Problem Statement

Alice is Jewellery dealer and he is involved in exporting and importing jewellery from various countries and sells them to his customers. Now Alice would like to expand his business and would like to make an online store. He has approached you for developing an application to build the Online Jewellery applicaftion. As Jewellery is exported and imported there is chance that it can be falsified in the transit. He has asked you to provide a solution for this.

Because of immutability and auditability features of blockchain technology, you have suggested him to build it on Ethereum platform. Now Develop an Online Jewellery application using solidity and any of the front-end Java scripting languages that you are aware of.

Here are the functional requirements from Alice.

- List of Jewellery that are available with Alice are to be displayed as a catalogue with below parameters

S. No	Jewellery Parameter	Value
1	Type	Ring / Chain / Stone
2	Price	200 / 300 / 400
3	Material	Gold / Platinum / Silver

- A Customer, who is interested in the Jewellery, can select the appropriate one and Purchase it.

- Once he authorizes the Purchases transaction against the item, it will be show as "Success"

- Ensure, the same item is not for other customers to "Purchase"

Note: The Customer should have an Ethereum account or an Address, so that from his Account, the transaction can be initiated.

Project Analysis:

Alice wants to build the online store to sell jewelleries. This can be easily develop e-commerce website using centralized technology. According to the problem statement, he wants to build a distributed system so that it never happened falsified in the transit.

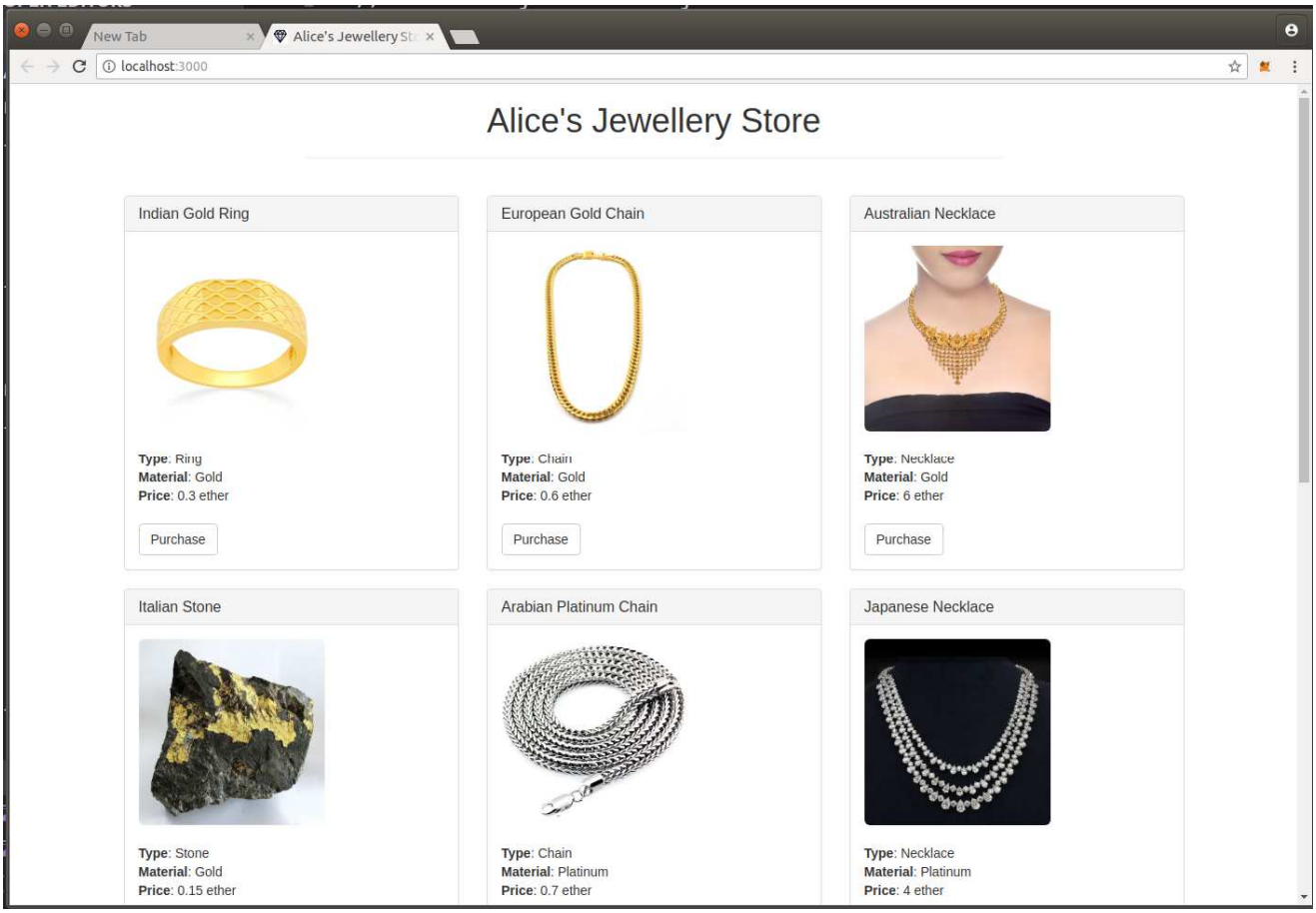
Since, the blockchain is the distributed technology with immutability shared ledger; it cannot be modified any transaction and so it is auditability. Ethereum dApp can be much efficiency way to handle Jewelleries selling. The buyers will have an Ethereum wallet address and it will use to purchase the jewelleries. The payment will be deposited to the dealer wallet. The transaction will be much faster and very low transaction fees to the buyers.

Alice's Jewellery Shop

For the simplicity, I named this distributed application (dApp) as Alice's Jewellery Shop. This is distributed web product to sell/buy jewelleries product. All transactions will be saved in the Ethereum network as database and for the audit propose.

For the proof of work, it uses json file to store the records of jewelleries which work as the back-end database. We can consider this json file as a table having the columns id, name, picture, type, material and price. For the simplicity I included only 12 records. We can add more records as well.

Alice's Jewellery Shop



As shown in the picture, the price is in ether. When user click a purchase button, the ether value also passed to the Metamask which makes it more interactive. Gas and gwei can be changed by user as needed for the speed of the transaction which will be paid by a user.

After sending the transaction, the ether will be deposited on the solidity contract address which is called "PurchaseJewellery".

For this environment, I am using javascript, truffle, ganache, solidity smart contract, bootstrap, html5.

Steps for the development

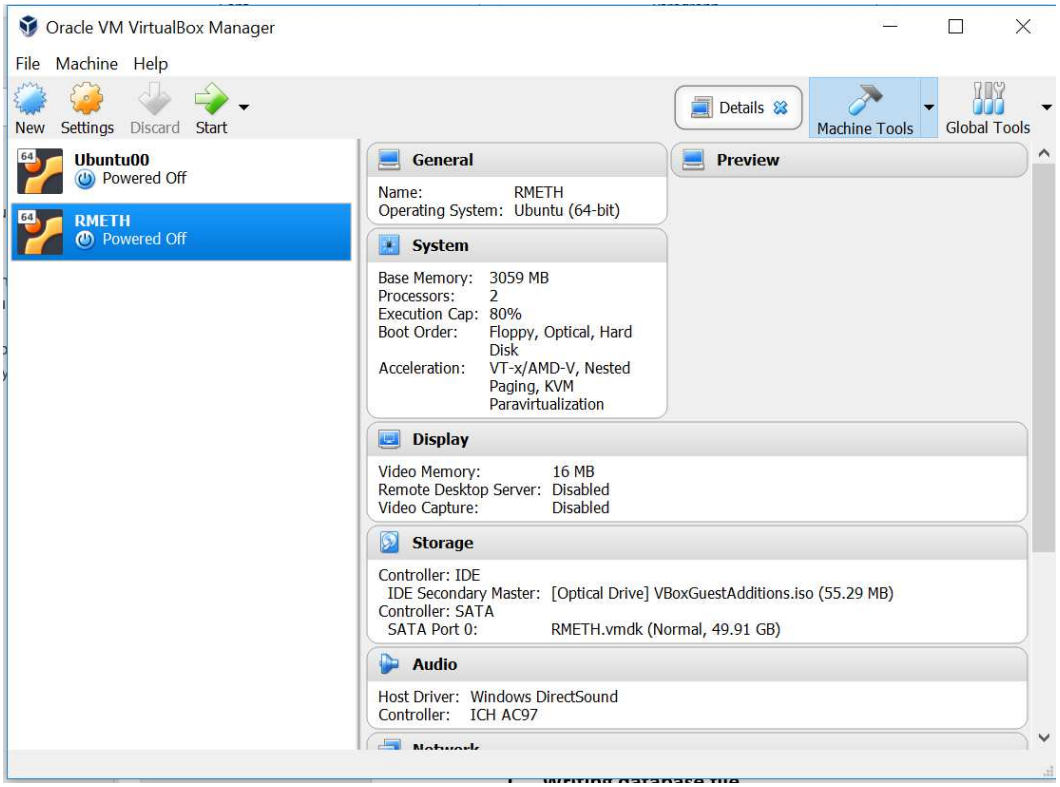
To build this dApp, we need to proceed many steps which I mentioned below.

A. Setting up the development environment
B. Creating a project using truffle
C. Copying standard folders and files templates
D. Creating a database
E. Writing smart contract in solidity language
F. Code Development
G. Setting up MetaMask and connecting Ganache CLI
H. Compiling and migrating the smart contract
I. Testing the smart contracts
J. Interacting with the dapp with browser

Now, I am going to describe in detail for each step

A. Setting up the development environment

1. I am using Ubuntu linux OS in Oracle VM. First, let's install Oracle VM and Ubuntu OS version 16.04 LTS.



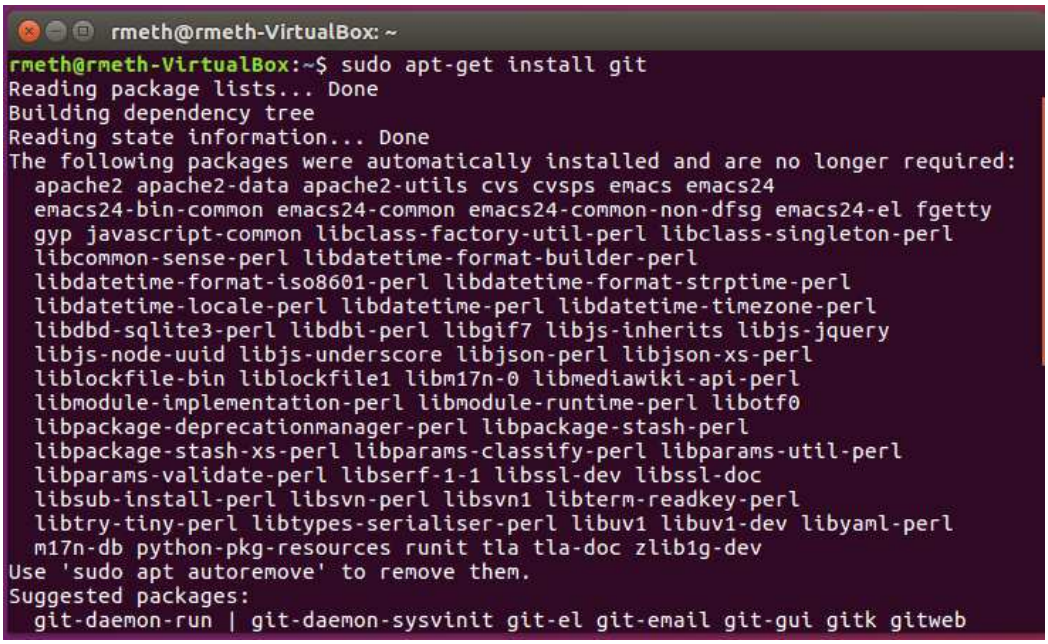
2. Installing Git:

   Before installing any command from ubuntu package manager, lets update and upgrade the package manager repository using below command.

   ```
   $ sudo apt-get update
   $ sudo apt-get upgrade
   ```

   Now, run below command to install git

   ```
   $ sudo apt-get intstall git
   ```



3. Install latest version of curl command

   The curl command will need to clone any GitHub data or setup files

   Run below commands to install latest curl version

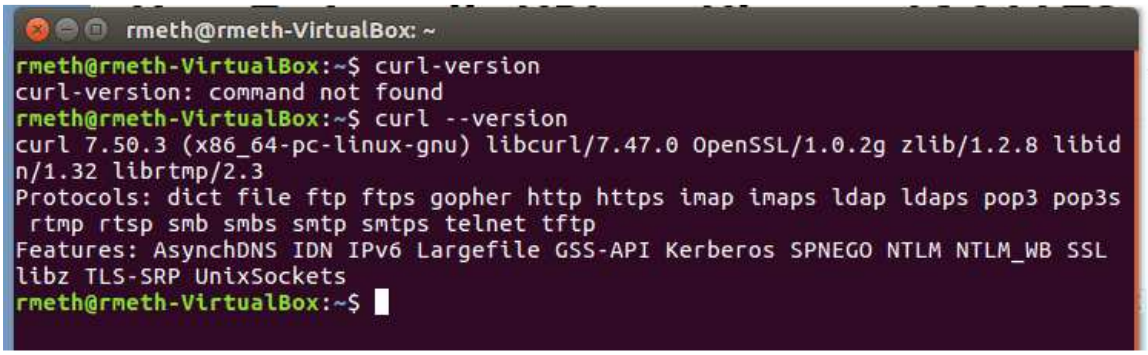
   ```
   $ wget http://curl.haxx.se/download/curl-7.50.3.tar.gz
   ```

   ```
   $ tar -xvf curl-7.50.3.tar.gz
   ```

```
$ cd curl-7.50.3/

$ ./configure

$ make

$ sudo make install
```

Once installed, we can verify curl using the command given below:

```
$ curl –version
```



We could simply use the apt-get command to install curl, but we might not get the latest version that we need.

