Hyperledger Fabric Library Lending Decentralized Application

Ledger pada aplikasi terdesentralisasi ini berisi informasi tentang buku-buku yang terdapat pada perpustakaan seperti status peminjaman, nomor ISBN, kategori buku, judul buku, dan berbagai informasi buku lainnya. Transparansi terlihat dari logging chaincode event yang dicatat setiap kali terjadi transaksi terhadap ledger. Tech Stack Front-End menggunakan framework Ionic + React yang dipadukan dengan Express REST API sebagai Back-End dengan Smart Contract Hyperledger Fabric.

Smart Contract Folder Structure

➔ application
➔ bin
➔ chaincode
➔ gateway
➔ network

files dari folder bin dapat di-download dari drive berikut https://drive.google.com/file/d/147x3oTvtDpO4hClYc0kiUzHdbe4zoFcy/view kemudian di-extract pada folder bin tersebut.

Alternatif Repository

https://github.com/darren-code/ulibrary

```
git clone https://github.com/darren-code/ulibrary
```

Prerequisites

- Pastikan prerequisites Hyperledger Fabric seperti docker, go, dan lainnya telah terinstal.
- Pastikan versi node berada di 8.9.0 dan versi npm 5.5.1, apabila belum maka dapat dengan menggunakan package n pada npm (untuk mengatur versi node)
  - `npm install -g npm@5.5.1`
  - `npm install -g n`
  - `n 8.9.0`
- Pastikan node_modules pada beberapa direktori
  - `./chaincode/library/`
  - `./application/`
  - `./gateway/Org1`
  - `./gateway/Org2`

  telah terinstal, apabila belum maka dapat dengan mengeksekusi perintah `npm i` pada direktori-direktori tersebut

Pertama-tama jalankan perintah berikut agar memastikan ketika docker-compose tidak terjadi timeout

```
export COMPOSE_HTTP_TIMEOUT=9000
```

Lalu navigate ke direktori network dan nyalakan network (asumsi posisi sedang berada di root directory dari project)

```
cd ./backend/network
./sample-network.sh up
```

Terkadang pertama kali proses instantiate chaincode akan mengembalikan error 500, apabila terjadi maka dapat dengan mengeksekusi perintah berikut

```
docker exec -it cli bash
peer chaincode instantiate -o orderer1.example.com:7050 --tls true --cafile
/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/order
er1.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n mycc -l node -v 1.0 -c
'{"Args":["InitLedger"]}' -P 'AND ('\''Org1MSP.peer'\'','\''Org2MSP.peer'\'')'
```

Apabila masih mengembalikan error 500 maka dapat mengulang perintah `peer chaincode instantiate` tersebut berulang kali hingga tidak mengembalikan error tersebut yang berarti instansiasi chaincode telah sukses.

```
Instantiating chaincode on peer2.org2...
+ peer chaincode instantiate -o orderer1.example.com:7050 --tls true --cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordere
rOrganizations/example.com/orderers/orderer1.example.com/msp/tlscacerts/tlsca.example.com-cert.pem -C mychannel -n mycc -l node -v 1.0 -c '{"A
rgs":["InitLedger"]}' -P 'AND ('\''Org1MSP.peer'\'','\''Org2MSP.peer'\'')'
2021-12-09 01:48:04.464 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default escc
2021-12-09 01:48:04.465 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vscc
Error: could not assemble transaction, err proposal response was not successful, error code 500, msg timeout expired while starting chaincode
mycc:1.0 for transaction
+ set +x
===================== Chaincode is instantiated on peer2.org2 on channel 'mychannel' =====================


===================== Sample network end-to-end test completed =====================
darren@hyperledger:~/hyperledger/rentee/network$ docker exec -it cli bash
root@d3992d7b5496:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chaincode instantiate -o orderer1.example.com:7050 --tls true --caf
ile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer1.example.com/msp/tlscacerts/tl
sca.example.com-cert.pem -C mychannel -n mycc -l node -v 1.0 -c '{"Args":["InitLedger"]}' -P 'AND ('\''Org1MSP.peer'\'','\''Org2MSP.peer'\'')'
2021-12-09 01:56:13.697 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default escc
2021-12-09 01:56:13.697 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vscc
Error: could not assemble transaction, err proposal response was not successful, error code 500, msg timeout expired while starting chaincode
mycc:1.0 for transaction
root@d3992d7b5496:/opt/gopath/src/github.com/hyperledger/fabric/peer# peer chaincode instantiate -o orderer1.example.com:7050 --tls true --caf
ile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/example.com/orderers/orderer1.example.com/msp/tlscacerts/tl
sca.example.com-cert.pem -C mychannel -n mycc -l node -v 1.0 -c '{"Args":["InitLedger"]}' -P 'AND ('\''Org1MSP.peer'\'','\''Org2MSP.peer'\'')'
2021-12-09 02:02:40.258 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 001 Using default escc
2021-12-09 02:02:40.258 UTC [chaincodeCmd] checkChaincodeCmdParams -> INFO 002 Using default vscc
root@d3992d7b5496:/opt/gopath/src/github.com/hyperledger/fabric/peer#
```

