How to use CARBAY

1 Introduction

This document contains detailed instructions to set up the Auction House Decentralized Application. If you are well-versed in the operation of decentralized applications and have the prerequisites installed, you can refer to the instructions from the README on GitHub or from Instructions for Running App (Section 3 of the document). This document was tested on a Windows 10 Home machine and was last verified on 27-August-2020. The other dependencies and their versions used for this project are stated below. Please read the complete document before starting the process.

- truffle v5.1.10
- Solidity v0.5.16
- Node ^12.18.0
- Web3.js v1.2.1
- lite-server ^2.3.0
- Ganache v2.4.0

2 Installation of Prerequisites

The prerequisites and dependencies to run the DApp are mentioned below:

Node.js: a JavaScript run-time environment. Click to install.

Truffle IDE*: a framework that allows you to easily develop, run and test smart contracts on the Ethereum Virtual Machine (EVM). Install the Truffle IDE globally from the windows terminal:

npm install -g truffle@v5.1.10

Liteserver*: is a lightweight development-only node server that serves a web app, opens it in the browser, refreshes when HTML or javascript changes, injects CSS changes using sockets, and has a fallback page when a route is not found. Install liteserver globally from the windows terminal:

npm install -g lite-server

Ganache: allows a developer to create an Ethereum blockchain locally to run tests, create accounts, execute commands, and inspect the state of the chain. Click to install.

MetaMask: is a browser extension that allows you to have crypto wallets and provides a gateway to decentralized applications (Web3 interfaces). Click to install.

** You need to have Node.js installed to install these packages.*

3 Instructions for Running DApp

Once the prerequisites are installed, follow the instructions below to run the DApp.