$ sudo apt-get install curl or **$ sudo apt install curl**

4. Installing verion 8.x Node.Js/Node Package Manager

Node package can be downloaded from the below URL

https://nodejs.org/en/download/package-manager/

Before install nodejs, get the latest nodejs setup file

```
curl -sL https://deb.nodesource.com/setup_8.x | $ sudo -E
bash -
```

```
rmeth@rmeth-VirtualBox: ~
rmeth@rmeth-VirtualBox:~$ curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash -
[sudo] password for rmeth:

## Installing the NodeSource Node.js 8.x LTS Carbon repo...


## Populating apt-get cache...

+ apt-get update
Hit:1 http://us.archive.ubuntu.com/ubuntu xenial InRelease
Hit:2 http://security.ubuntu.com/ubuntu xenial-security InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu xenial-updates InRelease
Ign:4 http://dl.google.com/linux/chrome/deb stable InRelease
Hit:5 http://us.archive.ubuntu.com/ubuntu xenial-backports InRelease
Get:6 http://dl.google.com/linux/chrome/deb stable Release [1,189 B]
Hit:7 http://ppa.launchpad.net/ethereum/ethereum/ubuntu xenial InRelease
Get:8 http://dl.google.com/linux/chrome/deb stable Release.gpg [819 B]
Get:9 http://ppa.launchpad.net/git-core/ppa/ubuntu xenial InRelease [17.5 kB]
Get:10 http://dl.google.com/linux/chrome/deb stable/main amd64 Packages [1,354 B]
Get:11 http://ppa.launchpad.net/git-core/ppa/ubuntu xenial/main amd64 Packages [3,296 B]
Get:12 http://ppa.launchpad.net/git-core/ppa/ubuntu xenial/main i386 Packages [3,300 B]
Get:13 http://ppa.launchpad.net/git-core/ppa/ubuntu xenial/main Translation-en [2,604 B]
Fetched 30.1 kB in 1s (22.9 kB/s)
Reading package lists... Done
```

Now, install nodejs using command

```
$ sudo apt-get install -y nodejs

$ sudo apt-get install -y build-essential
```

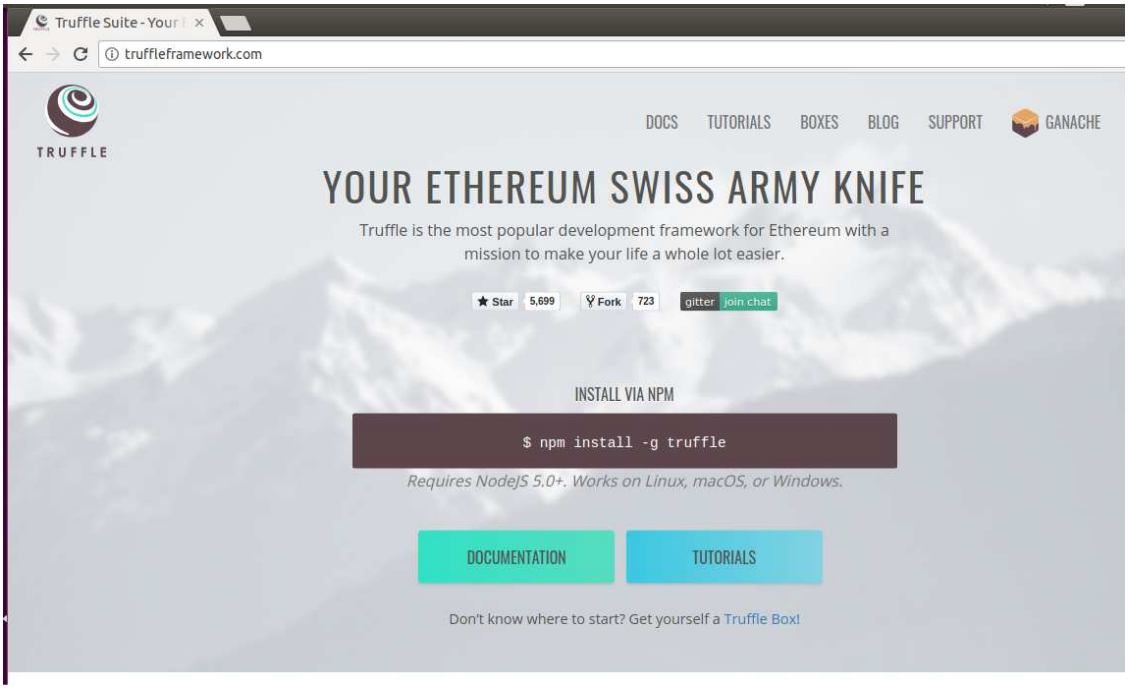
```
rmeth@rmeth-VirtualBox: ~

rmeth@rmeth-VirtualBox:~$ sudo apt-get install -y nodejs
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  apache2 apache2-data apache2-utils cvs cvsps emacs emacs24 emacs24-bin-common emacs24-common
  emacs24-common-non-dfsg emacs24-el fgetty gyp javascript-common libclass-factory-util-perl libclass-singleton-perl
  libcommon-sense-perl libdatetime-format-builder-perl libdatetime-format-iso8601-perl
  libdatetime-format-strptime-perl libdatetime-locale-perl libdatetime-perl libdatetime-timezone-perl
  libdbd-sqlite3-perl libdbi-perl libgif7 libjs-inherits libjs-jquery libjs-node-uuid libjs-underscore libjson-perl
  libjson-xs-perl liblockfile-bin liblockfile1 libm17n-0 libmediawiki-api-perl libmodule-implementation-perl
  libmodule-runtime-perl libotf0 libpackage-deprecationmanager-perl libpackage-stash-perl libpackage-stash-xs-perl
  libparams-classify-perl libparams-util-perl libparams-validate-perl libserf-1-1 libssl-dev libssl-doc
  libsub-install-perl libsvn-perl libsvn1 libterm-readkey-perl libtry-tiny-perl libtypes-serialiser-perl libuv1
  libuv1-dev libyaml-perl m17n-db python-pkg-resources runit tla tla-doc zlib1g-dev
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  nodejs
0 upgraded, 1 newly installed, 0 to remove and 3 not upgraded.
1 not fully installed or removed.
Need to get 12.6 MB of archives.
```

To check the version, we can run node -v or nodejs -v. The version should display 8.11.1

5. Installing Truffle Framework

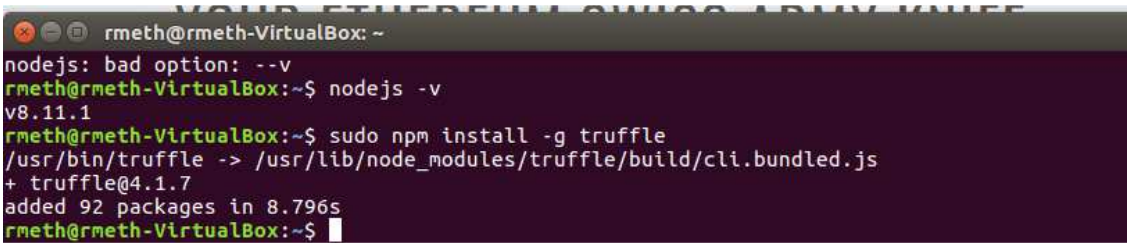
We can download and install truffle framework from the below link