Pada gambar tersebut diperlukan 2 kali eksekusi `peer chaincode instantiate` untuk dapat instansiasi chaincode secara sukses.

Apabila sebelumnya telah menjalankan `docker exec -it cli bash` maka lakukan perintah `exit` atau `Ctrl + D` terlebih dahulu untuk kembali ke environment utama.

Ketika network telah nyala dan chaincode telah terinstansiasi maka hal berikut-nya lakukan navigasi ke direktori ./gateway/Org1 dan hapus folder admin pada folder wallet lalu lakukan enroll admin dengan perintah berikut (asumsi direktori saat ini sedang berada pada network)

```
cd ../gateway/Org1
sudo rm -rf ./wallet/admin
node enrollAdmin.js
```

Lalu lakukan navigasi ke folder application dan jalankan server Express dengan perintah berikut (asumsi direktori saat ini sedang berada pada ./gateway/Org1)

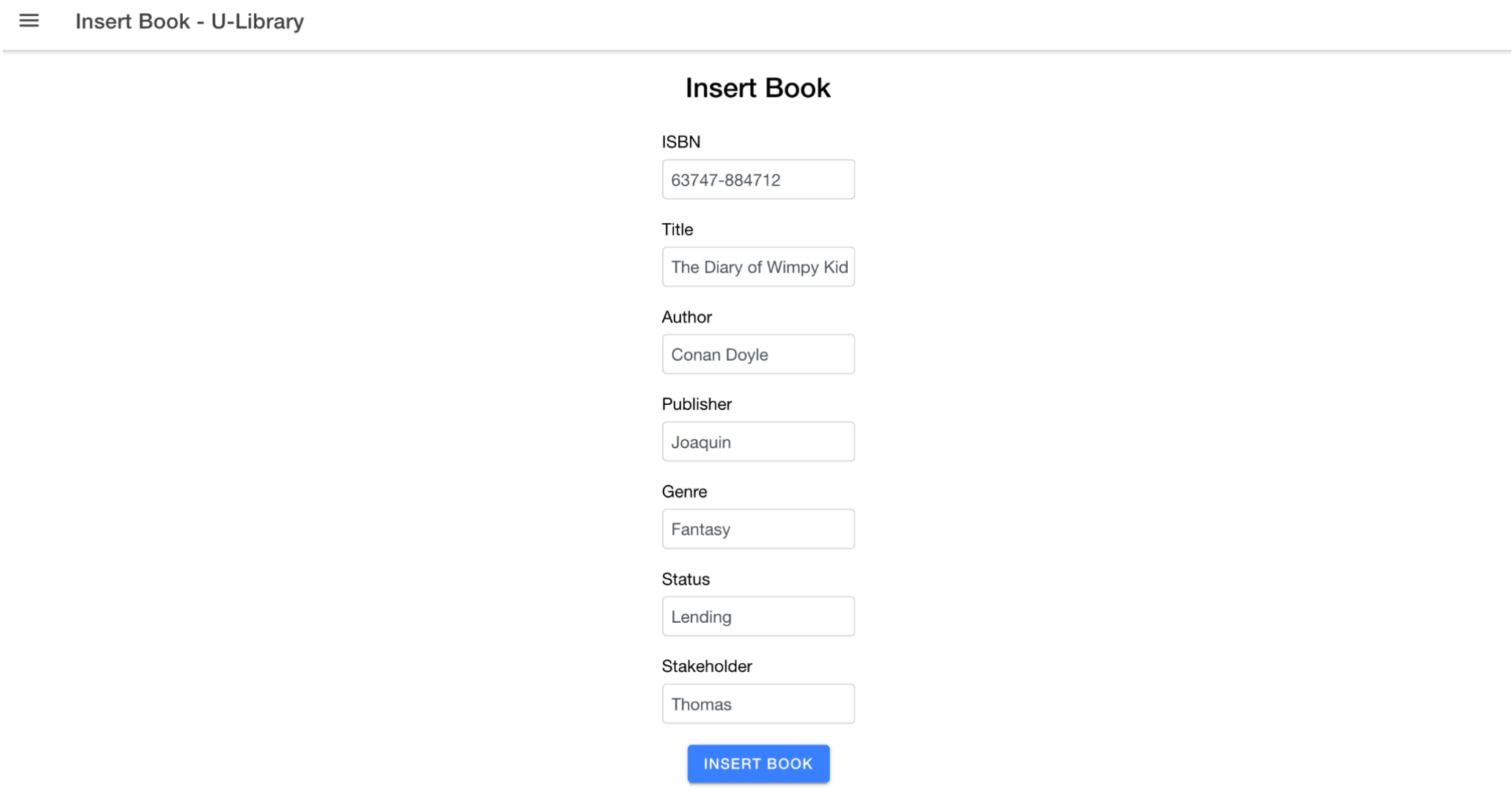
```
cd ../../application
node server.js
```

