Decentralised Application Project

Team: **Block Business**
Yihan Xiao
Hao Chen
Chenwei Qian
Li Yu

Abstract: a decentralised shopping application cart supports online business operations by using a block-chain to manage transaction data. All trades happen directly between buyers and sellers with no middleman to take a cut from each sale.

Contents :

Overview :

The Block Business shopping website provides a functional distributed shopping platform using block-chain technology with Ethereum, a widely-used block-chain-based platform, which can handle most of foundational problem. The project focuses on implementing the smart contract for trading, data storage and user interface. The main parts of this project including:

1. Smart contract

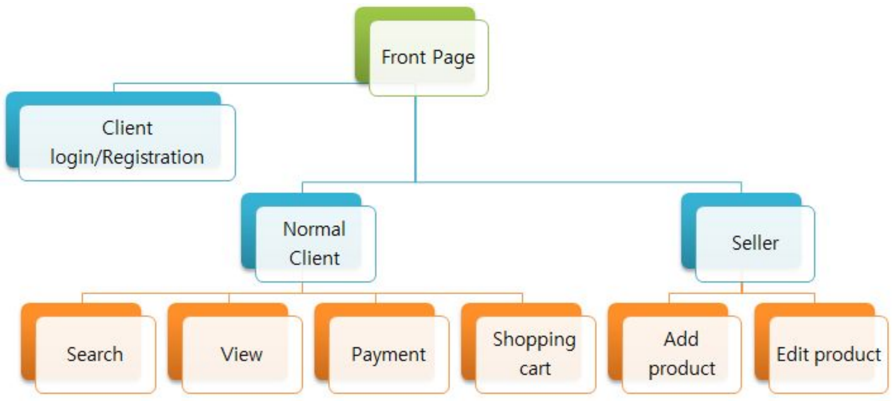
As a decentralised project, smart contract is something combined the traditional database and financial system. The monetary operation is naturally integrated in Ethereum with strong safety. User will use Ether as currency to trade in the application with others and all operation related to trade will be recorded and monitored by smart contract.
Generally speaking, traditional database are centralised, which brings heavy pressure to the server side to handle request from enormous users and easy to lose data if the server shutdown. Smart contract can solve this problem by storing all data on block-chain, decentralised that data are encrypted and stored among a network of nodes rather than single data server. Alternatively, the writing speed(mining speed of block) of Ethereum is rather slow due to drawbacks of block-chain. This will limit the idea that we can store everything we want in block-chain since we have to take the experience of users into account, for instance, no one will want to spend 10 minutes waiting for changing the mailing address.

2. Data storage

Each order's trading history is stored on block-chain so that once data is stored in it will be unchangeable(except for the status variable).
General information like user profile, product information and shopping cart will be modified frequently, which are stored in **mLab**, a web-based MongoDB database server.
To reduce the workload of the back-end server, product images are storing in a cloud service **Cloudinary.** These will be introduced in detail in functionality part.

3. Shopping system

The Block Business website provides a user friendly interface for visitors to explore products for trading. Each user can be a seller and a buyer. It includes user registration, profile management, product publishing and purchasing.
Searching products and filter them can help customer easily find the item he need. All features above aims to make the website more user-friendly and enhance user shopping experience.
The structure of the project is:

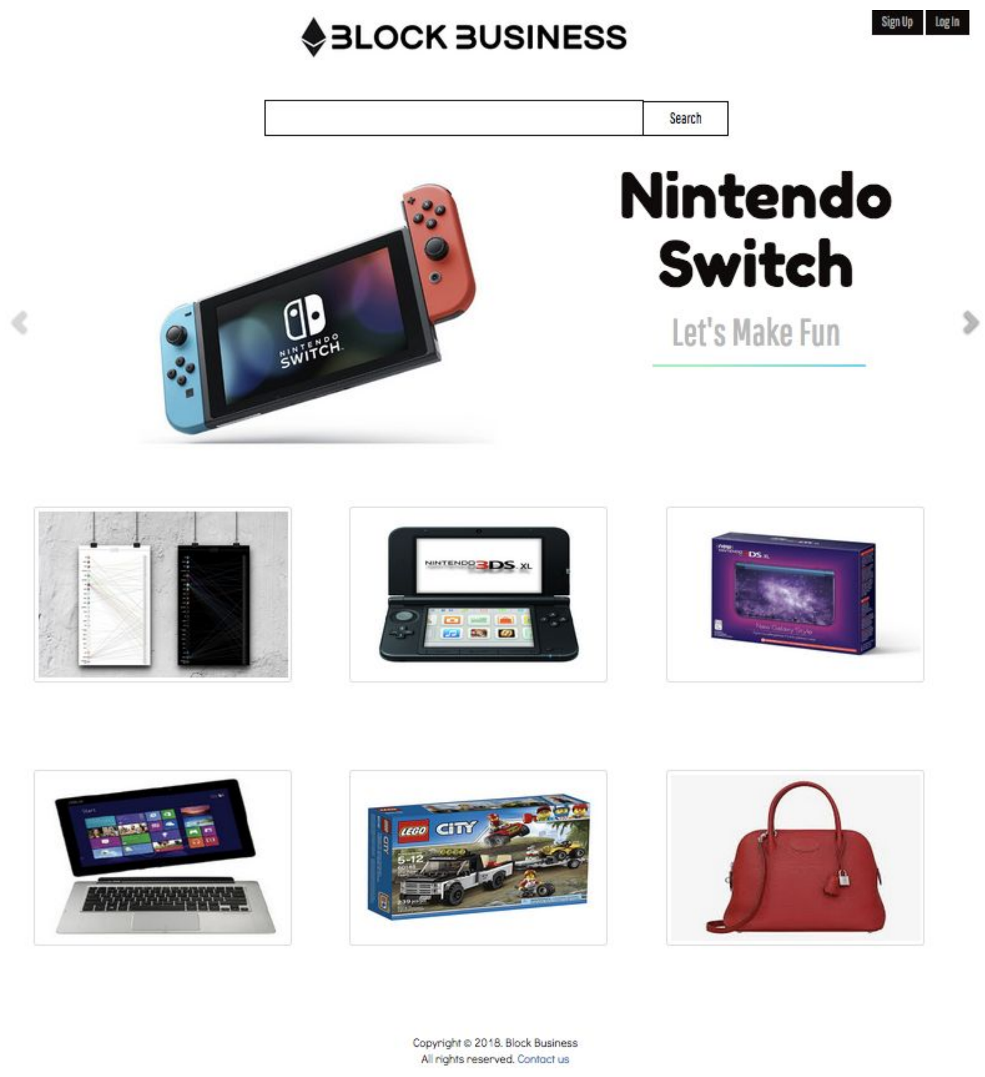


Functionalities & implementation:

1. Website functions

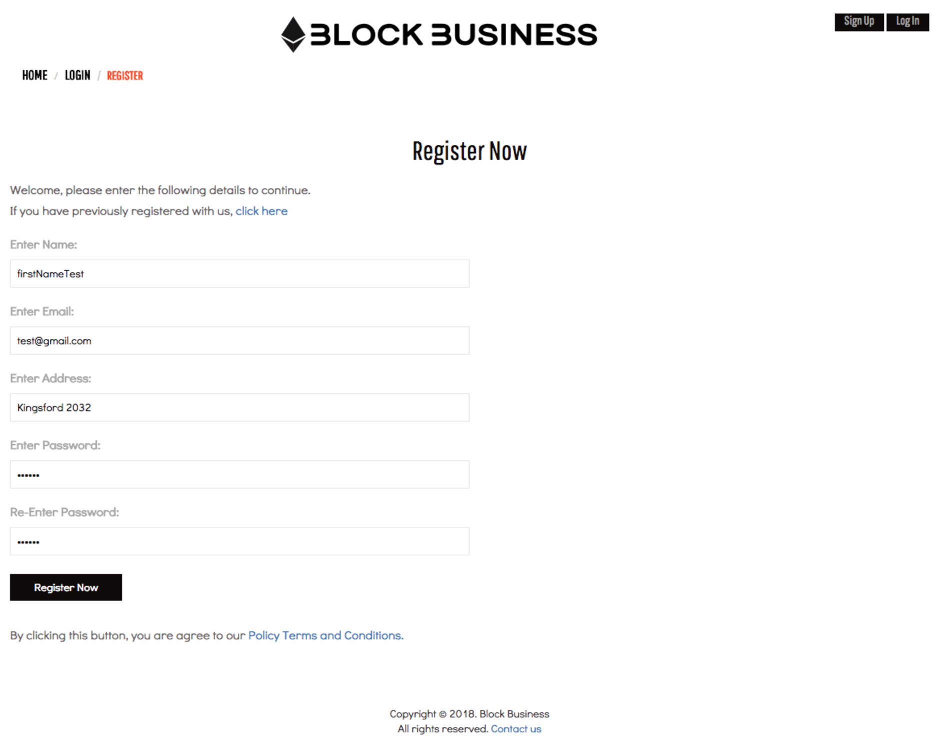
1) Welcome Home page

Any visitor of Block Business website access this welcome page. This home page display the latest products (from MongoDB Product Schema) post by sellers and an advertising area for some seller to make announcement of new products.


BLOCK BUSINESS
Sign Up
Log In
Search
Nintendo Switch
Let's Make Fun
Copyright © 2018. Block Business
All rights reserved. Contact us

2) User Register Module

The interface of our online shopping store of Block Business website for register and login. Here visitors could do the register and login function to continue their shopping travel. However, in our website visitors could also view the website, search for certain products and check one product's details. This is done by adding a specific session user when render some certain pages.



Sign Up Log In

BLOCK BUSINESS

HOME / LOGIN / REGISTER

Register Now

Welcome, please enter the following details to continue.
If you have previously registered with us, click here

Enter Name:
firstNameTest

Enter Email:
test@gmail.com

Enter Address:
Kingsford 2032

Enter Password:
••••••

Re-Enter Password:
••••••

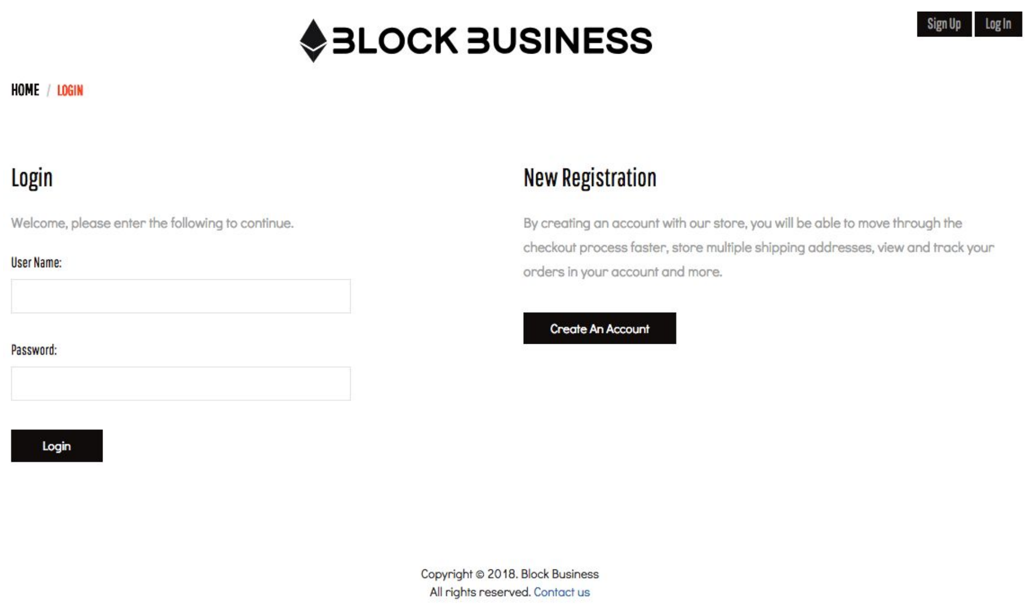
Register Now

By clicking this button, you are agree to our Policy Terms and Conditions.

Copyright © 2018. Block Business
All rights reserved. Contact us

3) User Login Module

In the Login pages we could also do the registration by clicking the right button, here the session will be created when you login and it can only be destroyed by clicking the Signout button on the welcome home page.



Sign Up Log In

BLOCK BUSINESS

HOME / LOGIN

Login

Welcome, please enter the following to continue.

User Name:

Password:

Login

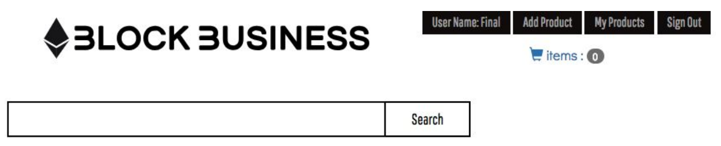
New Registration

By creating an account with our store, you will be able to move through the checkout process faster, store multiple shipping addresses, view and track your orders in your account and more.