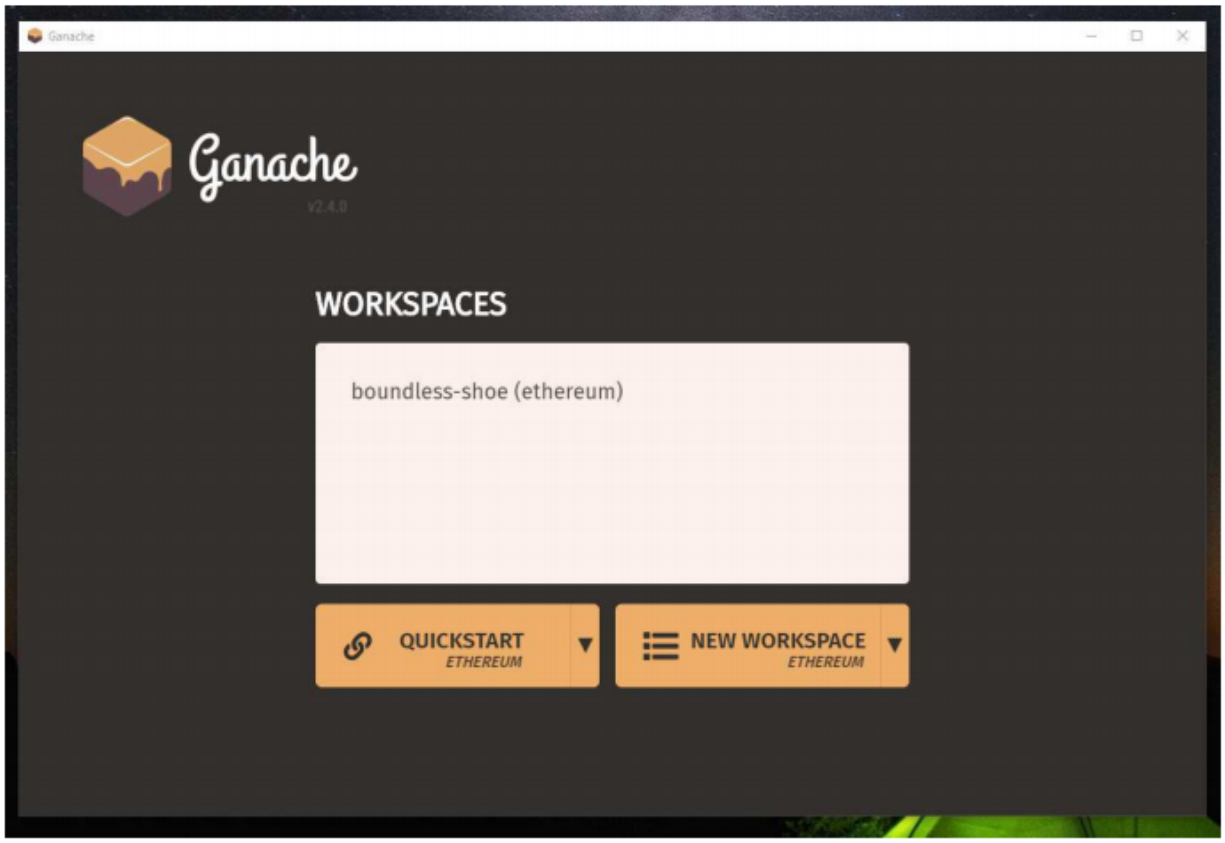
3.1 Running Ganache

Run and open Ganache. (If you have a Ganache environment already, you can skip this step)

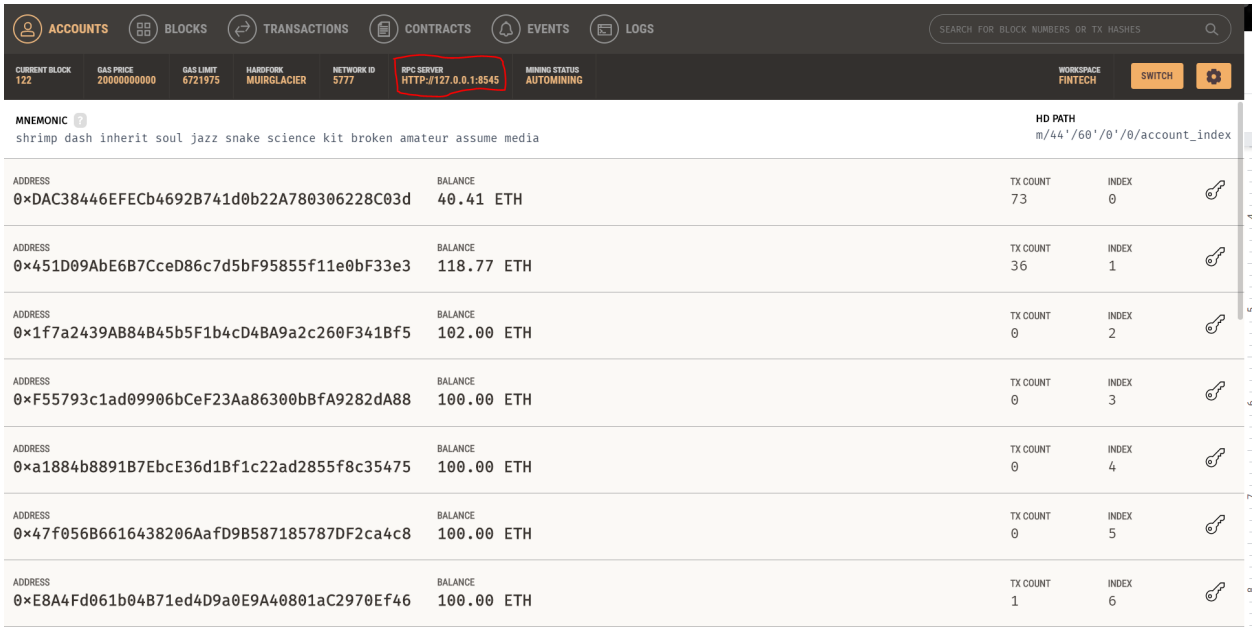
1. Click on the quickstart mode. In the Ganache Accounts tab you should see and have access to 10 addresses with 100 Testnet ether in each account.