http://truffleframework.com/

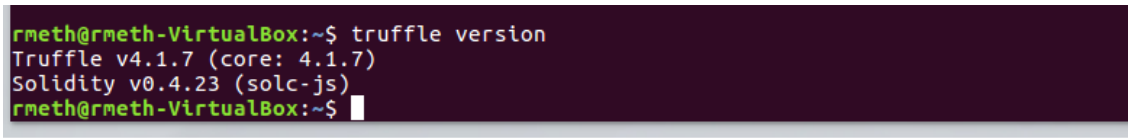


Run command

$ sudo npm install -g truffle

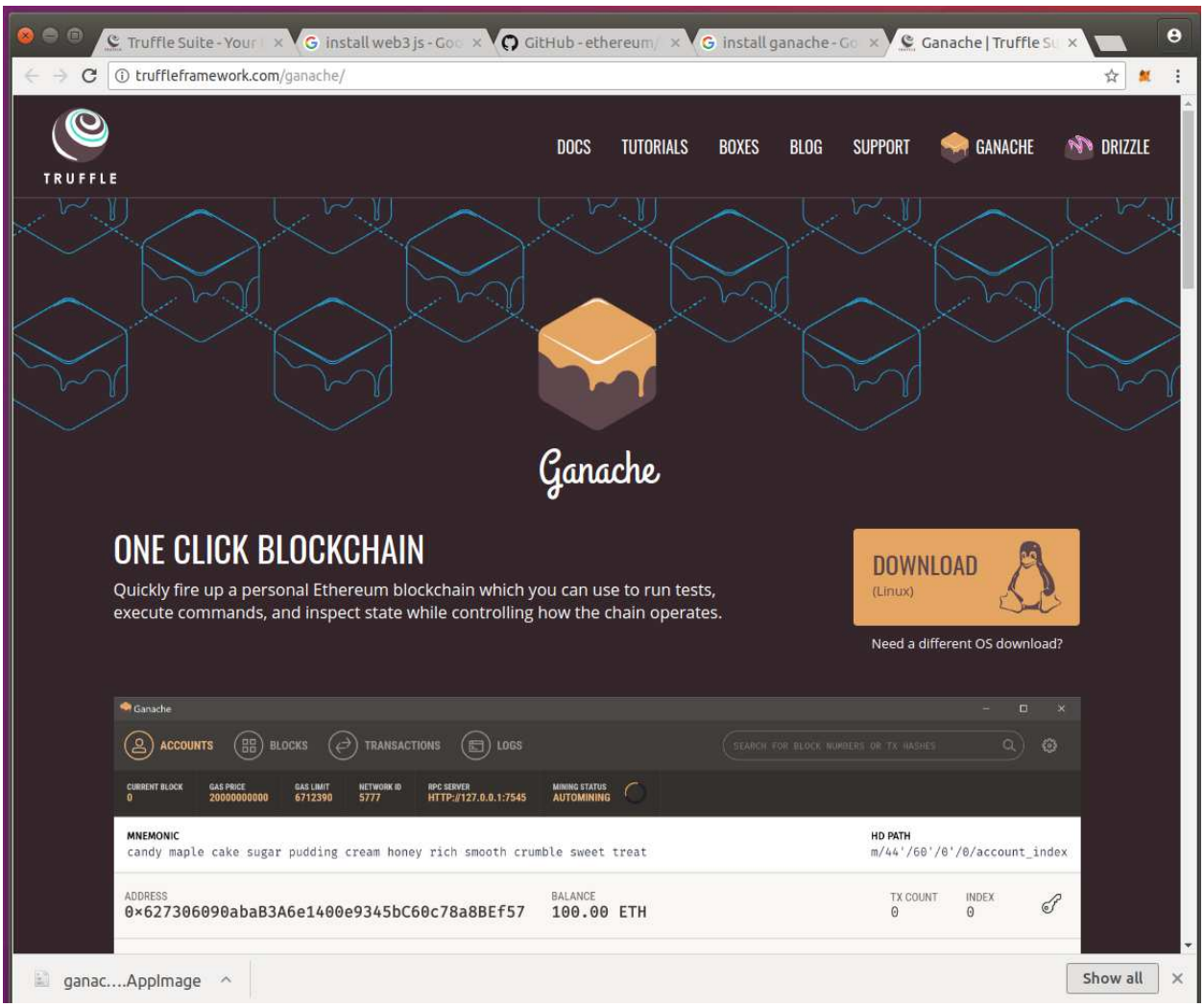


$ truffle version - display the version 4.1.7 and solidity 0.4.23



6. Setting up Ganache

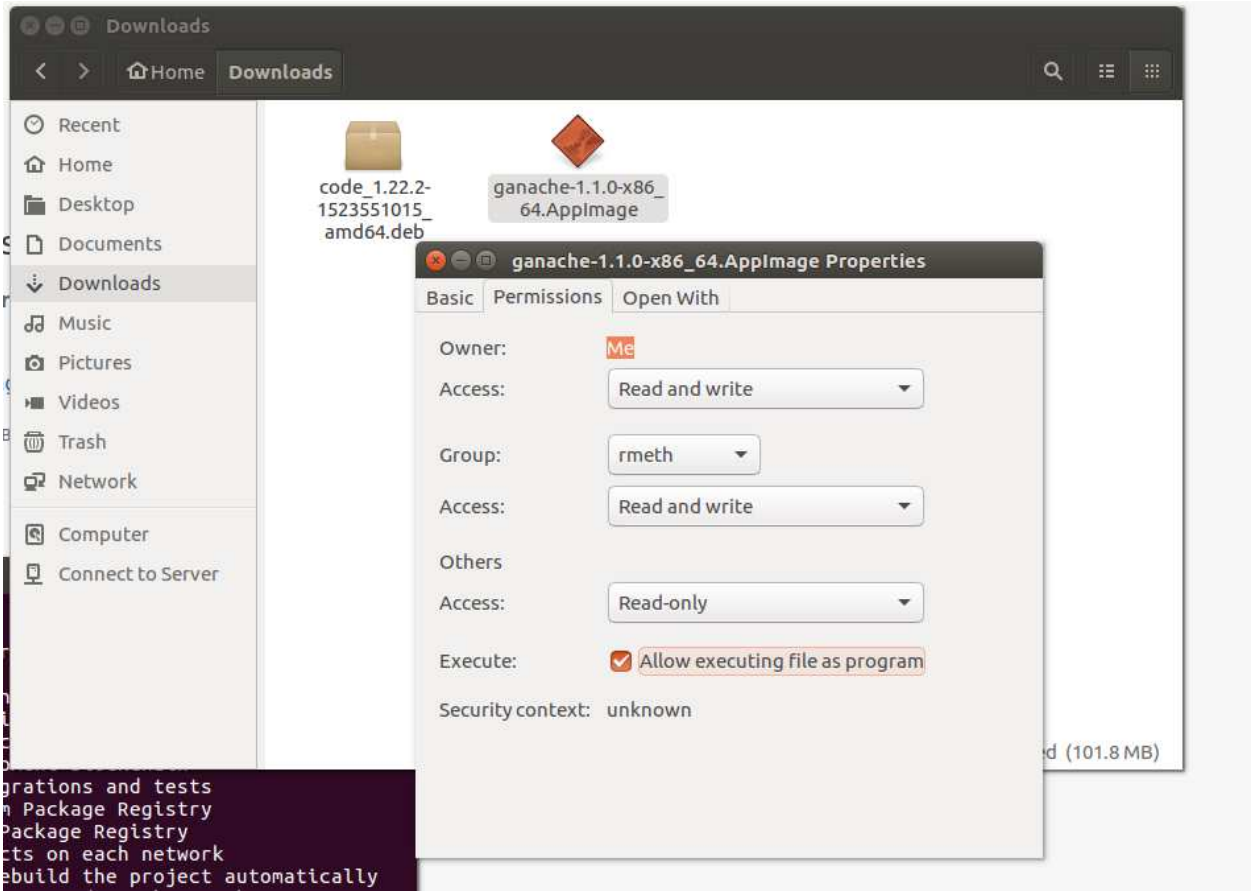
Go to truffleframework.com/ganache/ and click download (linux)



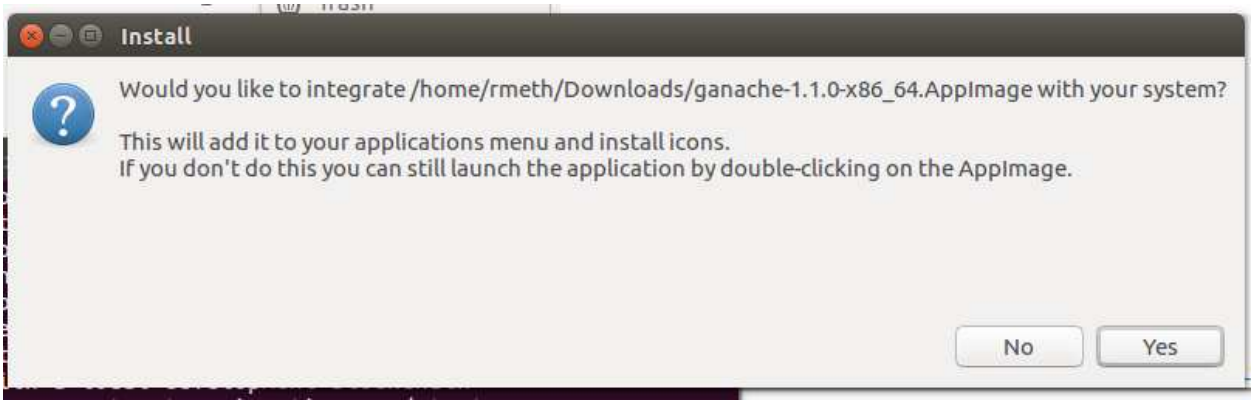
Go to download folder, you will find Ganache..*.AppImage file. This file doesn't simply install. First, we need to make this executable.

How to make executable?

Right click on the file > Properties > Permissions > Check Allow executing file as program



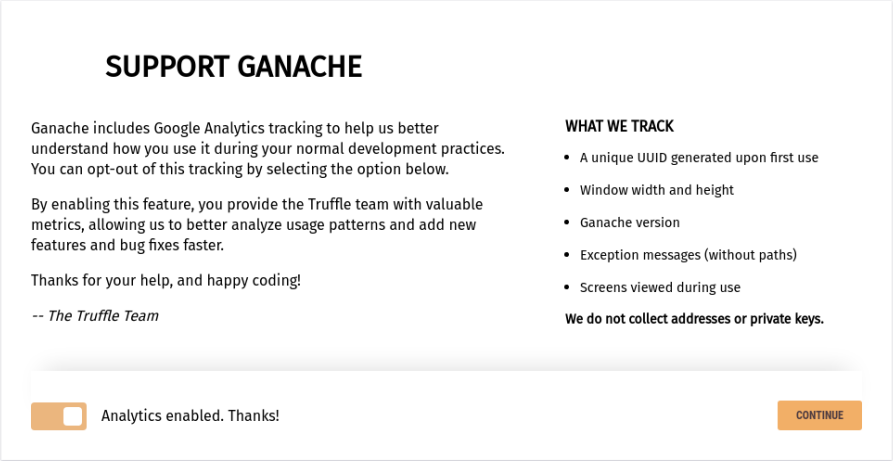
Now, double click on the ganache-*.AppImage file and select Yes on Install dialog box

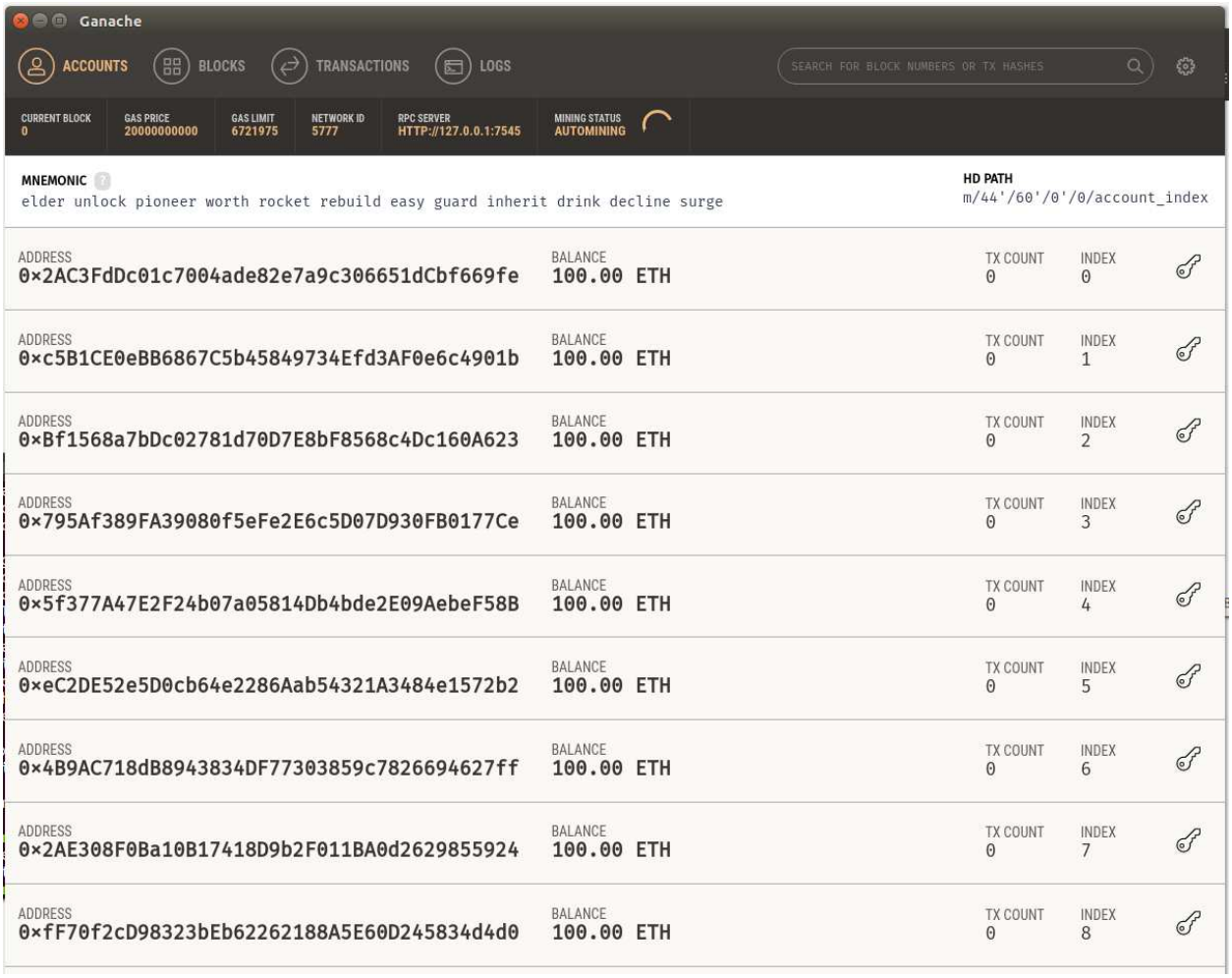


The installation will start now with following screen


Ganache
Ganache
v1.1.0


Ganache


SUPPORT GANACHE
Ganache includes Google Analytics tracking to help us better understand how you use it during your normal development practices. You can opt-out of this tracking by selecting the option below.
By enabling this feature, you provide the Truffle team with valuable metrics, allowing us to better analyze usage patterns and add new features and bug fixes faster.
Thanks for your help, and happy coding!
-- The Truffle Team
WHAT WE TRACK
A unique UUID generated upon first use
Window width and height
Ganache version
Exception messages (without paths)
Screens viewed during use
We do not collect addresses or private keys.
Analytics enabled. Thanks!
CONTINUE



By default, Ganache gives us 10 Ethereum accounts from index 0 to 9 with each of having 100 ETH balances for the testing purpose.

7. Installing Ganache-cli if we are not using Ganache UI

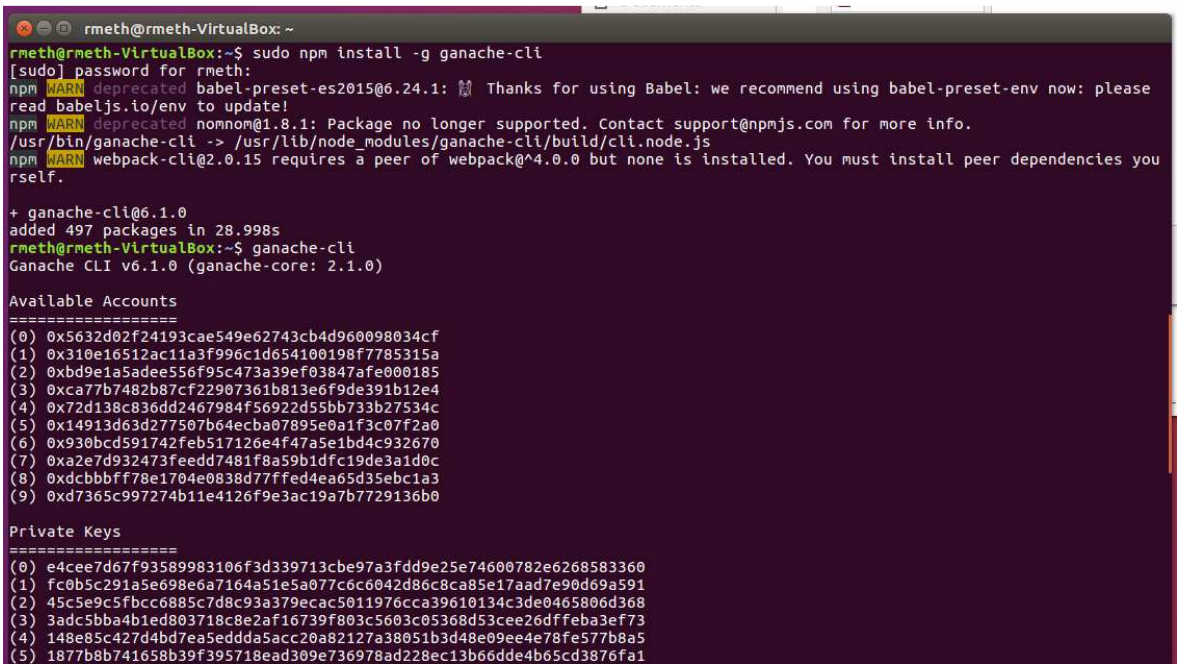
   We can install Ganache-cli instead of Ganache UI

