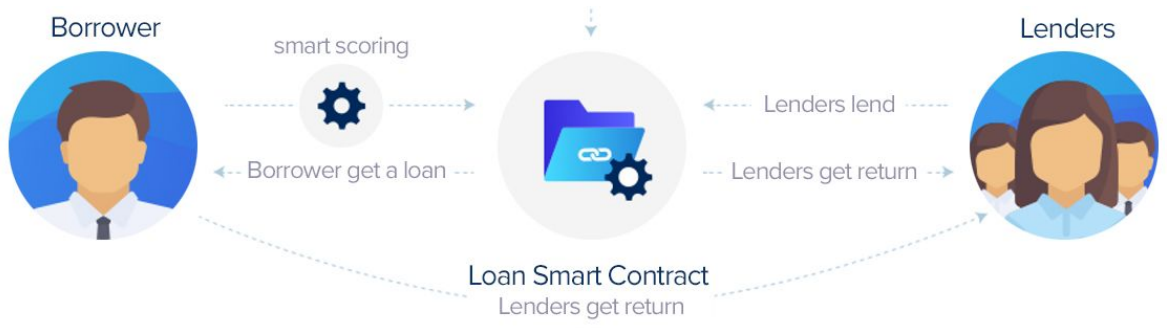
LoanChain- A blockchain based P2P platform



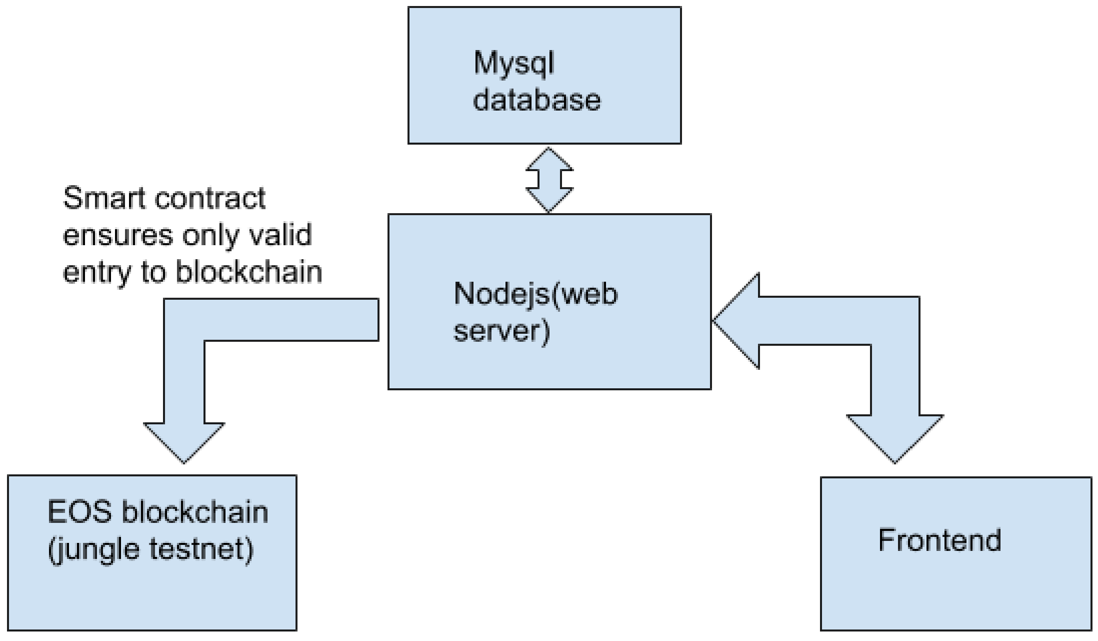
Key concepts/modules:

- Nodejs
- Expressjs
- EOS/EOSjs
- Pug
- Ajax calls
- Passport
- Express - Session
- Mysql

Key players in the platform:

- Borrower(actions: create profile, update profile, create lending)
- Lender (actions: create profile, update profile, create lendinfo)
- Admin_borrower(handles borrower profile creation/update/closing in the blockchain)
- Admin_lender(handles lender profile creation/update/closing in the blockchain)
- Admin_lending (handles lendings creation/update/closing in the blockchain)