Create An Account

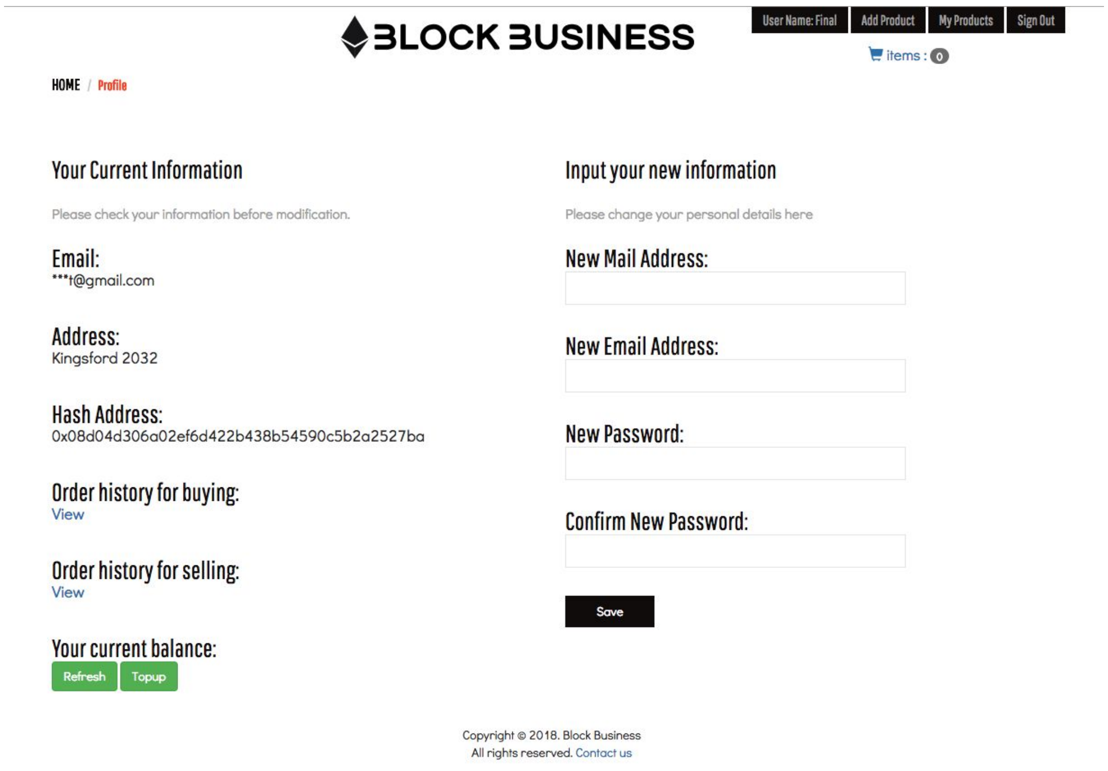
Copyright © 2018. Block Business
All rights reserved. Contact us

The button is shown as below.



4) User Profile Module

In the head navigation bar, the first button is displayed with the user's name. This button is directed to the user's profile page, which will show the user's details such like email address, the hash address obtained from the block-chain and his or her current balance and two port for his buying or ordering history.



Your Current Information

Please check your information before modification.

Email:
***t@gmail.com

Address:
Kingsford 2032

Hash Address:
0x08d04d306a02ef6d422b438b54590c5b2a2527ba

Order history for buying:
View

Order history for selling:
View

Your current balance:

Input your new information

Please change your personal details here

New Mail Address:

New Email Address:

New Password:

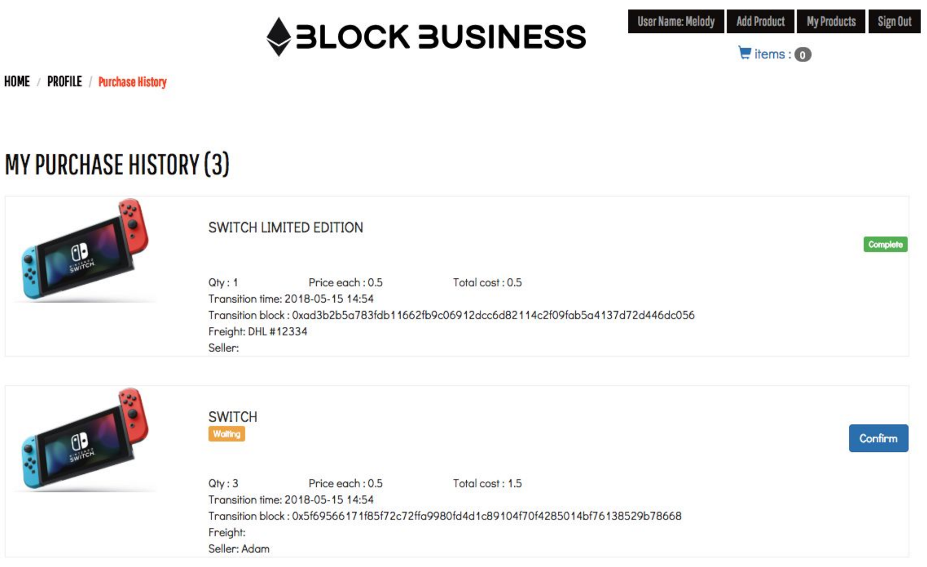
Confirm New Password:

Copyright © 2018. Block Business
All rights reserved. Contact us

This page also allow users to change their details by submitting the form in the right.
Click the "refresh" button, then it will show the current balance of this user.

5) User Purchase History Module

Our website allow users to view his or her purchase history, and in this page all products have two states. When one user purchased the product, this product will be shown here with a label delivering, here all the information about this product like transition time, transition block, freight, price and seller name.



MY PURCHASE HISTORY (3)

SWITCH LIMITED EDITION

Qty : 1 Price each : 0.5 Total cost : 0.5
Transition time: 2018-05-15 14:54
Transition block : 0xad3b2b5a783fdb11662fb9c06912dcc6d82114c2f09fab5a4137d72d446dc056
Freight: DHL #12334
Seller:

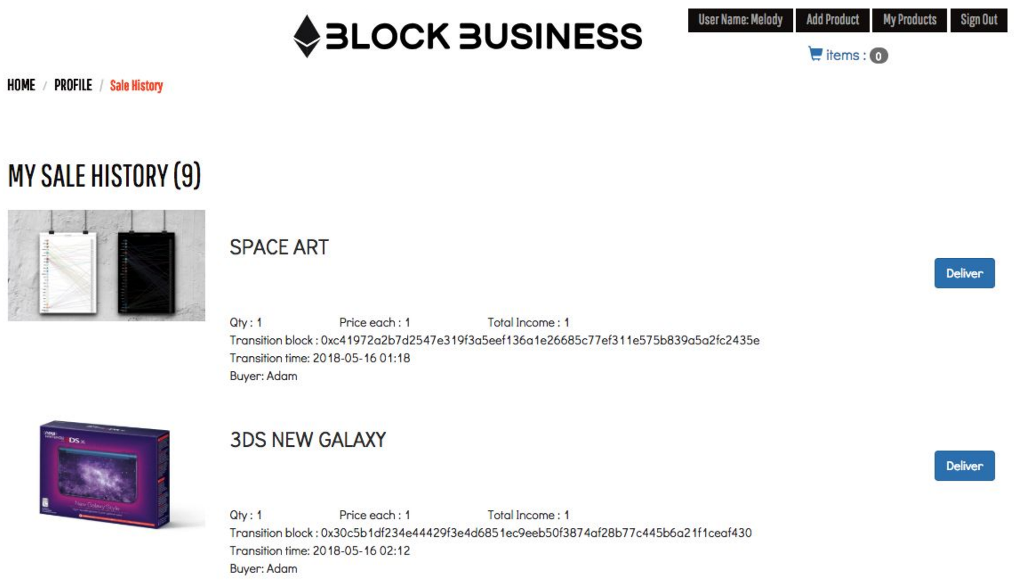
SWITCH

Qty : 3 Price each : 0.5 Total cost : 1.5
Transition time: 2018-05-15 14:54
Transition block : 0x5f69566171f85f72c72ffa9980fd4d1c89104f70f4285014bf76138529b78668
Freight:
Seller: Adam

When the user click the "Confirm" button, the label will be turned into complete, which means the products are received, and the money is going to the seller's account, and this process can't be reversed.

6) User Selling History Module

Our website allow users to view his or her sale history, and in this page all products also have two states.
When one user sells the product, this product will be shown here with a label delivered, and the seller also need to input the freight here and all information about this product will also be shown here such like price and buyer.



SPACE ART

Qty : 1 Price each : 1 Total Income : 1
Transition block : 0xc41972a2b7d2547e319f3a5eef136a1e26685c77ef311e575b839a5a2fc2435e
Transition time: 2018-05-16 01:18
Buyer: Adam

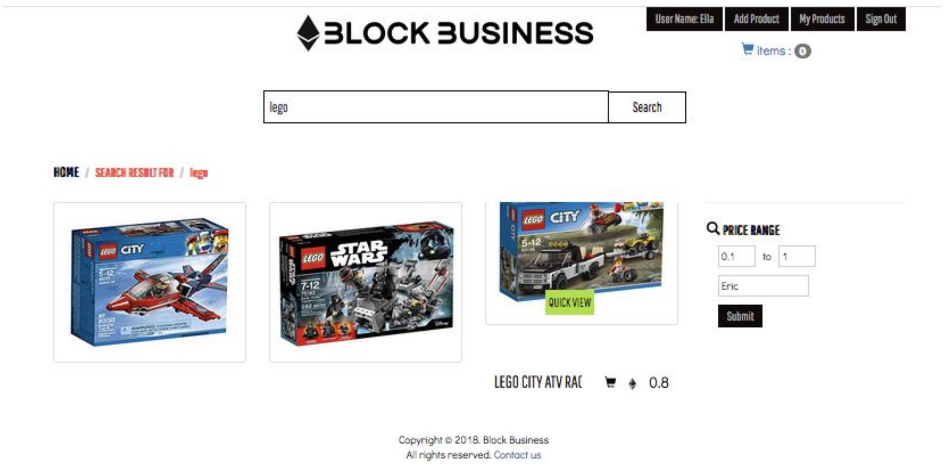
3DS NEW GALAXY

Qty : 1 Price each : 1 Total Income : 1
Transition block : 0x30c5b1df234e44429f3e4d6851ec9eeb50f3874af28b77c445b6a21f1ceaf430
Transition time: 2018-05-16 02:12
Buyer: Adam

When the customer clicked the "Confirm" button, the label here will be changed to the green one with "confirmed". The money will be given to your account, thus one whole process is completed.

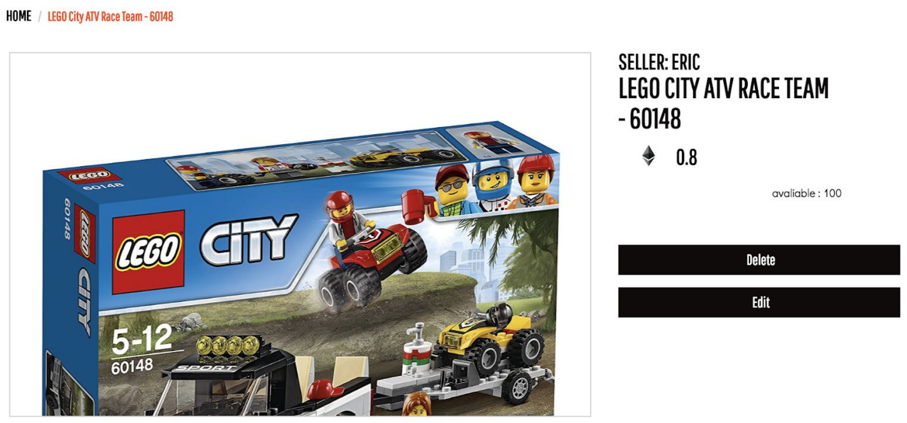
7) Search function Module

The search bar under the header can help users explore all products by their names. Even if a visitor is currently not a logged-in user, he still can search items on Block Business website. When mouse is hover over one item, the product will show its name, price and a quick view button. User can filter current results by set the price in a particular range. Furthermore, input a seller name in the filter area will show all products post by this seller.

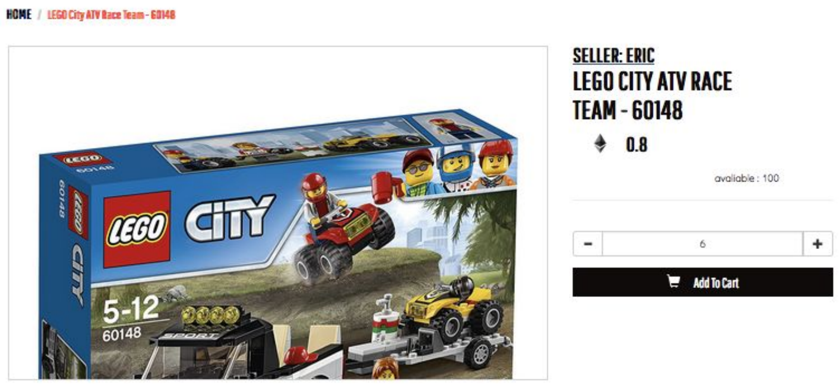


8) Single product display Module

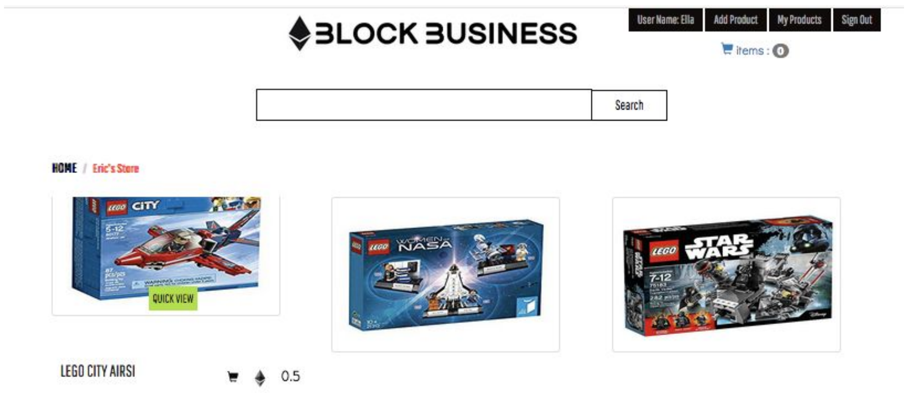
When different user visit a single product page, this item page will display different functions. If this product is posted by the current user, this page will "Delete" and "Edit" buttons for this seller to manage.