Lakukan navigasi ke direktori frontend untuk setup environment front-end aplikasi dengan memperbaharui node dan npm ke versi terbaru kemudian diikuti dengan instalasi ionic. Setelah instalasi sukses berikutnya dapat menjalankan (`npm install`) untuk melengkapi `node_modules` dari dependencies `package.json` sehingga dapat mengeksekusi `ionic serve` untuk menjalankan aplikasi.

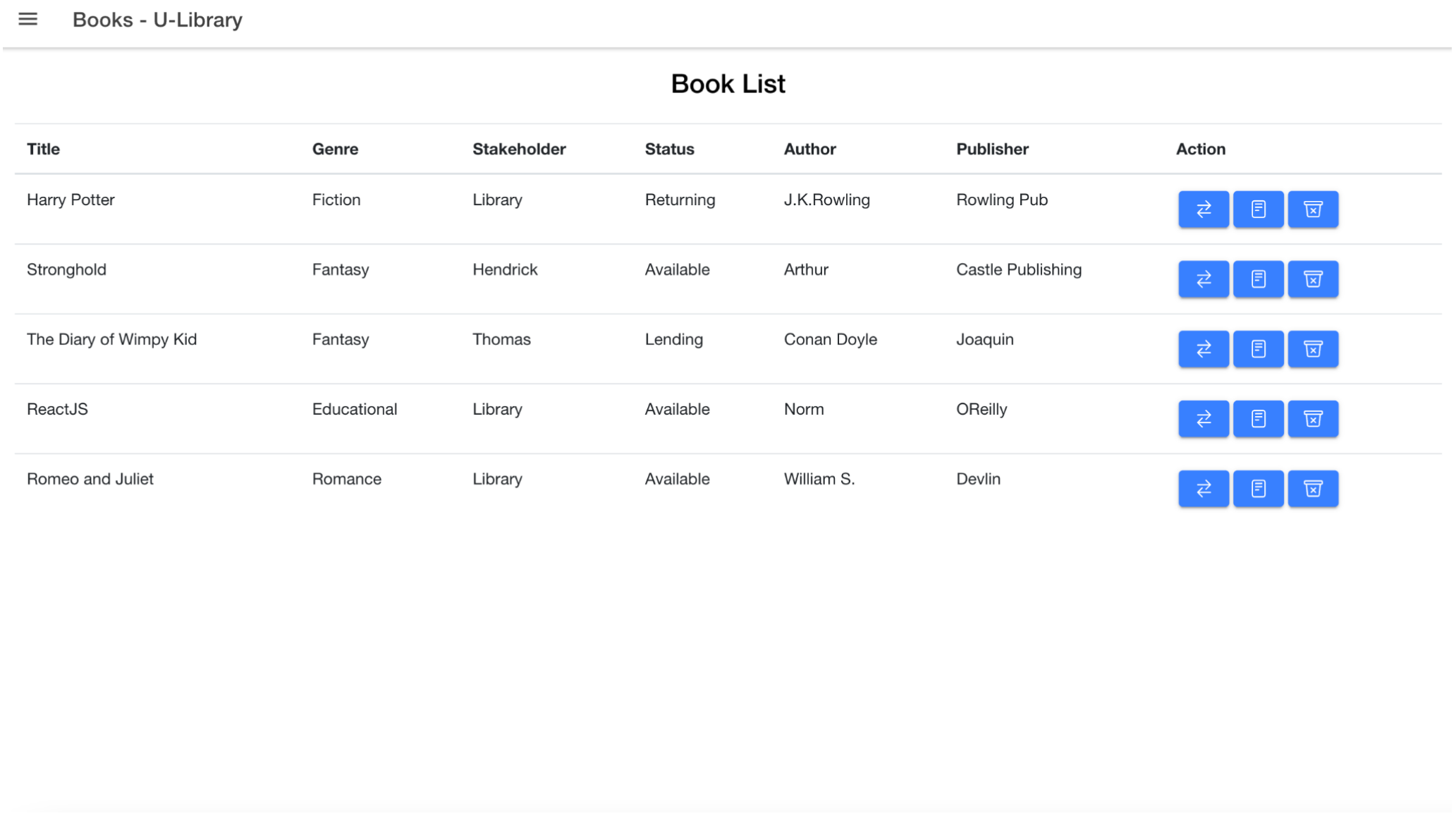
```
cd ../../frontend
npm install -g npm@latest
sudo n stable
npm install -g @ionic/cli
npm install
ionic serve
```

