CS731:-Blockchain Technology and Applications

Endsem Report

Team Id:-4

BorderPay.io Cross Border Contractual Payments

Documentation

Contents

§1 What are we building?

Making a complete system that allows creation for financial contracts between a company and it's contractors/employees, have a progressive payroll system that automatically updates and calculates salaries of the contractors/employees on a timely basis and ensure payments are made as per the contract signed by both parties viz. the employer and employee be it local or cross-border.

Assumptions:- Our team is well connected with banks and legal authorities around the globe to deliver a prompt and seamless experience to all your customers.

§2 Frontend

Technologies Used:- HTML+Reactjs+CSS

§2.1 Home Page

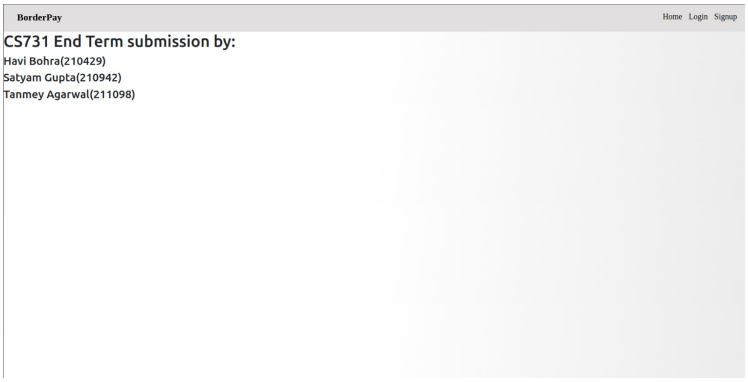

Figure 1

§2.2 Login Page

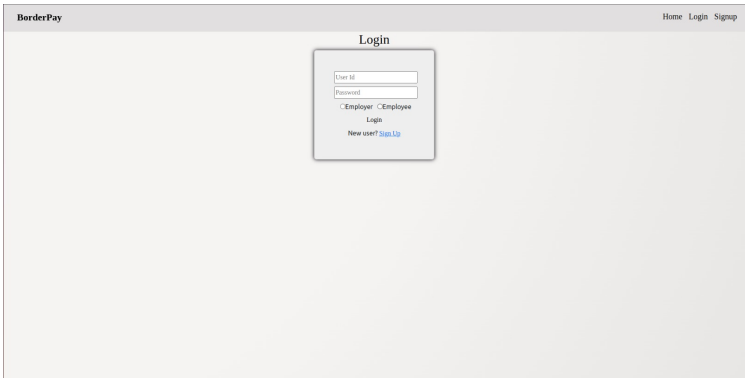

Figure 2

This page is responsible for authentication of user.

- React Form for User Login
- At backend connected with Peer0.Org1(Company's Peer) for authentication
- Asks type of User(Employee or Employer)

§2.3 Signup Page

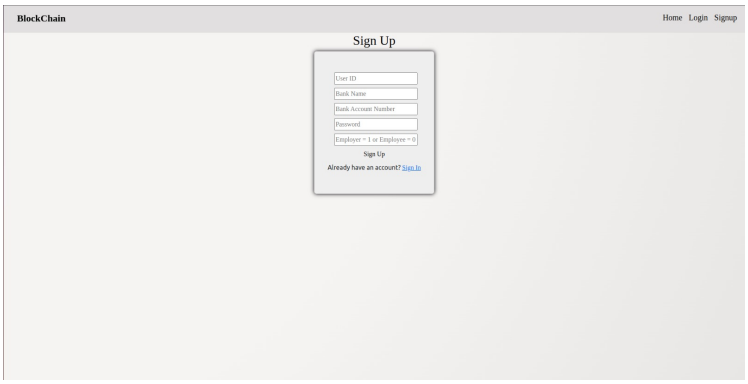

Figure 3

This page is reached by new user for registration in Company's Database

- React form for signup
- Asks for UserId, Bank Name, Account Number, Password and type of User
- Connected to Peer0.Org1(Company's Peer) for registering in **Users_Collection**.

