

# Programming assignment#3. Hyperledger Fabric

한양대학교 컴퓨터 소프트웨어 전공  
2014004066 서왕규

본 문서는 분산 컴퓨팅의 하이퍼레저 패브릭 과제에 대한 정보를 담고 있다. 1절에서는 실행과 환경, 2절은 주요 기능, 3절은 소프트웨어 디자인, 4절은 시나리오 수행과 결과를 나타낸다.

## 1. Develop Enviroment and setting

### 1.1 Enviroment

개발은 다음과 같은 환경에서 진행되었다.

```
npm version : 6.14.8  
node version : 10.23.0  
openjdk version : 1.8.0_275
```

과제의 주요 코드는 아래 파일에서 작성되었습니다.

```
chaincode : fabcar.java(=TradeItem.java 파일의 명칭만 변경했습니다.)  
index.ejs, routes/index.js  
enrollAdmin.js, invoke.js, query.js, registerUser.js : sdk code  
public/javascripts/jsfunc.js
```

### 1.2 Setting

○ add to ~/.profile

```
PATH="$HOME/bin:$HOME/.local/bin:$HOME/fabric_project_interface/bin:$PATH"  
  
export FABRIC_CFG_PATH=$HOME/fabric_project_interface/config/  
export ORDERER_CA=$HOME/fabric_project_interface/test-network/organizations/ordererOrganizations/example.com/orderers/orderer.example.com/msp/tlscacerts/tlsca.example.com-cert.pem  
export ORG1_PEER_TLS_ROOTCERT=$HOME/fabric_project_interface/test-network/organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt  
export ORG2_PEER_TLS_ROOTCERT=$HOME/fabric_project_interface/test-network/organizations/peerOrganizations/org2.example.com/peers/peer0.org2.example.com/tls/ca.crt  
  
#Environment variables for Org1  
export CORE_PEER_TLS_ENABLED=true  
export CORE_PEER_LOCALMSPID="Org1MSP"  
export CORE_PEER_TLS_ROOTCERT_FILE=$HOME/fabric_project_interface/test-network/
```

```
organizations/peerOrganizations/org1.example.com/peers/peer0.org1.example.com/tls/ca.crt
export CORE_PEER MSPCONFIGPATH=$HOME/fabric_project_interface/test-network/orga
nizations/peerOrganizations/org1.example.com/users/Admin@org1.example.com/msp
export CORE_PEER_ADDRESS=localhost:7051
```

○ Change enviroment file(제출한 과제에는 모두 수정되어 제출하였습니다.)

fabric\_project\_interface에서 수정이 필요하여 수정한 부분을 나열한다. 과제에는 모두 수정되어 제출하였기 때문에, 변경할 필요가 없습니다.

fabric.js 의 모든 파일	
const ccpPath = path.resolve(__dirname, '..', 'test-network', 'organizations', 'peerOrganizations', 'org1.example.com', 'connection-org1.json');	로 변경
ccp-template.json	
<pre>"certificateAuthorities": {   "ca.org\${ORG}.example.com": {     "url": "https://localhost:\${CAPORT}",     "caName": "ca-org\${ORG}",     "tlsCACerts": {       "pem": "\${CAPEM}"     },     "httpOptions": {       "verify": false     }   } }</pre>	
chaincode의 Tradeitem.js -> fabcar.js로 파일명 변경	

○ 제출한 과제 Home에 압축해제 및 실행(실행절차)

권한부여
cd ~
chmod -R 777 ~/fabric_project_interface
모듈 설치
cd ~/fabric_project_interface
npm install
네트워크 실행 및 체인코드 배포
cd ~/fabric_project_interface/test-network
rm -rf ../wallet/*
./network.sh down
./network.sh up createChannel -ca -s couchdb
./network.sh deployCC -ccl java -ccn fabcar -ccp '../chaincode/chaincode/java' -cci initLedger
서버 실행
cd ~/fabric_project_interface

```
npm start
```

○ 접속  
http://localhost:3000 으로 접속  
\*브라우저가 보증되지 않은 CA를 사용하는 https 접속을 차단하기 때문에, 반드시 http로 접속 부탁드립니다.


