.eyJhcHBJZCI6IjE1Y2Y4YWYzZWJlY2E0NDNkMzc3YmQ1YzI2NThjYWU5IiwiZXhwIjox
NTUxNDk0NDQyfQ.VEEzUW9pjO0KznzyrJ2B6G0lfF0NSR0RMqkN45Im1s
-D
{
    "requestNo":"1554093799",
    "quantity":5
}
```

Fig. 3. Example request to receive 5 VIDs from the blockchain

The response will include a list of 5 VIDs for the use of our information. We must then occupy those VIDs for putting our data there later. The following is one such way to occupy the VIDs:

```
POST /v1/vid/occupy HTTP/1.1
-H language: en_us
-H content-type: application/json;charset=utf-8
-H x-api-token:
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJhcHBJZCI6IjE1Y2Y4YWYzZWJlY2E0NDNkMzc3Y
mQ1YzI2NThjYWU5IiwiZXhwIjoxNTUxNDk0NDQyfQ.VEEzUW9pjO0KznzyrJ2B6
G0lfF0NSR0R-MqkN45Im1s
-D
{
    "requestNo":"1554094268",
    "vidList":[
        "0xe51e3d7d9d49005d2135caa60eac1daedd7a69c62b23893b5fe57b71a0dce1f8f1",
        "0xe51e3d7d9d49005d2135caa60eac1daedd7a69c62b23893b5fe57b71a0dce1f8f2"
    ]
}
```

Fig. 4. Example request to occupy two VIDs

The response should include a success message should all go well. Once data is hashed and ready to be uploaded the following request can be sent to the blockchain to upload that data given:

```
POST /v1/artifacts/hashinfo/create HTTP/1.1
-H language: en_us
-H content-type: application/json;charset=utf-8
-H x-api-token:
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9
.eyJhcHBJZCI6IjE1Y2Y4YWYzZWJlY2E0NDNkMzc3YmQ1YzI2NThjYWU5IiwiZXhwIjoxNTUxNDk0NDQyfQ
.VEEzUW9pjO0KznzyrJ2B6G0lfF0NSR0R-MqkN45Im1s
-D
{
    "data": [
        {
        "dataHash":
        "0x97b928405a07861f94fe272ad21f223197d94aea08eb65cf34c2c886006e4613",
        "vid":
        "0x0076c12826e26375b08450867fb1cb9e2ed3c4b33b33f35f87c29985149bfbc6"
        }
    ],
    "requestNo": "440581bc1e",
    "uid":
    "0xb06b816a0b55eef70995b1d4f86d648bc287fca6ced0b50efa0ac936cebce16c"
}
```

Fig. 5. Example request to store hashed data.

Then if there are no errors, the response should show a success message with a text receipt of what data was entered onto the blockchain. Our design will have different hashes of information being sent to separate VIDs to ensure that the data of the images is secure going forward.

C. Database

The database will be split between two separate methods to store all of the data for the entire project. First, there will be a MySQL server that will handle all login information hashes from the user through the server. There will be a table for just login information hashes tied to usernames, but also another table that will only store hashes of the image data from the drone pictures. This table that stores image hashes will also have the bucket URLs in a separate column. Both the login information hash and the image hashing will be done through the server. Examples of the SQL tables below:

ID	username	hash
1	galatia	ae80976b013f1a2a241b8d68e0b32e9475233d6f830940e43a8980975edd097e
2	fisherj	d004a78f868009fe0920265c18b89d6b3ebfb938f9a2fdd2a7472936b118826a
3	cupplesj	9a88e236f13cc8719993a009f622d39d822d13c61c8b45e9893a81e976d81c16

Fig. 6. Example data for the login table on MySQL

ID	hash	url
1	14458962759852348248951585l349	http://droneImage1.s3-us-west-2.amazonaws.com
2	854328268435842697536876518963	http://droneImage2.s3-us-west-2.amazonaws.com
3	951356478523642698535422379145	http://droneImage3.s3-us-west-2.amazonaws.com

Fig. 7. Example data for the Image Table of MySQL

The images themselves will be stored in Amazon S3 buckets. Before doing so, the drone images will be verified with the hashes sent from the drone on the server. Should they match up, the image will be sent to a bucket. The team will be using free credits to have access to Amazon Web Services.

D. Server

The server will be written in NodeJs and it will be hosted on AWS. It will authenticate users who access it via the web front end. There will multiple groups of users in the application that the server will track, farms, operators, admins and certifiers. Farms will have the ability to view submissions that they have made as well as view statuses for their submissions. A farm can register for the service using a page on the web app. An account will be made for their farm on our server. The server will also provide a portal for authenticated certifiers to view farms that have submitted images and accept or deny the farm's sustainability efforts. Only specific, certified, users will have the authority to certify farms. The privilege to certify farms will be given out sparingly, and by the discretion of the AgCheck team. The web app will have a sign in page where users can input their usernames and passwords, these will be sent to the server which will hash the password and look up the password hash in the database based on the username. If the hashes match then the server will generate a temporary web token which will be sent back to the web app to use for future requests.

The server will also accept connections with drones, which will be sending image and location data to the server. For a drone to be registered with the server, it will generate a private and public key, as well as a password. The public key and password will be sent to the server when a new drone is added to the system, this submission is authenticated by an operator. The server will then create a record of the drone in the database with its public key and name. The server will receive uploads from the drone, which includes images and their signatures. The server will verify that the image matches the signature by hashing the image and comparing it with the decrypted signature. If the hashes match then the image will be saved in an s3 bucket and the location of the image will be stored in the database in relation to the hash so that they can be referenced in the future. A node in VeChain will then be created for the now successfully uploaded data, and the hashes and location data will be sent to the node. The node will be accessed exclusively with the metadata of the submission, including creation date and farm name.

E. Front End

The front end will provide a landing page where any user can view farms that have opted in as well as information about the service. There will also be a link for farms to sign in as well as a link for certifiers to sign in. Farms will see a page that gives an overview of the data associated with their farm, such as inspection dates and the results of those inspections. They will be able to click on a specific date of inspection to view the drone images that were taken. Certifiers will see their past certifications that they have granted as well as submissions that are available for certifications. If they click on an available certification then they will be shown the farm's general information, as well as the images that were taken, the location of each image, and the date and time of the submission. They will have the option of selecting no-till, crop rotation and cover crop for certification and the ability to submit that acceptance. The server will receive the request and verify that the web token is valid and that it matches the certifier session. If it does then the server will make a record of the submission in the database and create a smart contract that is linked to the farm. Anyone can also search through drone imagery submitted to the website.

F. Drone

The drone will use a companion computer that can store the captured images and create a signature. The signature allows the server to verify that the image hasn't been tampered with since it was originally taken. The companion computer will be a Raspberry pi, which is connected to the drone's flight controller through a protocol called MavLink. To connect the drone with AgCheck, the companion computer will be originally configured with an operator's user name and password. It then uses these credentials, and it's desired name, to register with the server. While doing this, it creates a private and public key pair, and a password that it will store and use for future interactions with the server. The drone can then start capturing using the companion computer, and store the data until a wifi connection is available. The operator can then start the upload process. The companion computer will post the information using it's name and password which was created during registration.

IV. References

[1] Evich, Helena Bottemiller. “How a Closed-Door Meeting Shows Farmers Are Waking up on Climate Change.” POLITICO, https://www.politico.com/news/2019/12/09/farmers-climate-change-074024.