§2.4 Create Contract Page

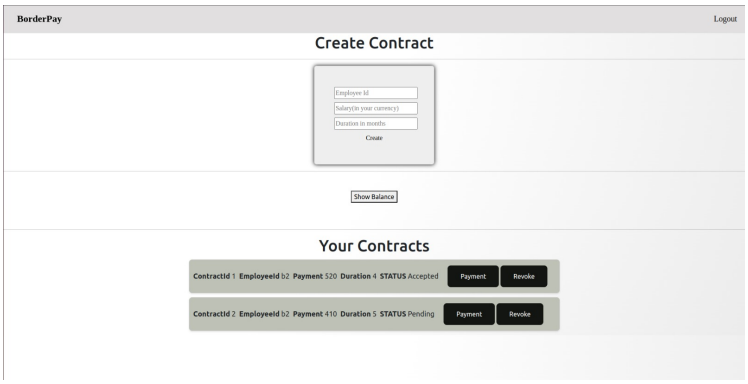

Figure 4

This page is for users signed in as **employer** to "Create", "View","Revoke" or to "Execute" contracts.

- At backend connected to Peer0.Org1(Company's Peer)

- Contracts are stored in Contract's Collection at Peer0.Org1(Company's Peer)
- **Revoke Button** connects to Peer0.Org1(Company's Peer) which changes Contract Status from Pending or Accepted to Revoked, which then can not be executed.
- **Payment Button** connects to Peer0.Org2(Banks Peer) for executing payments which returns TransactionID if payment successfully executed

§2.5 Accept Contract Page

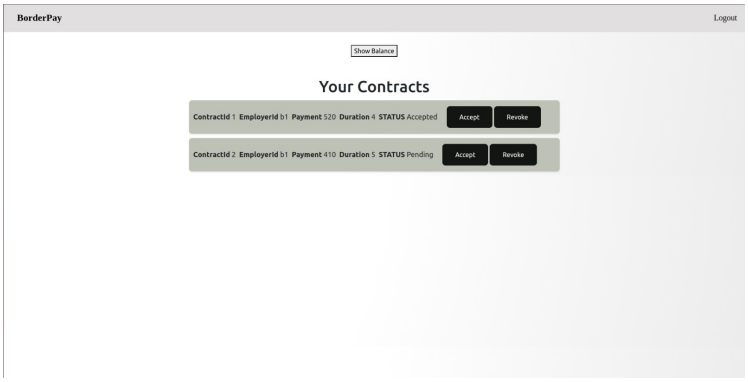

Figure 5

This page is for user signed in as **Employee**.

- **Accept Button** connects to company's peer which changes status of contract from "Pending" to "Accepted"
- **Revoke Button** connects to company's peer which changes status of contract from "Pending" to "Rejected"

§3 User Manual and Setup

§3.1 System Requirements

1. Hyperledger Fabric(with alll dependencies)
2. npm
3. go
4. Docker
5. Docker-Compose

§3.2 Instructions to Run

Follow steps as written in https://github.com/havibohra/BorderPay.Io/blob/main/README.md