2. Description of Feature

2.1. Description

중고 거래를 목적으로 하는 컨소시엄 블록체인 네트워크를 하이퍼 렛저 페브릭 프레임워크를 통해 구현하고, 이 블록체인 네트워크와 통신하고 사용자의 브라우저에 웹 페이지를 렌더링해주는 서버를 구현한다. 이 문서에서 소개하는 프로그램은 사용자가 자신의 물품을 등록하고, 판매하며 구매할 수 있고, 블록체인에 등록된 아이템과 내 아이템, 거래에 등록된 아이템과 종료된 거래기록을 확인할 수 있어야 한다. 다음 2.2절에서는 과제에서 제시한 기능에 대한 설명을 나열한다.

2.2. Features

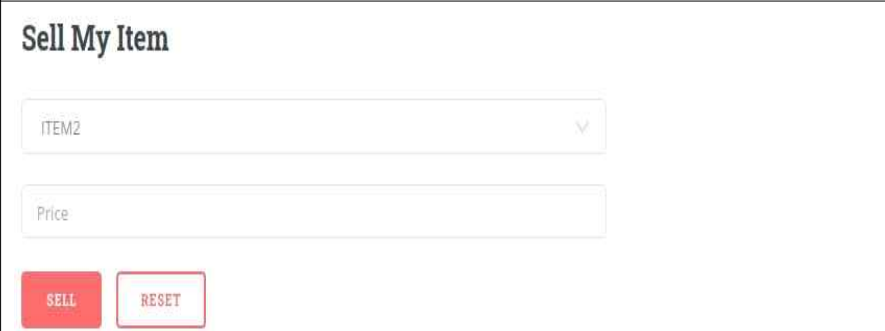
웹 어플리케이션을 통해 수행할 수 있는 구현된 기능을 나타낸다. 과제의 모든 조건을 포함하며, 과제에서 모호한 조건이나 구현에 있어 필요한 가정에 대한 구현, 테스트를 수행하기에 용이하도록 Balance와 같은 추가적으로 진행한 구현은 굵은 글씨로 표현된다.


기능 번호		F001
기능 명칭		관리자 등록
상세 설명	정의	관리자 등록
	세부 내용	○ 관리자의 지갑을 생성하고, 관리자를 등록 - 이후 admin으로 change user하는 것으로 관리자 계정 사용 가능
입력		None
사전 조건		네트워크 실행 및 서버 실행
사후 조건		Wallet에 관리자 지갑 생성
구현 화면		<b>Identity</b> 

기능 번호		F002
기능 명칭		사용자 등록 및 변경
상세 설명	정의	사용자 등록 및 변경
	세부 내용	○ 해당 사용자 지갑을 생성 ○ 해당 사용자 계정으로 전환
입력		사용자 이름
사전 조건		관리자 지갑 생성 이후
사후 조건		Wallet에 사용자 지갑 생성 사용자 계정으로 웹 어플리케이션 활용
구현 화면		<p><b>Name : wanggyu</b></p> <p><b>change user</b></p> <div>Enter user name</div> <p><b>CHANGE USER</b></p>

기능 번호		F003
기능 명칭		Token 수령
상세 설명	정의	Token 수령
	세부 내용	○ 관리자로부터 100token을 수령 ○ <b>Balance</b> 를 표시하는 UI 생성
입력		None
사전 조건		사용자 계정 등록 및 관리자 계정 등록
사후 조건		사용자에게 관리자가 100Token을 전송
구현 화면		<p><b>Balances : 200 Token</b></p> <p><b>earn 100Toekn</b></p> <p><b>EARN 100TOEKN</b></p>

기능 번호		F004
기능 명칭		아이템 등록
상세 설명	정의	아이템 등록
	세부 내용	<ul style="list-style-type: none"> <li>○ 아이템을 블록체인에 등록</li> <li>- 등록한 아이템은 해당 사용자의 소유가 됨</li> <li>- 아이템은 이름이 아닌 제조사를 기입 받고, 아이템 이름은 ITEM숫자로 명명됨</li> </ul>
입력		아이템 제조사 입력 - 과제 조건에서 <b>제조사</b> 를 입력받도록 규정
사전 조건		사용자가 전환된 상태
사후 조건		블록체인 아이템 등록, Sell my item에 등록된 아이템 판매 가능
구현 화면		

기능 번호		F005
기능 명칭		아이템 판매
상세 설명	정의	아이템 판매
	세부 내용	<ul style="list-style-type: none"> <li>○ 아이템 판매</li> <li>- Dropdown 형태로 내 아이템 목록 중 선택</li> <li>- 가격을 입력받아 해당 가격으로 등록</li> </ul>
사전 조건		아이템 등록
사후 조건		해당 아이템이 드롭다운 목록에서 제외 해당 아이템을 블록체인에 판매 상태로 기록
구현 화면		

