
Galaxy.Finance Final Report

Fall 2020

Sean Murphy

Website URL:<https://hardcore-visvesvaraya-c62dc3.netlify.app/>
GitHub:<https://github.com/seanmurphy355/Galaxy.Finance>

Quick Definitions List:

[Ethereum](#): Ethereum is a decentralized open source blockchain featuring smart contract functionality.

[Smart Contract](#): Is a computer program or a transaction protocol which is intended to automatically execute, control or document legally relevant events and actions according to the terms of a contract or an agreement.

[Web3](#): Is a collection of libraries that allow you to interact with a local or remote ethereum node using HTTP, IPC or WebSocket.

[Staking](#): The process of actively participating in transaction validation (similar to mining) on a proof-of-stake (PoS) blockchain.

[USDC](#): Is a stablecoin that is pegged to the U.S. dollar on a 1:1 basis.

[ERC20](#): ERC-20 tokens are tokens designed and used solely on the Ethereum platform.

I. Introduction

Galaxy.Finance is a web3 application that allows users to stake nUSDC (notUSDC), a fake asset that was created for the sole purpose of this project. nUSDC acts as a synthetic asset that attempts to emulate USDC which is a stable ERC20 asset that attempts to match the US dollar. When users stake their nUSDC on the Galaxy.fiance platform they are then able to earn the platform's native ERC20 token called Galaxy (GLXY). The Galaxy token itself acts as a governance token. In theory a governance token is able to be used by users to allow them to vote on what direction the platform should move. This is a relatively new idea in the cryptocurrency space that enables the decentralized governance of projects.

It should be noted that Galaxy.Finance was mainly created in-order to learn the following skills: 1. Coding ERC20 based smart contracts through the use of the solidity programming language, 2. Learning the process of creating, developing and testing; ERC20 based smart contracts (note: the development process is critical for ethereum based development due to smart contracts being immutable once deployed to the mainnet), 3. Developing against a local testnet and 4. Creating and developing a React application that utilizes web3.js. In essence this project's main purpose was to develop critical skills that are necessary for blockchain development. That being said none of the contracts that were developed in this project will be deployed to the mainnet due to fear that blockchain users could mistakenly interact with these contracts which could cause a loss of actual monetary assets.

II. User Perspective

When it comes to a user's perspective of Galaxy.Finance this is a rather complicated subject due to the contracts not being deployed on the blockchain. Because of this we can explore two different types of user perspectives. The first perspective being the theoretical perspective of Galaxy.Finance. In theory a user could stake their nUSDC on the platform and accumulate the native galaxy token. It should be noted that the native galaxy token would only be accumulated by users that decide to become validators (stakers). The galaxy token would then eventually be used by these validations in-order to enable decentralized governance over the Galaxy.Finance platform.

The second user perspective is the actual user perspective of what Galaxy.Finance does in its current state. Because none of these contracts will be deployed to blockchain users will need to set up their own at home blockchain testnet in-order to interact with the galaxy.finance application. Once a user has gone to the github repository and downloaded the project and installed all required dependencies they should be good to go. All a user would need to do after they installed the project properly is deploy the smart contracts to their local testnet and run the react application. After the react application is live the user should have some nUSDC available to them in their metamask account. This nUSDC can be used in-order to stake on the Galaxy.fiance application. While staked if the user runs the rewards script it should generate GLXY rewards and update their balance. If the user is currently unstaked and they run the generate rewards script there should be no rewards that are distributed.

Note: The reason for the two different user perspectives is mainly due to the fact that deploying these contracts to the testnet could cause actual harm for those that decide to interact with these contracts.

III. Usage Instructions

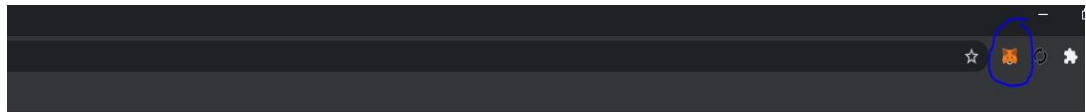
This section will provide users with an in-depth guide with how to correctly run this project on their local machine. Note: this section will go through all the steps required in-order to achieve the second user perspective.

Before doing anything else please clone the project's (github) and ensure that you install Node.js in the directory that you choose to store this file.