§4 Wireframe Diagram

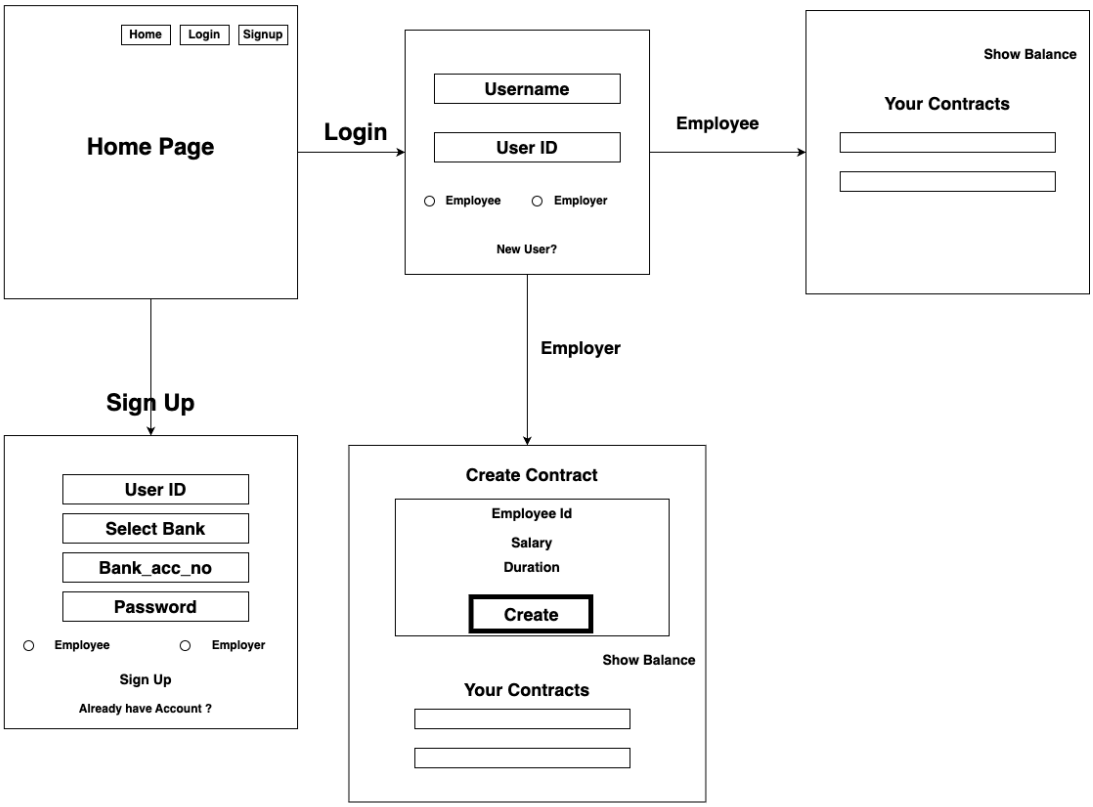

Figure 6: Wireframe Diagram

§5 Architecture Diagram

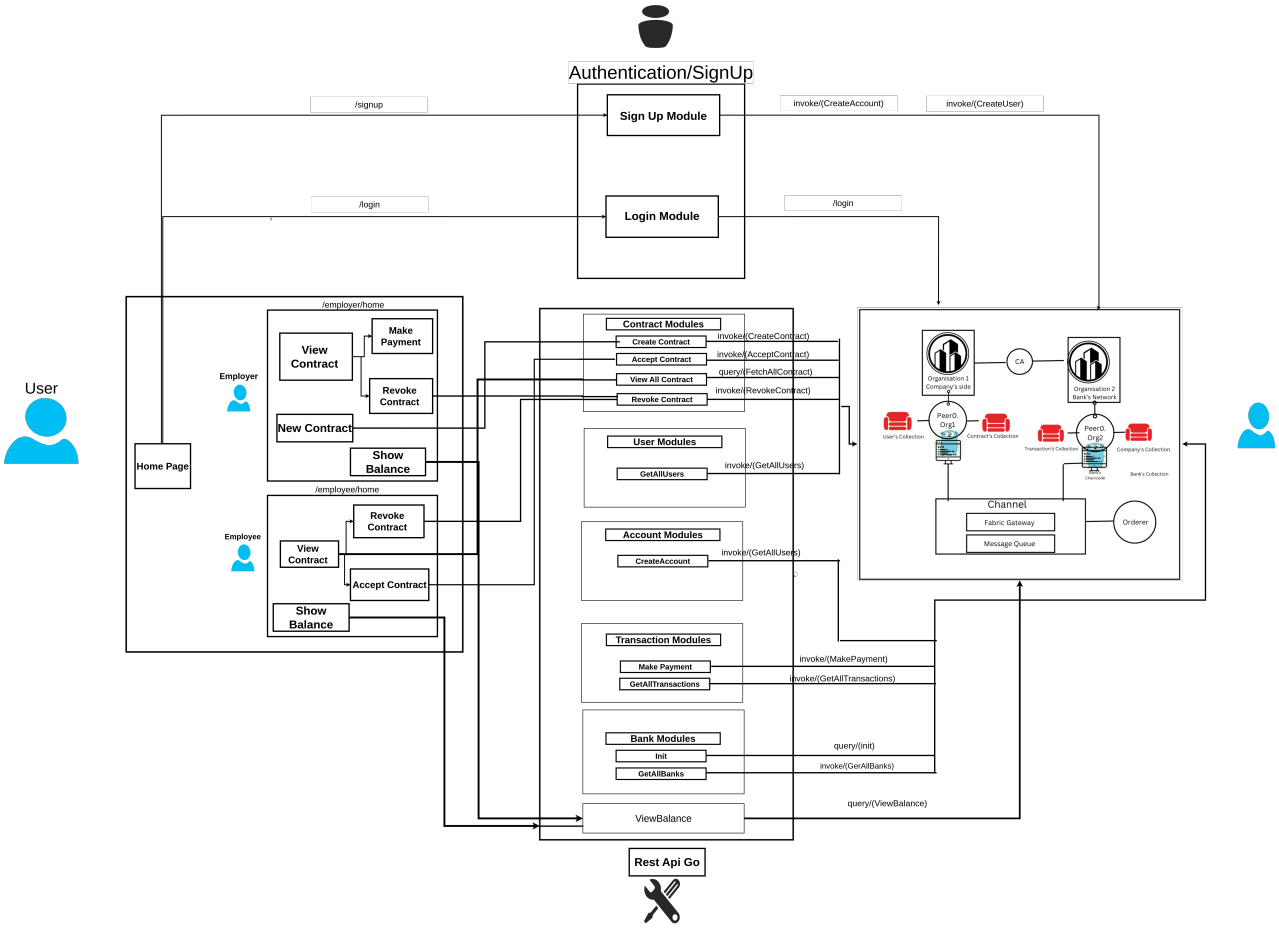

Figure 7

§6 Chaincodes Explained

§6.1 Payments Chaincode

§6.1.1 Account

```go
// Account represents a bank account
type Account struct {
    Account_No string  `json:"Account_No"`
    Bank_Name  string  `json:"Bank_Name"`
    Balance    float64 `json:"Balance"`
    Currency   string  `json:"Currency"`
}
```

Figure 8

This struct defines the attributes of an Account and how it is stored in bank's database.

§6.1.2 Bank

```go
type Bank struct {
    Bank_Name   string             `json:"Bank_Name"`
    Account_Map map[string]Account `json:"Account_Map"`
    Tax         float32            `json:"Tax"`
    Bank_Type   string             `json:"Bank_Type"`
    Currency    string             `json:"Currency"`
}
```

Figure 9

This struct defines the attributes of an Bank

§6.1.3 Init

```go
// InitLedger initializes the ledger with initial bank and account details
func (s *SmartContract) Init(ctx contractapi.TransactionContextInterface) error
```

Figure 10

Initialize banks creating initial accounts. This struct defines the attributes of an Bank.

§6.1.4 GetAccountBalance

Function returns balance in given account

```
// GetAccountBalance retrieves the balance of a bank account
func (s *SmartContract) GetAccountBalance(ctx contractapi.TransactionContextInterface, bank_name string, account_no string) (float64, error)
```

Figure 11

§6.1.5 GetAllBanks

```
// GetAllBanks retrieves all the banks from the ledger
func (s *SmartContract) GetAllBanks(ctx contractapi.TransactionContextInterface) ([]Bank, error)
```

Figure 12

Returns the current status of all the banks

§6.1.6 GetAllTransactions

```