기능 번호		F006
기능 명칭		아이템 구매
상세 설명	정의	아이템 구매
	세부 내용	<p>○ 판매 등록된 아이템의 구매</p> <ul style="list-style-type: none"> <li>- 다른 사용자가 판매를 등록한 아이템을 드롭다운으로 선택하여 구매</li> <li>- 구매 완료된 아이템은 목록에서 제외된다.</li> <li>- 자신이 판매 등록한 아이템은 보이지 않는다.</li> </ul>
입력		None
사전 조건		타 사용자의 아이템 판매 등록
사후 조건		해당 아이템의 가격만큼 토큰이 판매자에게 지불되며, 기능 F007에 거래기록이 나타나고, 아이템이 자신의 소유물이 된다.
구현 화면		

기능 번호		F007
기능 명칭		내 아이템 정보
상세 설명	정의	내 아이템 정보
	세부 내용	○ 내가 소유한 아이템을 표시
입력		None
사전 조건		사용자 전환 : 브라우저에 계정을 등록하지 않으면 리스트가 보이지 않는다.(웹 사이트 로그인과 유사)
사후 조건		None
구현 화면		

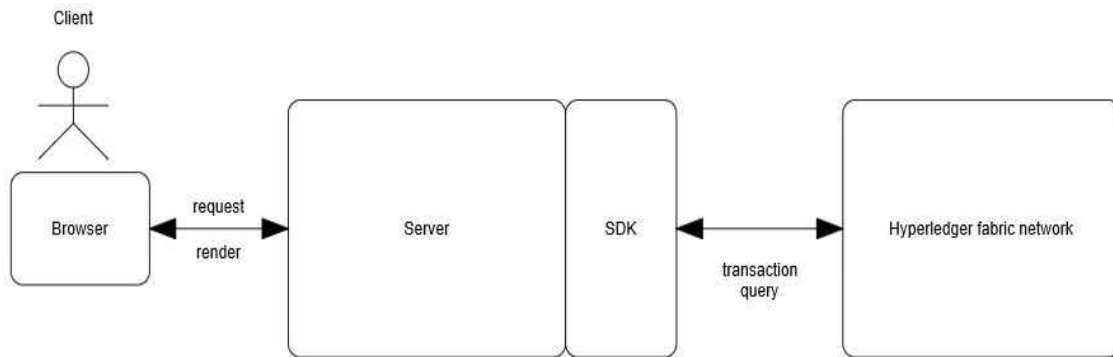
기능 번호		F008									
기능 명칭		등록된 모든 아이템 정보									
상세 설명	정의	등록된 모든 아이템 정보									
	세부 내용	○ 블록체인에 등록된 모든 아이템의 정보를 표시									
사전 조건		사용자 전환 : 브라우저에 계정을 등록하지 않으면 리스트가 보이지 않는다.(웹 사이트 로그인과 유사)									
사후 조건		None									
구현 화면		<div> <b>registered Items</b> <table> <thead> <tr> <th>Id</th><th>Owner</th><th>Name</th></tr> </thead> <tbody> <tr> <td>ITEM1</td><td>wanggyu</td><td>Samsung</td></tr> <tr> <td>ITEM2</td><td>kim</td><td>apple</td></tr> </tbody> </table> </div>	Id	Owner	Name	ITEM1	wanggyu	Samsung	ITEM2	kim	apple
Id	Owner	Name									
ITEM1	wanggyu	Samsung									
ITEM2	kim	apple									

기능 번호		F009															
기능 명칭		판매중인 아이템과 완료된 거래내역															
상세 설명	정의	판매중인 아이템과 완료된 거래내역															
	세부 내용	○ 판매중인 아이템을 on Sale로 표기 ○ 판매 완료된 기록을 Done으로 표기 - Owner에는 그 당시 구매한 사람의 이름을 표기															
사전 조건		사용자 전환 : 브라우저에 계정을 등록하지 않으면 리스트가 보이지 않는다.(웹 사이트 로그인과 유사)															
사후 조건		None															
구현 화면		<div><div>Items on sale</div><div><table><thead><tr><th>Id</th><th>Owner</th><th>Name</th><th>Price</th><th>Status</th></tr></thead><tbody><tr><td>ITEM3</td><td>kim</td><td>Nintendo</td><td>450</td><td>on Sale</td></tr><tr><td>ITEM2</td><td>kim</td><td>apple</td><td>100</td><td>Done</td></tr></tbody></table></div></div>	Id	Owner	Name	Price	Status	ITEM3	kim	Nintendo	450	on Sale	ITEM2	kim	apple	100	Done
Id	Owner	Name	Price	Status													
ITEM3	kim	Nintendo	450	on Sale													
ITEM2	kim	apple	100	Done													

### 3. Design

#### 3.1. Design of architecture

과제의 소프트웨어 아키텍처는 다음과 같다.