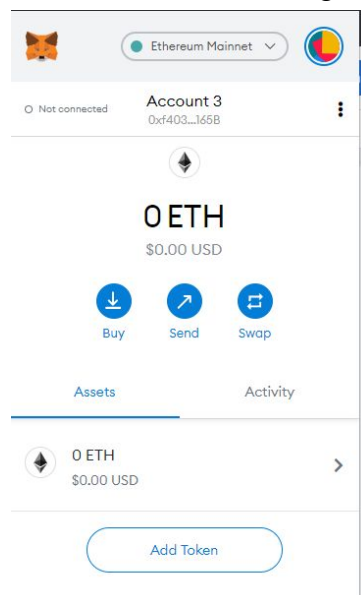
Critical dependencies: `npm install -g npm` and `npm install -g truffle`

A. Section One Setting Up a MetaMask Wallet

1. Download and Setup the Metamask Extension for google chrome. [Link](#)
 - a. A correct installation should result in fox icon pinned to chrome



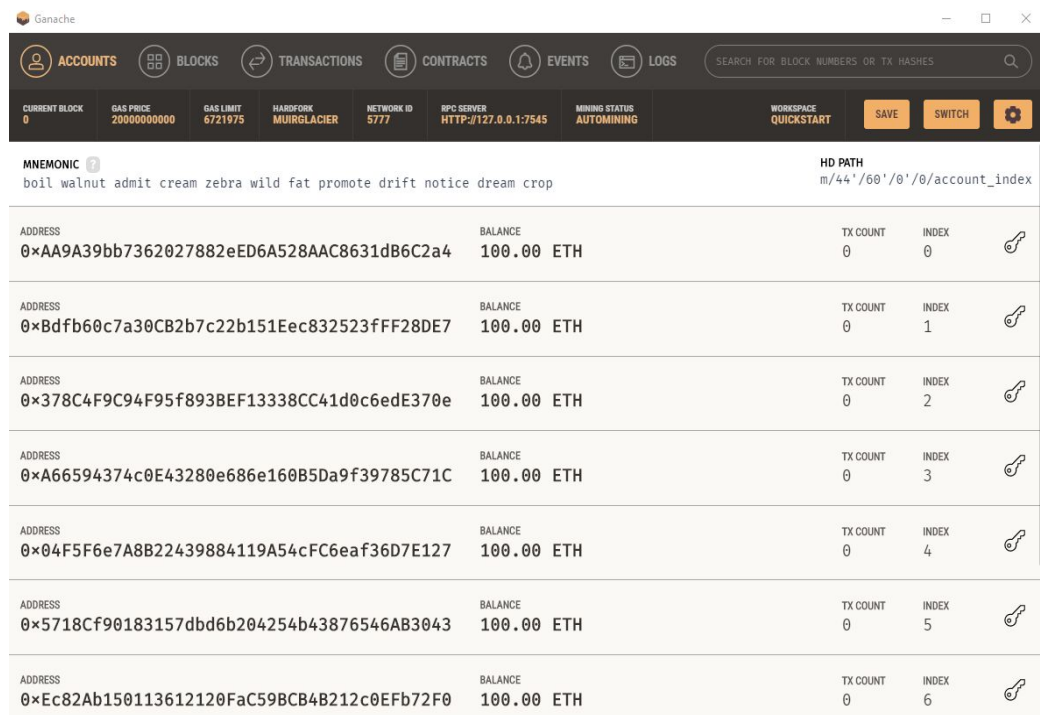
2. Once MetaMask has been installed you can open up the application and create a password.
3. After your password has been created metamask will provide you with a recovery phrase (note: you should store your account recovery phrase in a secure location.)
4. Once all the above steps are completed you are good to go. Note: you should have something that looks like the image below.



If Extra help is needed follow this guide: [YouTube Metamask Wallet Setup](#)

B. Section Two Setting Up a TestNet

1. Download Ganache <https://www.trufflesuite.com/ganache> this will be used as our at home blockchain.
2. When opening the application hit the QuickStart Ethereum option.
3. If you did the above steps correctly you should have something like the image below.



