Solien Gallery: Project Overview

Aaron, Gabe G., Gabe L., Nick

Identification

Project Name: Solien Gallery
Project Leader: Aaron Skepasts (ams10)
Team Members: Gabe Lebeau (glebeau), Gabe Grajeda (ggrajeda), Nick Hovsmith (nph2)
GitHub Source Code Repository: https://github.com/aaronskepasts/SolienGallery
Statement: The COS333 Instructors have access to the source code repository.

Elevator Speech

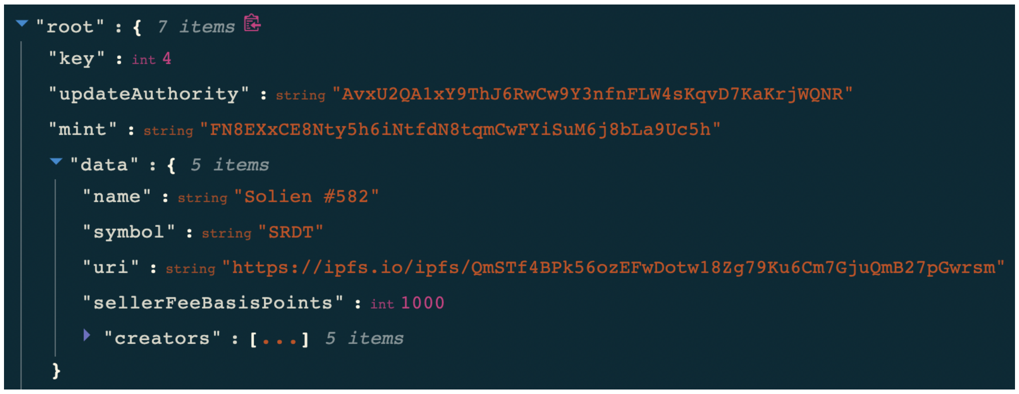
There is a lot of excitement around non-fungible tokens[1] (NFTs). People love to buy digital art and show them off to their friends, particularly on Twitter, but there is no standard way for people to showcase the NFTs that they own. Currently, most users display a screenshot of their favorite NFT on their Twitter banner (a large image on their profile). There are two issues with this. First, it is difficult to display multiple NFTs. People want to avoid the work of merging images together, and so many users choose just a single token to exhibit, even if they have several items that they want to show off. Second, it is not easy to verify ownership of the NFT; people can screenshot someone else's banner and claim the NFT as their own. We want to provide a service where users provide their wallet's public key (i.e., their account ID) and, after verification, create a gallery of the most prized NFTs in their wallet collection. The gallery of NFTs will be a token itself, and it will serve as a proof of ownership through the metadata stored in the token. Then, users will be able to upload images of their galleries to Twitter, solving the current issues with showcasing one's NFTs. For our project, we will restrict ourselves to Solien NFTs on the Solana blockchain.

Overview

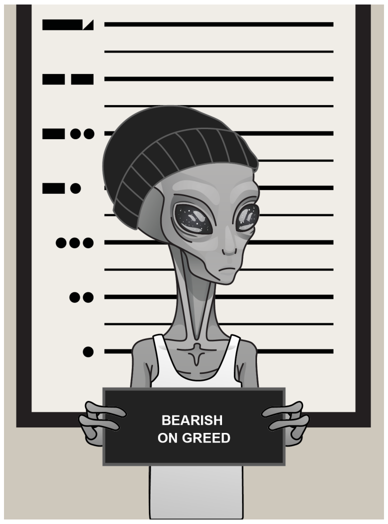
Our project is a web application for Solien NFT owners to create a gallery image of their favorite Solien NFTs. It is extremely important to understand what a Solien NFT is to understand the basis of our project. First, an NFT (non-fungible token) is a collection of data stored on a blockchain containing unique data related to a digital item, for example, an image. The data in an NFT proves that a digital item is unique and not interchangeable, and also provides the current owner and original creator of the NFT. NFTs are created and stored on digital blockchains like Ethereum or Solana, because these digital blockchains provide a secure datastore where no one except for the creator can modify the data of the NFT, including the data stating its creator, the record of ownership, and the digital item itself. Here is a little more information on NFTs provided by

[1] **From Wikipedia**: "A non-fungible token is a unit of data stored on a digital ledger, called a blockchain, that certifies a digital asset to be unique and therefore not interchangeable. NFTs can be used to represent items such as photos, videos, audio, and other types of digital files."