Or

Click Create New Workspace. A new window will pop up with 10 test wallets (some of which you might have used previously)



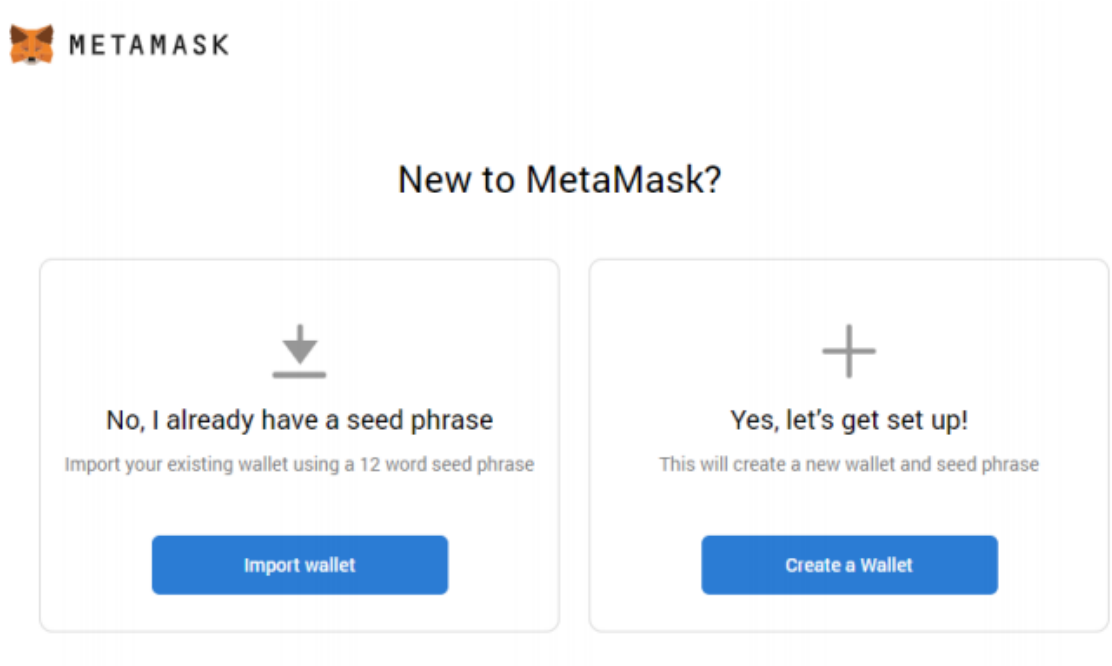
2. For either option above, you can go through different tabs to see more information about the blocks, transactions, etc. that have been processed. You can also link your project by clicking the contracts section and linking your Truffle project.



3.2 Connecting MetaMask to Ganache

1. Open Chrome browser (or Microsoft Edge).

2. Click on the MetaMask icon. This should open a new tab in your browser. Click Get Started and then and select: “No, I already have a seed phrase”.