4. Run the `npm install -g truffle` command in the Galaxy.finance project directory on your local machine.

C. Section Three Utilizing Solidity and Deploying Contracts

1. Download The solidity programming language in the directory that you are setting this project up in. If you are using visual studios code then all you must do is navigate to the extensions page and download solidity.
2. If you require further assistance please navigate to [link](#)
3. To ensure that truffle, ganache and solidity are all set up correctly run the following commands.
 - a. `truffle compile` this command will compile all sol files.
 - b. `truffle migrate` this command should migrate all contracts to your local blockchain.
 - c. `truffle test` this command will ensure to run all the tests in the project suite. Note that these test cases should not give back any errors.

```

Deploying 'StakerContract'
-----
> transaction hash: 0xd9304b5a0afa30bc05b8fdb2ecaff21fc0ddcb79359c98069a3c0de1166875ca
> Blocks: 0       Seconds: 0
> contract address: 0x1512fC6aABadf8Eb5181eBae41456EdA4EdED148
> block number:    5
> block timestamp: 1605662563
> account:         0x6f32557C8a8BD96FDc2A19f2E1947C1ABA01143D
> balance:         99.94720016
> gas used:        885576 (0xd8348)
> gas price:       20 gwei
> value sent:      0 ETH
> total cost:      0.01771152 ETH

> Saving migration to chain.
> Saving artifacts
-----
> Total cost:      0.04744784 ETH

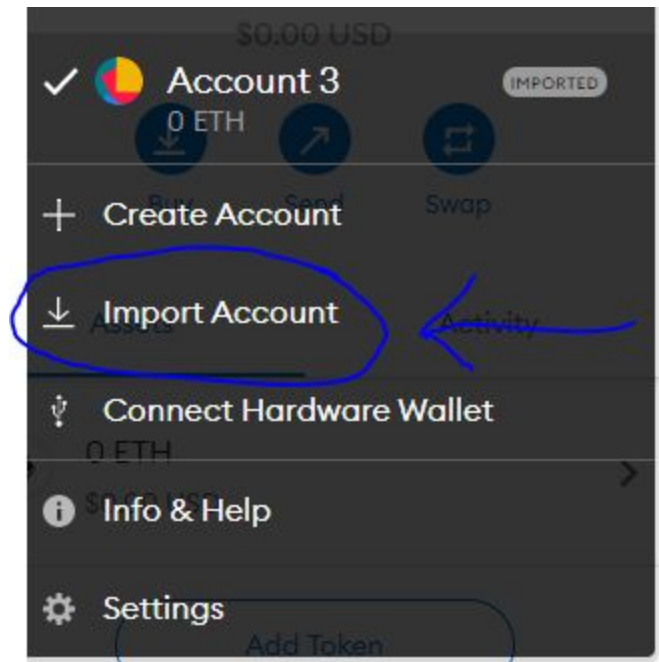
Summary
=====
> Total deployments: 4
> Final cost:      0.05195258 ETH

```

The above image is what a correct Migration test looks like: note the values you get should not be the same as the ones displayed in the image. The only value that should be the same is the total deployments.

D. Section Four Interacting with Galaxy.Finance

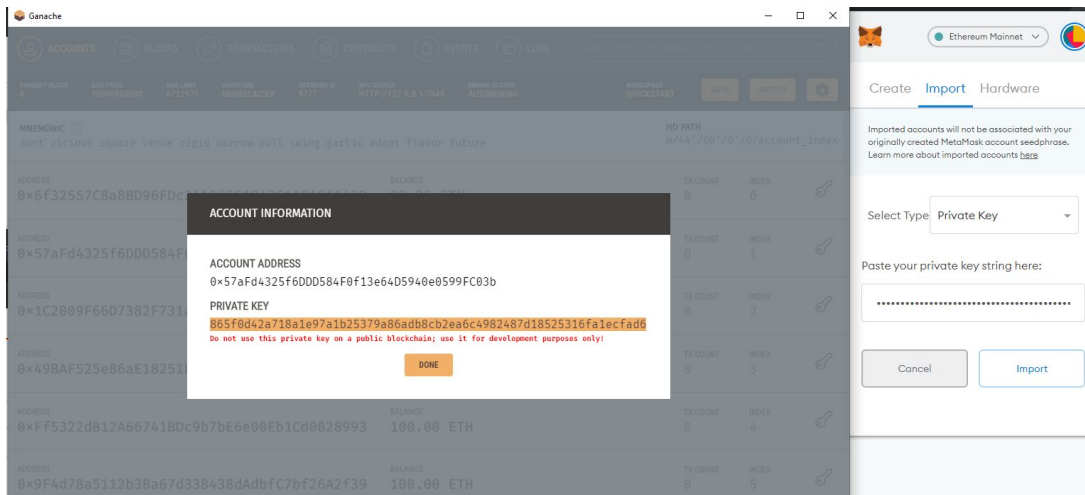
1. After Contracts have been deployed and all other sections have been accurately followed. It is now time to interact with the galaxy.finance application. First we will make sure that we have the proper metamask account. Navigate to the metamask menu and hit import account.
- 2.