Nodejs(Express 프레임 워크)를 통해, 사용자에게 view를 렌더링해주고 사용자의 요청을 받아 블록체인 네트워크의 정보를 읽어오거나, 거래를 보낸다. 블록체인 상에 거래를 보내는 Server는 사용자 계정의 지갑 파일을 가지고 있으며, 해당 사용자가 적절한 거래를 요청하면 서버가 블록체인에 이 거래를 요청하거나 수행한다.

### 3.2. Design of chaincode

블록체인 상에서 endorser와 committing peer가 수행하는 코드에 대한 설명이다. getter와 setter, 생성자, 또는 상속받는 함수의 설명은 제외하고 서술한다.

#### ○ Item.java

하나의 아이템의 정보를 담는 클래스이다. 이 클래스의 객체에 정보를 담아 ledger에 기록하여 하나의 아이템을 설명한다.

구분	명칭	설명
필드	name	아이템의 명칭 ITEM1과 같이 정해진 명칭을 사용
필드	owner	아이템의 소유주
필드	marketState	아이템이 현재 판매 상태인가를 나타내는 변수
필드	price	판매 상태에서의 가격

#### ○ ItemQueryResult.java

쿼리를 통해 아이템을 출력하기 위해 사용하는 객체

#### ○ Fabcar.java (=TradeItem.java에서 명칭만 바꾸었습니다.)

레저의 스토리지에 데이터를 기록하거나 읽는 체인코드이다. 아이템을 사고팔거나, 소유주를 옮기는 모든 기능의 구현이 되는 클래스이다.



스토리지	
키	설명
사용자 명칭	보유 토큰 기록
MaximumHistory	다음의 구매내역을 저장할 킷값을 생성하기 위해 사용
HISTORY숫자	이전 동안 발생된 구매내역을 기록. 배열을 값으로 갖는다.
ITEM숫자	아이템은 ITEM + 숫자의 형태로 등록. 아이템 객체를 값으로 갖는다.
MaximumItem	다음에 등록될 아이템 킷값을 생성하기 위해 사용

구분	명칭	설명
에러	ItemErrors	아이템을 찾을 수 없거나, 이미 존재할 경우 발생
에러	TokenErrors	전송하려는 토큰의 액수보다 보유량이 적을 경우 발생
함수	initLedger	admin 계정에 토큰 1000000개를 부여하고, 다른 값들을 초기화
		<pre> @Transaction() public void initLedger(final Context ctx) {     ChaincodeStub stub = ctx.getStub();     stub.putStringState("admin", "1000000");     stub.putStringState("MaximumHistory", "0");     stub.putStringState("MaximumItem", "0"); } </pre>
함수	registerItem	아이템을 렛저에 등록
		<pre> @Transaction() public Item registerItem(final Context ctx, final String name, final String owner) {     ChaincodeStub stub = ctx.getStub();     String max = stub.getStringState("MaximumItem");     String key = "ITEM" + max;     String stateOfKey = stub.getStringState(key);     int m = Integer.parseInt(max);     m++;     Item item = new Item(name, owner, 0);     stub.putStringState("MaximumItem", Integer.toString(m));     stateOfKey = genson.serialize(item);     stub.putStringState("ITEM" + Integer.toString(m), stateOfKey);     return item; } </pre>
함수	transfer	from에서 to로 amount의 토큰을 전송