Ethereum: https://ethereum.org/en/nft/. Solien is an organization dedicated to the "Solien Project", a project that has minted (created an NFT with a unique ID) ~5000 NFTs on the Solana blockchain. The digital item unique to each of these Solien NFTs is an image of a unique Alien (Sol-ien). This is an example of a Solien NFT stored on the Solana blockchain that our project leader, Aaron Skepasts, currently owns:



```
▼ "root" : { 7 items
    "key" : int 4
    "updateAuthority" : string "AvxU2QA1xY9ThJ6RwCw9Y3nfnFLW4sKqvD7KaKrjWQNR"
    "mint" : string "FN8EXxCE8Nty5h6iNtfdN8tqmCwFYiSuM6j8bLa9Uc5h"
  ▼ "data" : { 5 items
      "name" : string "Solien #582"
      "symbol" : string "SRDT"
      "uri" : string "https://ipfs.io/ipfs/QmSTf4BPk56ozEFwDotw18Zg79Ku6Cm7GjuQmB27pGwrsm"
      "sellerFeeBasisPoints" : int 1000
    ▶ "creators" : [...] 5 items
    }
```



The first picture is the relevant metadata of the NFT that Aaron owns. The most important data is the "mint," the unique hash of the NFT on the Solana blockchain, and "uri," the place where this image is stored. The second picture above is an image (from the uri) attached to the NFT, which is the unique Solien Alien. Below are more examples of other unique Solien Aliens in the collection:

7'0" 84
6'6" 78
6'0" 72
5'6" 66
5'0" 60
4'6" 54
4'0" 48
DECENTRALIZE EVERYTHING
YOU ARE NOT ALONE
Give Us The Data

Now that there is a basic explanation of what a Solien NFT is, here is a general description of our project: SolienGallery will be a webpage that allows owners of Solien NFTs to make a digital image of a gallery of the Solien NFTs they own. When a user signs onto our website, they will be prompted to sign into their Solana wallet. Once a user signs in, the back-end will run a query on the NFTs that they own and return their Solien NFTs into a grid-like view on a webpage. The user will then select the NFTs that they want to display in their gallery and we will provide them with a new gallery image containing all Solien NFTs they choose to display. Actually signing into the user's wallet is a stretch goal of ours, as we will need to implement unfamiliar libraries from Phantom. Thus, for the scope of this course, we will initially have the user provide their wallet ID, which unfortunately doesn't allow the user to verify they own the NFTs in their gallery but still provides us with the same data from their public wallet.

For the scope of this course, we plan to leave the project at that. We ultimately want to provide a way to verify that a user owns the items in their gallery wallet after it has been initially created. (Some users may create a gallery from Solien NFTs that they initially own, then sell the NFTs later.) This post-gallery-creation verification will entail either minting a new NFT or creating a database which verifies that a user owns the NFTs in their gallery. This will be important if we plan to continue working on the project after COS 333 and publicly release the project, but it is more reasonable for this course to proceed without wallet verification.

Requirements

People who own NFTs tend to display images of their NFTs on their Twitter and other social media accounts, in a manner similar to how rappers might show off an expensive car or jewelry. It's easy to upload a single image, however, there is currently no service that allows users to easily compile a given wallet's images directly from the blockchain into a gallery-style image. Instead, a user would need to pull the NFT images manually. They would then need to compile their own gallery using photo-editing software, which takes a significant amount of time, effort, and energy. Our system will provide an easy and quick solution to NFT owners who would like to compile a gallery of their NFTs. We have kept the scope of this project to Solien NFTs, since we know the CEO personally, and do not want to deal with illegal images.