HOME / LEGO City ATV Race Team - 60148
SELLER: ERIC
LEGO CITY ATV RACE TEAM - 60148
0.8
avaliable : 100
Delete
Edit

Otherwise, this page display "Add To Cart" function for the buyer. Potential buyer can choose any quantity based on the available stock to add into his cart.

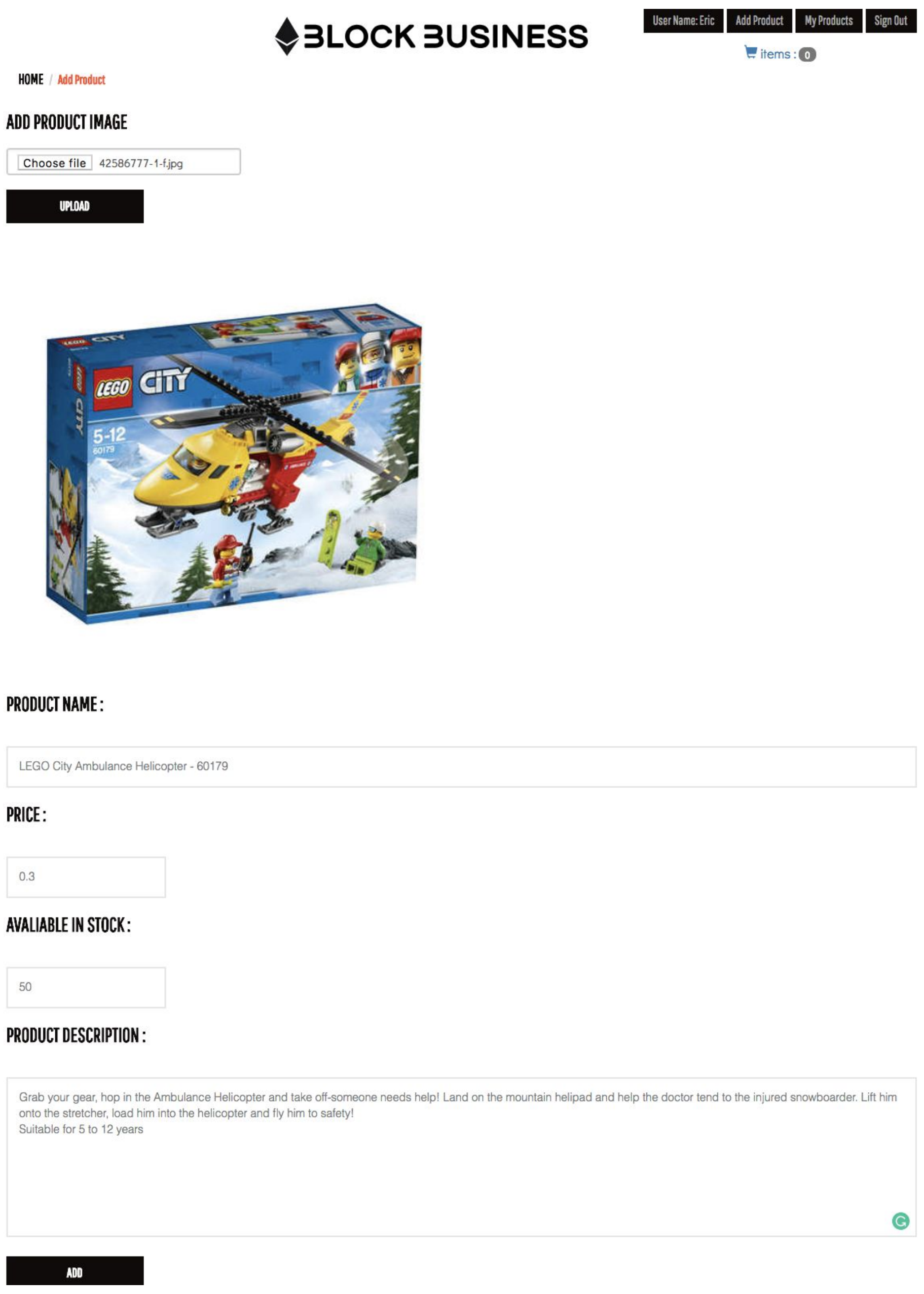

HOME / LEGO City ATV Race Team - 60148
SELLER: ERIC
LEGO CITY ATV RACE TEAM - 60148
0.8
avaliable : 100
-
6
+
Add To Cart

Clicking on the seller name link (above the product name) will redirect the logged-in user to the seller's store which display all products posted by this seller.



9) Add and Edit product Module

Any registered user of Block Business website can post products to sell. After choose a product image to upload to the Cloudinary, seller should fill in product name, price, available stock and product description. Then click on the "Add" button to add this product into MongoDB and then the page will be redirected to display this new posted product. Editing an existing item is like Adding products. The modified information will be update in MongoDB and available for all users to access.



PRODUCT NAME :

LEGO City Ambulance Helicopter - 60179

PRICE :

0.3

AVALIABLE IN STOCK :

50

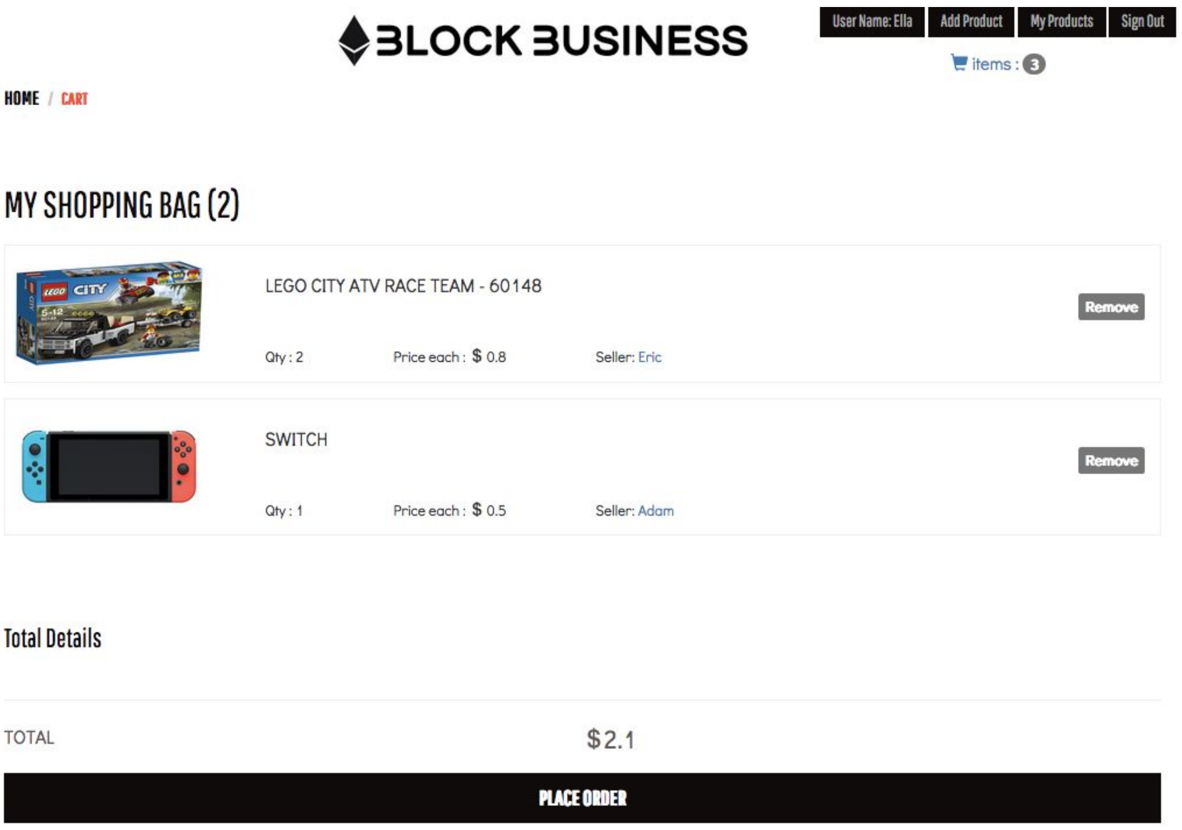
PRODUCT DESCRIPTION :

Grab your gear, hop in the Ambulance Helicopter and take off-someone needs help! Land on the mountain helipad and help the doctor tend to the injured snowboarder. Lift him onto the stretcher, load him into the helicopter and fly him to safety!
Suitable for 5 to 12 years

10) Cart Module

Any registered user of Block Business website can add products to cart and access his cart page. All items added to cart will be kept in MongoDB Carts Schema.
When loading the header, an Ajax request will be sent to the back-end to obtain amounts of all items in his cart of this login user. This total item number displays with a cart icon in the header area.

All items added by the current will be displayed in the cart page. "My shopping bag" will display numbers of all kinds of items, which is different to the items number in the header.
User can choose to remove any product in his cart. If the "Place Order" button is clicked, all items in this cart will send to block-chain to process trading.



MY SHOPPING BAG (2)

LEGO CITY ATV RACE TEAM - 60148
Qty : 2 Price each : $ 0.8 Seller: Eric

SWITCH
Qty : 1 Price each : $ 0.5 Seller: Adam

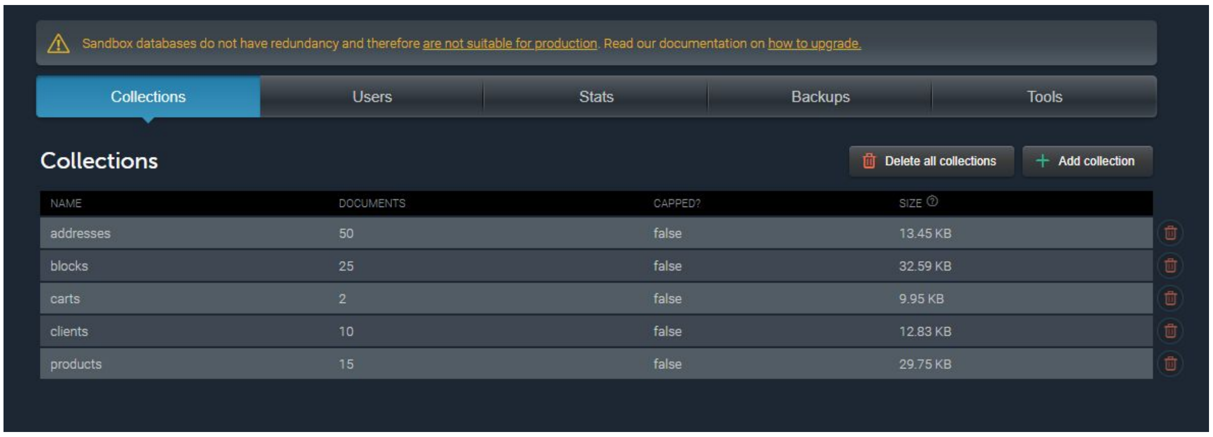
Total Details

TOTAL $2.1

2. Database and image hosting

1) mLab

To overcome the low interaction speed of block-chain, this website put static information and image in local database to increase the interactive speed and improve user experience. MongoDB is chosen to store data since its schema is flexible and easy to present at front-end since it naturally return json format result.
Local database is deployed at third party platform, "mLab". Since we use mongoose to define schema in MongoDB, All schema is shown below:



Sandbox databases do not have redundancy and therefore are not suitable for production. Read our documentation on how to upgrade.

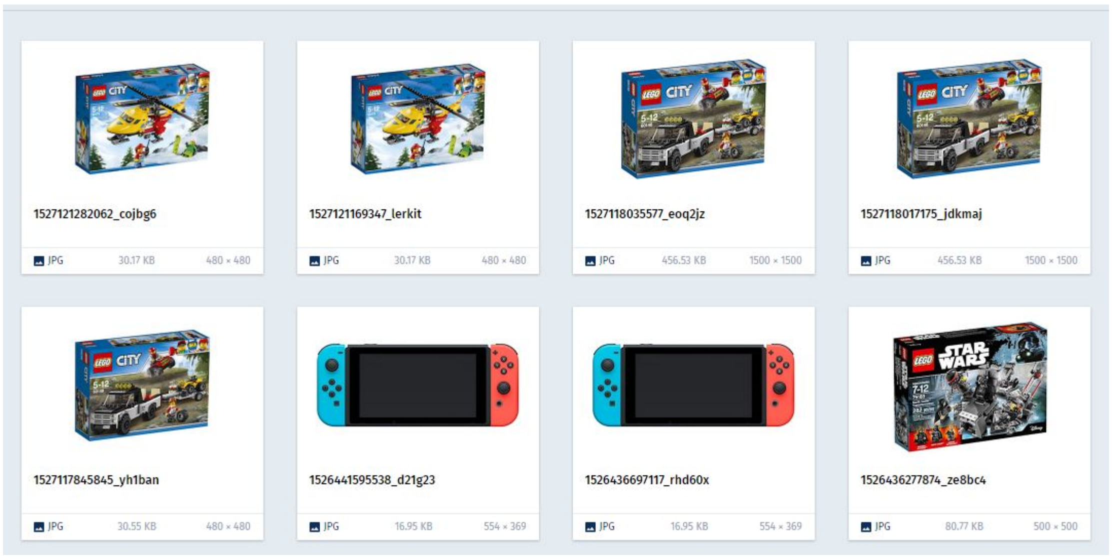
Collections

NAME	DOCUMENTS	CAPPED?	SIZE
addresses	50	false	13.45 KB
blocks	25	false	32.59 KB
carts	2	false	9.95 KB
clients	10	false	12.83 KB
products	15	false	29.75 KB

- Address: 50 block-chain hash address ready to assign to new user.
- Blocks: trade history for each user fetch from block-chain.
- Carts: Items of each user's shopping cart
- Clients: All user information, include username, password….
- Product: Information for each product, include product's price, description, image link….