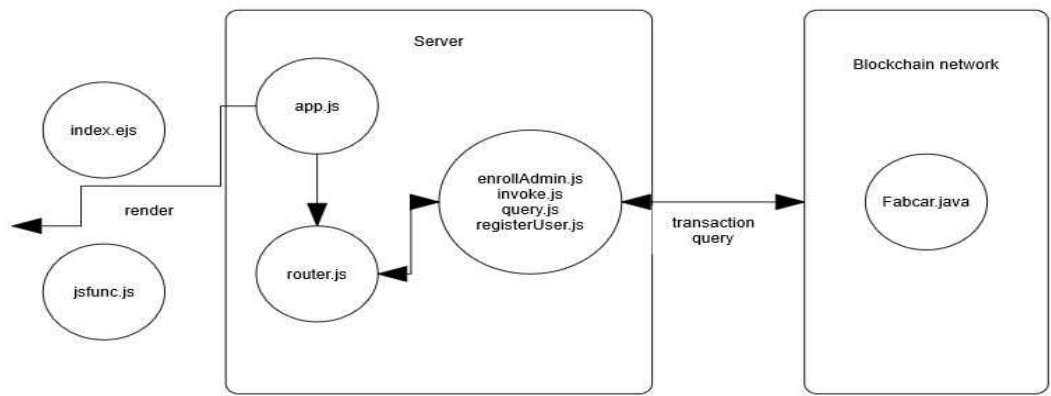
		<pre> @Transaction() public String transfer(final Context ctx, final String from, final String to, final int amount) {     ChaincodeStub stub = ctx.getStub();     String sender = stub.getStringState(from);     String receiver = stub.getStringState(to);     if (sender.isEmpty()) {         String errorMessage = String.format("%s has not balances", from);         System.out.println(errorMessage);         throw new ChaincodeException(errorMessage, TokenErrors.HAVING_LESS_TOKEN.toString());     }     if (receiver.isEmpty()) {         receiver = "0";     }     int senderBalances = Integer.parseInt(sender);     int rec = Integer.parseInt(receiver);     if (senderBalances &lt; amount) {         String errorMessage = String.format("%s has not balances", to);         System.out.println(errorMessage);         throw new ChaincodeException(errorMessage, TokenErrors.HAVING_LESS_TOKEN.toString());     }      senderBalances = senderBalances - amount;     rec = rec + amount;     stub.putStringState(from, Integer.toString(senderBalances));     stub.putStringState(to, Integer.toString(rec));     return "success"; } </pre>
함수	sellMyItem	<p>아이템의 판매 등록</p> <pre> @Transaction() public Item sellMyItem(final Context ctx, final String key, final int price) {     ChaincodeStub stub = ctx.getStub();     String stateOfKey = stub.getStringState(key);     if (stateOfKey.isEmpty()) {         String errorMessage = String.format("Item %s is not exists", key);         System.out.println(errorMessage);         throw new ChaincodeException(errorMessage, ItemErrors.ITEM_NOT_FOUND.toString());     }     Item item = genson.deserialize(stateOfKey, Item.class);     item.setMarketState(true);     item.setPrice(price);     stateOfKey = genson.serialize(item);     stub.putStringState(key, stateOfKey);     return item; } </pre>
함수	buyUserItem	아이템의 구매 등록

		<pre> @Transaction() public Item buyUserItem(final Context ctx, final String key, final String buyer) {     ChaincodeStub stub = ctx.getStub();     String stateOfKey = stub.getStringState(key);     String max = stub.getStringState("MaximumHistory");     int m = Integer.parseInt(max);     if (stateOfKey.isEmpty()) {         String errorMessage = String.format("Item %s is not exists", key);         System.out.println(errorMessage);         throw new ChaincodeException(errorMessage, ItemErrors.ITEM_NOT_FOUND.toString());     }     Item item = genson.deserialize(stateOfKey, Item.class);     transfer(ctx, buyer, item.getOwner(), item.getPrice());     ArrayList&lt;String&gt; hist = new ArrayList&lt;String&gt;();     hist.add(key);     hist.add(item.getName());     hist.add(buyer);     hist.add(item.getPrice().toString());     changeItemOwner(ctx, key, buyer);     m++;     String histSerialization = genson.serialize(hist);     stub.putStringState("HISTORY" + Integer.toString(m), histSerialization);     stub.putStringState("MaximumHistory", Integer.toString(m));      return item; } </pre>
함수	changeItemOwner	<p>buyUserItem에서 사용되는 함수로 구매자에게 아이템의 소유권을 양도</p> <pre> @Transaction() public Item changeItemOwner(final Context ctx, final String key, final String newowner) {     ChaincodeStub stub = ctx.getStub();     String stateOfKey = stub.getStringState(key);     if (stateOfKey.isEmpty()) {         String errorMessage = String.format("Item %s is not exists", key);         System.out.println(errorMessage);         throw new ChaincodeException(errorMessage, ItemErrors.ITEM_ALREADY_EXISTS.toString());     }     Item item = genson.deserialize(stateOfKey, Item.class);     Item newItem = new Item(item.getName(), newowner, 0);     stateOfKey = genson.serialize(newItem);     stub.putStringState(key, stateOfKey);     return item; } </pre>
함수	getMyItems	<p>자신의 아이템 목록을 조회</p> <pre> @Transaction() public String getMyItems(final Context ctx, final String owner) {     ChaincodeStub stub = ctx.getStub();     final String startKey = "ITEM1";     final String endKey = "ITEM99";     List&lt;ItemQueryResult&gt; queryResults = new ArrayList&lt;ItemQueryResult&gt;();      QueryResultsIterator&lt;KeyValue&gt; results = stub.getStateByRange(startKey, endKey);      for (KeyValue result: results) {         Item item = genson.deserialize(result.getStringValue(), Item.class);         if (item.getOwner().equals(owner)) {             queryResults.add(new ItemQueryResult(result.getKey(), item));         }     }      final String response = genson.serialize(queryResults);      return response; } </pre>
함수	getAllRegisteredIt	블록체인에 기록된 모든 아이템을 조회