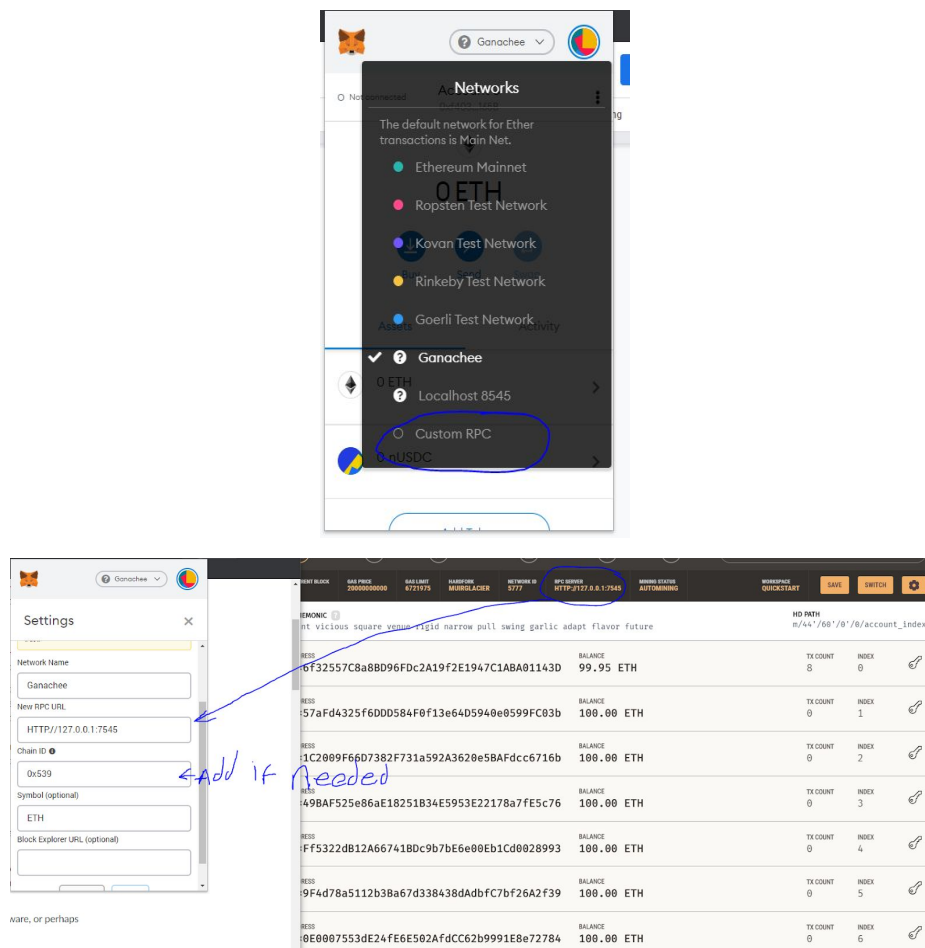
3. On ganache your blockchain owner will be deployed from the index zero account and the wallet that will generate nUSDC will be the index 1 account. Hit the key icon to the right in-order to obtain that account's private key. (Note never send real funds to these ganache accounts)

ADDRESS	BALANCE	TX COUNT	INDEX	
0x6f32557C8a8BD96FDc2A19f2E1947C1ABA01143D	99.95 ETH	8	0	
ADDRESS	BALANCE	TX COUNT	INDEX	
0x57aFd4325f6DD584F0f13e64D5940e0599FC03b	100.00 ETH	0	1	