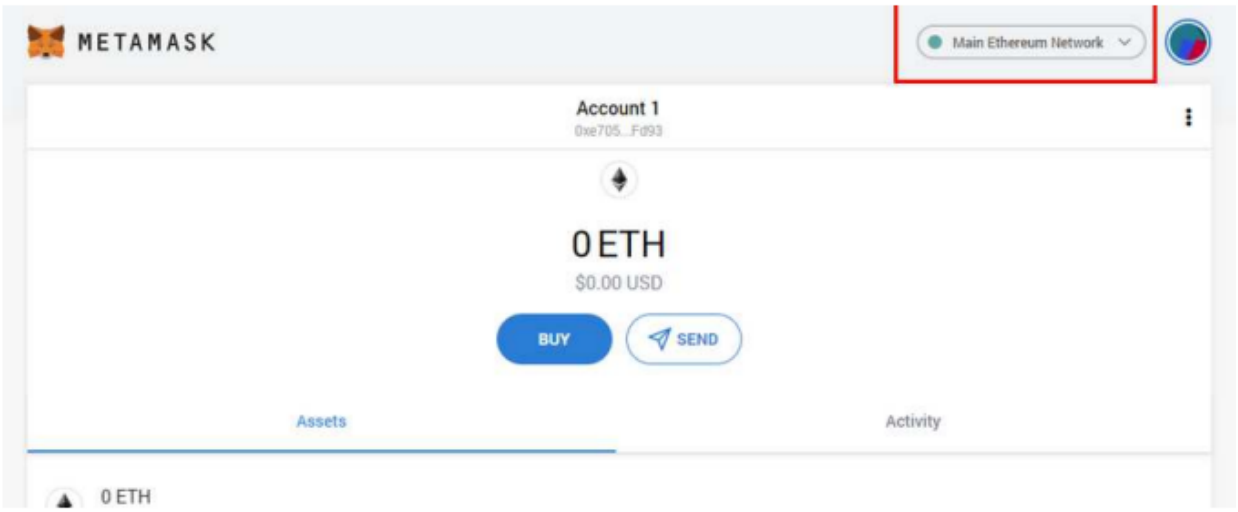


3. Click import using account seed phrase. Then click either I agree or No Thanks based on your preferences.

4. Copy the MNEMONIC from Ganache (the red box in the image in Section 3.1) and paste it into the text box under: “Enter your secret twelve-word phrase here to restore your vault. Seed phrase”.

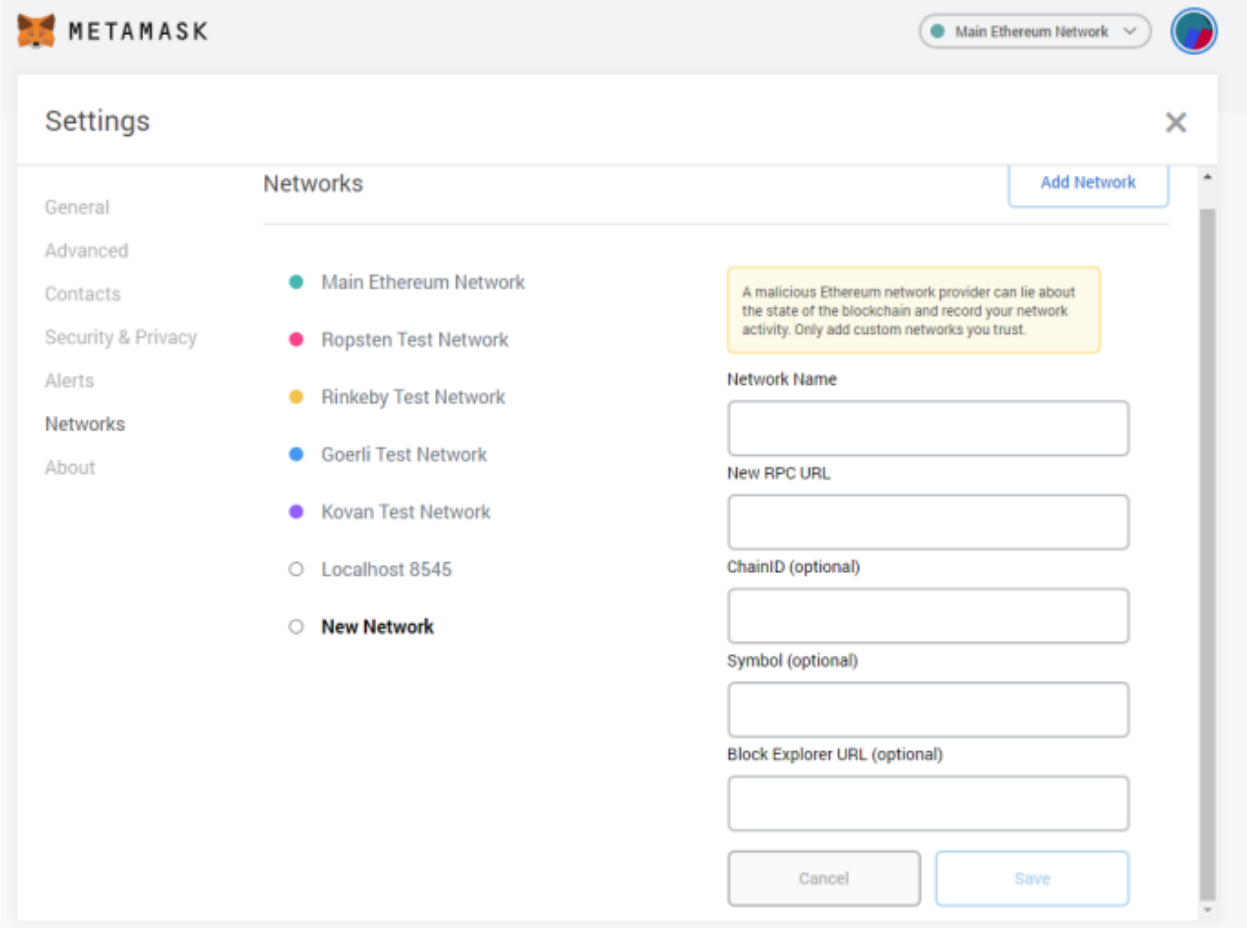
5. Confirm a password, agree to the Terms of Use, and then click “Import”.