Overview of the platform infrastructure


Mysql database
Smart contract ensures only valid entry to blockchain
Nodejs(web server)
EOS blockchain (jungle testnet)
Frontend

LoanChain- A blockchain based P2P platform

Key Components in detail

- **EOS Blockchain:**

  Tables in the blockchain:
  - Lenders: contains profile of lender including age, balance, lending score, verified(boolean) identified by a unique account name.
  - Borrowers: contains profile of borrower including age, income, cibil score, verified(boolean) identified by unique account name.
  - Lendings: contains lending created by a borrower which includes net_borrowed(loan amount), amount_left(amount yet to be lended), interest, tenure etc.
  - Lendinfo: contains borrower account name , lenders account name, principal(amount lended by lender to borrower), emi, penalty etc.

  Logic of the smart contract: smart contract takes following variable into account
  - age_limit = 21(for both lender and borrower
  - income_limit = 300000(for borrower)
  - lending_limit = 500000(for lender on this platform)
  - borrowing_limit = 500000(for borrower on this platform)
  - per_lending_limit = 20000(limit of amount to be lended by a lender to a borrower)
  - interest_limit = 30(max interest rate a borrower can offer)
  - max_months = 30(maximum months upto which a borrower can borrow)
  - cibil_limit = 500(minimum cibi score of borrower to create lending on the platform)

  Overview of smart contract:

  First of all borrower and lender create their respective profile using a unique eos account name. Borrower and lenders are then subjected to their profile validation and verification which is done by adminb for borrower and adminl for lender. If a lender add some balance to his/her account then it is reflected in his/her profile in the blockchain. A verified borrower can create lending specifying the net borrowing amount , tenure, interest etc. If a lender wishes to lend some money to the borrower he/she can lend at maximum 20% of loan amount upto INR 20000. When a lender makes a lending to a borrower it is registered in lendinfo table. When a borrower makes monthly payment, EMI amount is proportionally distributed to all the lenders as per the amount lended by them. If a borrower fails to pay the amount in time, penalty is added to borrower's lending and is proportionally distributed when borrower pays the penalty. Lending is automatically closed(erased) once a borrower pays loan amount in full with 0 penalty.

  Guidelines for writing a smart contract:
  - A table should not have more than 8 fields. It should not have more than 3 int_64 data types in it.(it gave error in 3 int_64 types too sometimes)
  - Scope of the table: whenever there is new entry to the table scope is to mentioned. Scope basically something which makes a table different from similar table of that type. For eg. in tic-tac-toe game to create a table of

existing games it is efficient to choose scope of the table as host of the game and challenger as the primary key. This will ensure that for a host there are no duplicate challengers.

- Secondary Indexes: It is additional indexing of the table to quickly and efficiently retrieve data from the table. Currently EOS multi-index table only supports one/two secondary index opposed to 16 secondary index as they claim. This is because secondary index and primary key should be atleast of int_64t types or more and not more than 3 data types can be of int_64t types ina table. Thus, choice of primary key and secondary index is very important to make database efficient.
- Whenever there is a new entry or update or erasing an entry to a table, payer for that action has to be named and that action should have the permission of the payer.
- The most efficient and quick way to write a smart contract is to first identify actors involved and then identify the tables needed for the contract. Then identify the actions to be done on each table like a new entry, updating an existing entry or erasing an entry, then identify prerequisites/checks/dependencies for the corresponding action such as what authorization is needed for the action or check in another table etc. After identifying all the actions table can be revised to make database more optimal and efficient.

Wallet keys-

1. Loanchain:
   PW5HyftCXhjXkMgaUewUzDEG4e61LDrE2KjCsZY7PRxmGwKENiPT9

Account keys-

1. loanchain-
   Private key:
   5Kk6Q4L4XR9pLEVwAEA1copNRXUXb2ivs2jg31qfmF54DeJYAeg
   Public key:
   EOS7kMU2iZ3jS3R6SiLmnrC2W7KDVDCcMwPKJKnobHKCy4eJqzGLL

2. adminl-
   Private key:
   5K5Yh3EV8yKX8eU9yc3FJHoZVYfHseaDy9WZkDiN38C68JnZZVD
   Public key:
   EOS8gw1PGH3dq15yhDngryWnX1dMyAncm3X6onkCN3vbA1GHNeYfb