   To install Ganache-cli

   $ npm install -g ganache-cli

   To run ganache

   $ ganache-cli



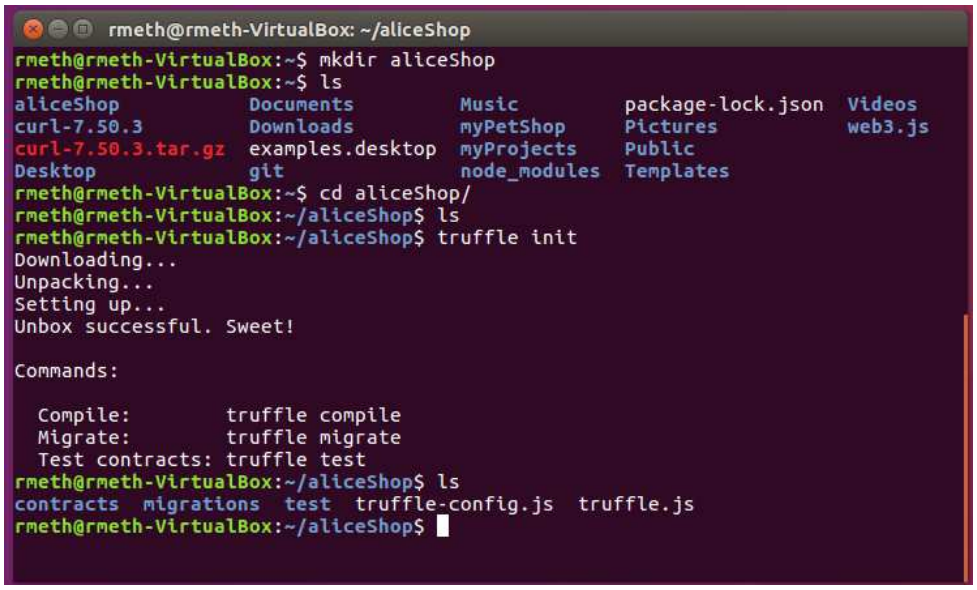
```
rmeth@rmeth-VirtualBox: ~
rmeth@rmeth-VirtualBox:~$ sudo npm install -g ganache-cli
[sudo] password for rmeth:
npm WARN deprecated babel-preset-es2015@6.24.1: 🙌  Thanks for using Babel: we recommend using babel-preset-env now: please
read babeljs.io/env to update!
npm WARN deprecated nomnom@1.8.1: Package no longer supported. Contact support@npmjs.com for more info.
/usr/bin/ganache-cli -> /usr/lib/node_modules/ganache-cli/build/cli.node.js
npm WARN webpack-cli@2.0.15 requires a peer of webpack@^4.0.0 but none is installed. You must install peer dependencies you
rself.

+ ganache-cli@6.1.0
added 497 packages in 28.998s
rmeth@rmeth-VirtualBox:~$ ganache-cli
Ganache CLI v6.1.0 (ganache-core: 2.1.0)

Available Accounts
==================
(0) 0x5632d02f24193cae549e62743cb4d960098034cf
(1) 0x310e16512ac11a3f996c1d654100198f7785315a
(2) 0xbd9e1a5adee556f95c473a39ef03847afe000185
(3) 0xca77b7482b87cf22907361b813e6f9de391b12e4
(4) 0x72d138c836dd2467984f56922d55bb733b27534c
(5) 0x14913d63d277507b64ecba07895e0a1f3c07f2a0
(6) 0x930bcd591742feb517126e4f47a5e1bd4c932670
(7) 0xa2e7d932473feedd7481f8a59b1dfc19de3a1d0c
(8) 0xdcbbbff78e1704e0838d77ffed4ea65d35ebc1a3
(9) 0xd7365c997274b11e4126f9e3ac19a7b7729136b0

Private Keys
==================
(0) e4cee7d67f93589983106f3d339713cbe97a3fdd9e25e74600782e6268583360
(1) fc0b5c291a5e698e6a7164a51e5a077c6c6042d86c8ca85e17aad7e90d69a591
(2) 45c5e9c5fbcc6885c7d8c93a379ecac5011976cca39610134c3de0465806d368
(3) 3adc5bba4b1ed803718c8e2af16739f803c5603c05368d53cee26dffeba3ef73
(4) 148e85c427d4bd7ea5eddda5acc20a82127a38051b3d48e09ee4e78fe577b8a5
(5) 1877b8b741658b39f395718ead309e736978ad228ec13b66dde4b65cd3876fa1
```

B. Creating a project using truffle

I am creating brand new project for this dApp and giving the folder name as aliceShop.

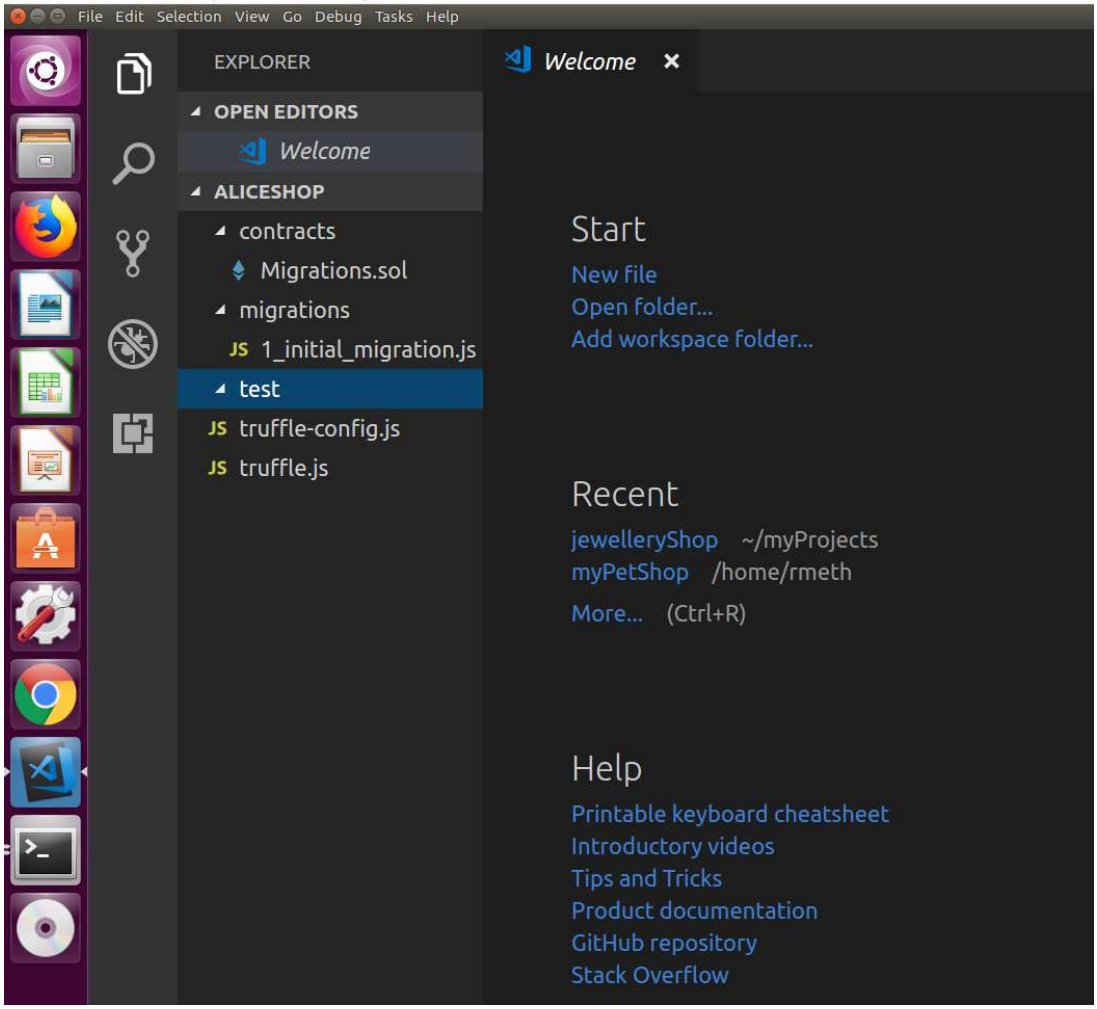
```
$ mkdir aliceShop
$ cd aliceShop
$ truffle init
```

Truffle init command creates an empty truffle project.



The empty truffle project will already have some files and folders

Below screen shot shows the folder structure that create by truffle empty project



- contracts/ : It contains the solidity source files for the smart contracts
- migrations/ : Truffle uses a migration system to handle smart contract deployment
- test/ : It contains the test files for both solidity and javascript tests
- truffle.js : Truffle configuration file to connect lite server

C. Copying standard folders and files templates

Since, there are already standard templates to develop dapp by using truffle framework, I am using those standard files and folder templates.

Copy node_modules or install from npm package to the root of the project.

Copy src folder including the sub folders like css, fonts, images, js and the file favicon.ico.

Css subfolder contains bootstrap.min.css and bootstrap.min.css.map

Copy all images into images folder

Js folder includes the standard js files like bootstrap.min.js, truffle-contract.js and web3.min.js

Copy file bs-config.json, package-lock.json and package.json

D. Creating a database

After analyzing the project and finalizing the data needs and considering the proof of work for this project, I am using json file which I have written under the src folder. It's called **jewellery.json**. I have also loaded all picture files under images folder.


```
[
 {
  "id": 0,
  "name": "Indian Gold Ring",
  "picture": "images/gold-ring-300.jpg",
  "type" : "Ring",
  "material": "Gold",
  "price": 0.30
 },
 {
  "id": 1,
  "name": "European Gold Chain",
  "picture": "images/gold-chain-400.jpg",
  "type" : "Chain",
  "material": "Gold",
  "price": 0.60
 },
 {
  "id": 2,
  "name": "Australian Necklace",
  "picture": "images/gold-necklace-5000.jpg",
  "type" : "Necklace",
  "material": "Gold",
  "price": 6.00
 },
 {
  "id": 3,
  "name": "Italian Stone",
  "picture": "images/gold-stone-200.jpg",
  "type" : "Stone",
  "material": "Gold",
  "price": 0.15
 },
```

```
{
  "id": 4,
  "name": "Arabian Platinum Chain",
  "picture": "images/platinum-chain-600.jpg",
  "type": "Chain",
  "material": "Platinum",
  "price": 0.70
},
{
  "id": 5,
  "name": "Japanese Necklace",
  "picture": "images/platinum-necklace-2600.jpg",
  "type": "Necklace",
  "material": "Platinum",
  "price": 4.00
},
{
  "id": 6,
  "name": "Chinese White Ring",
  "picture": "images/platinum-ring-450.jpg",
  "type": "Ring",
  "material": "Platinum",
  "price": 0.50
},
{
  "id": 7,
  "name": "African Stone",
  "picture": "images/platinum-stone-300.jpg",
  "type": "Stone",
  "material": "Platinum",
  "price": 0.40
},
{
  "id": 8,
  "name": "SilverSpring Chain",
  "picture": "images/silver-chain-200.jpg",
  "type": "Chain",
  "material": "Silver",
  "price": 0.30
```

```
  },
  {
    "id": 9,
    "name": "Scotish Necklace",
    "picture": "images/silver-necklace-2000.jpg",
    "type": "Necklace",
    "material": "Silver",
    "price": 0.24
  },
  {
    "id": 10,
    "name": "Russian Ring",
    "picture": "images/silver-ring-300.jpg",
    "type": "Ring",
    "material": "Silver",
    "price": 0.74
  },
  {
    "id": 11,
    "name": "Jamaican Stone",
    "picture": "images/silver-stone-400.jpg",
    "type": "Stone",
    "material": "Silver",
    "price": 0.43
  }
]
```

E. Writing smart contract in solidity language

In a contract folder, there is already Migrations.sol file from the truffle framework template. Let's change the solidity version to latest version for this file. I am using pragma solidity ^0.4.23; version which is the latest version while writing this project.

Create another solidity contract file for the jewellery purchase activity. I have named it PurchaseJewellery.sol and created under contracts folder with following code.

PurchaseJewellery.sol

```
pragma solidity ^0.4.23;

contract PurchaseJewellery {
   address public owner;
   address[12] public buyers;

   modifier restricted() {
      if (msg.sender == owner) _;
   }

   constructor() public {
      owner = msg.sender;
   }

   function buyItem(uint itemId) public payable returns(uint) {
      require(itemId >= 0 && itemId < 12);     //Only we have 12 items for sell
      buyers[itemId] = msg.sender;
      return itemId;
   }

   function getAllBuyers() public view returns (address[12]) {
      return buyers;
   }
}
```

F. Code Development

migration\1_initial_migration.js

This page has been taken from the template and we don't change anything.

```
var Migrations = artifacts.require("./Migrations.sol");

module.exports = function(deployer) {
  deployer.deploy(Migrations);
};
```

migration\2_initial_migration.js

This page I needed to create to migrate the PurchaseJewellery solidity file

```
var Purchase = artifacts.require("./PurchaseJewellery");

module.exports =function(deployer) {
   deployer.deploy(Purchase);
}
```

src\index.html

This is the main front end html page which interacts with JS files. I am using bootstrap for the style of the page.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- The above 3 meta tags *must* come first in the head; any other head content must
come *after* these tags -->
    <title>Alice's Jewellery Store</title>
```

```
    <!-- Bootstrap -->
    <link href="css/bootstrap.min.css" rel="stylesheet">

    <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media queries --
>
    <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
    <!--[if lt IE 9]>
      <script src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>
      <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
    <![endif]-->
  </head>
  <body>
    <div class="container">
      <div class="row">
        <div class="col-xs-12 col-sm-8 col-sm-push-2">
          <h1 class="text-center">Alice's Jewellery Store</h1>
          <hr/>
          <br/>
        </div>
      </div>

      <div id="itemsRow" class="row">
        <!-- Jewellery Records -->
      </div>
    </div>

    <div id="itemsTemplate" style="display: none;">
      <div class="col-sm-12 col-md-8 col-lg-4">
        <div class="panel panel-default panel-item">
          <div class="panel-heading">
            <h3 class="panel-title">Jewellery</h3>
          </div>
          <div class="panel-body">
            <!--<img alt="200x200" data-src="holder.js/200x200" class="img-rounded img-
center" style="width: 100%;" src="" data-holder-rendered="true">-->
            <img alt="200x200" width="200" height="200" class="img-rounded img-center"
src="">
            <br/><br/>
            <strong>Type</strong>: <span class="item-type"></span><br/>
```

```
          <strong>Material</strong>: <span class="item-material"></span><br/>
          <strong>Price</strong>: <span class="item-price"></span> ether<br/><br/>
          <button class="btn btn-default btn-purchase" type="button" data-
id="0">Purchase</button>
        </div>
      </div>
    </div>
  </div>

  <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
  <!-- Include all compiled plugins (below), or include individual files as needed -->
  <script src="js/bootstrap.min.js"></script>
  <script src="js/web3.min.js"></script>
  <script src="js/truffle-contract.js"></script>
  <script src="js/app.js"></script>
 </body>
</html>
```

src\js\app.js

app.js is the main java script page which interact with database and the user front-page.