4. Import the index 1 account and it should look like the image below.

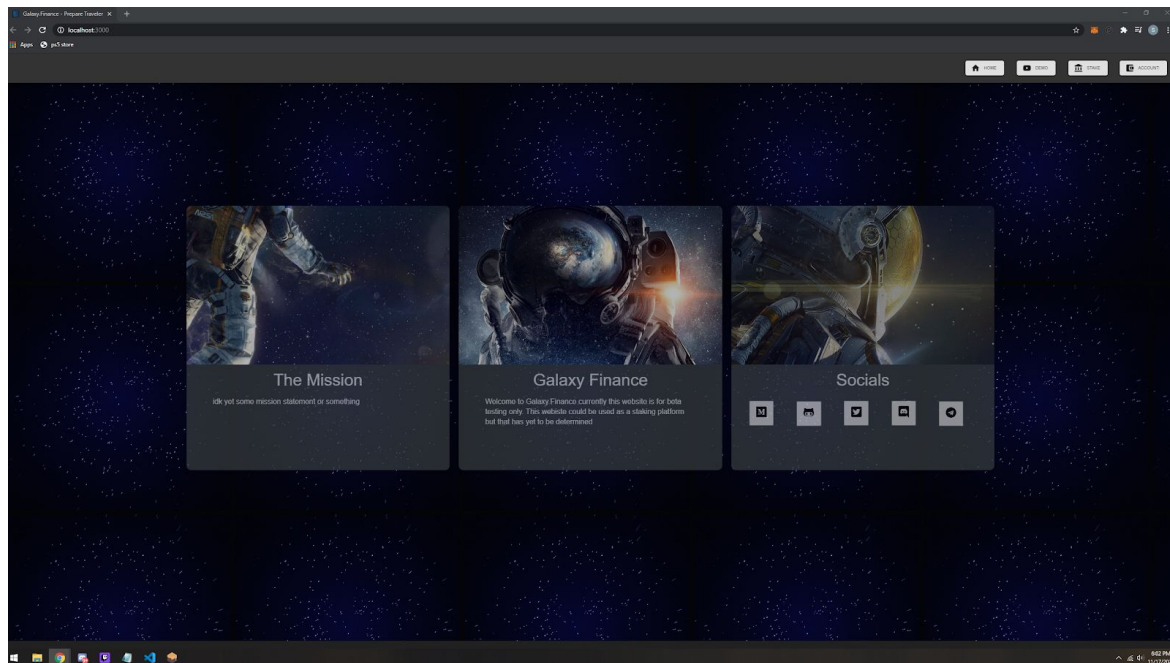


5. Finally on metamask add the Ganache network by hitting on the networks button on the top of the wallet. Then navigate to the custom RPC option. Follow the images provided below.

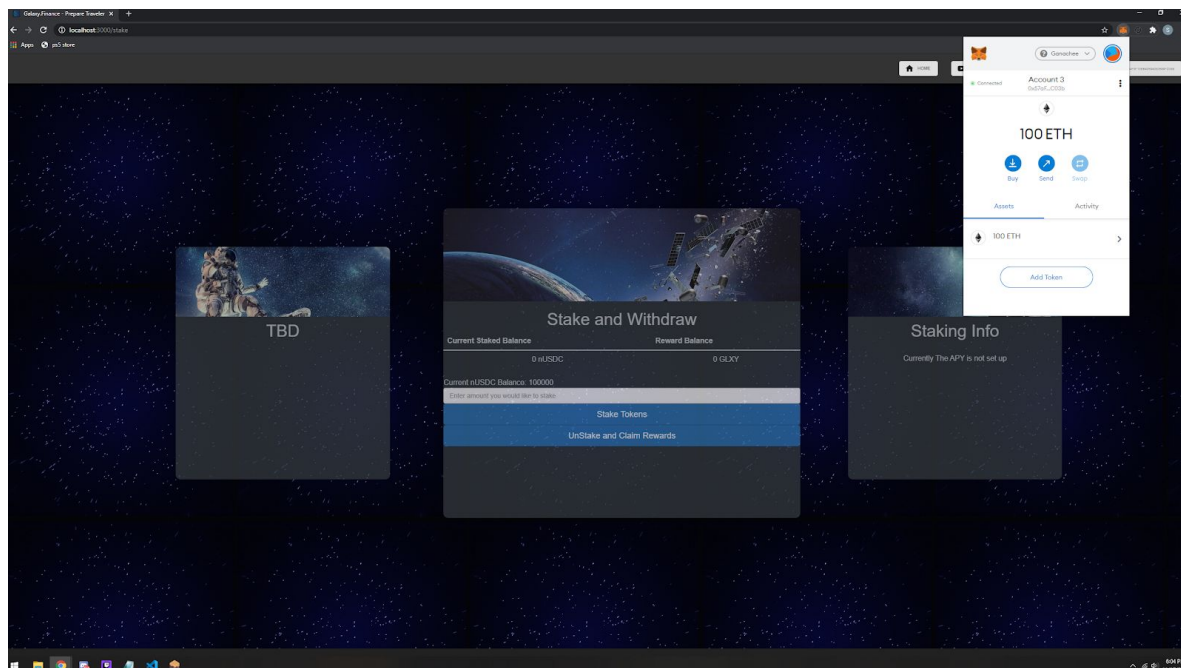


6. With your wallet now on the correct rpc server run the following command
`npm start` in-order start the galaxy.finance application.
7. The application should run on your local host port 3000. Below will contain some images of the application.