	ems	<pre> @Transaction() public String getAllRegisteredItems(final Context ctx) {     ChaincodeStub stub = ctx.getStub();     final String startKey = "ITEM1";     final String endKey = "ITEM99";     List&lt;ItemQueryResult&gt; queryResults = new ArrayList&lt;ItemQueryResult&gt;();      QueryResultsIterator&lt;KeyValue&gt; results = stub.getStateByRange(startKey, endKey);      for (KeyValue result: results) {         Item item = genson.deserialize(result.getStringValue(), Item.class);         queryResults.add(new ItemQueryResult(result.getKey(), item));     }      final String response = genson.serialize(queryResults);      return response; } </pre>
함수	getAllOrderedItems	<p>현재 판매 상태인 아이템 목록과 그동안 구매 완료된 내역을 조회("&amp;"구분자로 구분되어 reponse)</p> <pre> @Transaction() public String getAllOrderedItems(final Context ctx) {     ChaincodeStub stub = ctx.getStub();     final String startKey = "ITEM1";     final String endKey = "ITEM99";     List&lt;ItemQueryResult&gt; queryResults = new ArrayList&lt;ItemQueryResult&gt;();      QueryResultsIterator&lt;KeyValue&gt; results = stub.getStateByRange(startKey, endKey);      for (KeyValue result: results) {         Item item = genson.deserialize(result.getStringValue(), Item.class);         if (item.getMarketState()) {             queryResults.add(new ItemQueryResult(result.getKey(), item));         }     }     final String response = genson.serialize(queryResults);      final String startHistKey = "HISTORY1";     final String endHistKey = "HISTORY99";     List&lt;String&gt; queryHistResults = new ArrayList&lt;String&gt;();     QueryResultsIterator&lt;KeyValue&gt; resultOfHist = stub.getStateByRange(startHistKey, endHistKey);     for (KeyValue result: resultOfHist) {         queryHistResults.add(result.getStringValue());     }     final String responseHist = genson.serialize(queryHistResults);     return response + "&amp;" + responseHist; } </pre>
함수	getBalance	<p>사용자의 토큰 잔액을 조회</p> <pre> @Transaction() public int getBalance(final Context ctx, final String name) {     ChaincodeStub stub = ctx.getStub();     String stateOfName = stub.getStringState(name);     if (stateOfName.isEmpty()) {         String errorMessage = String.format("%s has not balances", name);         System.out.println(errorMessage);         throw new ChaincodeException(errorMessage, TokenErrors.HAVING_LESS_TOKEN.toString());     }     int m = Integer.parseInt(stateOfName);     return m; } </pre>

3.3. Design of server

nodejs를 통해 실행되는 서버는 사용자의 브라우저에 view를 렌더링해주고, 사용자의 요청을 받아 하이퍼레저 페브릭 네트워크에 거래를 보내는 역할을 한다.



브라우저를 통해 접속하는 사용자는 index.ejs와 jsfunc.js에 의해 view가 구성되고, view에서의 활동을 통해 서버로 보내는 요청은 app.js와 router.js에 의해 sdk파일을 수행하여 블록체인 네트워크에 거래를 요청하게 된다.

