***Endpoints:***

/getToken

Supported Methods: GET

Query Params: csrfToken

Description: This endpoint retrieves information the csrf token.

Supported Return Types: JSON

/getUser

Supported Methods: GET

Query Params:

Description: This endpoint retrieves information about the currently logged in user.

Supported Return Types: JSON

/getOtherUser

Supported Methods: GET

Query Params:

Description: This endpoint retrieves information about the owner of a purchased NFT

Supported Return Types: JSON

/getNFTs

Supported Methods: GET

Query Params:

Description: This endpoint retrieves information about all nfts from the database.

Supported Return Types: JSON

/getNFTsForSale

Supported Methods: GET

Query Params:

Description: This endpoint retrieves information about nfts that are listed as “for sale” from the database.

Supported Return Types: JSON

/login

Supported Methods: GET, POST

Query/Body Params: username, password

Description: This endpoint retrieves information about the login page as well as confirms a user is valid and logs them in

Supported Return Types: JSON

/signup

Supported Methods: POST

Body Params: username, password, password2

Description: This endpoint creates new users and saves them to the database

Supported Return Types: JSON

/reset

Supported Methods: POST

Query Params: username, currentPassword, password, password2

Description: This endpoint resets the users password to a new password

Supported Return Types: JSON

/logout

Supported Methods: GET

Query Params:

Description: This endpoint logs the user out and returns them to the login page

Supported Return Types: JSON

/addValue

Supported Methods: POST

Body Params: increaseAmount

Description: This endpoint adds or subtracts from a users balance by a certain amount

Supported Return Types: JSON

/addOtherValue

Supported Methods: POST

Body Params: domo id

Description: This endpoint adds to an owners balance when their nft is purchased

Supported Return Types: JSON

/maker

Supported Methods: GET, POST

Query/Body Params: name, value

Description: This endpoint retrieves information about the maker page as well as makes new nfts and saves them to the database

Supported Return Types: JSON

/forSale

Supported Methods: POST

Body Params: nft id

Description: This endpoint sets an nft from being “not for sale” to “for sale”

Supported Return Types: JSON

/shop

Supported Methods: GET, POST

Query/Body Params: nft id

Description: This endpoint retrieves information about the shop page as well as changes the owner of nfts and saves them to the database when bought

Supported Return Types: JSON

/delete

Supported Methods: DELETE

Body Params: nft id

Description: This endpoint deletes an nft from the database

Supported Return Types: JSON

/

Supported Methods: GET

Query Params:

Description: This endpoint return the user to the login page

Supported Return Types: JSON

***Function & Purpose:***

The site is a modified version of the DomoMaker assignment, in which Domos have been turned into NFT’s (non-fungible tokens). The site allows for users to create and sell their NFTs, and other users can buy said NFTs and gain ownership of them. Users will be required to add Etherium cryptocurrency to their account from the shop to purchase NFT’s, and users who sell their NFT’s will receive the value in ETH in their balance.

***Using React.js:***

React is being used to draw all of the DOM elements. It is split into 3 handlebars pages as well as 3 .jsx files for the login, maker, and shop respectively.

***Data Storage:***

Currently Mongo stores the user information (username & password) as well as the NFT information. More specifically the NFT’s name, monetary value, owner, and image.

***Using Handlebars:***

Handlebars is used to display the login, maker, and shop pages. Each are in their own .handlebars file that is updated with React as information changes.

***Above and Beyond:***

If I’m being completely honest, it’s tough to say I went very above and beyond. I’m happy with the implemented shop system, as it was pretty hard to get working, but I’m disappointed I was unable to get the image uploading system functioning along with it.