

Barter-X Documentation



Project Domain: Private Blockchain (E-Commerce)

Start Date: 6 th May 2020

End Date: 31 st August 2020

Mentors:

- Vishwesh Mirajkar (May - June)
- Riya Agarwal (July - August)

Abstract: Barter-x is a decentralized platform to barter trade your personal assets online more transparently. It is a Hyperledger Fabric based implementation.

FRONTEND

BARTERX WEB APPLICATION UI

Technologies Used: ReactJS, Redux, Bootstrap, React-Bootstrap, ES6, HTML, CSS

- React.Js -

ReactJS is an open-source JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. However, React is only concerned with rendering data to the DOM, and

so creating React applications usually requires the use of additional libraries for state management and routing. Redux and React Router are respective examples of such libraries.

How to install React on PC -Link - https://youtu.be/IbWXHfz91_Y

- HTML -

Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

- CSS -

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like

HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts.[3] This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file which reduces complexity and repetition in the structural content as well as enabling the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.

- React Bootstrap -

Bootstrap is a free and open-source CSS framework directed at

responsive, mobile-first front-end web development. It contains CSS and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components. React-Bootstrap replaces the Bootstrap JavaScript. Each component has been built from scratch as a true React component, without unneeded dependencies like jQuery. As one of the oldest React libraries, React-Bootstrap has evolved and grown alongside React, making it an excellent choice as your UI Foundation.

- Redux -

Redux is a predictable state container for JavaScript apps. It helps you write applications that behave consistently, run in different environments (client, server, and native), and are easy to test. On top of that, it provides a great developer experience, such as live code editing combined with a time traveling debugger. You can use Redux together with React, or with any other view library.

How to Install NodeJS and NPM on Windows 10 -

Link - https://youtu.be/X-FPCwZFU_8

To create a react app :

Use the following commands to create a React App.

1. `npx create-react-app my-app-name`
2. `cd my-app-name`
3. `npm start`

npm start: Runs the react application.

How To Run The Project:

- Before running the project make sure git and node is installed in your system.

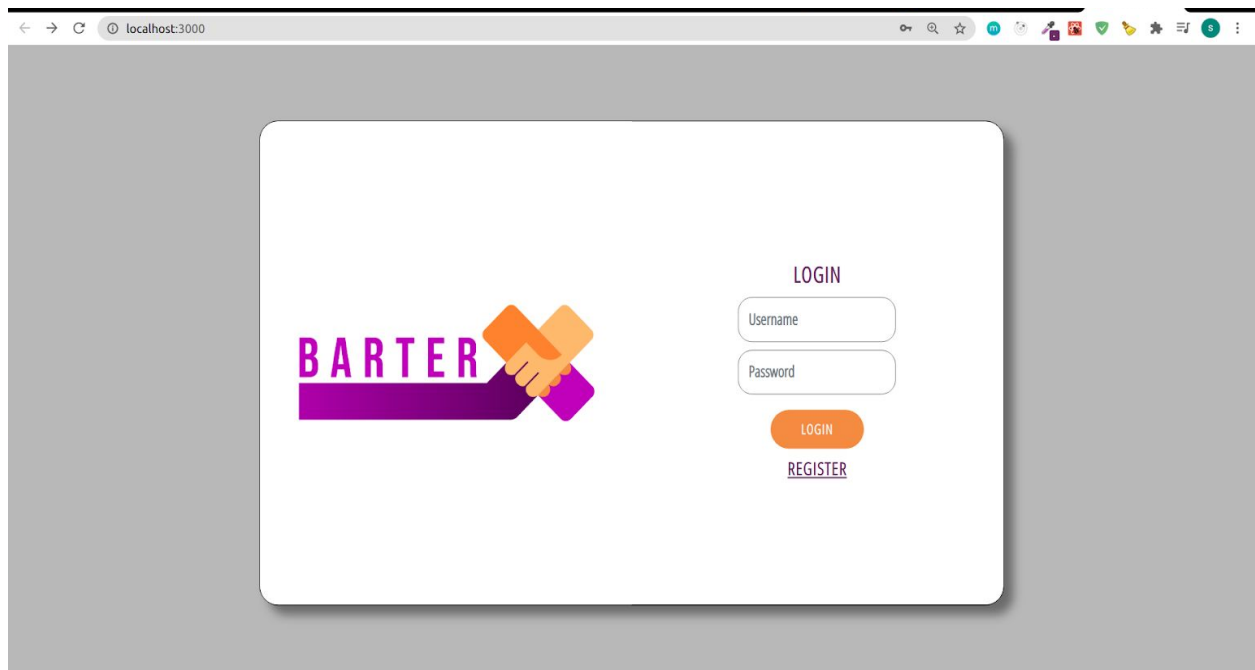
- Open your terminal/git bash and clone the new-theme branch from [//github.com/innobytes-net/BarterX-web-app.git](https://github.com/innobytes-net/BarterX-web-app.git) or Copy this: git command git clone -b new theme [//github.com/innobytes-net/BarterX-web-app.git](https://github.com/innobytes-net/BarterX-web-app.git) and paste in your terminal/git bash.
- Now install all the repositories which are used in the project.
- Change directory to cd BarterX-web-app and type npm install to install all your repositories.
- After installing all the repositories, run the project by running servers.
- Open another terminal and run npm start, to run the server on localhost:3000