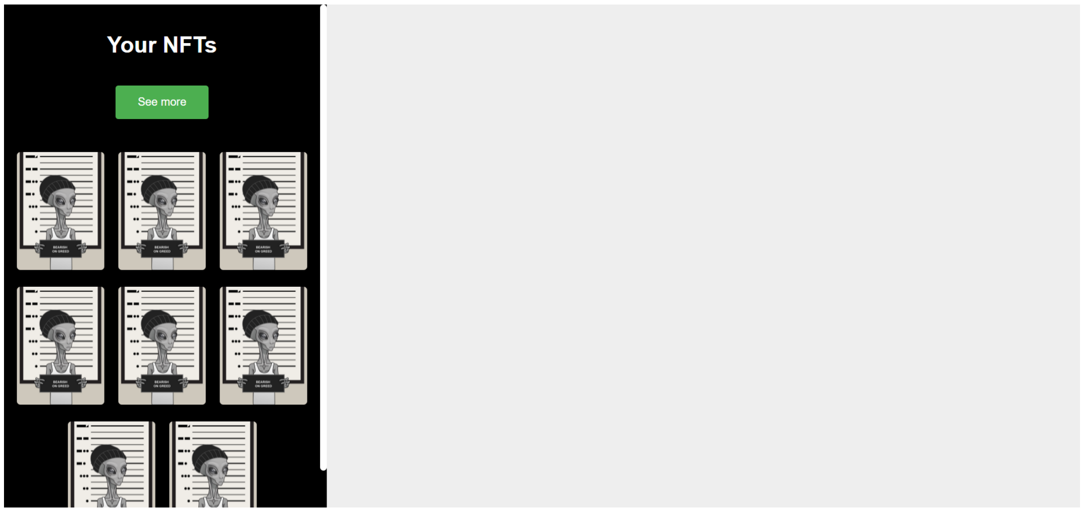
Functionality

The functional specifications for our project are relatively straightforward, as there will only be one main use case for our product. Indeed, we purposefully restricted ourselves to Solien NFTs so that we could handle the steep learning curve of blockchain technology without the complications of various use cases (e.g., NFTs with audio and video metadata).

There will be three distinct ways for the user to interact with our system:

1. ***Website Homepage***: The homepage will prompt users to enter their wallet's public key into a text field in the middle of the screen. (You can think of the public key as a username.) We want a clean and simple user interface here. As a stretch goal, we may have a rotating alien figure in the background of the homepage, as an allusion to the Solien NFTs. We would use the three.js library for the 3D animation.

2. ***Gallery Editor***: After entering their public key, users will be taken to the gallery editor, which will be a split-screen interface. On the left-hand side, users will be given a grid view of the Solien NFTs that they own. On the right-hand side, users will see the gallery that they are creating. They will be able to click on an NFT on the left-hand side to bring it onto the gallery, and vice versa. As a stretch goal, users will be able to change the background of the gallery, as well as the dimensions of the image. We are currently working on a prototype for the gallery editor, a screenshot of which can be found below.



3. ***Download Page***: Once the user is finished creating their gallery, they will be taken to the download page, where they will be given the option to save their image. We would like a clean and simple interface here, too. As a stretch goal, we will give users the option to mint their newly created gallery as its own NFT.

Design

First, we present an outline of the system flow for the MVP:

1. User enters their wallet's public key in the text field on the homepage.
2. Client program sends a request for images of the Solien NFTs associated with the key.
3. Server program handles the request, querying the blockchain for the Solien NFTs.***
4. Server program converts the list of NFTs into a list of image URLs, which it subsequently returns to the client.
5. Client program displays the images of the NFTs in a grid view on the Gallery Editor UI.
6. Client program handles the gallery editing until the user clicks the "Download" button.
7. Client program sends a request to the server to convert the gallery to an image.
8. Server program handles the request, compiling the images of the NFTs into a gallery image and returning the result to the client.***
9. Client program prompts the user to download the image onto their computer.

The steps marked with "***" are expected to be the most challenging to implement. They are described in more detail below, as modules.

To make the distinction between the three tiers explicit: Steps 1, 2, 5, 6, 7, and 9 relate to the front end, steps 3, 4, and 8 relate to the backend, and step 3 relates to data storage (read-only). If we achieve the stretch goal of allowing users to mint their own gallery NFT, then we will be writing to the blockchain in a final step. Our front end will be written largely in HTML, CSS, and JavaScript (JQuery). Our backend will be written in Javascript (web3.js) and maybe Rust. We will be using Express as the server-side application framework, and we will ultimately deploy on Heroku.