Beberapa transaksi yang dapat dilakukan pada aplikasi perpustakaan terdesentralisasi sebagai berikut

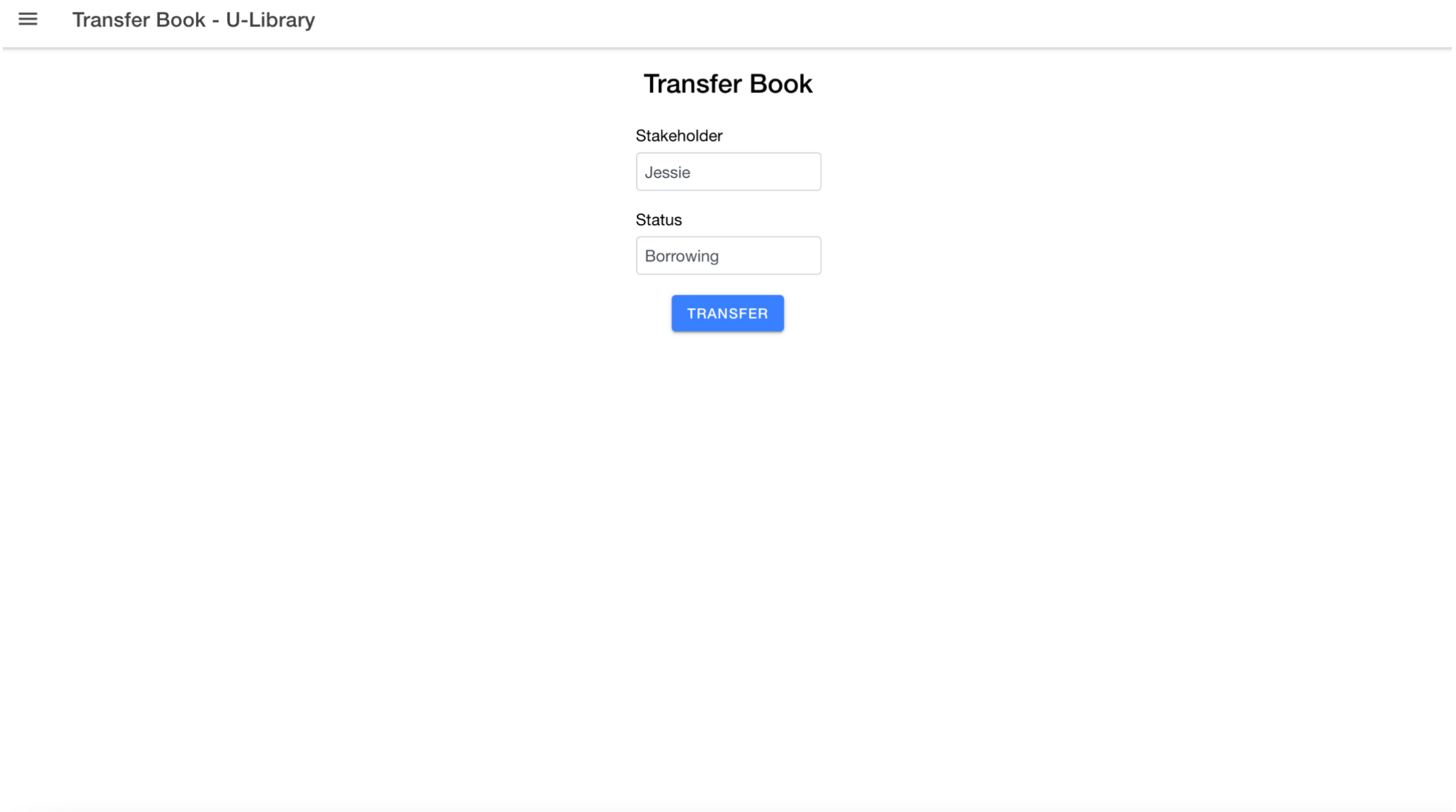
1. Insert Book



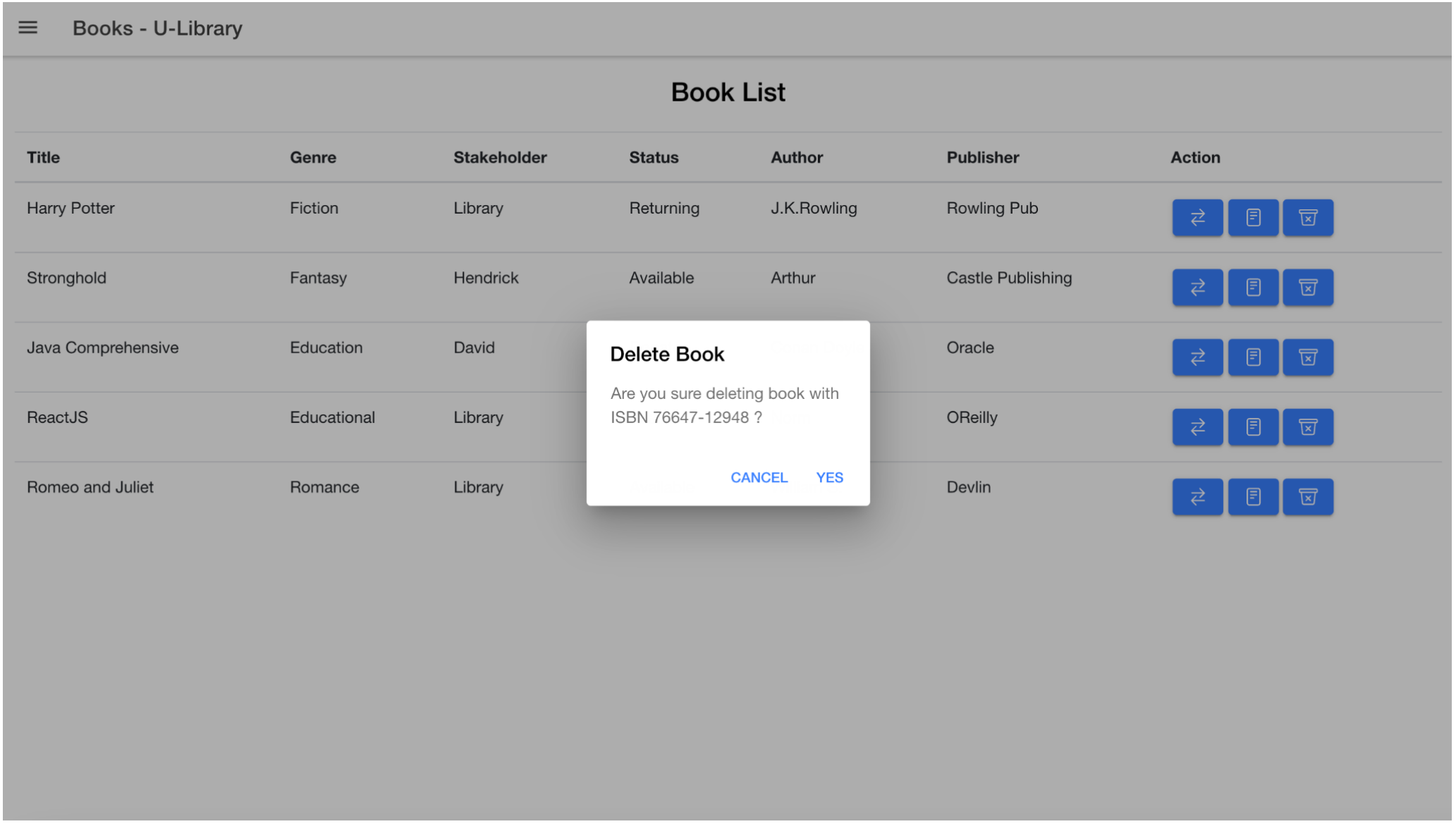
2. Book List


Books - U-Library
Book List
Title
Genre
Stakeholder
Status
Author
Publisher
Action
Harry Potter
Fiction
Library
Returning
J.K.Rowling
Rowling Pub
Stronghold
Fantasy
Hendrick
Available
Arthur
Castle Publishing
The Diary of Wimpy Kid
Fantasy
Thomas
Lending
Conan Doyle
Joaquin
ReactJS
Educational
Library
Available
Norm
OReilly
Romeo and Juliet
Romance
Library
Available
William S.
Devlin