Flow of the Website:

- Users:
 - Login
 - Sign up and create wallet
 - View products
 - Barter/Exchange Products
 - Add New products
 - Delete Products
 - Update Products
 - Check Transactions
 - Buy/Barter Products

Website Pages Screenshots:


Login Page:



Sign Page:

localhost:3000/register

Wallet created successfully




SIGN UP


☒ INDIVIDUAL ☐ ORGANISATION

SIGN UP


Profile:

localhost:3000/profile



Exchange  All Products

View Profile Edit Profile Wallet



Username

Organisation Name:


Affiliation Name:

Email:

localhost:3000/profile

BARTER Exchange All Products

View Profile Edit Profile Wallet



Username
mesaki

Organisation Name:
department1

Affiliation Name:
org1

Email:
d@gmail.com

Save Discard

Add New Products Page:

localhost:3000/myproducts

BARTER Exchange All Products

Add Product Update/Delete Product Product History

Fill in the details:

Product ID: 112P748	Product Name: Samsung Phone
Category: Electronics	Status: True
Owner: Srushiti	Price: 15000

Reset Add Product

PRODUCT ADDED

Edit New Products Page:

localhost:3000/myproducts

Add Product Update/Delete Product Product History

Enter product Id to delete/update :

112P748 Get Information

Click Edit to fill in the details and Update the information :

112P748

ID 112P748

Name Samsung Phone

Availability true

Category Electronics

Owner Srushti

Price 15000

Edit Reset Update Product Delete

Product History Page:

localhost:3000/myproducts

BARTER Exchange All Products

Add Product Update/Delete Product Product History

Enter product Id :