2) Cloudinary

Image hosting is also deployed on third party platform. We use Cloudinary as our image host so that we only need to store the URL for each product in MongoDB, which make product picture can easily be visited. From it, image upload function is implemented. The cloudinary library is as follows:



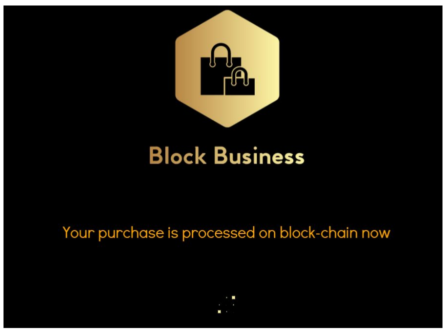
3. Smart Contract

The smart contract is kernel of our system. It handles the operation between the block-chain and user. Also, smart contract defines business logic and algorithm used to make the shopping website work at the back-end. Clients authority control is also handle by this part to keep user's privacy and data safety. The most important element is to store details of each trade and send/receive transactions.

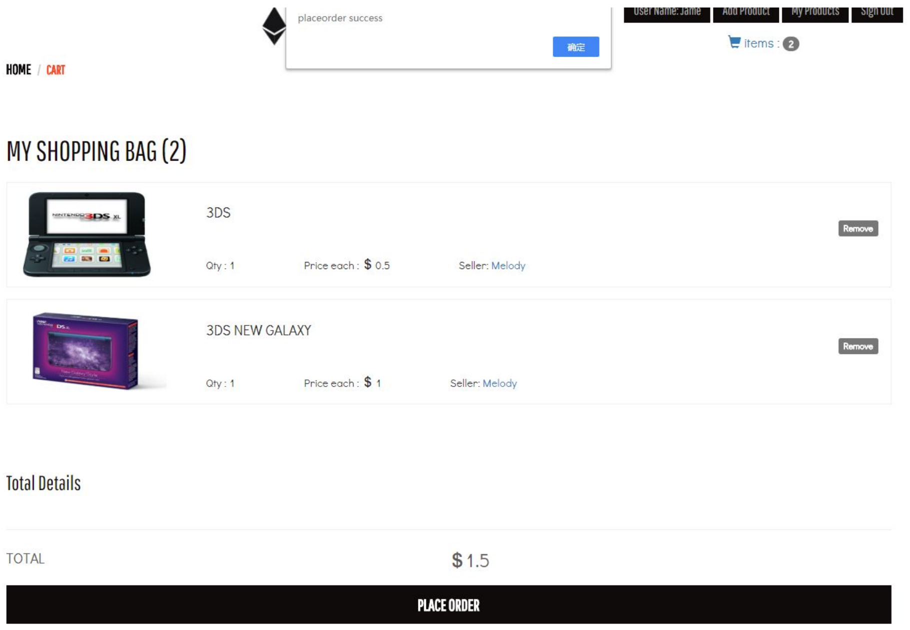
1) Place order with smart contract

When a user place order in the checkout page, the order will be recorded onto block-chain through smart contract. This includes several steps:

- Calculate the total price of items
- Look up the balance of user to check if it is enough to place order
- Encode all information of the order including the ID and price of items, information of both seller and buyer, timestamp and send to the smart contract.
- The smart contract returns a hash address and statues to webpage.



A loading page will show while processed order on block-chain by smart contract.



placeorder success

MY SHOPPING BAG (2)

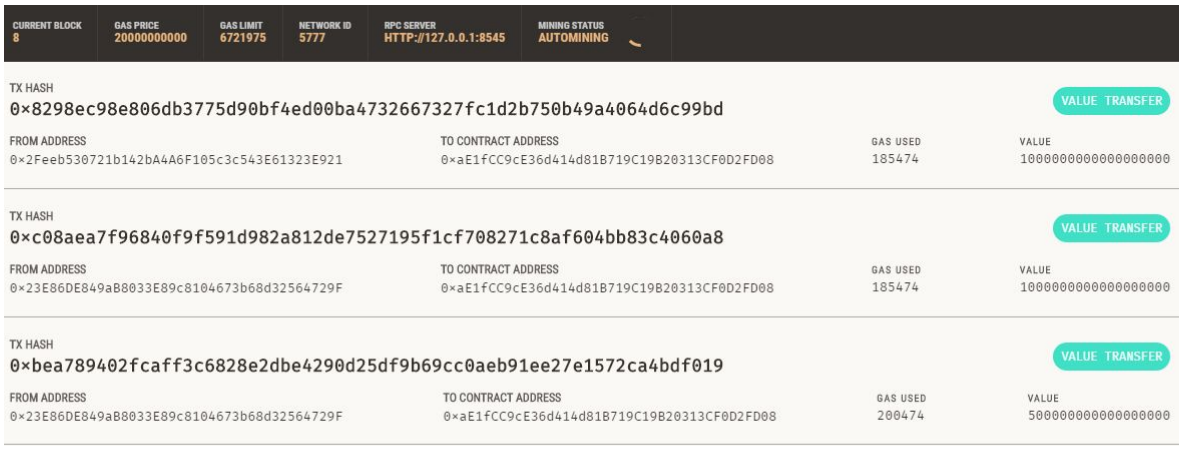
3DS
Qty : 1 Price each : $ 0.5 Seller: Melody

3DS NEW GALAXY
Qty : 1 Price each : $ 1 Seller: Melody

Total Details

TOTAL $ 1.5

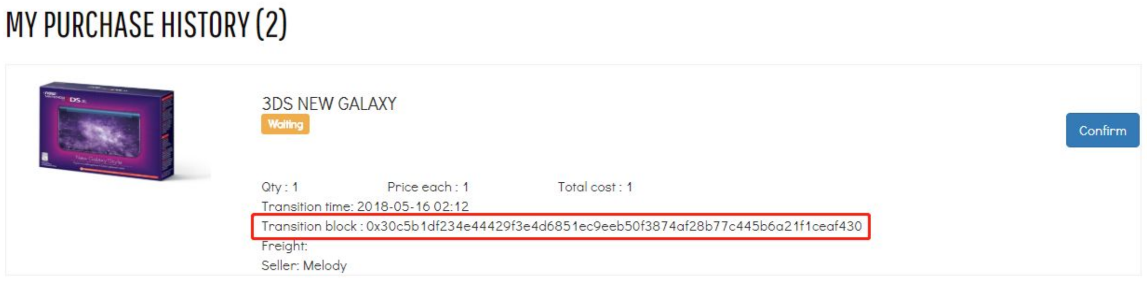
When smart contract returns with the statues and hash address of current trade, the page will automatically jump to the profile page of the user, from where the user can check the order made in the history page.
We can see the trade information on block-chain in Ganache after each transaction is made. Since all data on block-chain is encrypted, the details of encoded data is not accessible and all we can see is the hash address.



CURRENT BLOCK	GAS PRICE	GAS LIMIT	NETWORK ID	RPC SERVER	MINING STATUS
8	20000000000	6721975	5777	HTTP://127.0.0.1:8545	AUTOMINING

```
TX HASH
0×8298ec98e806db3775d90bf4ed00ba4732667327fc1d2b750b49a4064d6c99bd                              VALUE TRANSFER
FROM ADDRESS                                   TO CONTRACT ADDRESS                              GAS USED    VALUE
0×2Feeb530721b142bA4A6F105c3c543E61323E921     0×aE1fCC9cE36d414d81B719C19B20313CF0D2FD08       185474      1000000000000000000

TX HASH
0×c08aea7f96840f9f591d982a812de7527195f1cf708271c8af604bb83c4060a8                              VALUE TRANSFER
FROM ADDRESS                                   TO CONTRACT ADDRESS                              GAS USED    VALUE
0×23E86DE849aB8033E89c8104673b68d32564729F     0×aE1fCC9cE36d414d81B719C19B20313CF0D2FD08       185474      1000000000000000000

TX HASH
0×bea789402fcaff3c6828e2dbe4290d25df9b69cc0aeb91ee27e1572ca4bdf019                              VALUE TRANSFER
FROM ADDRESS                                   TO CONTRACT ADDRESS                              GAS USED    VALUE
0×23E86DE849aB8033E89c8104673b68d32564729F     0×aE1fCC9cE36d414d81B719C19B20313CF0D2FD08       200474      500000000000000000
```

The hash address of each trade is shown in the history of purchase/sale page for buyer/seller as below



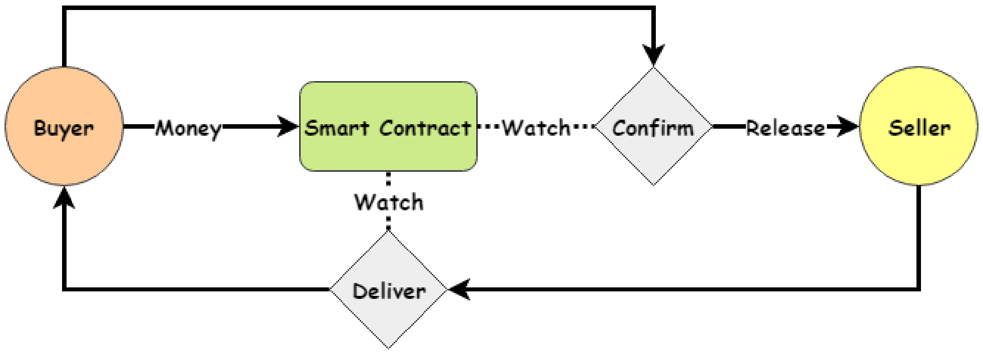
MY PURCHASE HISTORY (2)

3DS NEW GALAXY

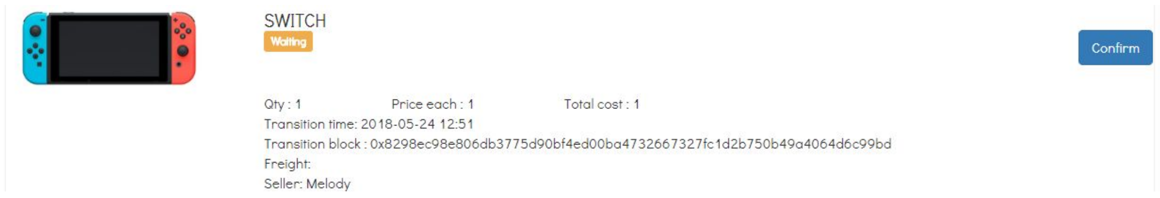
Qty : 1 Price each : 1 Total cost : 1
Transition time: 2018-05-16 02:12
Transition block : 0x30c5b1df234e44429f3e4d6851ec9eeb50f3874af28b77c445b6a21f1ceaf430
Freight:
Seller: Melody

2) Status monitor and escrow payment

When an order is placed with smart contract, the money(Ether) is not transferred to the seller directly. This is reasonable because the buyer has not received the items when the order is placed. In our design, the smart contract will take over the money(Ether) and release to the seller when buyer confirms. Therefore, we introduce status and status monitor function into smart contract, by which it is possible to record and watch the change of status of each trade.
The status of a single trade includes:



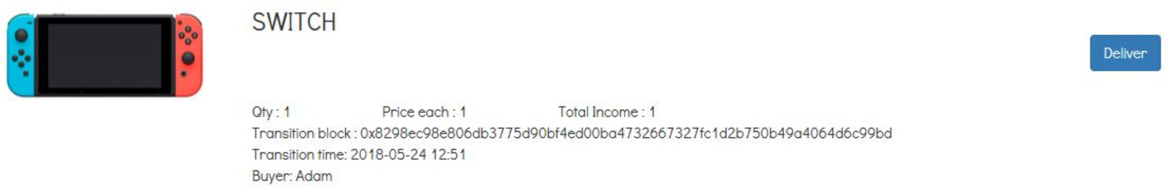
The "Confirm" and "Deliver" status are monitored and recorded by the smart contract. The process is shown below.



SWITCH
Waiting

Qty : 1 Price each : 1 Total cost : 1
Transition time: 2018-05-24 12:51
Transition block : 0x8298ec98e806db3775d90bf4ed00ba4732667327fc1d2b750b49a4064d6c99bd
Freight:
Seller: Melody

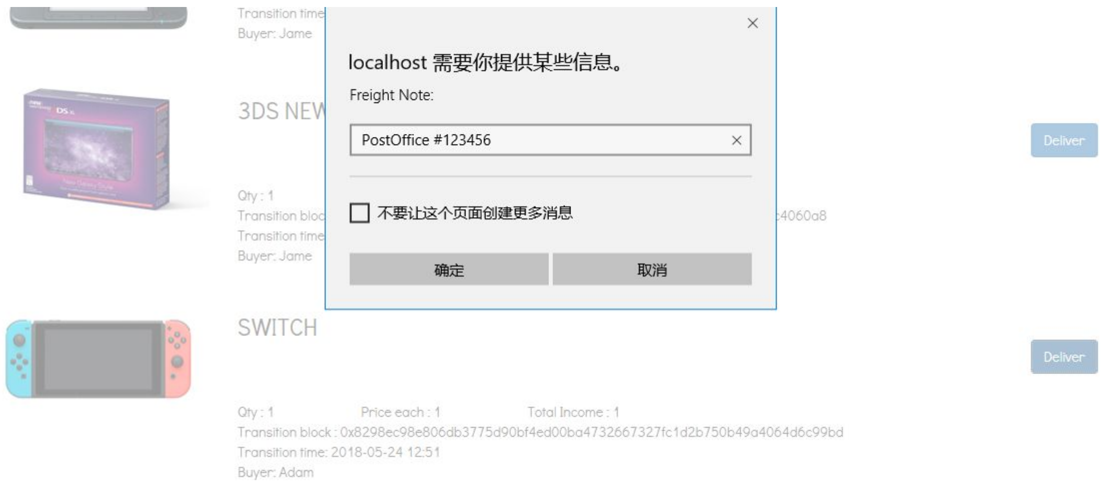
The orange notice label “Waiting” shows the seller has not made delivery so far. The blue button at right side is to confirm the trade to release money(Ether) to the seller.
From the perspective of seller, the page of sale history is shown below.