// GetAllTransactions retrieves all the transactions from the ledger
func (s *SmartContract) GetAllTransactions(ctx contractapi.TransactionContextInterface) ([]Transaction, error)
```

Figure 13

Returns a list of all transactions happened in ledger.

§6.1.7 GetBankDetails

```
// GetBankDetails retrieves the account details of a bank
func (s *SmartContract) GetBankDetails(ctx contractapi.TransactionContextInterface, nm string) ([]Account, error)
```

Figure 14

Gives account details of a particular bank.

§6.1.8 CreateAccount

```
// CreateAccount creates a new bank account
func (s *SmartContract) CreateAccount(ctx contractapi.TransactionContextInterface,bank_name string, currency string, account_no string) (string, error)
```

Figure 15

This function creates account for users who are doing transactions for first time.

§6.1.9 Execute

```
// execute executes a financial transaction
func execute(ctx contractapi.TransactionContextInterface, src Bank, src_bank_acc_no string, receiver Bank, receiver_bank_acc_no string, amount float64, rate float64) (string, error)
```

Figure 16

This function changes the balance of respective

§6.1.10 MakePayment

```
// MakePayment initiates a payment transaction
func (s *SmartContract) MakePayment(ctx contractapi.TransactionContextInterface, src_bank string, src_bank_acc_no string, receiver_bank string, receiver_bank_acc_no string, amount float64) (string, error)
```

Figure 17

Checks all errors and finally make payment.

§6.2 Company's Chaincode

§6.2.1 User

```
type User struct {
    UserID            string `json:"UserID"`
    Password          string `json:"Password"`
    UserType          int    `json:"UserType"`
    BankName          string `json:"BankName"`
    BankAccountNumber string `json:"BankAccountNumber"`
}
```

Figure 18

Defines all the attributes of datatype User.