112P748 Get Information

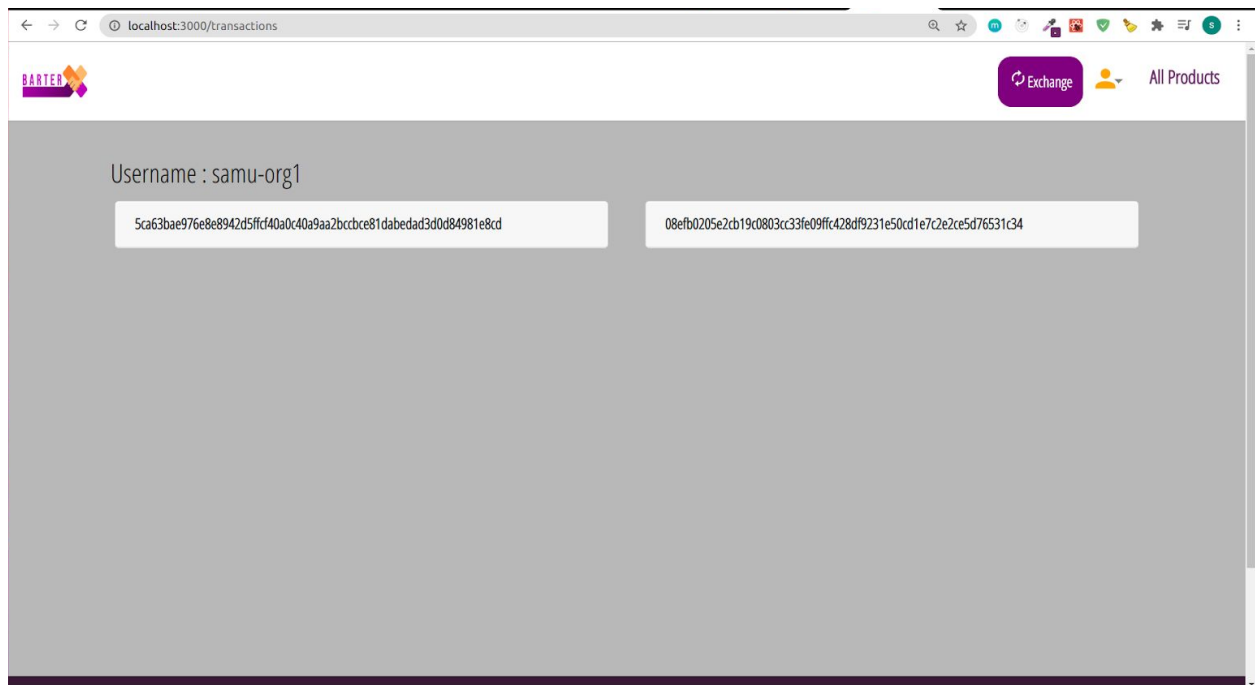
112P748
Name: Samsung Phone

Owner Name : Srushti
Category : Electronics
Status : true
Owner Name : Srushti
Price : 20000
Timestamp : 2020-09-04 13:11:34.242 +0000 UTC
Is Delete : false

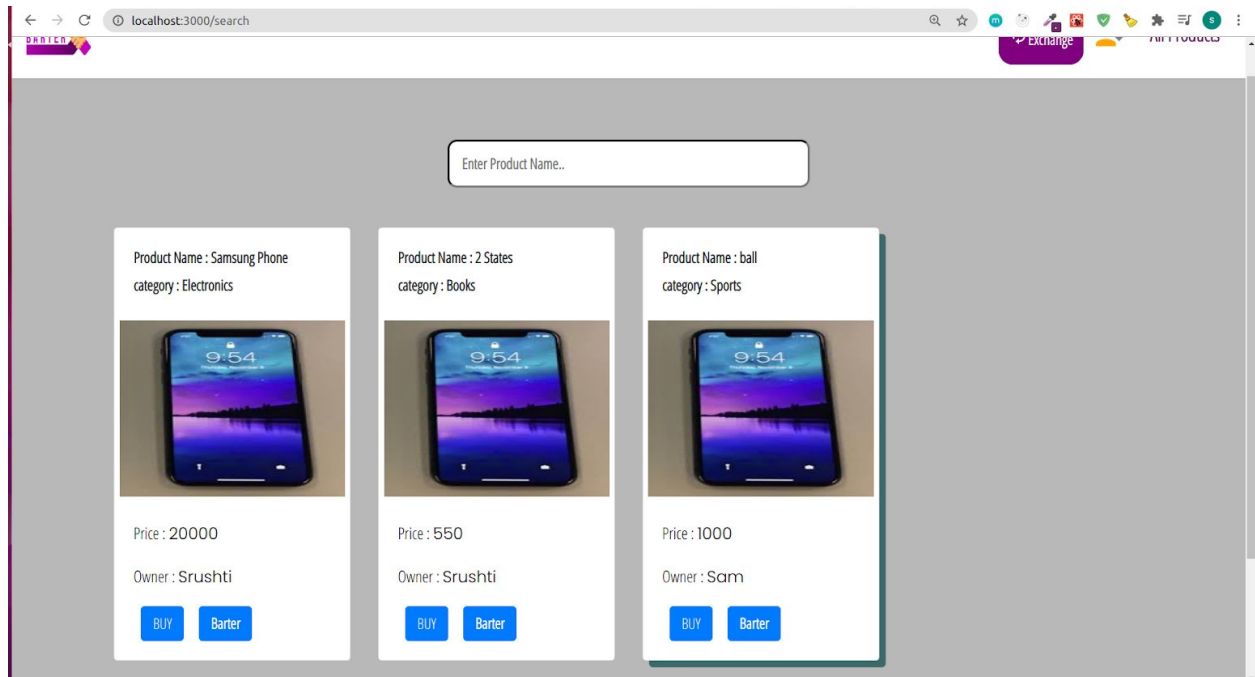
112P748
Name: Samsung Phone

Owner Name : Srushti
Category : Electronics
Status : true
Owner Name : Srushti
Price : 15000
Timestamp : 2020-09-04 13:09:05.313 +0000 UTC
Is Delete : false