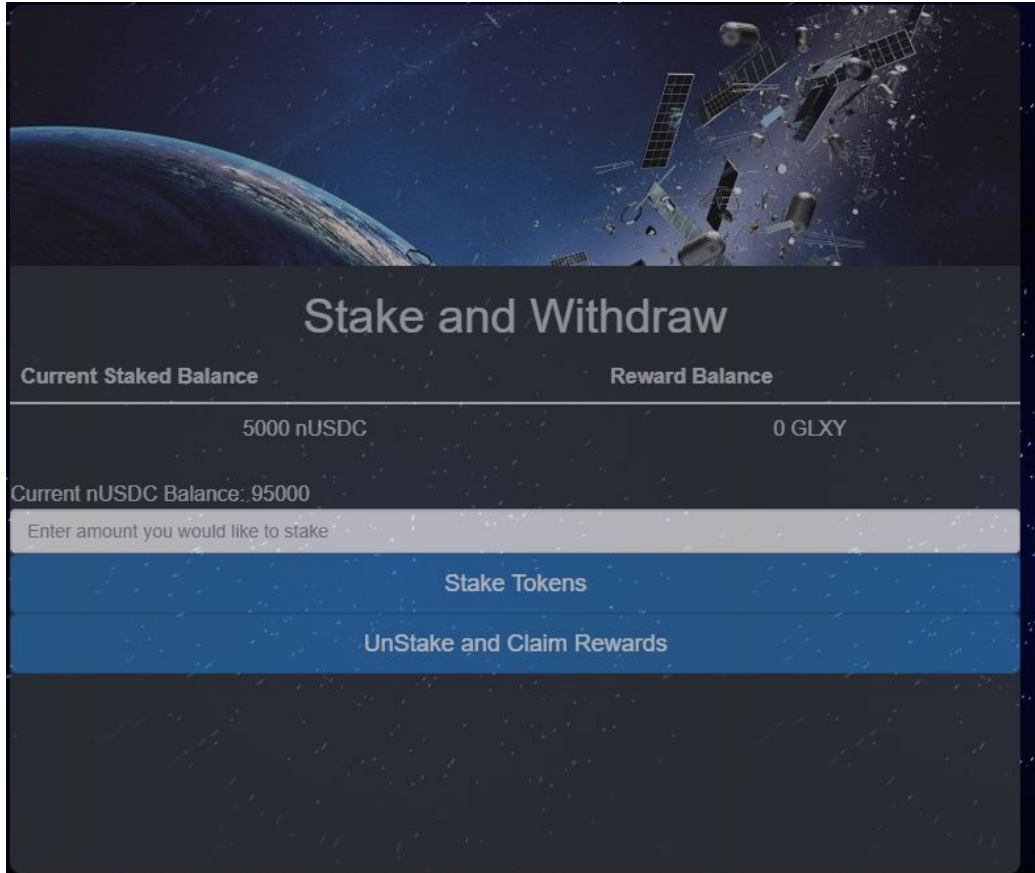
1.Home Page



2.Staking Page



3.Staked Funds



The interface features a dark theme with a space-themed background showing a planet and satellites. The title "Stake and Withdraw" is centered at the top. Below it, two columns display balances: "Current Staked Balance" at 5000 nUSDC and "Reward Balance" at 0 GLXY. A text label "Current nUSDC Balance: 95000" is positioned to the left of a light gray input field labeled "Enter amount you would like to stake". Below the input field are two blue buttons: "Stake Tokens" and "UnStake and Claim Rewards".