3. adminb-
   Private key:
   5Hpi4jXWt9tvX3pgoa6Lq7VRyXmHWAjJqQif9ipoy4LEfehA1Pa

Public key:
EOS8MLGtibzpwt9tGxCTs4ZhUPj56hdPEthvmEDN5hGyBUW126Uh
h

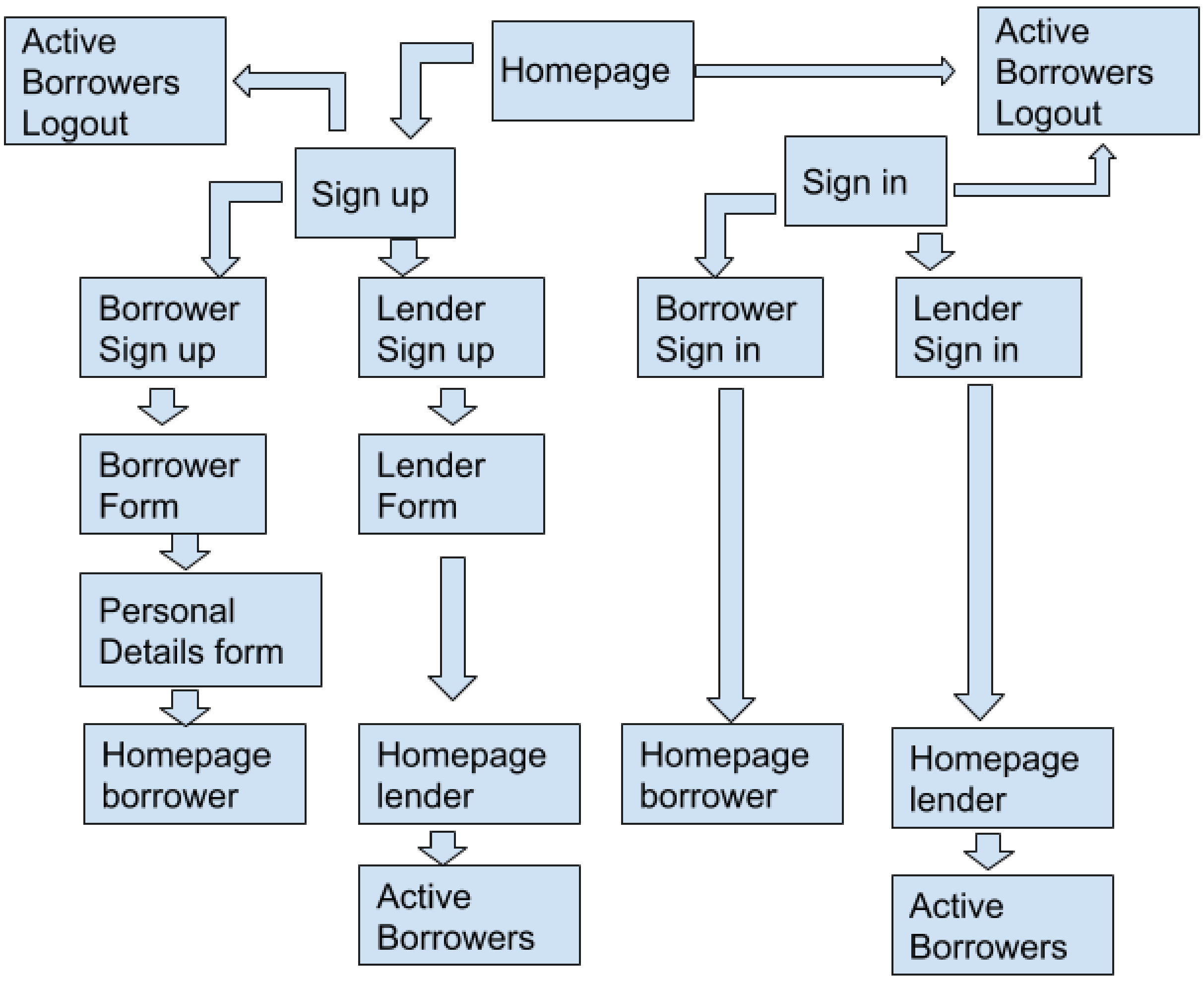
4. adminlen-
Private key:
5JebU9LKJDQ48Zg6omsQvVAKhGc7j1AE9YwMwgeG7cRvXALNoC
Y
Public key:
EOS6taTAVtCYrNHKZ3xea8j3xd4Y9mH8krUtb5puqgjRQggU2eXKG