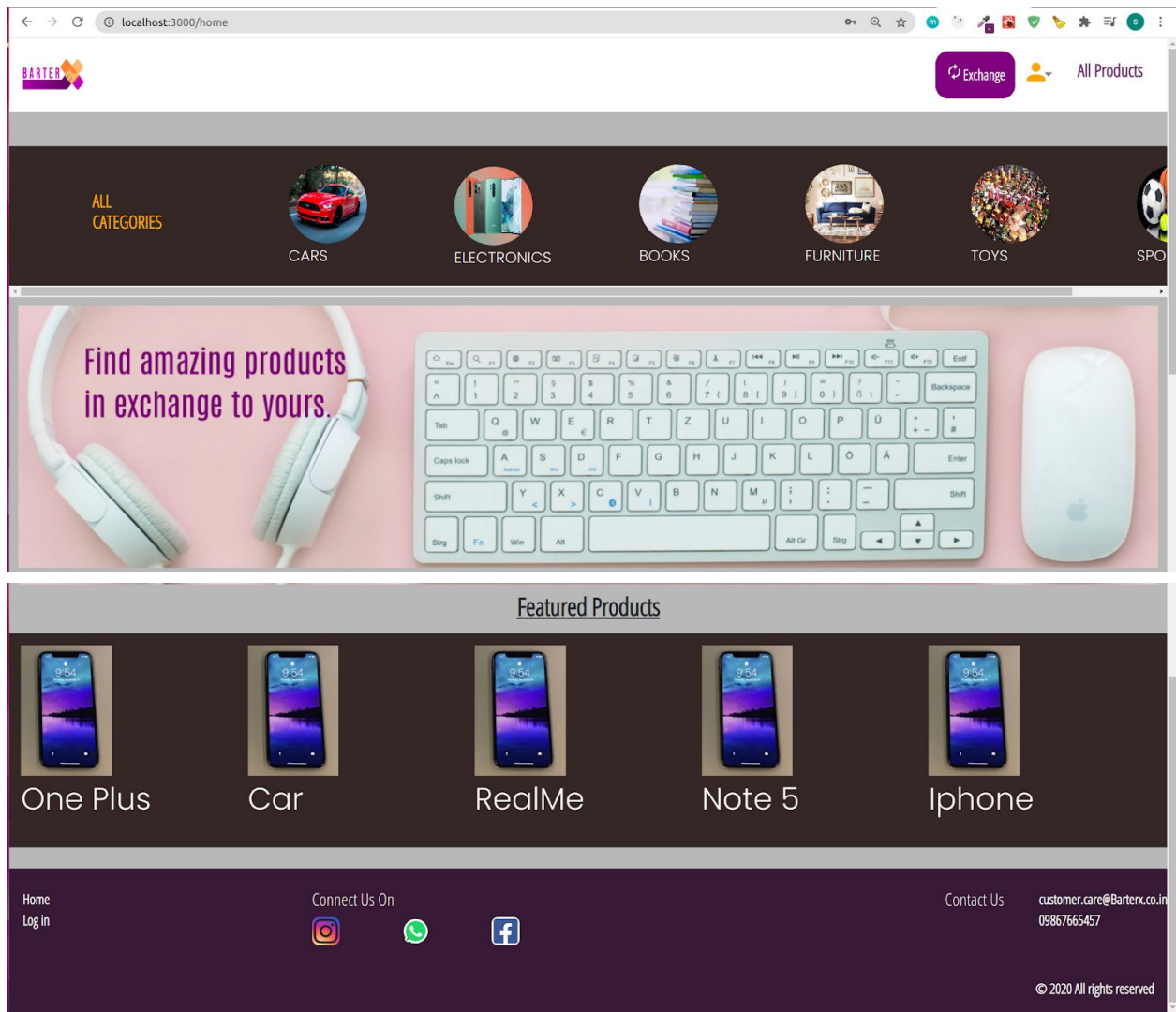
User Transactions:



All Products Page:



Home Page:



Chat/Negotiations:

localhost:9090/chat.html?username=srush

Your socket ID: CxL1LaJ_4tQAsbL2AAAA

Send To


Your Product


Customer Product

Doc HASH

Send

localhost:3000/barter

 [Exchange](#) [All Products](#)

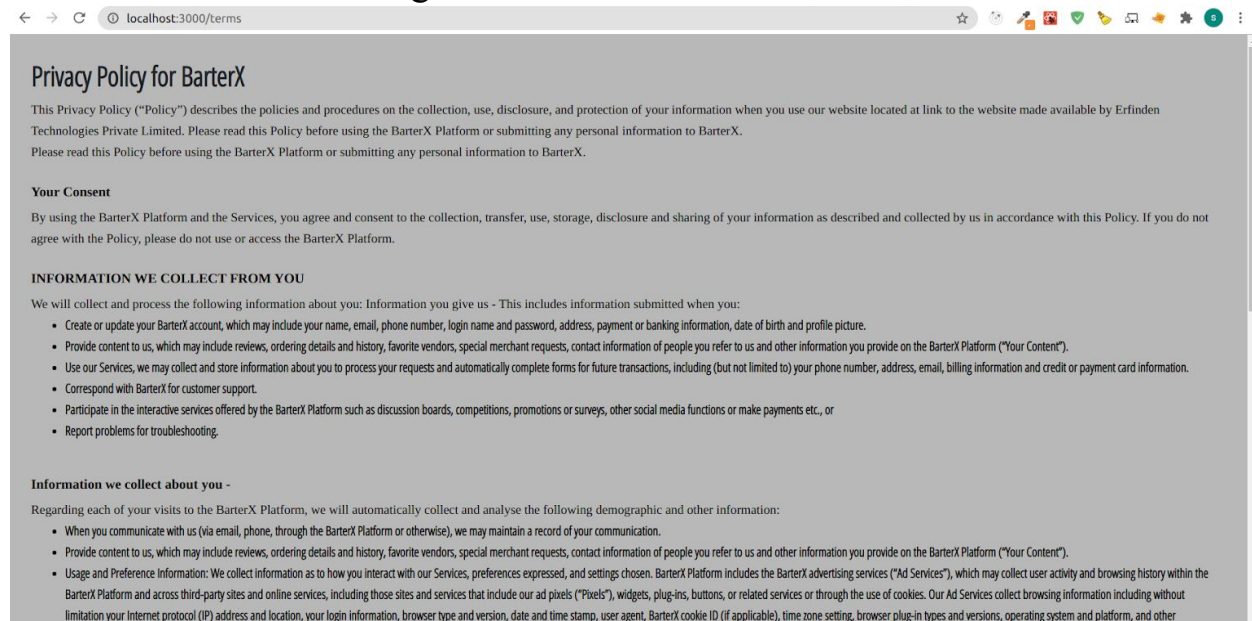
 ChatCord

Username

Join Chat

Buy

Terms and Conditions Page:



Steps for Database Access:

Couchdb:

1. Install couchdb
2. To check the database and its entries: http://127.0.0.1:5984/_utils/
3. Refer <https://www.tutorialkart.com/couchdb-tutorial/couchdb-get-database-list/> for more information

MongoDB:

1. Install mongodb on machine
2. Start the mongodb server
3. Command to check database: show databases
4. Command to check Collections: show collections
5. For more information refer to: <https://docs.mongodb.com/>

Steps to integrate UI pages with Backend:

1. Create UI page/component
2. To route the url to particular page add a Route in App.js
3. To Connect the pages with backend apis follow the following steps:
 - a. Create a function in store/actions/ .js file
 - b. Create a reducer function in store/reducers/ .js file

- c. Call the function created in actions from the UI page as and where necessary
- d. Use redux store to store the data retrieved from the backend
- e. For more information refer:
<https://redux.js.org/basics/basic-tutorial>

References:

- 1. <https://reactjs.org/docs/getting-started.html>
- 2. <https://getbootstrap.com/docs/4.5/getting-started/introduction/>
- 3. <https://docs.github.com/en>
- 4. <https://redux.js.org/basics/basic-tutorial>
- 5. <https://www.w3schools.com/>
- 6. <https://docs.npmjs.com/>