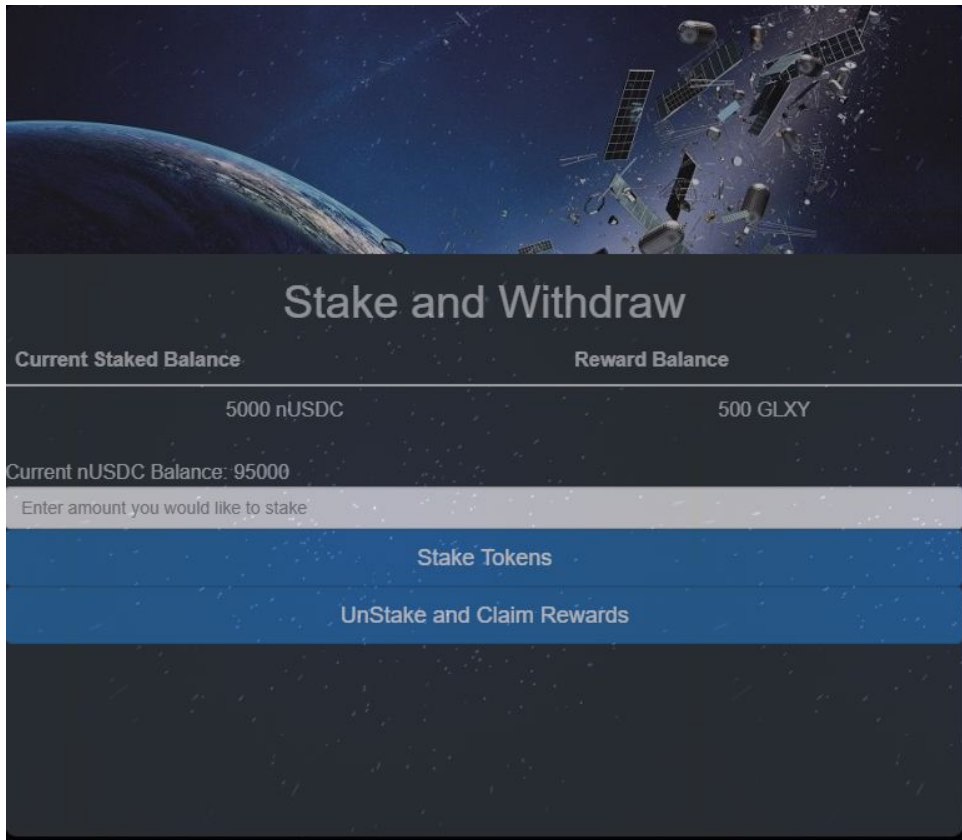
Current Staked Balance	Reward Balance
5000 nUSDC	0 GLXY

Current nUSDC Balance: 95000

Stake Tokens

UnStake and Claim Rewards

4.Rewards issued



This interface is identical to the previous one, but the "Reward Balance" has been updated to 500 GLXY, indicating that rewards have been issued. The "Current Staked Balance" remains at 5000 nUSDC, and the "Current nUSDC Balance" remains at 95000.

Current Staked Balance	Reward Balance
5000 nUSDC	500 GLXY

Current nUSDC Balance: 95000

Stake Tokens

UnStake and Claim Rewards

Final Usage notes: Once the user has achieved deployment of all contracts, set-up a metamask account on the proper rpc server and launch the application; The application will run flawlessly in regards to staking and un-staking user funds. That being said when it comes to distributing rewards to a staked user you must run the reward script. The command to run the reward script is `truffle exec script/go.js`. This step is a bit more complicated so please watch the demo video that is linked on the websites page for more in-depth information in regards to this.

IV. Systems Used

The Galaxy.Finance application uses several different systems in order for the application to work properly. One key component of the Galaxy.Finance application is the metamask wallet. Metamask is a distributed web bridge that allows users to run ethereum dApps (decentralized applications). Essentially metamask allows users to interact with a blockchain application without running a full ethereum node. This allows for users that hold crypto currencies to interact with different blockchain ecosystems purely through the use of their browser. Galaxy.Finance was developed by creating two separate ERC20 (ethereum based) cryptocurrencies (GLXY and nUSD). Once these tokens are created and distributed a user can store them in their metamask wallet and interact with the different contracts on galaxy.finance that requires the user to hold nUSD in-order to be considered a validator on the application. Thus the use of Metamaks is integral if a user would like to interact with the Galaxy.finance application.