```
//Author: Rajendra Maharjan
App = {
 web3Provider: null,
 contracts: {},

 init: function() {
  // Calling Json file for all items to bind
  $.getJSON('../jewellery.json', function(data) {
    var itemsRow = $('#itemsRow');
    var itemTemplate = $('#itemsTemplate');

    for (i = 0; i < data.length; i ++) {
      itemTemplate.find('.panel-title').text(data[i].name);
      itemTemplate.find('img').attr('src', data[i].picture);
      itemTemplate.find('.item-type').text(data[i].type);
```

```
      itemTemplate.find('.item-price').text(data[i].price);
      itemTemplate.find('.item-material').text(data[i].material);
      itemTemplate.find('.btn-purchase').attr('data-id', data[i].id);

      itemsRow.append(itemTemplate.html());
    }
  });

  return App.initWeb3();
},

initWeb3: function() {
  //Connecting to Network provider
  if(typeof web3 !== 'undefined') {
    App.web3Provider = web3.currentProvider;
  } else {
    App.web3Provider = new Web3.providers.HttpProvider('http://localhost:7545');
  }

  web3 = new Web3(App.web3Provider);

  return App.initContract();
},

initContract: function() {
  $.getJSON('PurchaseJewellery.json', function(data){
    var purchaseArtifact = data;
    App.contracts.PurchaseJewellery = TruffleContract(purchaseArtifact);
    //Set the provider for this contract
    App.contracts.PurchaseJewellery.setProvider(App.web3Provider);

    return App.markAsPurchased();
  });
  return App.bindEvents();
},

bindEvents: function() {
  $(document).on('click', '.btn-purchase', App.handlePurchase);
},
```

```
  markAsPurchased: function(buyers, account) {
    var purchaseInstance;

    App.contracts.PurchaseJewellery.deployed().then(function(instance) {
      purchaseInstance = instance;
//getAllBuyers return as view, use call method
      return purchaseInstance.getAllBuyers.call();
    }).then(function(buyers){
      for(i=0;i<buyers.length;i++){
        if(buyers[i] !== '0x0000000000000000000000000000000000000000') {
          $('.panel-item').eq(i).find('button').text('Success').attr('disabled',true);
        }
      }
    }).catch(function(err){
      console.log(err.message);
    });
  },

  handlePurchase: function(event) {
    event.preventDefault();

    var itemId = parseInt($(event.target).data('id'));
    var itemPrice;
    //Implementing Price
    //Get Item Price from json file
    $.getJSON('../jewellery.json', function(data) {
      itemPrice=data[itemId].price;
      //console.log("Price from json: " + data[itemId].price);
    //});

    var purchaseInstance;
    web3.eth.getAccounts(function(error, accounts) {
      if(error) {
        console.log(error);
      }

      var account = accounts[0];
```

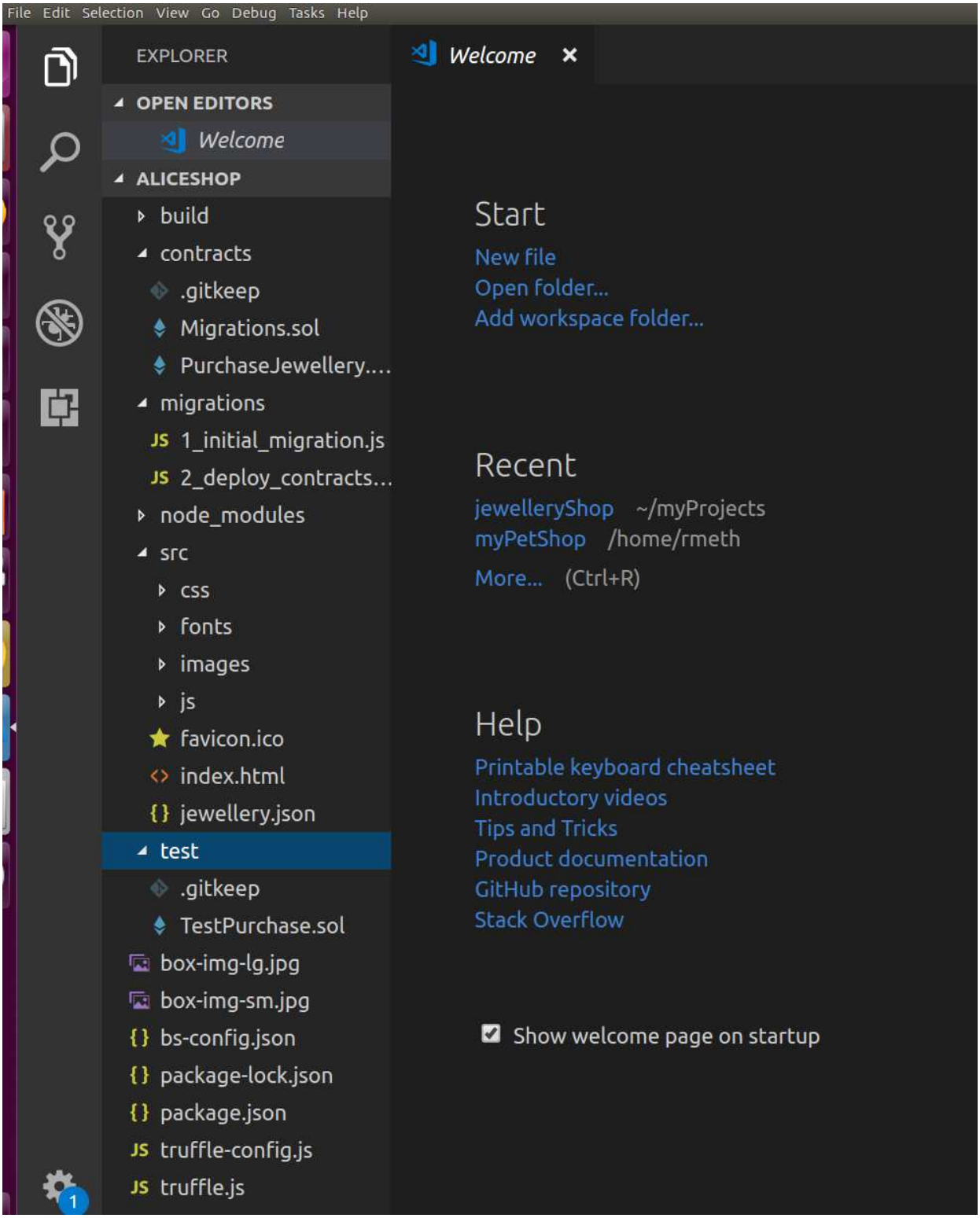
```
      App.contracts.PurchaseJewellery.deployed().then(function(instance) {
        purchaseInstance = instance;
        //Execute Purchase as a  transaction by sending account
        //return purchaseInstance.buyItem(itemId, {from: account, to:
'0x79bba8e1299CF25C2d71005bc73F65519c551dA7', value: web3.toWei(itemPriceInEther,
'ether'), gas: 21000});
        return purchaseInstance.buyItem(itemId, {from: account, value: web3.toWei(itemPrice,
'ether'), gas: 100000});
      }).then(function(result){
        return App.markAsPurchased();
      }).catch(function(err){
        console.log(err.message);
      });
    });
  });
  }
};

$(function() {
  $(window).load(function() {
    App.init();
  });
});
```

truffle.js

I am using ganache.cli and its port is 8545 to be connected from metamask for local network ganache-cli. If we are going to use ganache UI, we need to use 7545 ports.

```
module.exports = {
  // See <http://truffleframework.com/docs/advanced/configuration>
  // for more about customizing your Truffle configuration!
  networks: {
    development: {
      host: "127.0.0.1",
      port: 8545,
      network_id: "*" // Match any network id
    }
  }
};
```

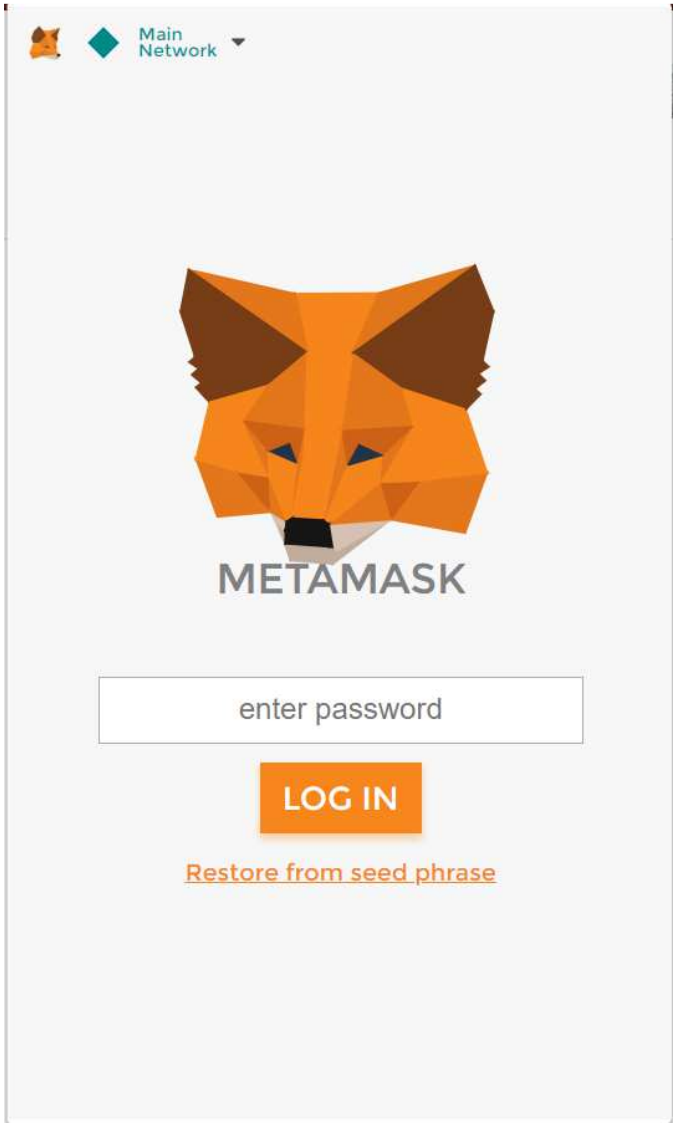
Project files structure



G. Setting up MetaMask and connecting Ganache CLI

MetaMask is a bridge that allows to visit the distributed web from the browser. It allows us to run Ethereum dApps right in our internet browser without running a full Ethereum node. I am using MetaMask to connect to the ganache-cli as the local network and to do the payment transactions.

We can install this MetaMask as the Chrome add- on or Firefox add-on. I am using chrome for this project.

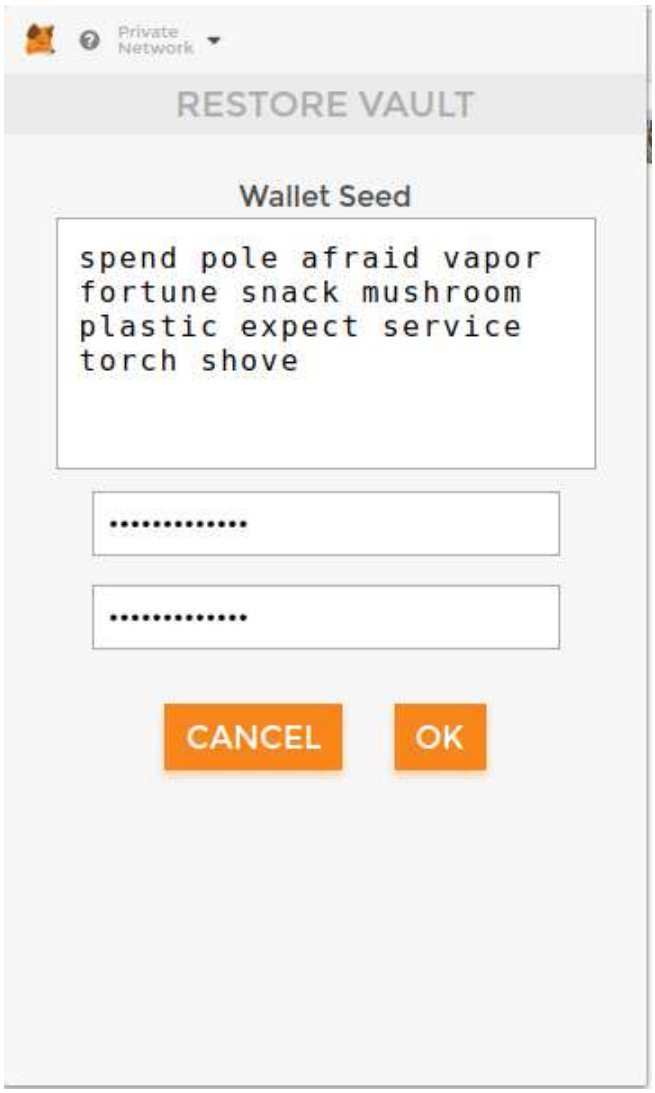


Before connecting to local host 8545, we need to start the ganache-cli on the terminal as shown below screen shot.