BACKEND-FUNCTIONALITIES

1. Chaincode functions:

1.1. Create Product:

1.1.1. Arguments

`"{"Args":["Create","productID","name","category","available","Owner","Price"]}"`

1.1.2. Description:

Creates a product on the world state in line with the product structure defined in chaincode. Corresponding api call is /products/store

1.2. Read :

1.2.1. Arguments:

`"{"Args":["Read2","key"]}"`

- 1.2.2. Description:
Performs a universal read on all types of structure instances, invokes the Read function of chaincode

1.3. Update:

- 1.3.1. Arguments:
`"{"Args":["Update2","key","index","value"]}"`
- 1.3.2. Definition:
This calls the Update function defined in chaincode and updates the index of the corresponding key value pair retrieved using the "key" argument. The value of index is updated to the argument "value".Api call
`/products/update`

1.4. Create Wallet:

- 1.4.1. Arguments:
`"{"Args":["Create_wall","wallet_id","owner_id","balance"]}"`
- 1.4.2. Description:
Called inside the createWallet api call ,it creates a an account based wallet for new user

1.5. Final :

- 1.5.1. Arguments:
`"{"Args":["Final","ProductID1","ProductID2"]}"`
- 1.5.2. Description:
Retrieves account based wallet Ids of the owners associated with this product.Calculates the difference between the products as cost of barter. Calls the approve function of the ERC20 transaction flow to validate transaction, finally the 'Transfer' function is called to perform transfer of tokens to beneficiary.Api call is
`/products/transfer`

1.6. User Products:

1.6.1. Arguments:

`\{"Args":["Rich","UserID1"]}\`

1.6.2. Description:

Requires couchDB containers and performs a rich query to retrieve products owned by the user. API call is /products/userproducts

1.7. Products by Category:

1.7.1. Arguments:

`\{"Args":["Category","Category_name"]}\`

1.7.2. Description:

Returns all products belonging to the entered category (Like FMCG, electronics, automobiles)

1.8. Index:

1.8.1. Arguments:

`\{"Args":["Index","startIndex","endIndex"]}\`

1.8.2. Description:

Retrieves all products by product IDs lexically whose ProductIDs are between [startIndex, endIndex). Can be set to default at P0 and max no. of products on the server side. API call is /products/index

1.9. Get product history

1.9.1 Arguments:

`\{"Args":["Read2","key"]}\`

1.9.2 Description:

Performs a universal read on all types of structure instances, invokes the Read function of chaincode

2. Tokenisation:

The ERC20 standard is followed for the tokenisation . The BalanceOf function of ERC20 is renamed to Read which has a multipurpose read role. The Totalsupply function retrieves the token balance of the token owner. The balance of owner can be increased by calling the Create_tok function(api call /buy/init) with required parameters or by writing a new function that increases the Totalsupply (performs the function of mining)

The transfer function and allow function emit events on the chaincode side which are caught by the event listener function defined inside event.js file . These events can serve as real time notifications to the users

Three of the eight ERC20 functions are described below:

2.1. Allow:

2.1.1. Arguments:

`"{"Args":["Allow","wallet_id", "amount"]}"`

2.1.2. Description:

Defined in the chaincode this function allows user to buy tokens. It is recommended that this function be encapsulated inside a payment gateway. Api call is /buy/tokens

2.2. Approve:

2.2.1. Arguments:

`"{"Args":["Approve","wallet_id", "amount"]}"`

2.2.2. Description:

Approves the transaction request submitted by the passed walletId to carry out a transfer of tokens with the passed cost of transaction. Returns a boolean value

2.3. Create_tok:

2.3.1. Arguments:

```
"{"Args":["Create_tok","symbol", "totalSupply",  
"description", "creator"]}"
```

2.3.2. Description:

Creates a token with the signature of token owner and initialises the Totalsupply attribute . Api call is /buy/init

2.4. Transfer:

2.4.1. Arguments:

```
"{"Args":["Transfer","from", "to", "amount"]}"
```

2.4.2. Description:

Transfers tokens from one wallet Id to another , it is called inside the 'Final' function of the chaincode after the validation recommended by the ERC20 transaction flow is carried out.

References:

NOTE: The videos in the link demonstrates the version 1.4 installation of Hyperledger Fabric and NOT 2.0.