4. Result

절차	내용	실행 사진 및 세부 설명
1	관리자 등록 및 관리자 계정 잔고 확인	<div>Identity</div> <div>ENROLL ADMIN</div>
		Enroll Admin 버튼을 클릭하여 관리자 지갑 생성
		<div>Name : admin</div> <div>change user</div> <div>Enter user name</div> <div>CHANGE USER</div>
		<div>Balances : 1000000 Token</div>
		1000000 토큰 확인

3	사용자 wanggyu 전환	<div><div>Name : wanggyu</div><div>change user</div><div><div>Enter user name</div></div><div>CHANGE USER</div></div>												
4	ITEM1, ITEM2, ITEM3 등록	<div><div>My Items</div><div><table><tr><th>Id</th><th>Owner</th><th>Name</th></tr><tr><td>ITEM1</td><td>wanggyu</td><td>Samsung</td></tr><tr><td>ITEM2</td><td>wanggyu</td><td>Apple</td></tr><tr><td>ITEM3</td><td>wanggyu</td><td>Mssoft</td></tr></table></div><div>register my item을 통해 등록</div></div>	Id	Owner	Name	ITEM1	wanggyu	Samsung	ITEM2	wanggyu	Apple	ITEM3	wanggyu	Mssoft
Id	Owner	Name												
ITEM1	wanggyu	Samsung												
ITEM2	wanggyu	Apple												
ITEM3	wanggyu	Mssoft												
5	ITEM2 판매(가격 150 token) 등록	<div><div>Sell My Item</div><div><div>ITEM2</div></div><div><div>150</div></div><div><div>SELL</div><div>RESET</div></div><div><div>Items on sale</div><div><table><tr><th>Id</th><th>Owner</th><th>Name</th><th>Price</th><th>Status</th></tr><tr><td>ITEM2</td><td>wanggyu</td><td>Apple</td><td>150</td><td>on Sale</td></tr></table></div></div></div>	Id	Owner	Name	Price	Status	ITEM2	wanggyu	Apple	150	on Sale		
Id	Owner	Name	Price	Status										
ITEM2	wanggyu	Apple	150	on Sale										

6	사용자 chulsu 전환 및 500token 받음(earn token 5회)	<div> Name : chulsu </div> <div> change user </div> <div> <input type="text" value="Enter user name"/> </div> <div> CHANGE USER </div> <div> Balances : 500 Token </div> <div> earn 100Toekn </div> <div> EARN 100TOEKN </div> <div> earn 100 toekn을 통해 500 token admin에게서 수령 </div>						
7	chulsu의 ITEM2 구매	<div> Buy Users Item </div> <div> <div>ITEM2</div> <div>▼</div> </div> <div> <div>BUY</div> <div>RESET</div> </div> <div> My Items </div> <div> <table> <thead> <tr> <th>Id</th> <th>Owner</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>ITEM2</td> <td>chulsu</td> <td>Apple</td> </tr> </tbody> </table> </div> <div> Name : chulsu </div> <div> change user </div> <div> <input type="text" value="Enter user name"/> </div> <div> CHANGE USER </div> <div> Balances : 350 Token </div> <div> earn 100Toekn </div> <div> EARN 100TOEKN </div>	Id	Owner	Name	ITEM2	chulsu	Apple
Id	Owner	Name						
ITEM2	chulsu	Apple						

		<div><div>Items on sale</div><table><tr><th>Id</th><th>Owner</th><th>Name</th><th>Price</th><th>Status</th></tr><tr><td>ITEM2</td><td>chulsu</td><td>Apple</td><td>150</td><td>Done</td></tr></table></div> <div><div>registered Items</div><table><tr><th>Id</th><th>Owner</th><th>Name</th></tr><tr><td>ITEM1</td><td>wanggyu</td><td>Samsung</td></tr><tr><td>ITEM2</td><td>chulsu</td><td>Apple</td></tr><tr><td>ITEM3</td><td>wanggyu</td><td>Mssoft</td></tr></table></div>	Id	Owner	Name	Price	Status	ITEM2	chulsu	Apple	150	Done	Id	Owner	Name	ITEM1	wanggyu	Samsung	ITEM2	chulsu	Apple	ITEM3	wanggyu	Mssoft
Id	Owner	Name	Price	Status																				
ITEM2	chulsu	Apple	150	Done																				
Id	Owner	Name																						
ITEM1	wanggyu	Samsung																						
ITEM2	chulsu	Apple																						
ITEM3	wanggyu	Mssoft																						
8	구매한 ITEM2(가격 200 token) 판매 등록	<div><div>Items on sale</div><table><tr><th>Id</th><th>Owner</th><th>Name</th><th>Price</th><th>Status</th></tr><tr><td>ITEM2</td><td>chulsu</td><td>Apple</td><td>200</td><td>on Sale</td></tr><tr><td>ITEM2</td><td>chulsu</td><td>Apple</td><td>150</td><td>Done</td></tr></table></div> <div>이전에 wanggyu에게서 chulsu가 구매한 이력 확인과 현재 철수가 판매중인 상태 목록 확인 가능</div>	Id	Owner	Name	Price	Status	ITEM2	chulsu	Apple	200	on Sale	ITEM2	chulsu	Apple	150	Done							
Id	Owner	Name	Price	Status																				
ITEM2	chulsu	Apple	200	on Sale																				
ITEM2	chulsu	Apple	150	Done																				
9	사용자 kim 전환 및 400 Token 받음	<div><div>Name : kim</div><div>change user</div><div><div>Enter user name</div></div><div>CHANGE USER</div><div>Balances : 400 Token</div><div>earn 100Toekn</div><div>EARN 100TOEKN</div></div>																						
10	ITEM4 등록	<div><div>My Items</div><table><tr><th>Id</th><th>Owner</th><th>Name</th></tr><tr><td>ITEM4</td><td>kim</td><td>Google</td></tr></table></div>	Id	Owner	Name	ITEM4	kim	Google																
Id	Owner	Name																						
ITEM4	kim	Google																						