Another key system that was used during the development cycle of the galaxy.finance application was the ganache/truffle testing suite. Ganache is a personal blockchain that is used for rapid Ethereum and Corda distributed application development. The ganache tool is commonly used across the entire development cycle allowing a developer to continuously: develop, deploy and test decentralized applications in a safe and secure environment. Ganache is part of the truffle suite which is a world class development environment that provides blockchain developers with tools that can be used as a testing framework, and asset pipeline for blockchains using the Ethereum Virtual Machine (EVM). Diving into how these tools were specifically used in conjunction with creating the galaxy.finance; Ganache was used as a local blockchain that the truffle suite could talk to when contracts were to be deployed to the projects local testnet. An example of this would be once smart contracts are ready to be deployed on the mainnet the developer must migrate the contracts onto the ethereum mainnet; Through the use of truffle and ganache when the nUSDC, Glxy and staker contracts were ready to be deployed they were instead deployed to the Ganache testnet. This development cycle allowed the creation of smart contracts that were used in galaxy.finance to be done in a safe and secure manner. Another key feature that the truffle development suite offers is tests. The test feature that truffle offers allowed the smart contracts that were involved with building galaxy.finance to be tested before deploying them to the testnet. This is critical because once smart contracts are deployed to the blockchain they are immutable. All and all the truffle test suite was an integral part of the development cycle of this project.

The solidity programming language is an essential system that was used in-order to develop the galaxy.finance application. Solidity is an object oriented programming language that is used for writing smart contracts. The Galaxy.Finance ecosystem consists of 3 major smart contracts that make up the overall functionality of the website. Each smart contracts carried out a specific role in regards to functionality of the Galaxy.Finance application; nUSDC acts the platforms fake USDC currency, while Galaxy(GLXY) acts as the applications native governance token and the staking contract acts as a validator contract that provides GLXY rewards to users who decide to become validators on the application. The development and testing of these immutable contracts was critical during the development of the galaxy.finance application. The reason being is because once these contracts are deployed to the mainnet they become immutable. That being said these smart contracts are still immutable when deployed to the testnet; The only difference being that when a smart contract is deployed to the testnet it can be continuously redeployed with no harm potential harm to the ethereum mainnet.

Web3.js also played a critical role in the creation of the galaxy.finance application. Web3.js at its core is a collection of libraries that allows users to interact with a local or remote ethereum node, through the use of HTTP or IPC connections. Essentially web3 enables communication between crypto users that hold funds in some sort of ethereum based wallet such as metamask, ledger or myetherwallet to communicate with online web applications that utilize the ethereum network. Galaxy.Finance's utilization of the web3 enables users to interact with the ecosystem's smart contracts.

Galaxy.finance utilized React.js which helped bring the galaxy.finance web application to life. React.js is an open source, front end, javascript library that is used for building user interfaces (UI components). React.js was utilized in-order to build most of the UI components that a user would interact with while on the galax.finance page. Some libraries that were used in tandem with React.js were material UI and Bootstrap. Both of these frameworks allowed for development of the galaxy.fiance's UI to be done in a somewhat streamlined manner.

Note: The approach of using libraries to create a somewhat streamlined UI provided valuable insight as to how powerful the available react libraries are.

V. Tools Used

- [Visual Studios Code](#): Visual Studio Code is a free source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.
- [React](#): React is an open-source, front end, JavaScript library for building user interfaces or UI components.
- [NodeJS](#) : Node.js is an open-source, cross-platform, back-end, JavaScript runtime environment that executes JavaScript code outside a web browser.
- [Web3](#): is a collection of libraries that allow you to interact with a local or remote ethereum node using HTTP, IPC or WebSocket.
- [Material UI](#): Provides optional CssBaseline components.
- [BootStrap](#): Provides optional CssBaseline components.
- [Solidity](#) : Solidity is an object-oriented programming language for writing smart contracts.
- [Truffle](#) : a development environment, testing framework and asset pipeline for Ethereum, aiming to make life as an Ethereum developer easier.
- [Ganache](#) : Personal blockchain for rapid Ethereum and Corda distributed application development.
- [MetaMask](#) : web3 wallet application.

VI. Conclusion

Overall the goal of Galaxy.Finance was to be a learning experience from a development perspective as well a portfolio project to display blockchain development skills to potential employers. This project provided quite the learning experience in terms of what it entails to create a blockchain application that works on a testnet. The Galaxy.Finance application can also be seen as a strong addition to my portfolio; Displaying general skills of blockchain development. This application was never meant to be the most user friendly mainly because it was never meant to be deployed to the blockchain. In conclusion though this application is not the most user friendly when it comes to interacting with it; The development of the Galaxy.Finance application achieved the goals that were set out for the project.