SWITCH

Qty : 1 Price each : 1 Total Income : 1
Transition block : 0x8298ec98e806db3775d90bf4ed00ba4732667327fc1d2b750b49a4064d6c99bd
Transition time: 2018-05-24 12:51
Buyer: Adam

The blue button at right side is to make delivery to the buyer. Once click, the seller will be asked to enter the index number of freight.



Transition time
Buyer: Jame

3DS NEW

Qty : 1
Transition bloc
Transition time
Buyer: Jame

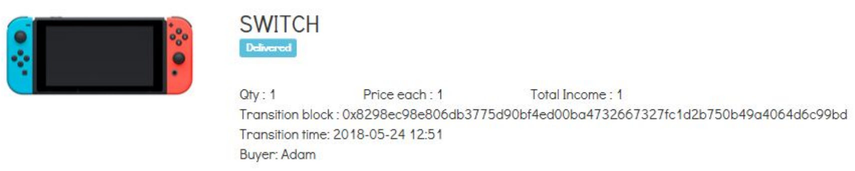
localhost 需要你提供某些信息。
Freight Note:
PostOffice #123456
☐ 不要让这个页面创建更多消息

4060a8

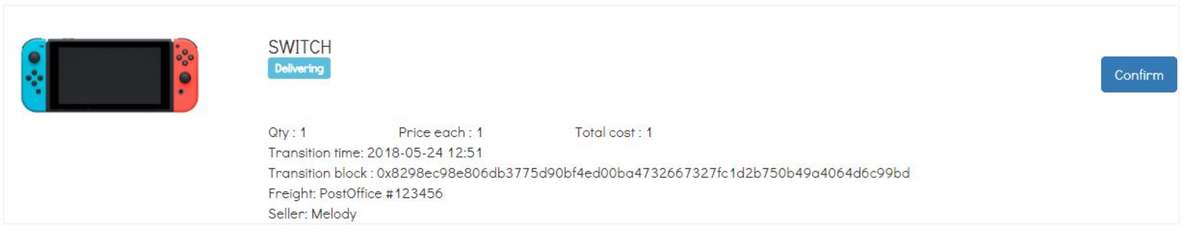
SWITCH

Qty : 1 Price each : 1 Total Income : 1
Transition block : 0x8298ec98e806db3775d90bf4ed00ba4732667327fc1d2b750b49a4064d6c99bd
Transition time: 2018-05-24 12:51
Buyer: Adam

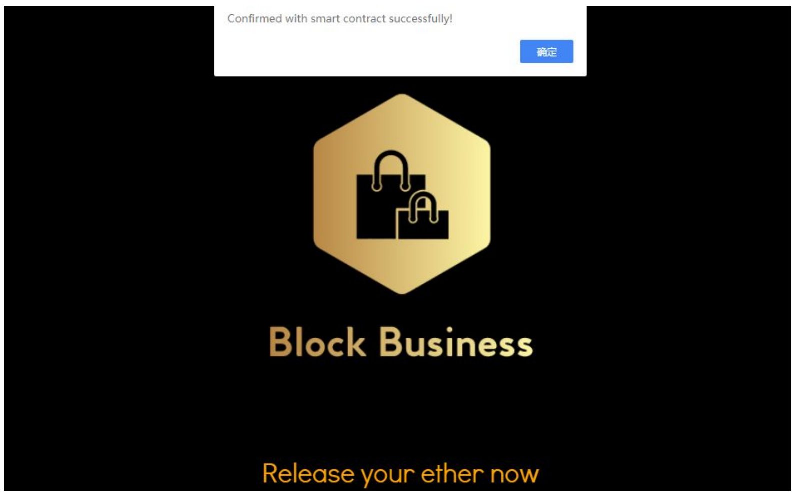
The information of freight input by seller will be recorded by smart contract and sent to the buyer. After making delivery, the status on the page of seller will change:


SWITCH
Delivered
Qty : 1 Price each : 1 Total Income : 1
Transition block : 0x8298ec98e806db3775d90bf4ed00ba4732667327fc1d2b750b49a4064d6c99bd
Transition time: 2018-05-24 12:51
Buyer: Adam

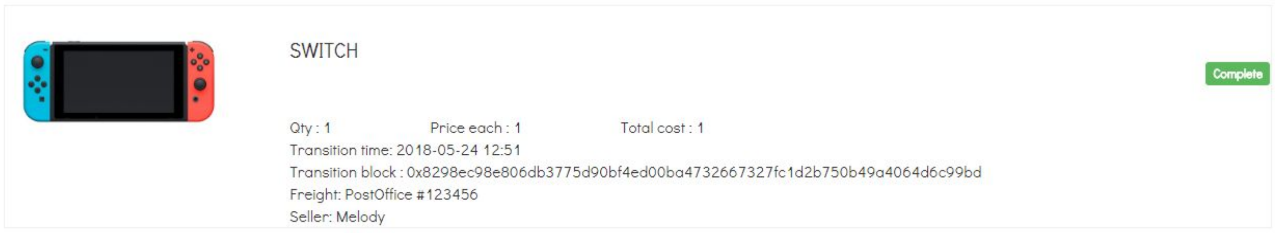
The light blue status notice label “Delivered” indicates this item has been delivered to the buyer. And the "Deliver" button is disable to prevent duplicate delivery.
The buyer will find the change of status then.


SWITCH
Delivering
Confirm
Qty : 1 Price each : 1 Total cost : 1
Transition time: 2018-05-24 12:51
Transition block : 0x8298ec98e806db3775d90bf4ed00ba4732667327fc1d2b750b49a4064d6c99bd
Freight: PostOffice #123456
Seller: Melody

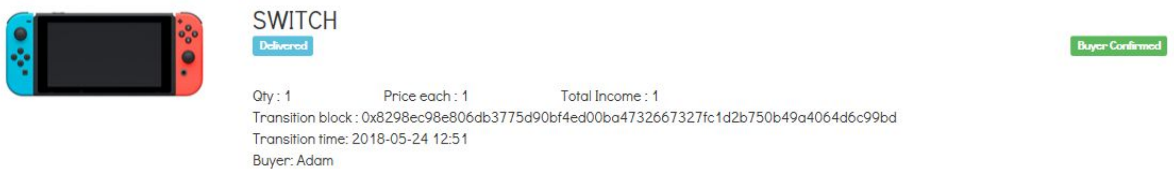
The notice label now is changed to light blue one “Delivering” and the information of freight is displayed. If the buyer choose to confirm the trade, the money will be released to the seller.


Confirmed with smart contract successfully!

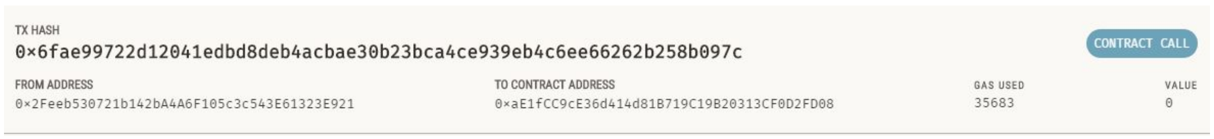
Then the status on buyer side will now change to **Complete**.


SWITCH
Complete
Qty : 1 Price each : 1 Total cost : 1
Transition time: 2018-05-24 12:51
Transition block : 0x8298ec98e806db3775d90bf4ed00ba4732667327fc1d2b750b49a4064d6c99bd
Freight: PostOffice #123456
Seller: Melody

The status on seller side will change to **Buyer Confirmed**.


SWITCH
Delivered
Buyer Confirmed
Qty : 1 Price each : 1 Total Income : 1
Transition block : 0x8298ec98e806db3775d90bf4ed00ba4732667327fc1d2b750b49a4064d6c99bd
Transition time: 2018-05-24 12:51
Buyer: Adam

Now the trade is finally complete and seller will receive the money(Ether) released from smart contract. We can see the release process of smart contract from the record of Ganage:

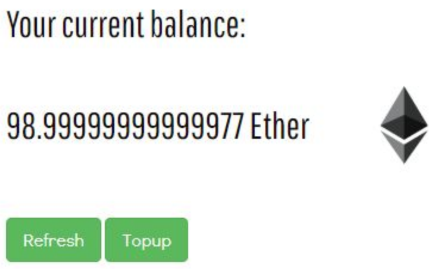

TX HASH
0x6fae99722d12041edbd8deb4acbae30b23bca4ce939eb4c6ee66262b258b097c

CONTRACT CALL

FROM ADDRESS	TO CONTRACT ADDRESS	GAS USED	VALUE
0x2Feeb530721b142bA4A6F105c3c543E61323E921	0xaE1fCC9cE36d414d81B719C19B20313CF0D2FD08	35683	0

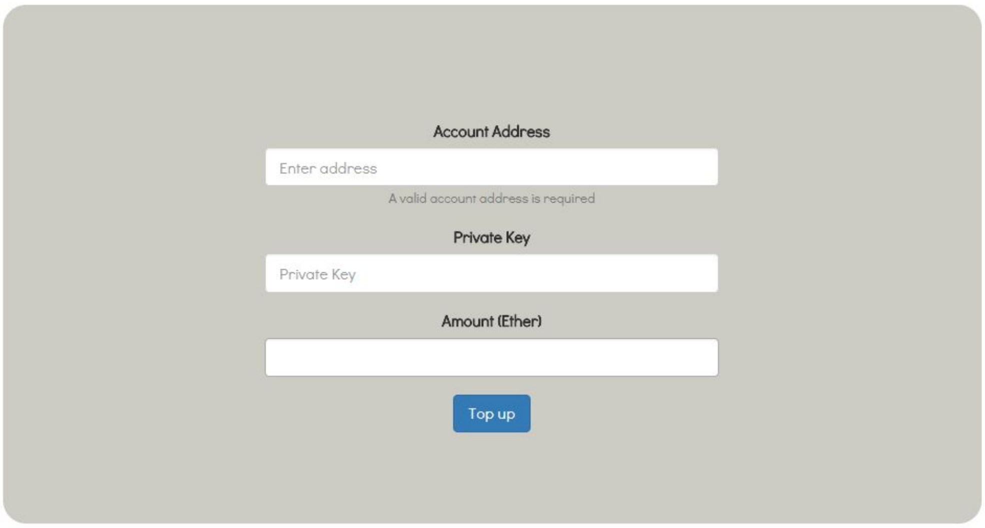
3) Top-Up

User can top-up balance from the **Profile** page as shown below.

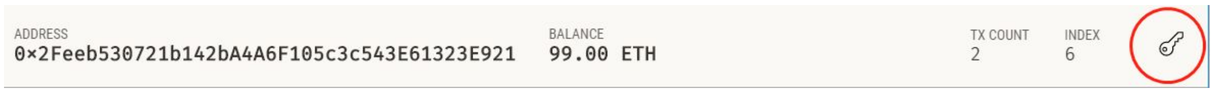

Your current balance:

98.99999999999977 Ether

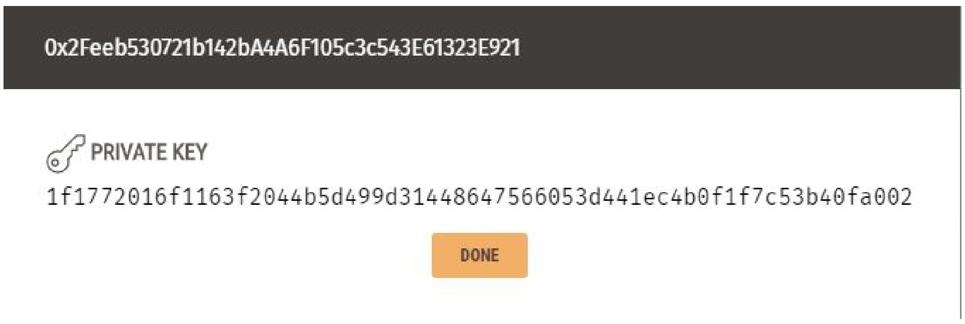
Clicking on **Topup**, it will jump to a specific page for Top-up.



User will need to insert the account the Ether is transferred from, and the private key to unlock the account and the amount of Ether. The private key is actually not necessary when using Ganache because all addresses in Ganache is set to be unlocked by default. However, in practical application with Ethereum, it is required to unlock the account before any further manipulation.
The private key could be found in Ganache interface. Every address will have its own private key and you can simply copy-and-paste.