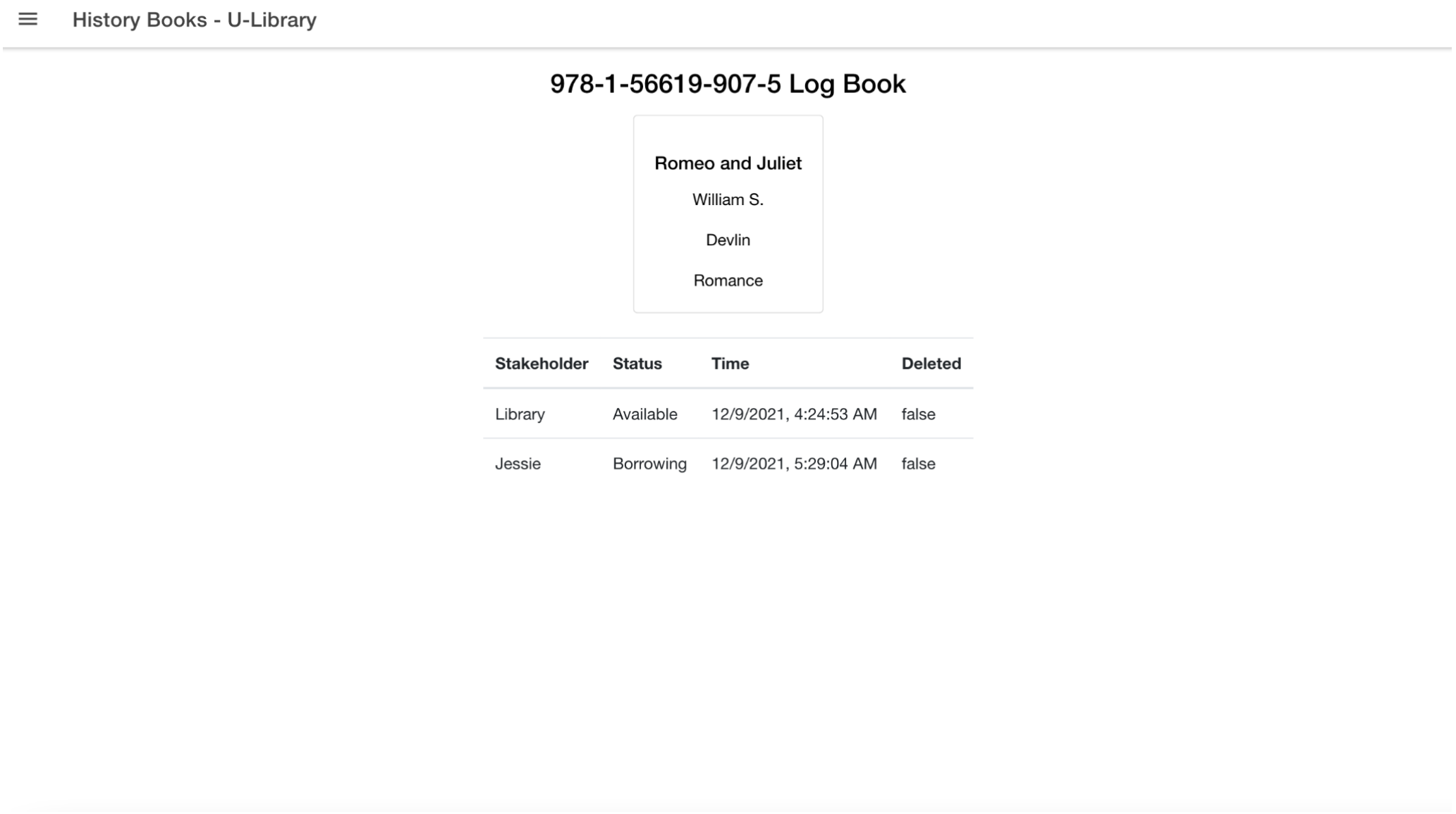
3. Transfer Book



4. Delete Book



5. Book Information



Aspek transparansi dapat dilihat dari logging chaincode event ketika menjalankan server Express sebelumnya (`node server.js`)

```
>>>setUserContext...
Connect to Fabric gateway with userid: admin
>>>utils.submitTx...TransferBook (978-1-56619-907-5,Jessie,Borrowing)
Event payload: {"Author":"William S.","Genre":"Romance","Holder":"Jessie","ISBN":"978-1-56619-907-5","Publisher":"Devlin","Status":"Borrowing"
,"Title":"Romeo and Juliet","docType":"book"}
Block Number: 13, Transaction ID: 5bf371871a571e0ff82728895060093d25fd0063dc4936c73d88fcfbc131893c, Status: VALID
----------------------------------
2021-12-09T05:29:07.078Z - info: [TransactionEventHandler]: _strategySuccess: strategy success for transaction "5bf371871a571e0ff8272889506009
3d25fd0063dc4936c73d88fcfbc131893c"
Transaction submitted successfully; Response:  true

>>>setUserContext...
Connect to Fabric gateway with userid: admin
>>>utils.evalTx...GetBookHistory (978-1-56619-907-5)
Transaction submitted successfully; Response:  "[{\"TxId\":\"5842cba1c336793bd0c0dfa675ed0be40c86dc50891ede60f888421eccd7f896\",\"IsDelete\":\
"false\",\"timestamp\":\"12/9/2021, 4:24:53 AM\",\"Value\":{\"ISBN\":\"978-1-56619-907-5\",\"Genre\":\"Romance\",\"Holder\":\"Library\",\"Titl
e\":\"Romeo and Juliet\",\"Status\":\"Available\",\"Author\":\"William S.\",\"Publisher\":\"Devlin\",\"docType\":\"book\"}},{\"TxId\":\"5bf371
871a571e0ff82728895060093d25fd0063dc4936c73d88fcfbc131893c\",\"IsDelete\":\"false\",\"timestamp\":\"12/9/2021, 5:29:04 AM\",\"Value\":{\"Autho