Useful links:
Developer portal
API Documentation

- **Nodejs(Web Server):** Nodejs is one of the many server side scripting language. It creates web-server for client to access the app.This is the core component of the platform which glues the other components together. It basically interact with database, blockchain and user interface to produce the intended flow of data/information. For this particular application we employed use of ExpressJS module along with modules such as pug, passport, mysql, eosjs, eosjs-ecc etc.
    - **ExpressJS**- ExpressJS is a light-weight web application framework to help organize web application on the server side. It basically helps you manage everything, from routes, to handling requests and views. Useful Link
    - **Pug-** It is one of the templating language that expressjs supports. It makes a lot easier to render a customized page for different users. For eg. Page of active borrower is different for a lender than to a user who is not signed in as lender. It also makes life a lot easier when it comes to rendering a dynamic table on the client side. For eg. rendering a list of active borrower whenever a user refresh/click on the page of active borrower. Useful link
    - **Passport-** Passport along with express-session and express-mysql-session helps to create and maintain session for a particular user. Useful Link
    - **Mysql-** This module is to interact with the database. Basically to send query and receive result.
    - **Eosjs**- It is a nodejs module used to interact with blockchain from the server-side. It can be used to connect to the blockchain and publish the transaction. Useful link
    - **Eosjs-ecc-** It is another module of eos used to generate keypair to be used for the creation of new user account. Useful link
- **Mysql Database:** Mysql is a structured collection of data. It is fast, reliable and simple to use although Nosql database such as mongoDB is recommended to use for future applications. Tables for this particular application are lender_login which includes all the lenders information including aadhar card, pancard, date of birth,

balance etc., borrower_login which includes all the borrower's information including aadhar, pancard, date of birth, Loan details, employment details, financial details etc., lendings which includes lending information of the applied loan by borrower such as loan amount, interest, tenure, purpose etc. and lendinfo table which includes information of borrower, lender ,amount lended by a borrower to a lenders, emi etc.. There is one addition table called sessions to store the session ID of current logged in user. Entry to table lending is only if borrower is verified which means if verified variable at borrower_login table is set to 1.

- **Frontend: Diagram below shows the routing of the web application on client side**



- **Borrower signup process**: It involves filing two form one after the other. First form consists of basic borrower login information, aadhar card and pancard. Second form consists of borrower loan details financial details, employment details etc.
- **Lender signup process**: It involves filling only one from consisting of lender's login information, aadhar, pancard, date of birth etc.
- **Borrower Homepage features**: A borrower can check for his/her loan eligibility irrespective of his current status of document verification and loan

application. When a borrower first log in to the page, he/she will see a document verification button which will serve the purpose of uploading and verification of document. Since this is a POC application on click it will automatically set the verification field in the borrower_login table to 1 and also in the blockchain. Once document is verified, a borrower can apply for a loan. After his/her loan application is accepted he/she can see the current status of the loan by clicking the current loan button. Current loan will show the status of his loan.

- **Lender Homepage feature**: A lender similar to borrower can verify his/her documents by a button click. He/she can choose to add balance by giving input the amount. Add balance button will trigger a transaction of adding balance on lender's profile on blockchain as well as on the database. Ideally, it should be redirecting to bank's page and only after successful completion of payment balance should be added to the blockchain and database but since it is a POC this will happen on a button click. A lender should be seeing his current lendings to different borrower but this feature is yet to be enabled. After adding balance a lender can browse through the lists of active borrowers.
- **Active borrower page**: This page will show the list of all the borrower and progress they made in their loan. For a lender this page will show the option to 'invest', and route to go back to 'home' and 'logout' and for other it will not. Rest of the page will be same. If a lender wants to invest he/she just need to enter the amount to be invested and click on the button 'invest'.