ADDRESS	BALANCE	TX COUNT	INDEX
0x2Feeb530721b142bA4A6F105c3c543E61323E921	99.00 ETH	2	6

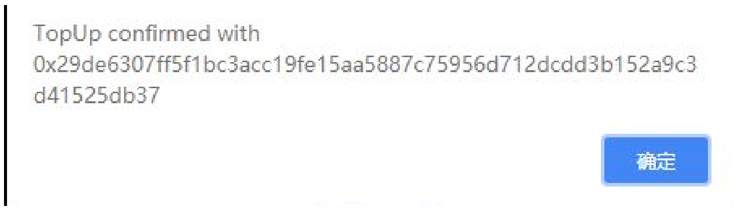


0x2Feeb530721b142bA4A6F105c3c543E61323E921

PRIVATE KEY

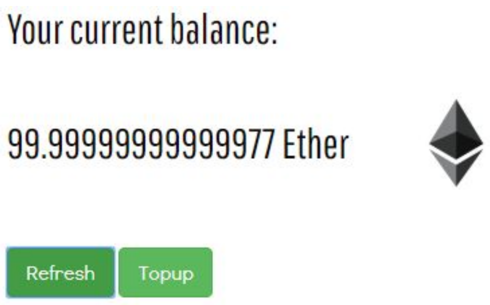
1f1772016f1163f2044b5d499d31448647566053d441ec4b0f1f7c53b40fa002

Click on “Top up” to submit the transaction to block-chain and a hash address of the transaction will be returned if it is processed successfully.



TopUp confirmed with 0x29de6307ff5f1bc3acc19fe15aa5887c75956d712dcdd3b152a9c3d41525db37

Go back to the profile page and refresh balance.



Your current balance:

99.99999999999977 Ether

4. Parts not implemented

The advanced data processing module has not been implemented so far. Basically, we do not have enough time to dig into more complicated functionality with smart contract as expected. The advanced data processing as we depicted in the proposal including price tracking and items recommendation is highly related to data mining tech, which is rather sophisticated and we have no sufficient time to design and test algorithms and generating enough sales data.

Implementation challenges:

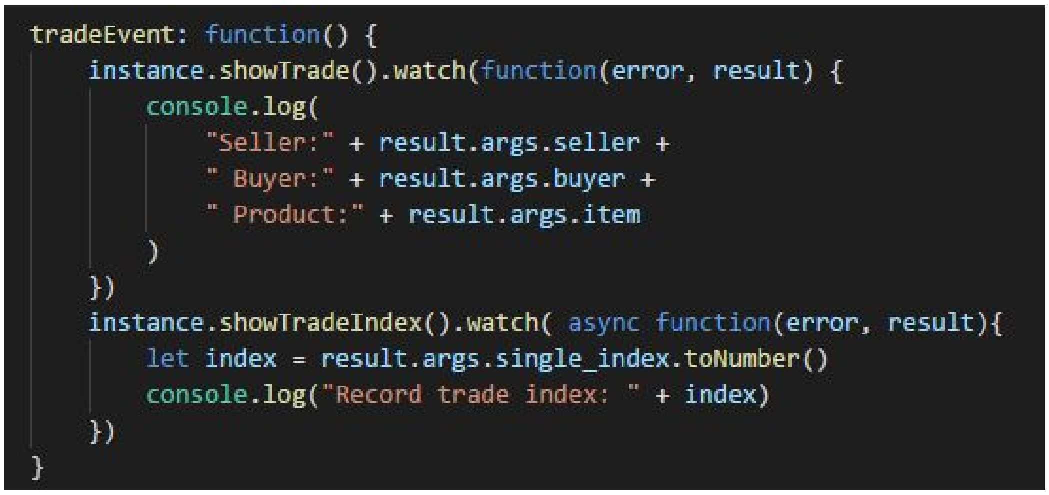
1. Design of smart contract

Store information on block-chain is a costly choice. Unlike traditional database, every writing operation will consume a mount of Ether(usually named Gas in term of Ethereum). It is important to choice what to store and how to store efficiently. In our implementation, we store the trade information into smart contract and keep tracking the change of status in every unfinished trade. We use a mapping data structure in Solidity, similar to dictionary, to keep the data. It is easy to look up and manipulate each trade with this structure. The key of mapping is the index of current trade in contract and the index is a self-increasing sequence as normally used in database. The smart contract will return the index as the identity of a trade to server and further manipulation is based on this index.

2. Event of Solidity

Calling functions inside smart contract implemented with Solidity is different with traditional API. The first problem is Solidity functions will not return any value besides the hash receipt to the caller which is modules written in Javascript. So as to catch the expected return values, we use event triggered in Solidity to inform the caller. When a transaction is done inside a Solidity function, an event is triggered and the values of variables will be sent to server side. Instead of catching the return value of a Solidity function, we actually try to catch the value registered as an event.

An example of event implementation is as below:



```
tradeEvent: function() {
    instance.showTrade().watch(function(error, result) {
        console.log(
            "Seller:" + result.args.seller +
            " Buyer:" + result.args.buyer +
            " Product:" + result.args.item
        )
    })
    instance.showTradeIndex().watch( async function(error, result){
        let index = result.args.single_index.toNumber()
        console.log("Record trade index: " + index)
    })
}
```

The flexible use of event makes the process control clear and easy to handle. For example, a transaction will trigger the change of several status on the page of both buyer and seller automatically.

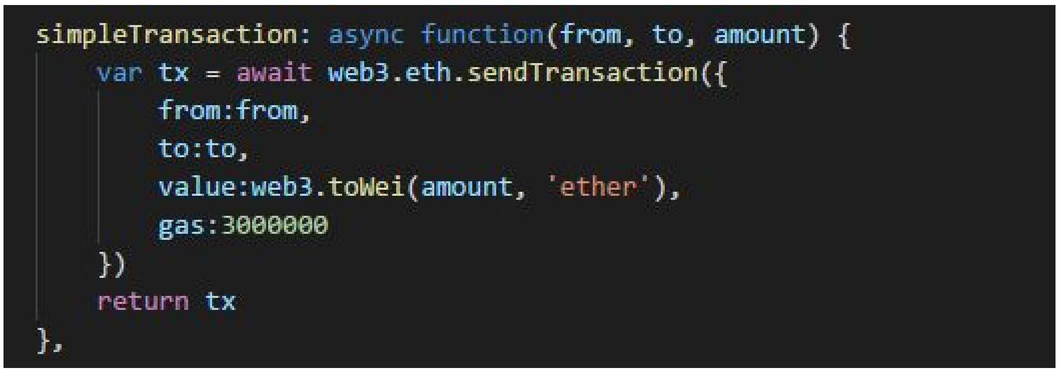
3. Asynchronism

Many of the interactions between server and smart contract are asynchronous, which means the server will not get messages instantly from smart contract since the mining of block of take time. Javascript is popular for its event-driven system which brings great efficiency for handling request. However, the management of asynchronism is troublesome usually and may lead to “Hell of callback” when multiple functions are waiting , or forming a nested waiting queue (a function is waiting in the wrap of another function and so on).

We make fully use of await/async feature introduced into Javascript by ES7. All asynchronous functions we implement will return a Promise as below:

```
var promise1 = new Promise(function(resolve, reject) {
  setTimeout(resolve, 100, 'foo');
});

console.log(promise1);
// expected output: [object Promise]
```

The **Promise** will indicate the result when ready in resolve status. The keyword await will tell the compiler an asynchronous operation exists and we need to wait for it to return.



```
simpleTransaction: async function(from, to, amount) {
    var tx = await web3.eth.sendTransaction({
        from:from,
        to:to,
        value:web3.toWei(amount, 'ether'),
        gas:3000000
    })
    return tx
},
```

User documentation/manual:

1. Initialisation:

1) Install NPM

The project is based on **Node.js**, which requires NPM environment to install all needed package. You need to install Node.js and NPM in the first place.
Start terminal at the root directory.
Run:

```
npm install
```

This will install all packages the project needs according to the **package.json** file in root path.

2) Install Webpack

Webpack is used to pack all files related to block-chain. You may need to run the following commands:
Run:

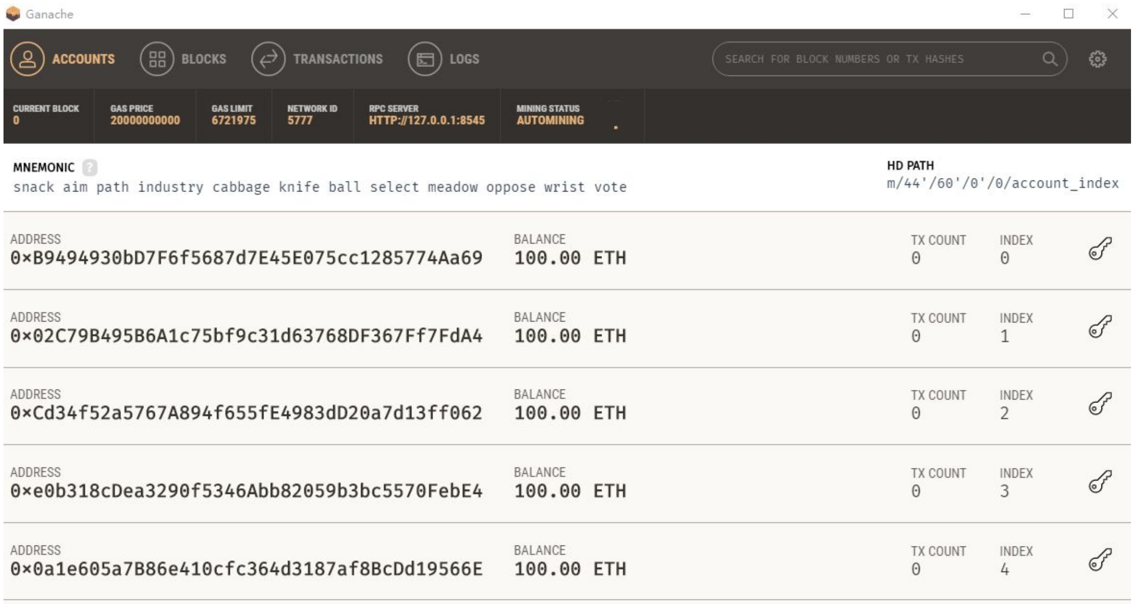
```
npm install -g webpack
npm install -g webpack-cli
```

3) Install TestRPC(Ganache)

Download and install following the link:

http://truffleframework.com/ganache/

Start Ganache and you will see the interface:



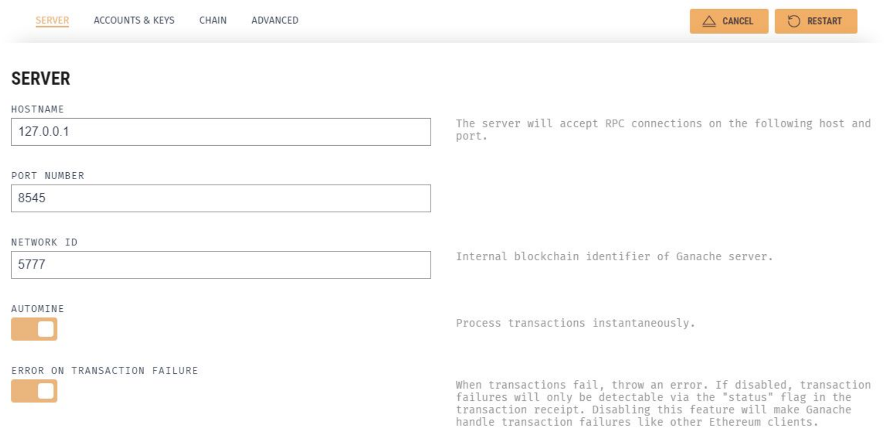
CURRENT BLOCK	GAS PRICE	GAS LIMIT	NETWORK ID	RPC SERVER	MINING STATUS
0	20000000000	6721975	5777	HTTP://127.0.0.1:8545	AUTOMINING

MNEMONIC
snack aim path industry cabbage knife ball select meadow oppose wrist vote

HD PATH
m/44'/60'/0'/0/account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0×B9494930bD7F6f5687d7E45E075cc1285774Aa69	100.00 ETH	0	0
0×02C79B495B6A1c75bf9c31d63768DF367Ff7FdA4	100.00 ETH	0	1
0×Cd34f52a5767A894f655fE4983dD20a7d13ff062	100.00 ETH	0	2
0×e0b318cDea3290f5346Abb82059b3bc5570FebE4	100.00 ETH	0	3
0×0a1e605a7B86e410cfc364d3187af8BcDd19566E	100.00 ETH	0	4

Click the configure icon at the top-right, next to the search bar. Then fill in all information of SERVER page as below. The Port Number is extremely important, and make sure it is set to **8545** otherwise the webpage will fail to connect.



SERVER

HOSTNAME: 127.0.0.1 — The server will accept RPC connections on the following host and port.

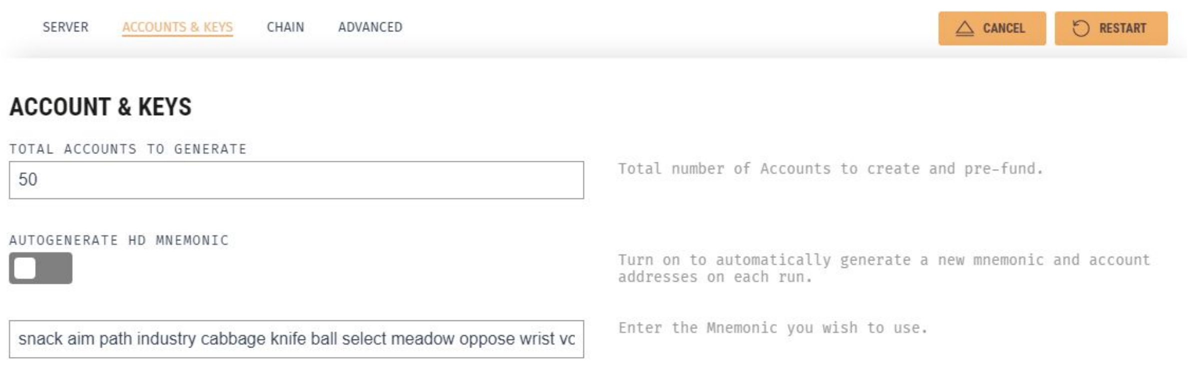
PORT NUMBER: 8545

NETWORK ID: 5777 — Internal blockchain identifier of Ganache server.