```
rmeth@rmeth-VirtualBox: ~
rmeth@rmeth-VirtualBox:~$ ganache-cli
Ganache CLI v6.1.0 (ganache-core: 2.1.0)

Available Accounts
==================
(0) 0x4dfc803345c929c71829b53f7b34a8dd847efa4b
(1) 0x102c2964cf9aec63165c8b87961649cb9d78449a
(2) 0x045a68bea5a167034154fcd1cb8a666b2015f70a
(3) 0x7a2465f6dd6a189d5377be68e2944889bb380c63
(4) 0xc98b987f6ead22532e891d61dfeecb3a90d92803
(5) 0xf8621fafe7cb597660857237cfd58062a22ab293
(6) 0x52e0919c181be4c8d89845659e4c0ca6dcd3ff6d
(7) 0x8a2af0b4502ccb406f6a720cb21d6aa42f64bb58
(8) 0x0bdfa9f04ddbfbd7a9f2b8e9ded5e802cdf71204
(9) 0x04fdce704d4f9b0ee9010308b706224baf431fa2

Private Keys
==================
(0) 26a80f32eee633120d24a4b08ad29426ec2b20b862ddd5e1b36b8e430d5d361f
(1) 9c9507b895755d3954559886cb774513309561eea2c29045c99f1b2a731198db
(2) 6affe88cce8b702300c55c0463e0426ea5c8cf09c16caf11e419246a471ac1f1
(3) f164ee5e18500c27dfdd290dccb2c493bd8c25e1f1e21f031e5fae0324aa3cc3
(4) dd354a84c77e7402829d760f1dc18236b2eda3bba51ff545a3ad06ba16873686
(5) 0a6cc1eda656145a6a78cc100d314f6ab6b087de6e6bcac2e34b17e932109d74
(6) 5148a19eeb8aee4be795d8dc68a2fd090b3f00d9a8bce47f4ef14b4757a890fa
(7) e35af37f5273ad95426ecb484eb5c1e31ebf6d5763b330426a02f8bbdba2c6b1
(8) 272be84555add0a7d8e7709bb3bc3d91fc11ac73e2d9aee8910c0863d262b93d
(9) 3ab5d96874b854801a06c5d0c8ed5ca2764750361c501f4bf15b3706e429cca8

HD Wallet
==================
Mnemonic:      spend pole afraid vapor fortune snack mushroom plastic
 expect service torch shove
Base HD Path:  m/44'/60'/0'/0/{account_index}

Listening on localhost:8545
```

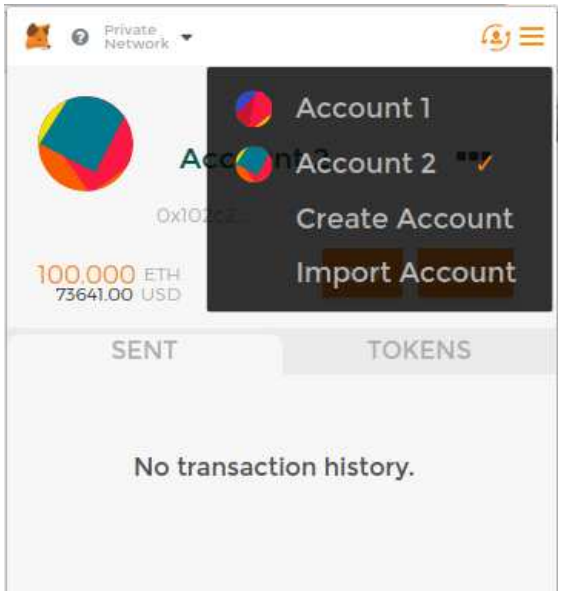
Copy all this public, private and mnemonic into a text file and save it because we need these public and private keys to test the project using Metamask.

Now, open Metamask and click on Restore from Seed Phrase. I want to use the same addresses those we created in ganache cli. Copy the mnemonic that we copied from terminal and paste in Wallet Seed text box. Enter password and confirm this.



Hit OK after we done.

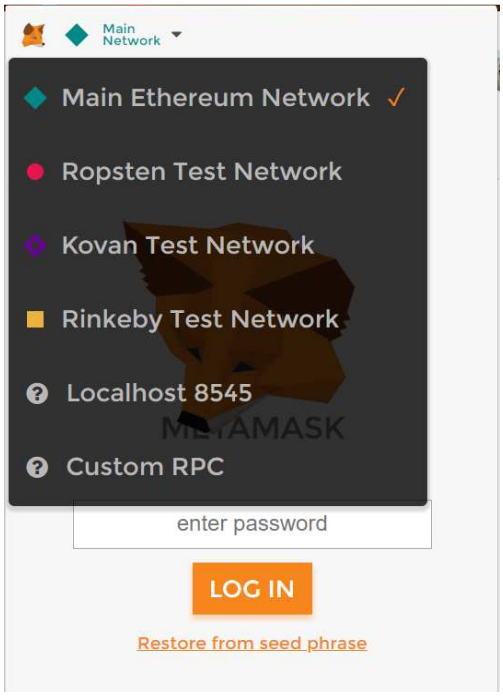
By default, now it will connect to ganache cli and gets the first address wallet from ganache. For the testing purpose, by default it gives 100 ethers per address. We can add others address just by clicking create account as shown below screen shot.



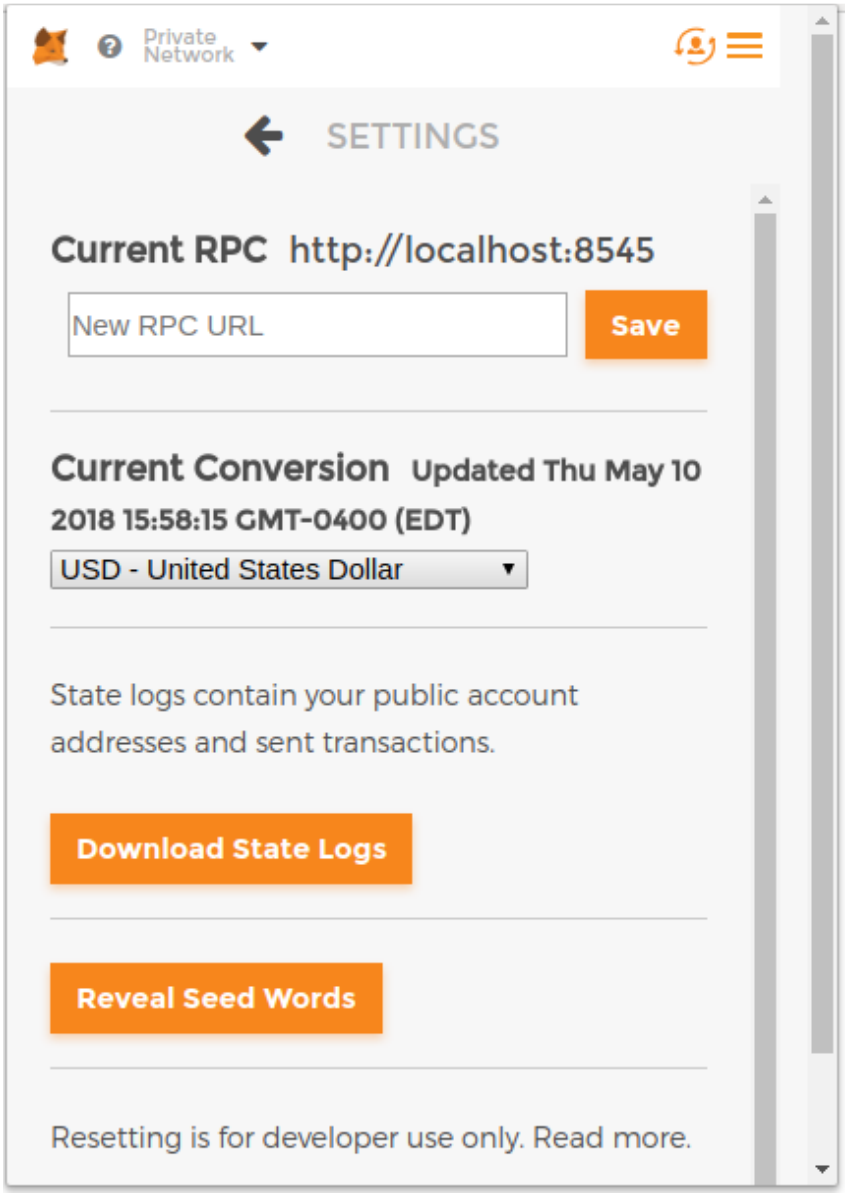
If we disconnected Metamask from ganache and we want to re-use same addresses wallet, we need to start ganache cli using the same old mnemonics as shown in below command

```
$ ganache-cli -m "mnemonic"
```

And then connect ganache from Metamask by clicking localhost 8575 as shown in below screen shot.



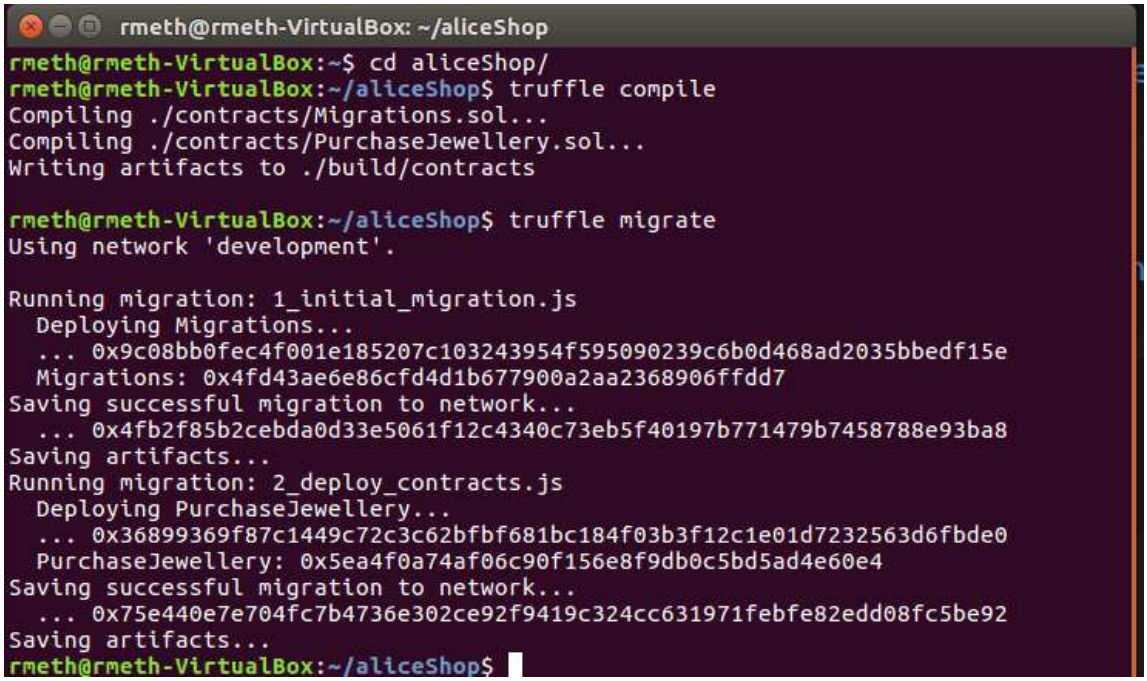
We will see the RPC settings of MetaMask is 8545. If we are using Ganache UI, we need to change to 7545 ports.



H. Compiling and migrating the smart contract

Let's start another terminal and go to the project folder, and then compile the code. If there is no error, let's migrate using following commands.

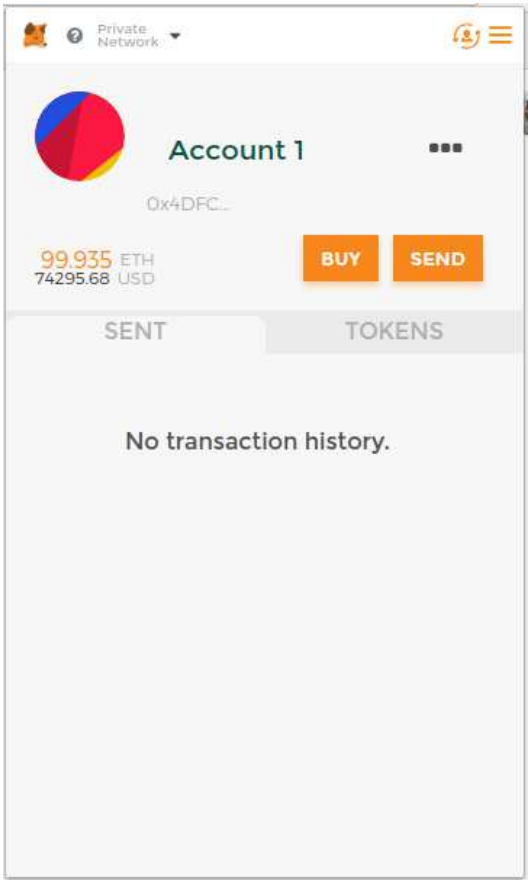
```
$ cd aliceShop
$ truffle compile
$ truffle migrate
```



The truffle migrate command deploys the smart contracts. Since we have two solidity files/contracts, it deploys two contracts and give us two contract addresses. Above screen is in the compiler screen. Let's look also in the ganache-cli terminal running screen which I have attached below.