§6.2.2 Transaction

```
type Transaction struct {
    Transaction_ID  string  `json:"Transaction_ID"`
    Sender_Bank     string  `json:"Sender_Bank"`
    Sender_Acc_No   string  `json:"Sender_Acc_No"`
    Receiver_Bank   string  `json:"Receiver_Bank"`
    Receiver_Acc_No string  `json:"Receiver_Acc_No"`
    Amount          float64 `json:"Amount"`
    Currency        string  `json:"Currency"`
}
```

Figure 19

Defines all the Datatype of datatype transaction.

§6.2.3 CreateUser

```
//Functions related to Users
func (s *SmartContract) CreateUser(ctx contractapi.TransactionContextInterface, id string, psw string, bname string, baccNo string, t int) error
```

Figure 20

Create User for the first time.

§6.2.4 Login

```
// Login Function
func (s *SmartContract) LoginFunc(ctx contractapi.TransactionContextInterface, id string, psw string, Utype int) (bool, error)
```

Figure 21

Help to login already created account.

§6.2.5 GetAllUsers

```
func (s *SmartContract) GetAllUsers(ctx contractapi.TransactionContextInterface) ([]User, error)
```

Figure 22

Returns a list of all users registered.

§6.2.6 GetBankDetails

```
//Given user id this function returns the bank details of the user
func (s *SmartContract) GetBankDetails(ctx contractapi.TransactionContextInterface, id string) (string, string, error)
```

Figure 23

Return whole user given user id.

§6.2.7 CreateContract

```
func (s *SmartContract) CreateContract(ctx contractapi.TransactionContextInterface, employerId string, employeeId string, payment int, duration int) error
```

Figure 24

Creates a new Contract by employer.

§6.2.8 AcceptContract

```
func (s *SmartContract) AcceptContract(ctx contractapi.TransactionContextInterface, contractID string) error
```

Figure 25

Help employee accept the contract.

§6.2.9 RevokeContract

```
func (s *SmartContract) RevokeContract(ctx contractapi.TransactionContextInterface, id_of_Contract string) error
```

Figure 26

Help both employer and employee to revoke contract.

§6.2.10 ViewAllContract

```
func (s *SmartContract) ViewAllContracts(ctx contractapi.TransactionContextInterface) ([]Contract, error)
```

Figure 27

Gives all list of contracts signed by both parties.

§6.2.11 FetchAllContracts

```
func (s *SmartContract) FetchAllContractsbyUserID(ctx contractapi.TransactionContextInterface, UserID string) ([]Contract, error)
```

Figure 28

Fetch contracts related to particular user.

§7 API endpoints

localhost for frontend:3000 localhost for api:3002

'http://localhost:3002/invoke', new URLSearchParams([['', ''], ['channelid', 'mychannel'], ['chaincodeid', 'paytest'], ['function', 'MakePayment'], ['args', b1], ['args', bn1], ['args', b2], ['args', bn2], ['args', pay]]));

```
'http://localhost:3002/query', {
        params: {
        'channelid': 'mychannel',
        'chaincodeid': 'basictest',
        'function': 'GetBankDetails',
        'args': logeduserid
        }
      });

'http://localhost:3002/query', {
        params: {
        'channelid': 'mychannel',
        'chaincodeid': 'basictest',
        'function': 'GetBankDetails',
        'args': logeduserid
        }
      });

'http://localhost:3002/query', {
          params: {
          'channelid': 'mychannel',
          'chaincodeid': 'basictest',
          'function': 'FetchAllContractsbyUserID',
          'args': logeduserid
          }
        });

'http://localhost:3002/invoke',
        new URLSearchParams([
          ['', ''],
          ['channelid', 'mychannel'],
          ['chaincodeid', 'basictest'],
          ['function', 'AcceptContract'],
          ['args', contract_id],
        ])
      );

'http://localhost:3002/invoke',
        new URLSearchParams([
```

```
        ['', ''],
        ['channelid', 'mychannel'],
        ['chaincodeid', 'basictest'],
        ['function', 'RevokeContract'],
        ['args', contract_id],
      ])
    );
```

§8 Future Prospects

1. Include multiple organisations to handle different Banks
2. Include a system to add new banks in hyperledger
3. Allote each bank its own certifying authorities