AUTOMINE — Process transactions instantaneously.

ERROR ON TRANSACTION FAILURE — When transactions fail, throw an error. If disabled, transaction failures will only be detectable via the "status" flag in the transaction receipt. Disabling this feature will make Ganache handle transaction failures like other Ethereum clients.

Click on ACCOUNT&KEYS next to SERVER and you shall see a page as below. Copy the keys from **address_keys.txt** in the root path and paste in the keys field here. The addresses generated by Ganache are different on different PC and this will cause problem that the address assigned to user may conflict with each other. We use the identical key to ensure the consistency of address pool. Set the number of address to **50**.



ACCOUNT & KEYS

TOTAL ACCOUNTS TO GENERATE: 50 — Total number of Accounts to create and pre-fund.

AUTOGENERATE HD MNEMONIC — Turn on to automatically generate a new mnemonic and account addresses on each run.

snack aim path industry cabbage knife ball select meadow oppose wrist vc — Enter the Mnemonic you wish to use.

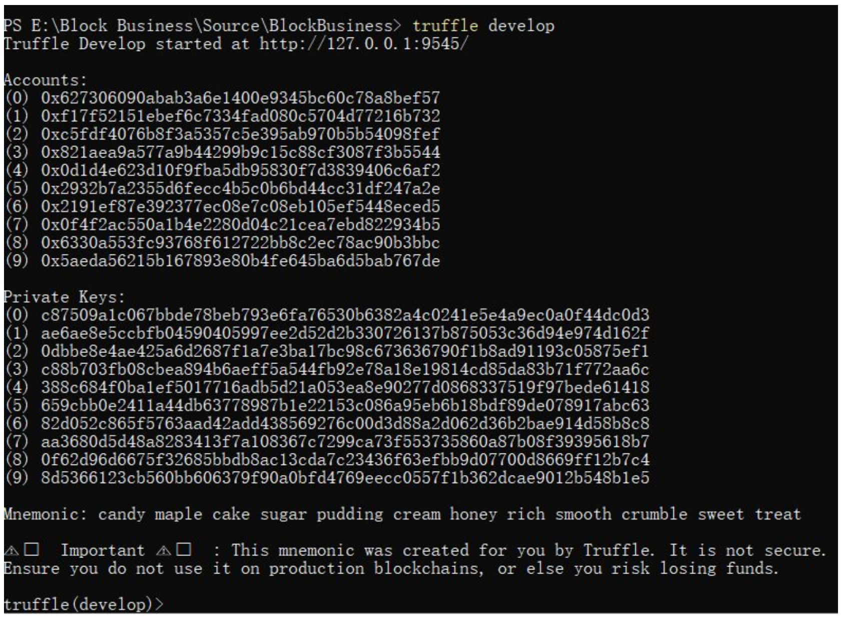
Finally, click RESTART at top-right to restart Ganache.

4) Compile Smart Contract

Start the terminal from root and run:

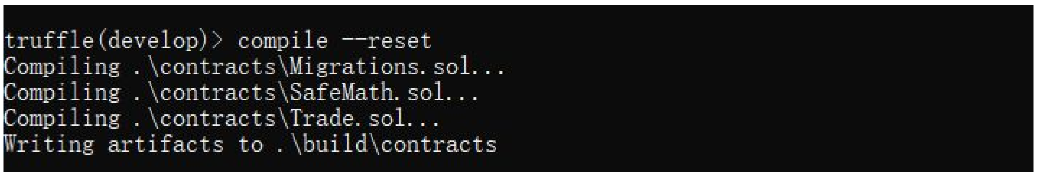
truffle develop

You should see the following message:



```
PS E:\Block Business\Source\BlockBusiness> truffle develop
Truffle Develop started at http://127.0.0.1:9545/

Accounts:
(0) 0x627306090abab3a6e1400e9345bc60c78a8bef57
(1) 0xf17f52151ebef6c7334fad080c5704d77216b732
(2) 0xc5fdf4076b8f3a5357c5e395ab970b5b54098fef
(3) 0x821aea9a577a9b44299b9c15c88cf3087f3b5544
(4) 0x0d1d4e623d10f9fba5db95830f7d3839406c6af2
(5) 0x2932b7a2355d6fecc4b5c0b6bd44cc31df247a2e
(6) 0x2191ef87e392377ec08e7c08eb105ef5448eced5
(7) 0x0f4f2ac550a1b4e2280d04c21cea7ebd822934b5
(8) 0x6330a553fc93768f612722bb8c2ec78ac90b3bbc
(9) 0x5aeda56215b167893e80b4fe645ba6d5bab767de

Private Keys:
(0) c87509a1c067bbde78beb793e6fa76530b6382a4c0241e5e4a9ec0a0f44dc0d3
(1) ae6ae8e5ccbfb04590405997ee2d52d2b330726137b875053c36d94e974d162f
(2) 0dbbe8e4ae425a6d2687f1a7e3ba17bc98c673636790f1b8ad91193c05875ef1
(3) c88b703fb08cbea894b6aeff5a544fb92e78a18e19814cd85da83b71f772aa6c
(4) 388c684f0ba1ef5017716adb5d21a053ea8e90277d0868337519f97bede61418
(5) 659cbb0e2411a44db63778987b1e22153c086a95eb6b18bdf89de078917abc63
(6) 82d052c865f5763aad42add438569276c00d3d88a2d062d36b2bae914d58b8c8
(7) aa3680d5d48a8283413f7a108367c7299ca73f553735860a87b08f39395618b7
(8) 0f62d96d6675f32685bbdb8ac13cda7c23436f63efbb9d07700d8669ff12b7c4
(9) 8d5366123cb560bb606379f90a0bfd4769eecc0557f1b362dcae9012b548b1e5

Mnemonic: candy maple cake sugar pudding cream honey rich smooth crumble sweet treat

⚠☐  Important ⚠☐  : This mnemonic was created for you by Truffle. It is not secure.
Ensure you do not use it on production blockchains, or else you risk losing funds.

truffle(develop)>
```

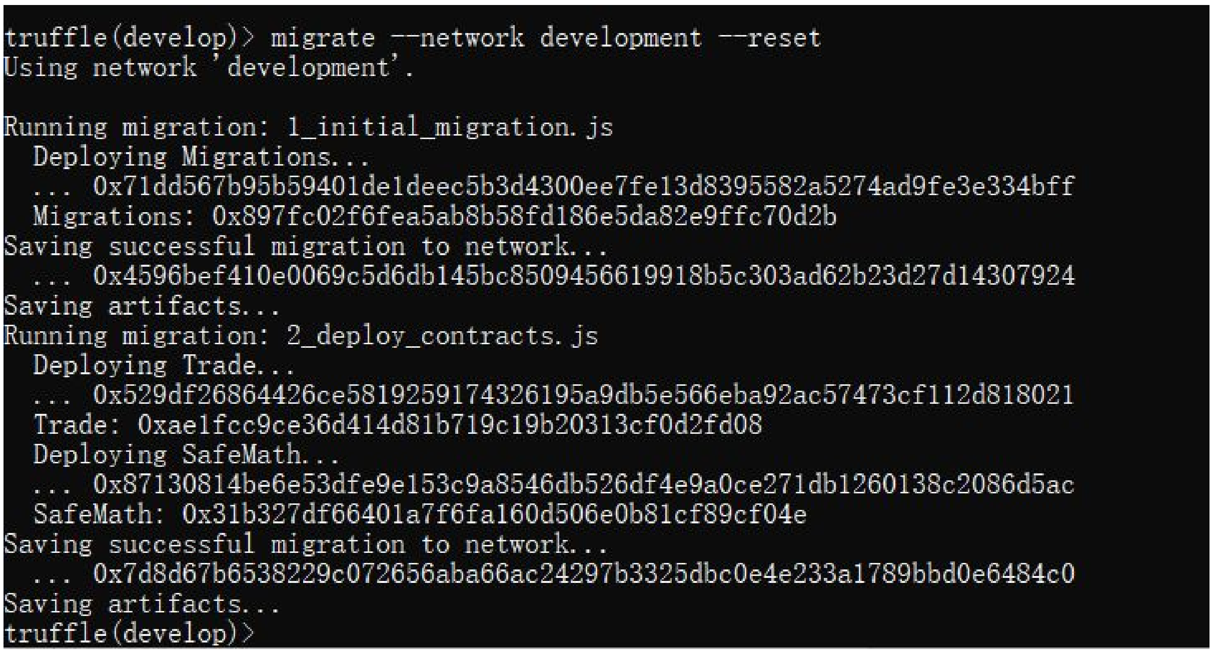
Run: **compile --reset**



```
truffle(develop)> compile --reset
Compiling .\contracts\Migrations.sol...
Compiling .\contracts\SafeMath.sol...
Compiling .\contracts\Trade.sol...
Writing artifacts to .\build\contracts
```

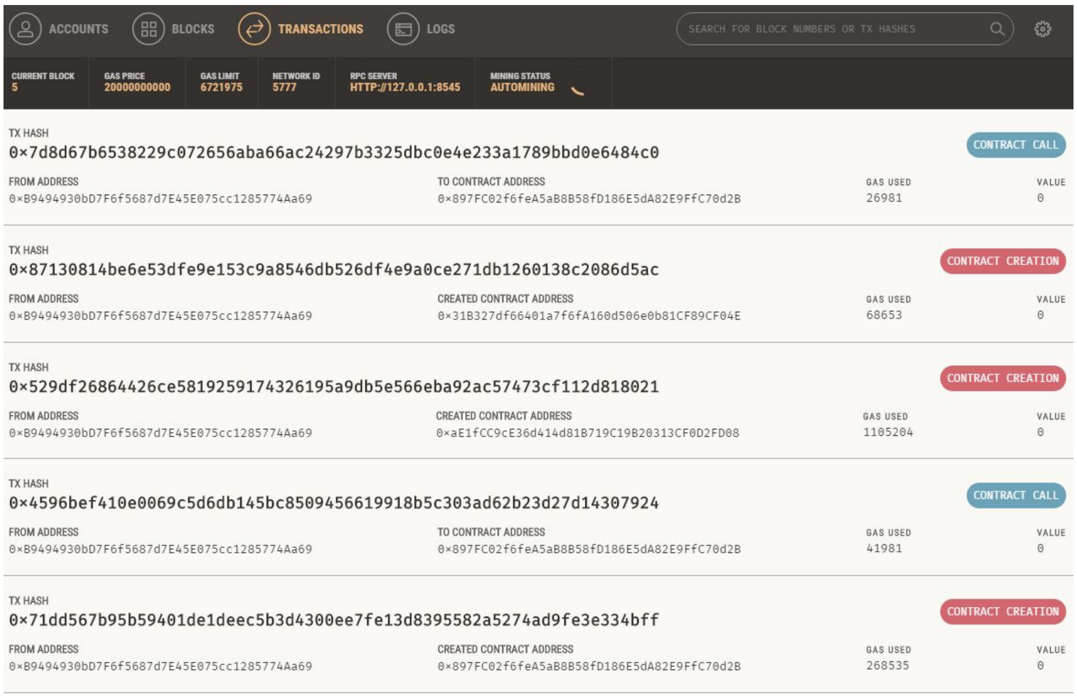
This will make truffle to compile all Solidity code in Contract directory. When it is done, a new directory named **build/contracts** will be generated in root path. All compiled contracts will be placed here in **json** format.