Our backend has great potential for modular design. For example, we may very possibly need to change our implementation for querying the blockchain; if our queries are too slow, then we may want to cache some results for future use. Ultimately, our backend will consist of two modules:

1. ***solana_search.js***: This module will define a utility function that takes in a container object with a wallet's public key and outputs a list of the NFTs associated with the key. We pass in a container object rather than just the key itself, because we may want to pass in more data later in the project. Similarly, we do not only return the list of image URLs that we need, because we might want to use the other metadata in the NFT. This module relates to Step 3 in the list above.

2. ***image_util.js***: This module will define a utility function that takes in an object specifying a gallery and outputs an image of the gallery. In our initial implementation, the gallery object will likely be a thin wrapper for a list of images, but in later implementations, we may pass in more data, like the dimension and resolution of the image, as well as customizations for the background of the gallery. This module relates to Step 8 in the list above.

We will be using GitHub to manage our code repository. For our server side, we will be using Heroku. The clients will be a user's local machine's internet browser. As of right now, we do not have plans to support mobile browsing.

Milestones

For the MVP of our project, we aim to create a website for users to create galleries out of the Solien NFTs on a specific user's wallet. What this means is that users would go to our website and type in a wallet number, prompting us to traverse the Solana blockchain and return the Solien NFTs contained within that wallet. The user would then be able to select the individual images out of which they would like our website to create a gallery. Put even more generally, the input to our website would be a wallet ID, and the output to the user would be a gallery of the images they selected to include out of the pool of Solien NFTs contained within the wallet they specified.

We plan on working according to the following schedule. Each is labeled as either broadly a front end or a backend goal, and each entry roughly corresponds to the remaining weeks in the semester (they are in chronological order). Additionally, we have generally assigned Aaron Skepasts and Gabriel Lebeau to backend tasks, and Gabriel Grajeda and Nick Hovsmith to front-end tasks:

MVP:

1. Simulate a "Hello, World" traversal of our full-stack architecture, linking the front end to the backend (FE and BE).
2. Successfully make queries on the Solana blockchain itself, returning the contents of a user's wallet on the blockchain (BE).
3. Develop U/I to allow the user to visually customize their gallery according to their preferences, with mechanisms for users to edit the layout of the gallery (FE).
4. Connect the blockchain querying mechanism to the front end (FE and BE).
5. Successfully make queries on the Solana blockchain to return data specifically relating to Solien NFTs (BE).
6. Translate the images returned from the blockchain queries into images displayable to the client on the front end (FE).
7. Allow for the user to download the gallery they created onto their local device (FE).
8. Launch more robust end-to-end testing of our ecosystem (FE and BE).
9. Finalize the functionality and appearance of the U/I and the connection between the front end and the blockchain (FE and BE).

It is critical to note that, in our MVP, there is no step that verifies that the wallet input by the user is owned by the user.

Stretch Goals:

10. Optimize the way we traverse the blockchain when obtaining wallet data using caching and sub-linear[2] search (BE).
11. Verify that the user owns the wallet they are searching for (BE).
12. Mint an NFT out of the gallery we've created and, in doing so, write to the blockchain. This step is necessarily contingent upon completing step 11 (BE).

We look forward to taking steps towards completing our MVP, and we are excited to pursue these stretch goals, as well!

Risks

One risk is that there is a steep learning curve associated with cryptocurrency and writing programs that communicate with and navigate along the blockchain. We attempted to address this issue by proposing an achievable minimum viable product. Additionally, we have already begun doing research on this topic and have made promising findings. Another risk that we are taking is learning new programming languages. Some of the languages that we may be learning, either from scratch or from some limited level, include HTML, CSS, Javascript, and Rust. Additionally, we will be learning how to use new tools and frameworks, including Heroku, Node JS, and Express JS. Outside of learning related risks, one risk we face that depends upon things outside of our control relates to Twitter. Twitter is expected to introduce a tool for users to link to their verified wallets. While we are not relying on this feature, it would make one of our stretch goals easier, as we would be able to utilize this as our own verification tool rather than creating one from scratch. In light of these risks, we are prepared to take the necessary steps to be able to utilize these concepts, tools, and languages in our project. We look forward to the learning that will ensue as a result.

[2] "Linear" with respect to the number of transactions on the blockchain.