r\":\"William S.\",\"Genre\":\"Romance\",\"Holder\":\"Jessie\",\"ISBN\":\"978-1-56619-907-5\",\"Publisher\":\"Devlin\",\"Status\":\"Borrowing\
",\"Title\":\"Romeo and Juliet\",\"docType\":\"book\"}}]"

>>>setUserContext...
Connect to Fabric gateway with userid: admin
>>>utils.submitTx...DeleteBook (76647-12948)
Event payload: {"ISBN":"76647-12948"}
Block Number: 14, Transaction ID: fa40fd642c7f7b21f3dbefbf4411d1bdef47b5643a025f36384882889d372cfc, Status: VALID
----------------------------------
2021-12-09T05:35:44.094Z - info: [TransactionEventHandler]: _strategySuccess: strategy success for transaction "fa40fd642c7f7b21f3dbefbf4411d1
bdef47b5643a025f36384882889d372cfc"
Transaction submitted successfully; Response:  true
```

Aspek skalabilitas terlihat dari seiring dengan berkembangnya data transaksi mampu mencegah terjadi throttling dengan meningkatnya jumlah transaction per second dengan benchmarking Hyperledger Caliper. Karena setiap peer akan memiliki salinan ledger yang sama secara terdesentralisasi dan terdistribusi sehingga tidak ada sebuah titik yang menjadi pusat transaksi choke point.

Konsensus ordering transaksi pada Hyperledger Fabric memastikan setiap peer dalam jaringan memiliki daftar transaksi yang sama pada distributed ledger masing-masing peer. Agar bisa mendapatkan ledger pertama-tama transaksi harus melewati kebijakan endorsement dan juga peers yang melakukan endorsement terhadap chaincode transaksi. Setelah sudah terdapat pada ledger, kemudian transaksi akan diperiksa untuk terakhir kalinya dari setiap langkah validasi peer. Hal demikian membuat arsitektur Hyperledger Fabric menjadi Byzantine Fault Tolerant.