Run: **migrate --network development --reset**



```
truffle(develop)> migrate --network development --reset
Using network 'development'.

Running migration: 1_initial_migration.js
  Deploying Migrations...
  ... 0x71dd567b95b59401de1deec5b3d4300ee7fe13d8395582a5274ad9fe3e334bff
  Migrations: 0x897fc02f6fea5ab8b58fd186e5da82e9ffc70d2b
Saving successful migration to network...
  ... 0x4596bef410e0069c5d6db145bc8509456619918b5c303ad62b23d27d14307924
Saving artifacts...
Running migration: 2_deploy_contracts.js
  Deploying Trade...
  ... 0x529df26864426ce5819259174326195a9db5e566eba92ac57473cf112d818021
  Trade: 0xae1fcc9ce36d414d81b719c19b20313cf0d2fd08
  Deploying SafeMath...
  ... 0x87130814be6e53dfe9e153c9a8546db526df4e9a0ce271db1260138c2086d5ac
  SafeMath: 0x31b327df66401a7f6fa160d506e0b81cf89cf04e
Saving successful migration to network...
  ... 0x7d8d67b6538229c072656aba66ac24297b3325dbc0e4e233a1789bbd0e6484c0
Saving artifacts...
truffle(develop)>
```

This will migrate the smart contract defined in Solidity onto Ganache.
You shall see the details showing the deployment of smart contract on Ganache from the interface of Ganache in the Transaction page:



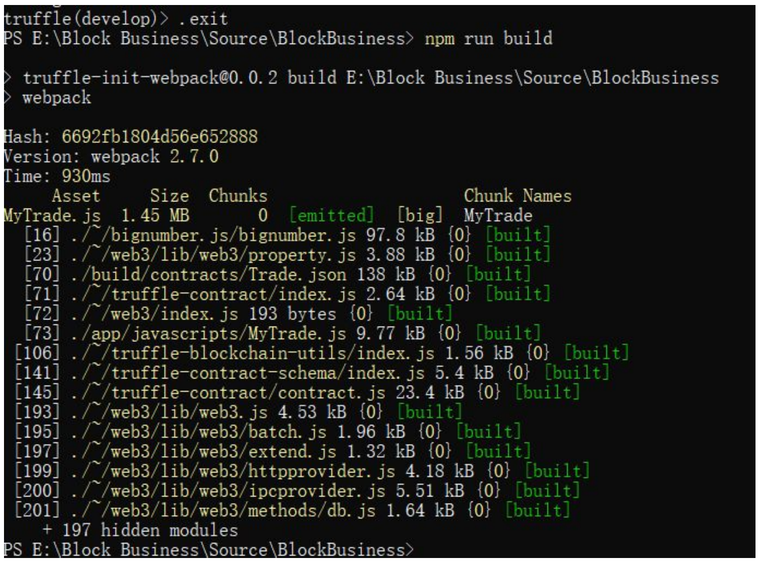
```
CURRENT BLOCK  GAS PRICE    GAS LIMIT  NETWORK ID  RPC SERVER             MINING STATUS
5              20000000000  6721975    5777        HTTP://127.0.0.1:8545  AUTOMINING

TX HASH                                                                                   CONTRACT CALL
0×7d8d67b6538229c072656aba66ac24297b3325dbc0e4e233a1789bbd0e6484c0
FROM ADDRESS                                  TO CONTRACT ADDRESS                          GAS USED   VALUE
0×B9494930bD7F6f5687d7E45E075cc1285774Aa69    0×897FC02f6feA5aB8B58fD186E5dA82E9FfC70d2B   26981      0

TX HASH                                                                                   CONTRACT CREATION
0×87130814be6e53dfe9e153c9a8546db526df4e9a0ce271db1260138c2086d5ac
FROM ADDRESS                                  CREATED CONTRACT ADDRESS                     GAS USED   VALUE
0×B9494930bD7F6f5687d7E45E075cc1285774Aa69    0×31B327df66401a7f6fA160d506e0b81CF89CF04E   68653      0

TX HASH                                                                                   CONTRACT CREATION
0×529df26864426ce5819259174326195a9db5e566eba92ac57473cf112d818021
FROM ADDRESS                                  CREATED CONTRACT ADDRESS                     GAS USED   VALUE
0×B9494930bD7F6f5687d7E45E075cc1285774Aa69    0×aE1fCC9cE36d414d81B719C19B20313CF0D2FD08   1105204    0

TX HASH                                                                                   CONTRACT CALL
0×4596bef410e0069c5d6db145bc8509456619918b5c303ad62b23d27d14307924
FROM ADDRESS                                  TO CONTRACT ADDRESS                          GAS USED   VALUE
0×B9494930bD7F6f5687d7E45E075cc1285774Aa69    0×897FC02f6feA5aB8B58fD186E5dA82E9FfC70d2B   41981      0

TX HASH                                                                                   CONTRACT CREATION
0×71dd567b95b59401de1deec5b3d4300ee7fe13d8395582a5274ad9fe3e334bff
FROM ADDRESS                                  CREATED CONTRACT ADDRESS                     GAS USED   VALUE
0×B9494930bD7F6f5687d7E45E075cc1285774Aa69    0×897FC02f6feA5aB8B58fD186E5dA82E9FfC70d2B   268535     0
```

At last, run: **.exit**
You will exit from truffle command-line and back to the root path.

5) Package files with Webpack

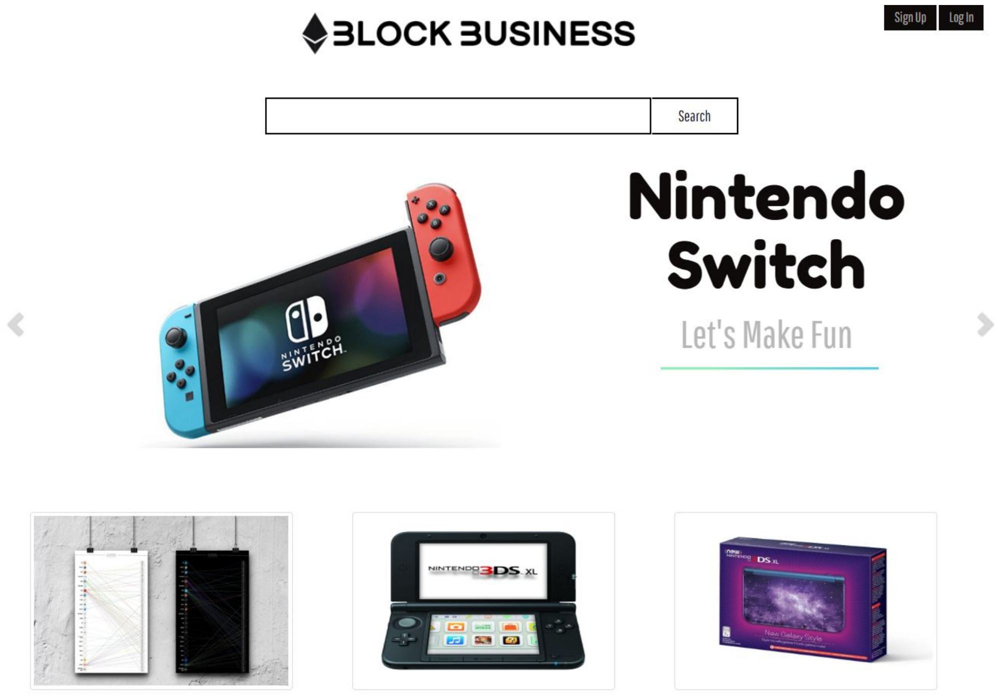
After all steps above, the smart contract is ready to use and we need to include them into a Javascript module for webpage to use.

Run: **npm run build**



```
truffle(develop)> .exit
PS E:\Block Business\Source\BlockBusiness> npm run build

> truffle-init-webpack@0.0.2 build E:\Block Business\Source\BlockBusiness
> webpack

Hash: 6692fb1804d56e652888
Version: webpack 2.7.0
Time: 930ms
     Asset     Size  Chunks                    Chunk Names
MyTrade.js  1.45 MB       0  [emitted]  [big]  MyTrade
  [16] ./~/bignumber.js/bignumber.js 97.8 kB {0} [built]
  [23] ./~/web3/lib/web3/property.js 3.88 kB {0} [built]
  [70] ./build/contracts/Trade.json 138 kB {0} [built]
  [71] ./~/truffle-contract/index.js 2.64 kB {0} [built]
  [72] ./~/web3/index.js 193 bytes {0} [built]
  [73] ./app/javascripts/MyTrade.js 9.77 kB {0} [built]
 [106] ./~/truffle-blockchain-utils/index.js 1.56 kB {0} [built]
 [141] ./~/truffle-contract-schema/index.js 5.4 kB {0} [built]
 [145] ./~/truffle-contract/contract.js 23.4 kB {0} [built]
 [193] ./~/web3/lib/web3.js 4.53 kB {0} [built]
 [195] ./~/web3/lib/web3/batch.js 1.96 kB {0} [built]
 [197] ./~/web3/lib/web3/extend.js 1.32 kB {0} [built]
 [199] ./~/web3/lib/web3/httpprovider.js 4.18 kB {0} [built]
 [200] ./~/web3/lib/web3/ipcprovider.js 5.51 kB {0} [built]
 [201] ./~/web3/lib/web3/methods/db.js 1.64 kB {0} [built]
    + 197 hidden modules
PS E:\Block Business\Source\BlockBusiness>
```

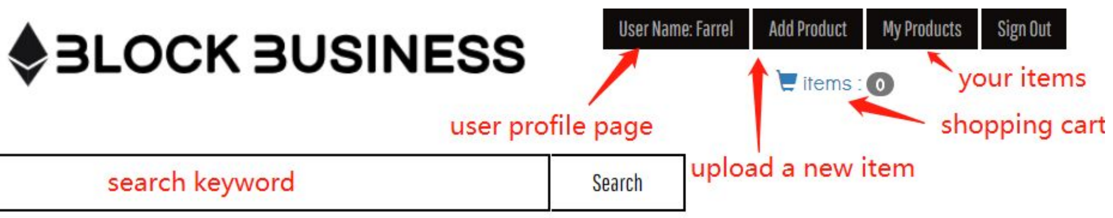
6) Start http://localhost:3000 with browser and you shall see the front page of the project.



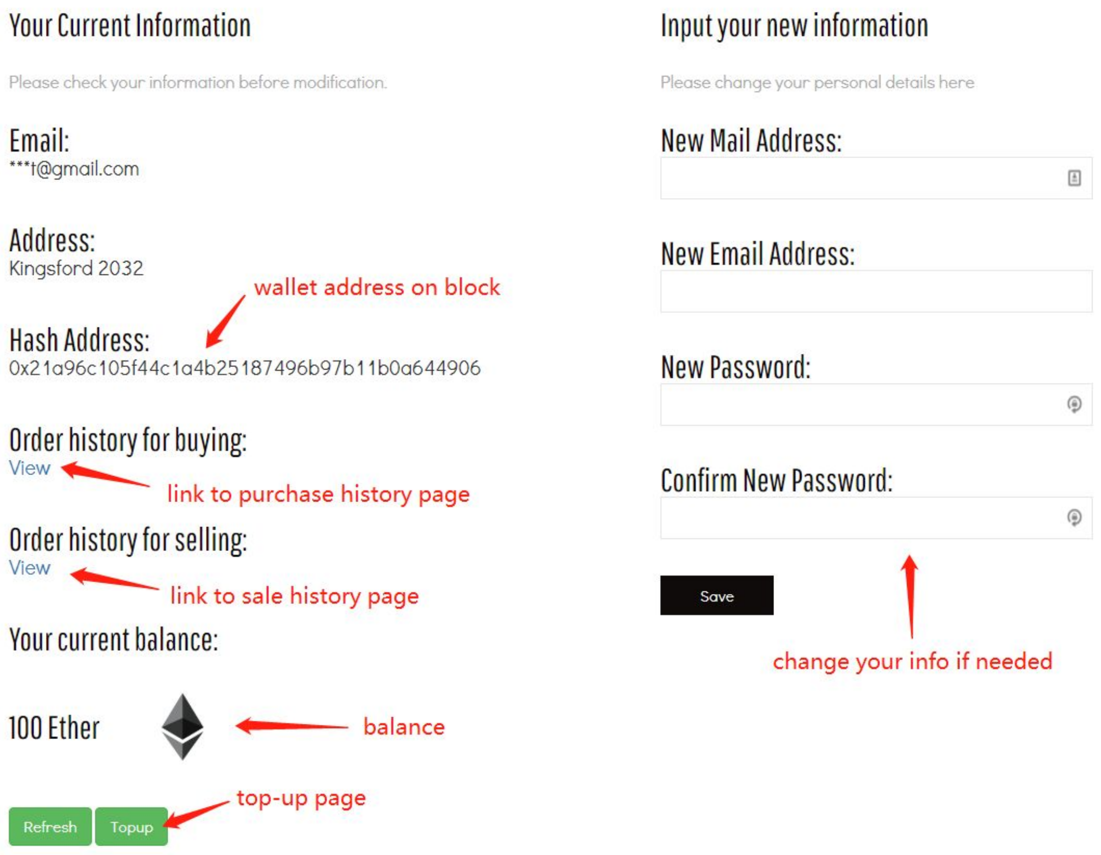
2. Guide for site

After setting up the project, it is easy to use the website just as a normal online shopping cart. All functions provided has been introduced in functionality section and all of them are easy to access from page interface.
You can use the following test users to login, or simply register a new user as you wish.

Name	Password
Adam	123
Jame	123
Farrel	123
Melody	111
Kate	111



The header of index page is as shown above. All functions are obvious to see and use.
The profile page of user is like below.


Your Current Information

Please check your information before modification.

Email:
***t@gmail.com

Address:
Kingsford 2032

Hash Address:
0x21a96c105f44c1a4b25187496b97b11b0a644906

Order history for buying:
View

Order history for selling:
View

Your current balance:

100 Ether

Input your new information

Please change your personal details here

New Mail Address:

New Email Address:

New Password:

Confirm New Password:

wallet address on block
link to purchase history page
link to sale history page
balance
top-up page
Refresh
Topup
Save
change your info if needed

Reference:

1. Bootstrap https://getbootstrap.com/docs/3.3/css/
2. Jquery Cheat Sheet https://oscarotero.com/jquery/
3. Cloudinary API https://cloudinary.com/documentation
4. Mlab online Mongodb https://mlab.com/
5. Truffle https://github.com/trufflesuite/truffle
6. Truffle-box http://truffleframework.com/boxes/webpack
7. Web3 0.2x https://github.com/ethereum/wiki/wiki/JavaScript-API
8. Express https://www.tutorialspoint.com/nodejs/nodejs_express_framework.htm
9. Loading page module Please-wait https://github.com/Pathgather/please-wait
10. Spinkit for animated icon https://github.com/tobiasahlin/SpinKit
11. Experiment on Remix https://remix.ethereum.org/#optimize=false&version=soljson-v0.4.21+commit.dfe3193c.js
12. Solidity tutorial https://ethereumbuilders.gitbooks.io/guide/content/en/solidity_tutorials.html
13. Solidity online course https://coursetro.com/courses/20/Developing-Ethereum-Smart-Contracts-for-Beginners
14. Webpack tutorial https://webpack.js.org/guides/getting-started/