6. To connect your wallet to the local blockchain that you locally deployed on Ganache, you need to change the network of MetaMask to match it with that of Ganache. The network that Ganache is running on can be seen at the RPC server box in the Accounts tab of Ganache (green box in Section 3.1).



7. Within the MetaMask extension on your browser, click the dropdown arrow located next to the text that says “Main Ethereum Network”. The dropdown is located at the top-right hand corner. Then, select Custom RPC.

8. Copy RPC server address from Ganache (by default, the server address is HTTP://127.0.0.1:8545) and paste it to New RPC URL. Give this network a name if you’d like and then click Save. Close the pop up.



9. By default, you should be connected to the RPC. If not, connect to it by opening the dropdown and selecting 127.0.0.1:8545. Now, the MetaMask wallet interface is connected to the chain and wallets initialized on Ganache.

3.3 Running the DApp

1. If you found this tutorial document online, clone the GitHub repo using the command: git clone this repo .

Or

If you downloaded the application package zip file directly, navigate to the Auction folder Page 4 located in the root directory.

2. Open a terminal from the root directory of the Auction folder. If you have a Windows machine, you can double-click the cmd.exe file to run a command-line terminal.

3. Once the terminal is open, test the contract functions using unit tests specified in test/TestAuction.sol (not essential) using the following command:

truffle test

4. Compile the contracts in the contracts folder using the command:

truffle compile

5. Migrate the contract to the local server on Ganache:

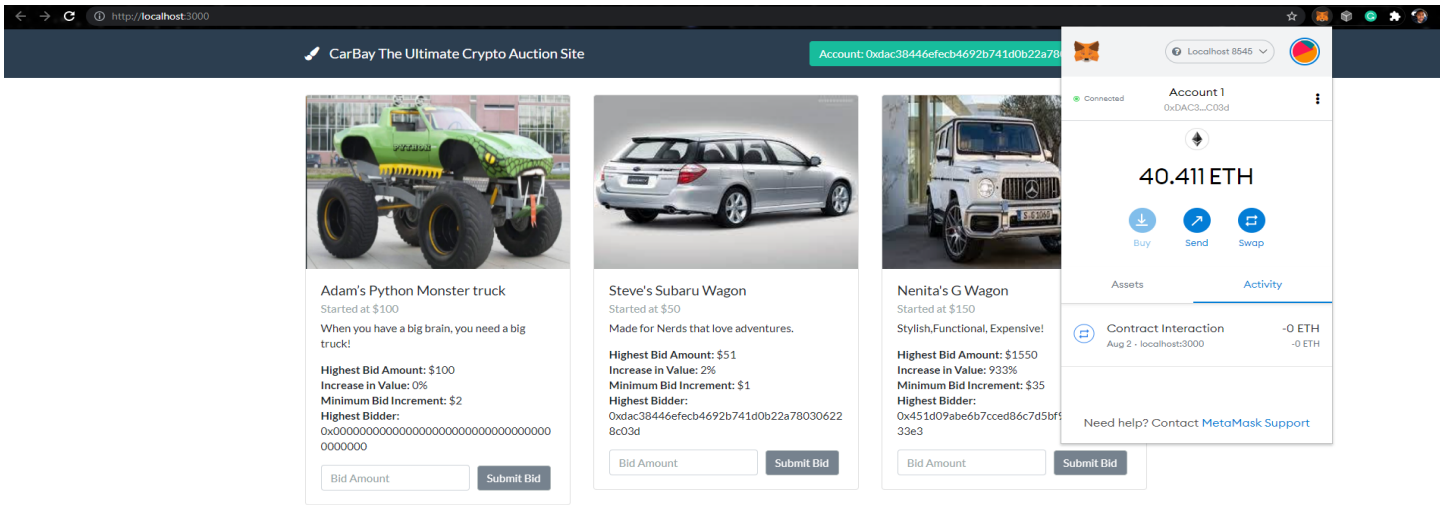
truffle migrate

6. Start the local web server:

npm run dev

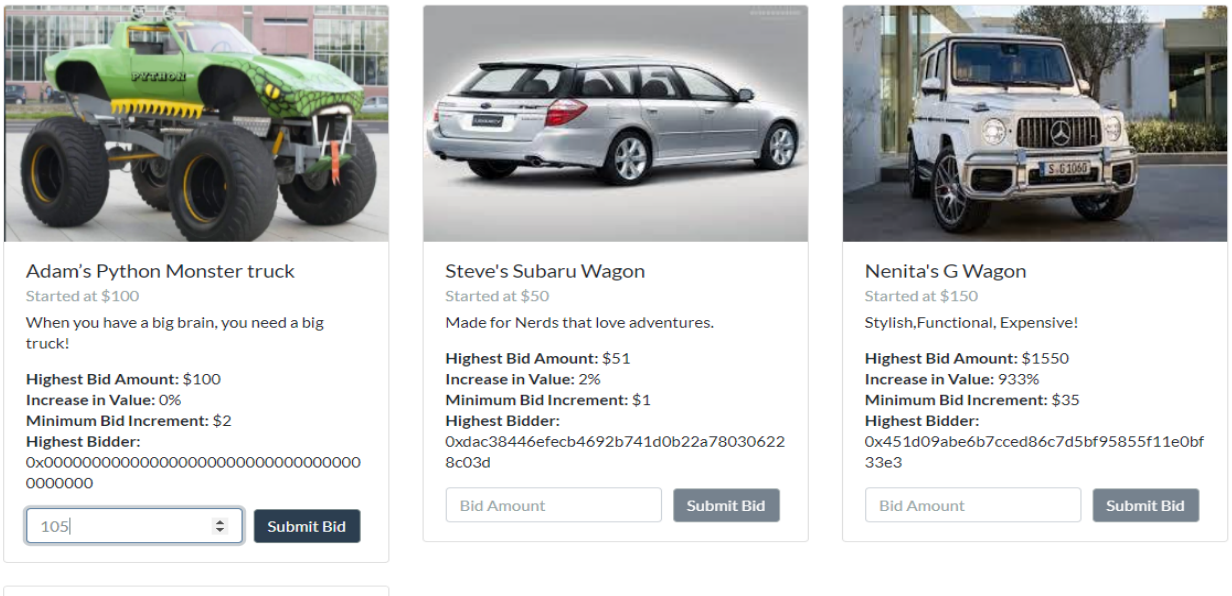
The dev server will launch and automatically open a new browser tab with the DApp.