11	ITEM4 판매(가격 90 token) 등록	<div>Items on sale</div> <table><tr><th>Id</th><th>Owner</th><th>Name</th><th>Price</th><th>Status</th></tr><tr><td>ITEM2</td><td>chulsu</td><td>Apple</td><td>200</td><td>on Sale</td></tr><tr><td>ITEM4</td><td>kim</td><td>Google</td><td>90</td><td>on Sale</td></tr><tr><td>ITEM2</td><td>chulsu</td><td>Apple</td><td>150</td><td>Done</td></tr></table>	Id	Owner	Name	Price	Status	ITEM2	chulsu	Apple	200	on Sale	ITEM4	kim	Google	90	on Sale	ITEM2	chulsu	Apple	150	Done									
Id	Owner	Name	Price	Status																											
ITEM2	chulsu	Apple	200	on Sale																											
ITEM4	kim	Google	90	on Sale																											
ITEM2	chulsu	Apple	150	Done																											
12	kim의 ITEM2 구매	<div>My Items</div> <table><tr><th>Id</th><th>Owner</th><th>Name</th></tr><tr><td>ITEM2</td><td>kim</td><td>Apple</td></tr><tr><td>ITEM4</td><td>kim</td><td>Google</td></tr></table> <div>Items on sale</div> <table><tr><th>Id</th><th>Owner</th><th>Name</th><th>Price</th><th>Status</th></tr><tr><td>ITEM4</td><td>kim</td><td>Google</td><td>90</td><td>on Sale</td></tr><tr><td>ITEM2</td><td>chulsu</td><td>Apple</td><td>150</td><td>Done</td></tr><tr><td>ITEM2</td><td>kim</td><td>Apple</td><td>200</td><td>Done</td></tr></table>	Id	Owner	Name	ITEM2	kim	Apple	ITEM4	kim	Google	Id	Owner	Name	Price	Status	ITEM4	kim	Google	90	on Sale	ITEM2	chulsu	Apple	150	Done	ITEM2	kim	Apple	200	Done
Id	Owner	Name																													
ITEM2	kim	Apple																													
ITEM4	kim	Google																													
Id	Owner	Name	Price	Status																											
ITEM4	kim	Google	90	on Sale																											
ITEM2	chulsu	Apple	150	Done																											
ITEM2	kim	Apple	200	Done																											
결과	잔여 잔고	<div>Name : chulsu</div> <div>change user</div> <div><div>Enter user name</div><div>CHANGE USER</div></div> <div>Balances : 550 Token</div> <div>earn 100Toekn</div> <div><div>EARN 100TOEKN</div></div> <div>Name : kim</div> <div>change user</div> <div><div>Enter user name</div><div>CHANGE USER</div></div> <div>Balances : 200 Token</div>																													

Name : wanggyu

change user

CHANGE USER

Balances : 150 Token

earn 100Toekn

EARN 100TOEKN

registered Items

Id	Owner	Name
ITEM1	wanggyu	Samsung
ITEM2	kim	Apple
ITEM3	wanggyu	Msoft
ITEM4	kim	Google

Items on sale

Id	Owner	Name	Price	Status
ITEM4	kim	Google	90	on Sale
ITEM2	chulsu	Apple	150	Done
ITEM2	kim	Apple	200	Done