(Make the changes accordingly while installation or else it gives version mismatch error)

1. To understand the basic concepts of blockchain
<https://www.youtube.com/watch?v=UqQMSVfugFA&list=PLsyebzWxl7oY6tZmnZ5S7yTDxyu4zDW->
2. Installation of Hyperledger fabric with all prerequisites and running the first network
https://www.youtube.com/watch?v=W8OR_MBTRO0&list=PLnC4m7jmtnwgcgUvo_wqEzOpDhnHGQAaD4

3. Explanation to BYFN
<https://www.youtube.com/watch?v=rtuwuo1dcTY>
4. Explanation for Basic architecture of Hyperledger Fabric(No need to watch the entire playlist)
https://www.youtube.com/watch?v=TOlv1nRDdfs&list=PLyqSpQzTE6M8wy_JBTgplS_HGuOYU1qkm
5. Building your own first network from scratch(when to write the code)
https://www.youtube.com/watch?v=MPNkUqOKhVE&list=PLjsqymUqgpSTGC4L6ULHC_B_Mqmy43Oclh
6. The Hyperledger fabric Documentation
<https://hyperledger-fabric.readthedocs.io/en/release-2.0/>
7. This sort of explains the role of agent in Hyperledger Cello(which gets up the nodes running)
<https://www.youtube.com/watch?v=hSoVLmkH74Q&feature=youtu.be>
8. This article addresses the currency conversion problem that can be bypassed by bartering blockchains
<https://medium.com/@mintdice/bartering-issues-solved-by-blockchain-e51f18f4e4a6>
9. Difference between OLX type exchange services and services based on blockchain
<https://yourstory.com/mystory/is-blockchain-the-future-of-c2c-marketplaces-q456c7tgsr>
10. The blockchain startup MYTC for trading
<https://steemit.com/cryptocurrencies/@kored/mytc-is-a-decentralized-platform-based-on-blockchain-technology>
<https://medium.com/@jonsen484848/mytc-trading-platform-using-blockchain-e587550596dd>
11. BarterTrade white paper
<https://bartertrade.io/whitepaper.pdf>
12. Kubernetes and hyperledger fabric implementation
<https://github.com/IBM/android-kubernetes-blockchain>
13. Understanding the step by step processes in Hyperledger Fabric (about all the peers, ordering service, channels, the network, etc)
https://youtu.be/U_0X11A16ts

14. Account based wallet chaincode in Java
<https://medium.com/coinmonks/hyperledger-fabric-account-based-wallet-java-chaincode-8cbf80a6fb82>
15. "System Chaincodes in Hyperledger Fabric — VSCC, ESCC, LSCC, CSCC" by Pavan Adhav <https://link.medium.com/Xq6Ug0GXt6>
16. What's new in fabric 2.0 <https://www.youtube.com/watch?v=HkX-JpwPjP4&t=1923s>
17. Amazon workshop. Building and deploying an application for Hyperledger Fabric on Amazon Managed Blockchain
<https://github.com/aws-samples/non-profit-blockchain>
18. Fabcar (medium)
<https://medium.com/@kctheservant/deep-dive-into-fabcar-revised-57cb54642572>
19. Writing Chaincode in Go
<https://github.com/hyperledger/fabric-contract-api-go/blob/master/tutorials/getting-started.md>
20. **Tutorial Golang** relevant videos are **1 to 8,12,14,18 to 21**
<https://www.youtube.com/watch?v=nSYFfWijl8U&list=PLQVvva0QuDeF3hP0wQoSxpkqgRcgxMqX>
21. FabTokens github
<https://github.com/hyperledger/fabric-samples/tree/v2.0.0-alpha/fabtoken>
22. difference-between-node-peer-and-user-in-hyperledger-fabric
<https://stackoverflow.com/questions/60305387/difference-between-node-peer-and-user-in-hyperledger-fabric>
23. in-hyperledger-fabric-what-is-difference-between-org1msp-member-org1msp-peer
<https://stackoverflow.com/questions/54419670/in-hyperledger-fabric-what-is-difference-between-org1msp-member-org1msp-pee/54423840>
24. Hyperledger Fabric 2.0 series
<https://www.youtube.com/watch?v=SJTdJt6N6Ow&list=PLSBNVhWU6KjW4qo1RIImR7cVvV8XIIILub6>