```
rmeth@rmeth-VirtualBox: ~
eth_getBlockByNumber
eth_getBlockByNumber
eth_getBlockByNumber
net_version
eth_accounts
eth_accounts
net_version
net_version
eth_sendTransaction
eth_getBlockByNumber

  Transaction: 0x9c08bb0fec4f001e185207c103243954f595090239c6b0d468ad2035bbedf15e
  Contract created: 0x4fd43ae6e86cfd4d1b677900a2aa2368906ffdd7
  Gas usage: 277462
  Block Number: 1
  Block Time: Thu May 10 2018 14:47:12 GMT-0400 (EDT)

eth_newBlockFilter
eth_getFilterChanges
eth_getTransactionReceipt
eth_getCode
eth_uninstallFilter
eth_sendTransaction

  Transaction: 0x4fb2f85b2cebda0d33e5061f12c4340c73eb5f40197b771479b7458788e93ba8
  Gas usage: 42008
  Block Number: 2
  Block Time: Thu May 10 2018 14:47:13 GMT-0400 (EDT)

eth_getTransactionReceipt
eth_getBlockByNumber
eth_accounts
net_version
net_version
eth_sendTransaction

  Transaction: 0x36899369f87c1449c72c3c62bfbf681bc184f03b3f12c1e01d7232563d6fbde0
  Contract created: 0x5ea4f0a74af06c90f156e8f9db0c5bd5ad4e60e4
  Gas usage: 303508
  Block Number: 3
  Block Time: Thu May 10 2018 14:47:13 GMT-0400 (EDT)

eth_newBlockFilter
eth_getFilterChanges
eth_getTransactionReceipt
eth_getCode
eth_getBlockByNumber
eth_uninstallFilter
eth_sendTransaction

  Transaction: 0x75e440e7e704fc7b4736e302ce92f9419c324cc631971febfe82edd08fc5be92
  Gas usage: 27008
  Block Number: 4
  Block Time: Thu May 10 2018 14:47:13 GMT-0400 (EDT)

eth_getTransactionReceipt
eth_getBlockByNumber
eth_getBlockByNumber
eth_getBlockByNumber
eth_getBlockByNumber
eth_getBlockByNumber
eth_getBlockByNumber
eth_getBalance
eth_getBalance
```

Also, look into the Metamask. We will see, it has used first address to deploy address and it has spent some gas during the contracts deployment. Here, earlier we had 100 ethers but now reduced to 99.935 ether.



I. Testing the smart contracts

To test the smart contracts, I created three different solidity code functions. One is testUserCanPurchase() which make sure whether user can do purchase action or not. Second one is testGetBuyerByItemId() that returns wallet address of buyer and make sure whether it is equal to the same item buyer or not. Third one is testGetBuyerAddressByItemIdInArray() method which make sure that it returns all buyers address or not.

Same way from the solidity code, we can also use java script file for the testing our project as well. But for the proof of work, I am using only from solidity code.

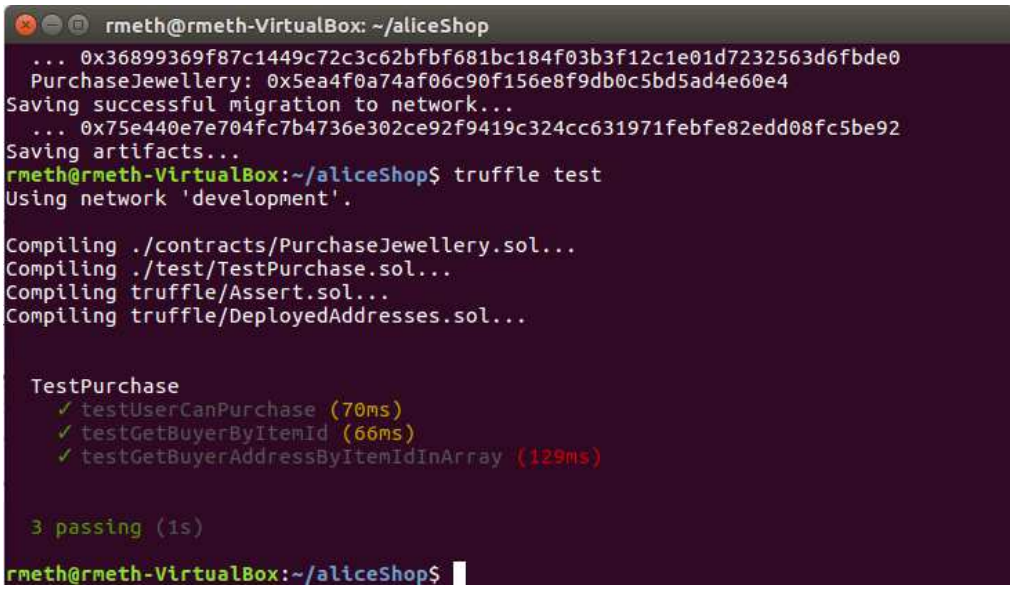
test\TestPurcahse.sol

```
//Author: Rajendra Maharjan
//1. Testing for User can purchase or not
//2. Testing whether it records buyer address or not
//3. Testing whether all buyers return or not.

pragma solidity ^0.4.23;

import "truffle/Assert.sol";
import "truffle/DeployedAddresses.sol";
import "../contracts/PurchaseJewellery.sol";

contract TestPurchase {
  PurchaseJewellery objPurchase =
PurchaseJewellery(DeployedAddresses.PurchaseJewellery());

  function testUserCanPurchase() public {
    uint returnId = objPurchase.buyItem(2);
    uint expected = 2;
    Assert.equal(returnId, expected, "Purchase of item 2nd should be recorded");
  }

  function testGetBuyerByItemId() public {
    address expected = this;
    address buyer = objPurchase.buyers(2);
    Assert.equal(buyer, expected, "Owner of 2nd item should be recorded");
```

```
    }

    function testGetBuyerAddressByItemIdInArray() public {
      address expected = this;

      address[12] memory buyers = objPurchase.getAllBuyers();

      Assert.equal(buyers[2], expected, "Owner of 2nd Item should be recorded");
    }
  }
```

For testing, run truffle test command from the project terminal

$ truffle test



J. Interacting with dApp browser

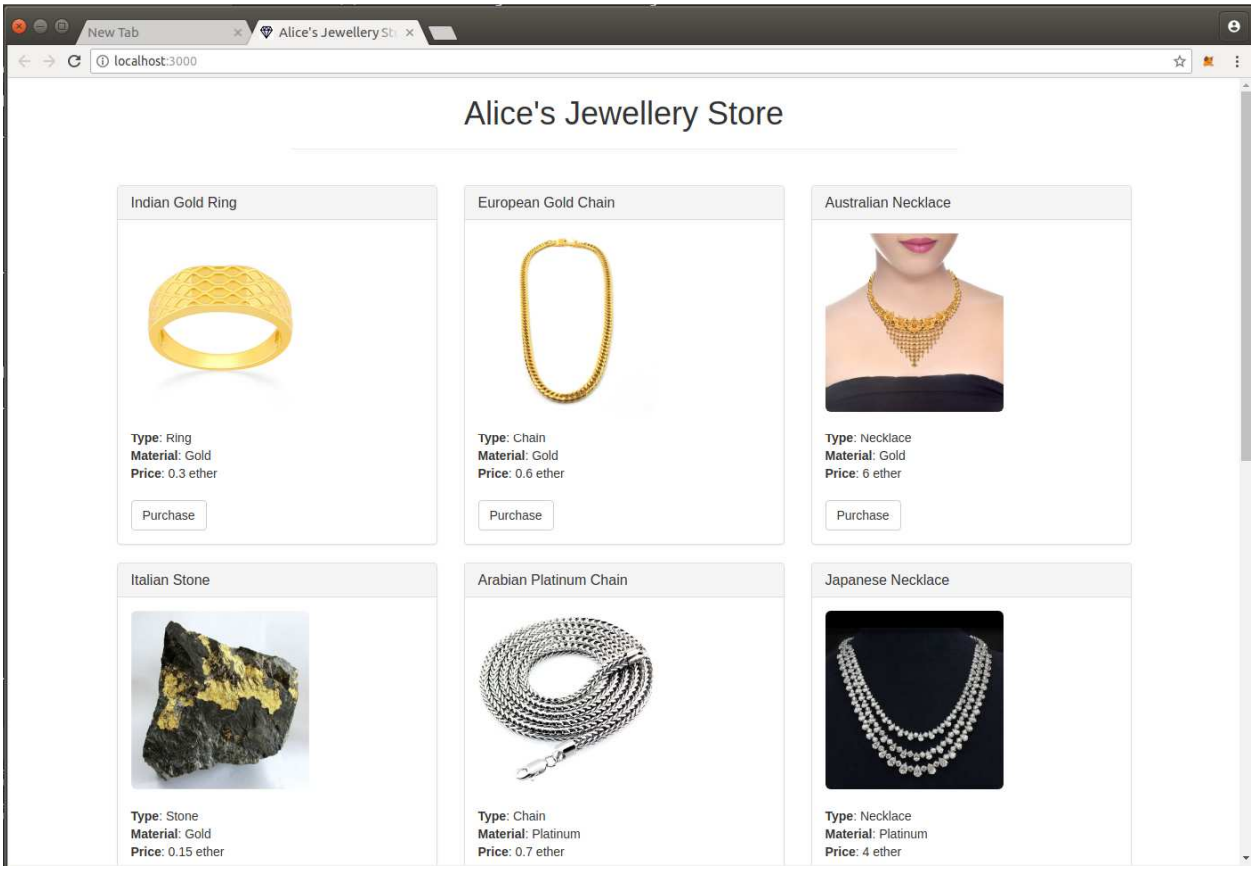
We have finished coding, compiling, migration and testing. Now, lets run the application using command npm run dev.

```
$ npm run dev
```

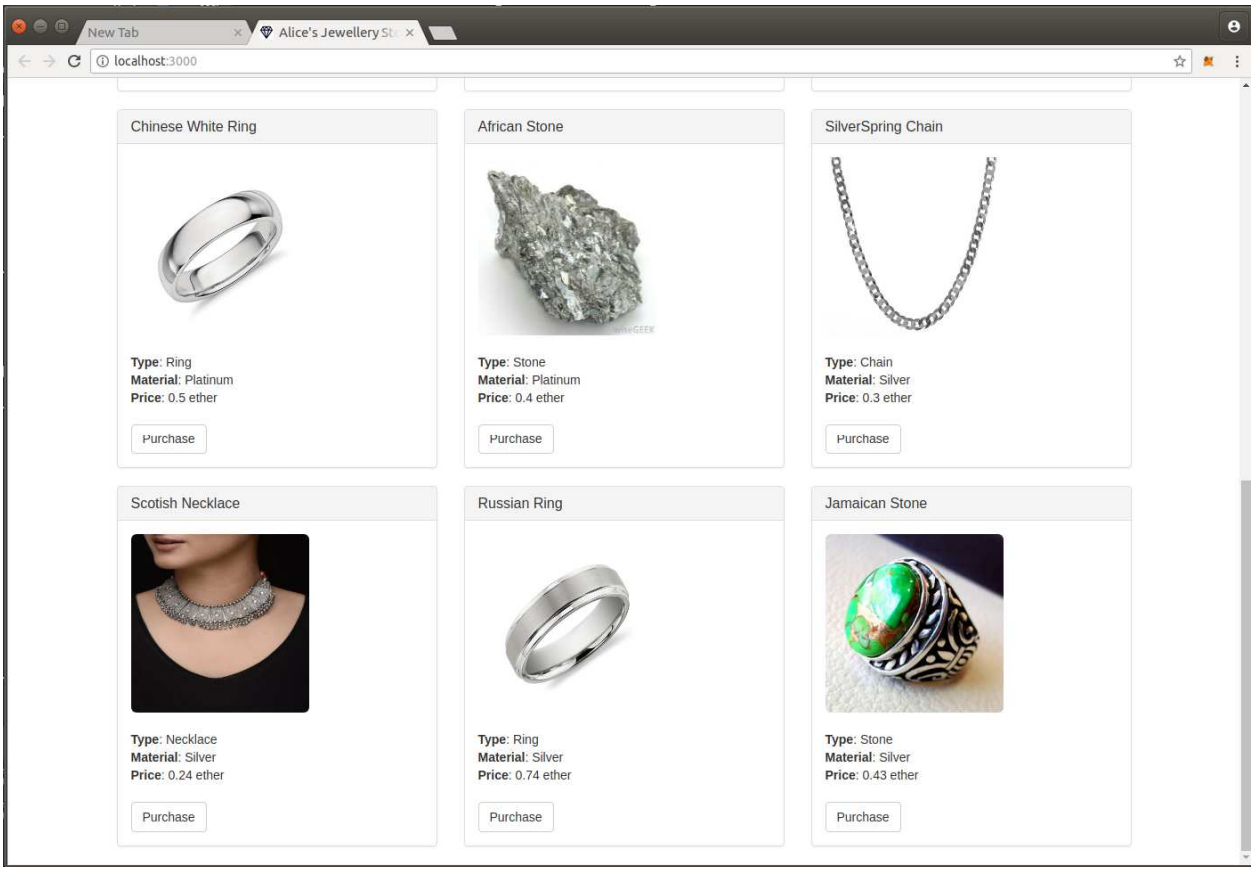
```
rmeth@rmeth-VirtualBox: ~/aliceShop