7. When the website opens, a MetaMask pop-up should appear requesting approval to allow the app to connect to your wallet. Check the last 4 digits of the account in the MetaMask popup and make sure that they match one of the accounts in Ganache. Once you confirm this, click Next and then click Connect.

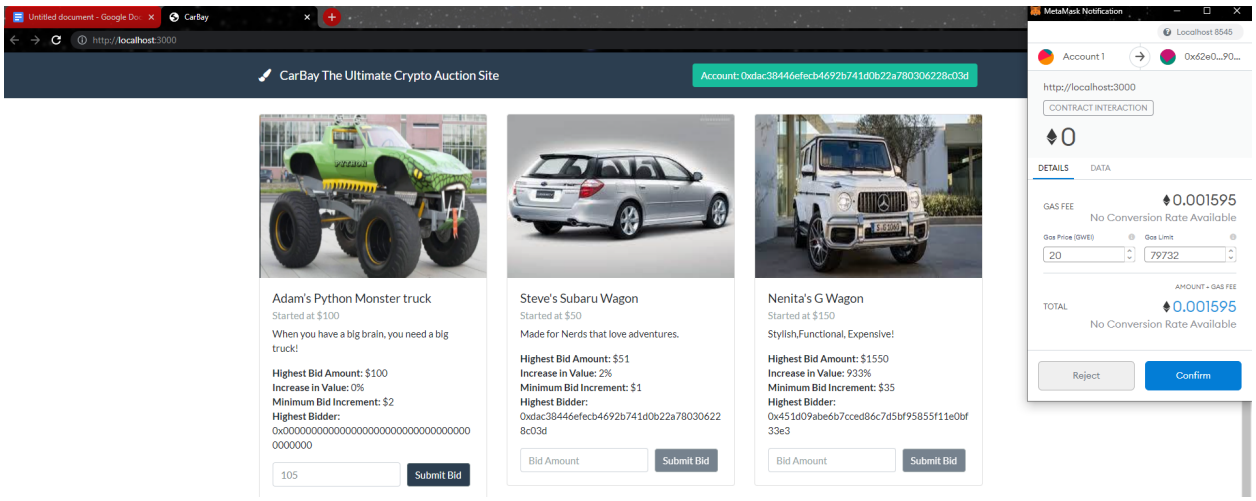


8. You should be able to see your account at the top right-hand corner of the app web page. If not, refresh the page.

9. The auction house has 4 items initialized by default. The web page displays descriptive and auction-related information about each item. You should be able to bid on items by entering integer values in the input boxes. By default, all “Submit Bid” buttons are disabled on the web page. In order to submit a bid on an item, you need to enter a value that is greater than the highest bid amount plus the minimum bid increment. You cannot submit bids on items where you are already the highest bidder.



10. Once you input a valid bid and click “Submit Bid”, a MetaMask popup will appear, requesting your approval for the transaction. Click Confirm to approve the transaction. After clicking confirm, the highest bid amount, highest bidder, and increase in value should update to reflect your bid. The transaction will also be reflected in the ”Blocks and Transactions” tabs in Ganache.



11. In case you do run into errors with MetaMask, refer to Section 4 for support

4 Error Debugging

Here, we present some commonly encountered errors and potential solutions for them:
• Error while compiling and migrating the contract: Make sure you are using the specified versions of dependencies.

• Error after compiling and migrating the contract: Compiling and Migrating the contract will create a new folder directory build/contracts with .json files for the smart contract Auction.sol and Migrations.sol. These .json files contain the “contract” information used to run the DApp locally. Delete these files before re-running the DApp.

• Alert:exception thrown in contract: The contract is designed to reject non-integer (strings and null value) and invalid values. Bids that are lower than the base price (first run) or highest bid amount and bids where the difference between your bid and the highest bid is less than the minimum increment are considered invalid. Try entering a valid value for bidding.

• Transaction failure with MetaMask: If you encounter any issue with MetaMask, reinstall it and follow the same procedure stated in section 3.2.