rmeth@rmeth-VirtualBox:~/aliceShop$ npm run dev

> pet-shop@1.0.0 dev /home/rmeth/aliceShop
> lite-server

** browser-sync config **
{ injectChanges: false,
  files: [ './**/*.{html,htm,css,js}' ],
  watchOptions: { ignored: 'node_modules' },
  server:
   { baseDir: [ './src', './build/contracts' ],
     middleware: [ [Function], [Function] ] } }
[Browsersync] Access URLs:
 --------------------------------------
       Local: http://localhost:3000
    External: http://10.0.2.15:3000
 --------------------------------------
          UI: http://localhost:3001
 UI External: http://10.0.2.15:3001
 --------------------------------------
[Browsersync] Serving files from: ./src
[Browsersync] Serving files from: ./build/contracts
[Browsersync] Watching files...
18.05.10 15:23:24 304 GET /index.html
18.05.10 15:23:24 304 GET /css/bootstrap.min.css
18.05.10 15:23:24 304 GET /js/web3.min.js
18.05.10 15:23:24 304 GET /js/bootstrap.min.js
18.05.10 15:23:24 304 GET /js/truffle-contract.js
18.05.10 15:23:24 304 GET /js/app.js
18.05.10 15:23:25 304 GET /jewellery.json
18.05.10 15:23:25 200 GET /PurchaseJewellery.json
18.05.10 15:23:25 304 GET /images/gold-ring-300.jpg
18.05.10 15:23:25 304 GET /images/gold-chain-400.jpg
18.05.10 15:23:25 304 GET /images/gold-necklace-5000.jpg
18.05.10 15:23:25 304 GET /images/gold-stone-200.jpg
18.05.10 15:23:25 304 GET /images/platinum-chain-600.jpg
18.05.10 15:23:25 304 GET /images/platinum-necklace-2600.jpg
18.05.10 15:23:25 304 GET /images/platinum-ring-450.jpg
18.05.10 15:23:25 304 GET /images/platinum-stone-300.jpg
18.05.10 15:23:25 304 GET /images/silver-chain-200.jpg
18.05.10 15:23:25 304 GET /images/silver-necklace-2000.jpg
18.05.10 15:23:25 304 GET /images/silver-ring-300.jpg
18.05.10 15:23:25 304 GET /images/silver-stone-400.jpg
```

This command opens the jewellery site in the default browser. In my case, Chrome is the default browser and the site opens in Chrome as shown in below screen.

First screen of the dapp site

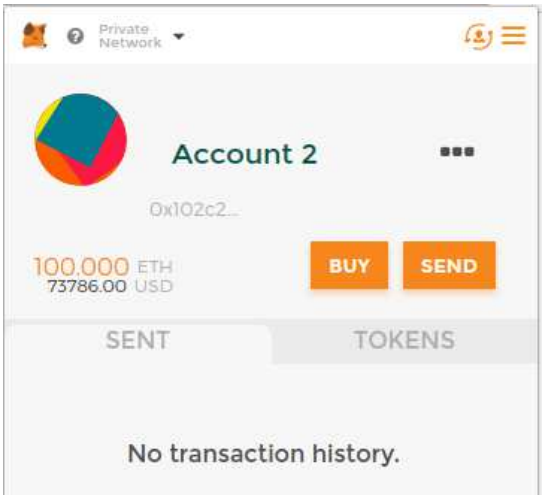


If we dragged down, there are more items.



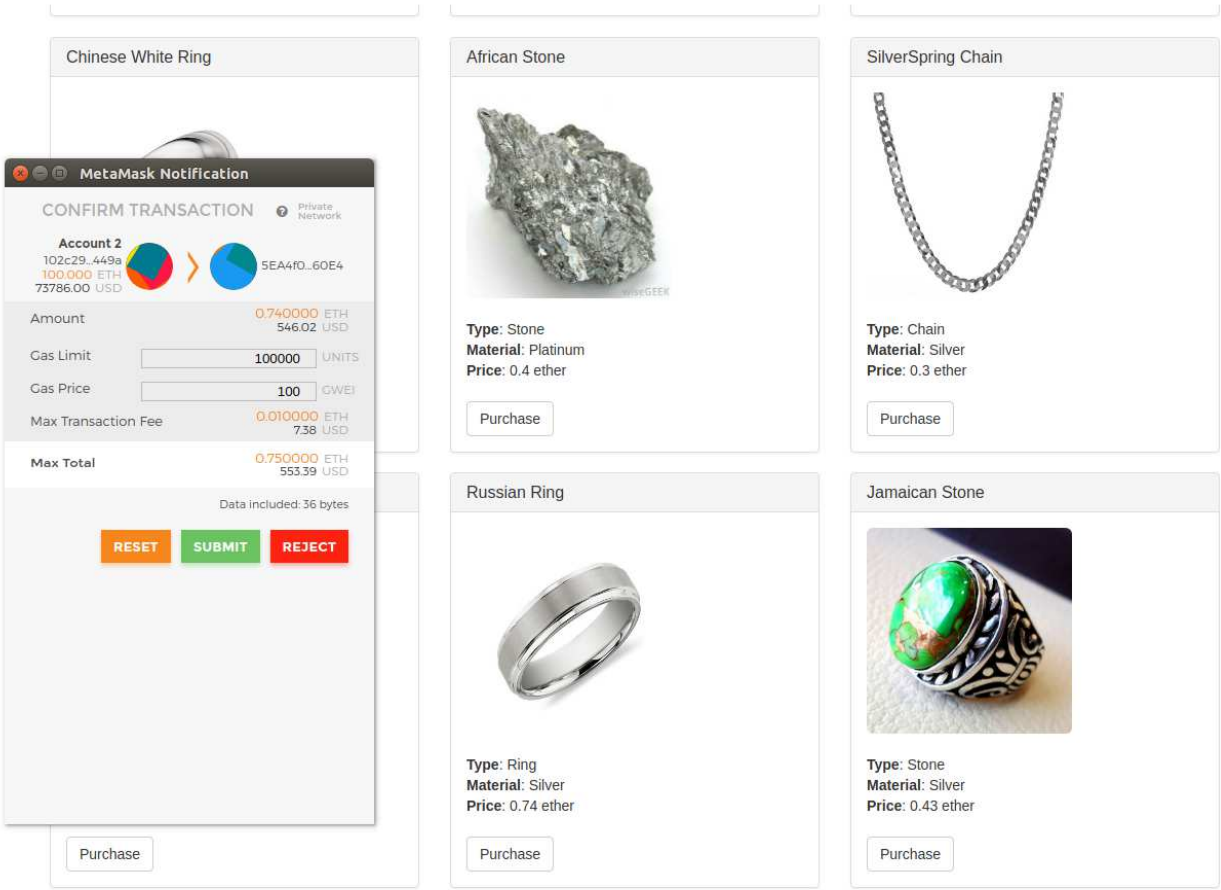
Let's do some purchase activity.

Before purchase item, let's use second account in MetaMask. If not there, let's add one more account going on Create Account option.

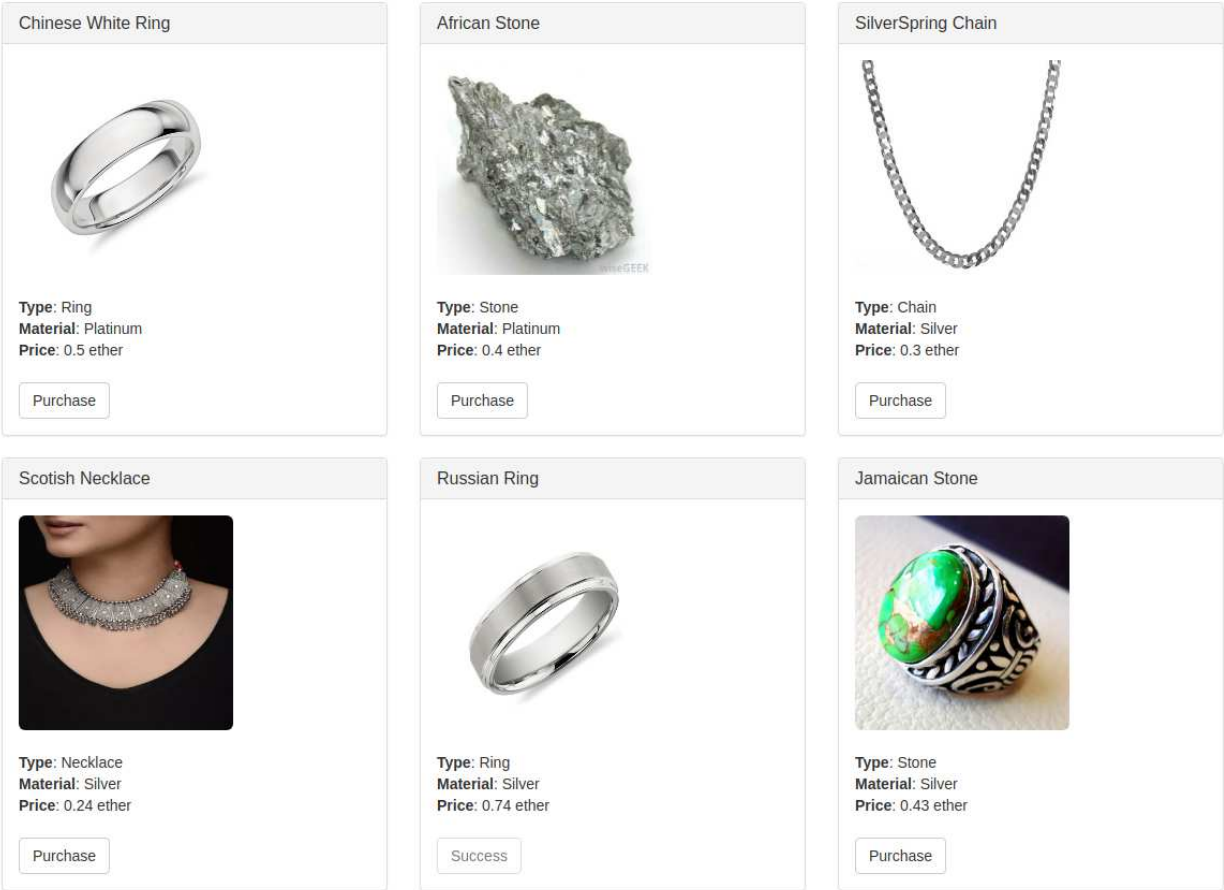


I am connected as Account2. I am going to click on Russian Ring which price is 0.74 ether. Once I clicked on Purchase button, the MetaMask Notification popped up. Here the Amount on MetaMask will be same as the price of the Russian Ring. I am passing the ether value to MetaMask using following code in web.js file

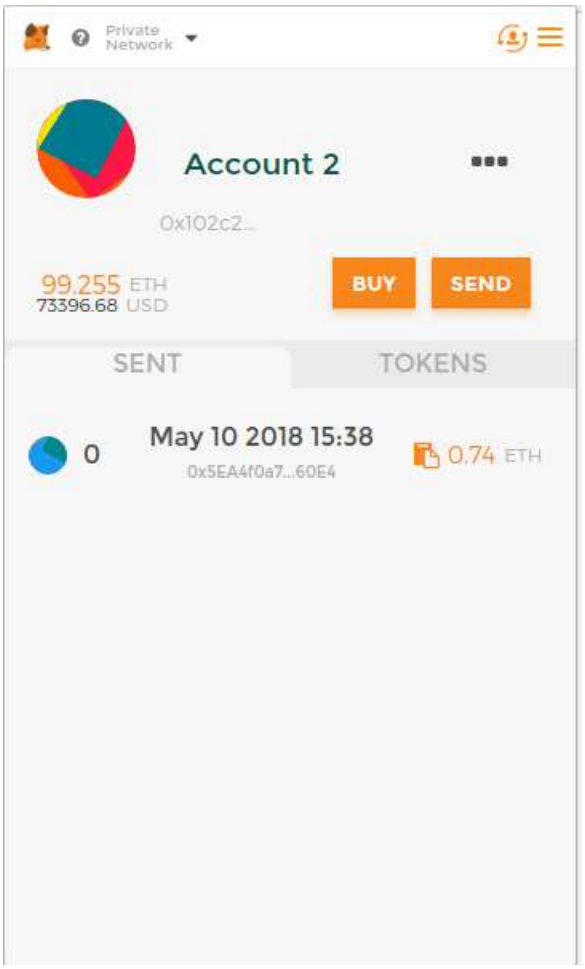
```
return purchaseInstance.buyItem(itemId, {from: account, value:
web3.toWei(itemPrice, 'ether'), gas: 100000});
```



As soon as submit the transaction from MetaMask, the button of the item will change to Success so that no one can purchase again.



The MetaMask transactions display as below screen shot.



Conclusions:

For the simplicity, I have used json file with only 12 records as the database, but we can use mango db or other kind of distributed database for larger project. This project gives us the idea and kind of ready project in distributed technology which makes user to do payment using the digital currency. This project only accepts ether for now, but we can extend this with any kind of digital currency like Litecoin, bitcoin or any another small value digital coin.

We can make it better more user friendly by adding filters by material type or others category or so. We can display the current dollars value with the amount of ether as well so that the users are aware with how much money they are paying right now without needing any calculations

Appendix

For the source code

https://github.com/rmpasha/eth-dapp-jewelleryShop