SUMMER INTERNS TEAM CSBBT - 4

presents

e-KYC

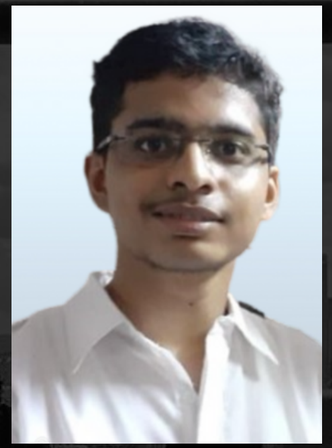
One-stop web-platform that modernizes the Know-Your-Customer process in banking sector, created using MERN stack and Web3 Blockchain technology.

Under the guidance of

Managers - Santosh Vasa and Sushma Yaramalla, Senior Coach - Jyothi A

Team CSBBT - 4

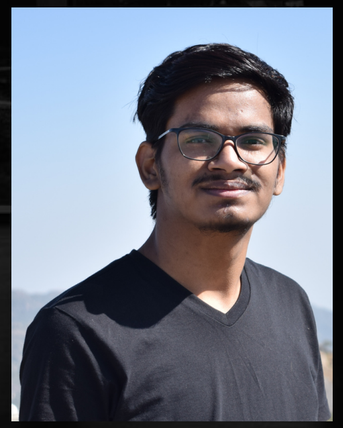
Aryan Kharbanda
IIIT Hyderabad



Hariom Vyas
SVNIT Surat

Sakshi Jain
SVNIT Surat

Nikhil E
IIIT Hyderabad



Rahul Singhadia
NIT Warangal

Prathamesh Bonde
DAIICT Gandhinagar

Chinmay Joshi
NIT Warangal

Teja

E-KYC

Problem Statement

KYC : Know Your Customer

A customer may need to submit his KYC documents across different departments and LOBs of the organisation. The complete process is manual, repetitive and time-consuming.

Our goal is to design a one-time e-KYC Document Repository to simplify the current KYC process thus reducing the time and effort (working-hours) that go into the process.

Flaws in Current System

- The current physical KYC procedure has become redundant
- Customer ends up submitting same documents across different LOBs repetitevly
- Consumes more time and energy
- The auditing of the documents is delayed
- This not only costs the customers but also costs the bank.

Solution

A blockchain-based Document Repository!

Webapp that facilitates the user to upload the document

Documents can be verified by the verifying team at the organisation

Once verified, the same documents can be used by different LOBs for different purposes.

User permission is required by the LOB to access the documents

The solution demands accessibility and transparency along with security.

Blockchain, as a technology, serves all these purposes and is the best fit for the solution

Why Blockchain?

Immutable

Data once added cannot be changed

Distributed

Data is distributed on different nodes

Instant Traceability

Any addition can be easily checked

No Down-Time

Even if one node is not working, others will work

Hack-Proof

Almost impossible to change data on blockchain

Crytographic-ally Secure

Each transaction is encrypted

Choosing the Blockchain Network

We made use of the Ethereum Blockchain network for this project

(We explored and built prototypes of different Blockchains, covered later in Application's Alternative Flows..)

Key Features

- Decentralized network :
  - not managed by any centralized authority
- Distributed public ledger :
  - every participant has an identical copy of the ledger
- Incorporates smart contract functionality :
  - code written for sending transactions
- Allows storage of data and creation of decentralized applications

Tech Stack

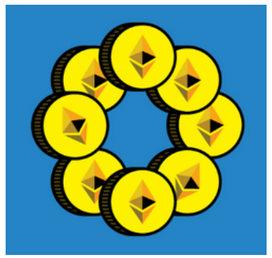
ethereum

SOLIDITY

METAMASK

IPFS

Ganache



Ropsten TestNet

HEROKU

mongoDB.

mongoose

Express JS

React JS

node JS

Actors

Three types of users are involved in KYC process:

- Customers - Request For KYC
- Admins :
  - Verifier - Approve/Reject KYC Request
  - LOBs - Access KYC Documents when needed

Assumptions

Customers can upload documents only in the pdf format.

Customer uploads single copy of a particular document (eg Aadhar); Can only re-upload a document if the previous one is rejected.

The verification process is single-step, a verifier does it manually and it is centralized for the bank.

The credentials for all actors will be provided by the organisation when deployed, register page is added for the sake of demo and ease in testing.

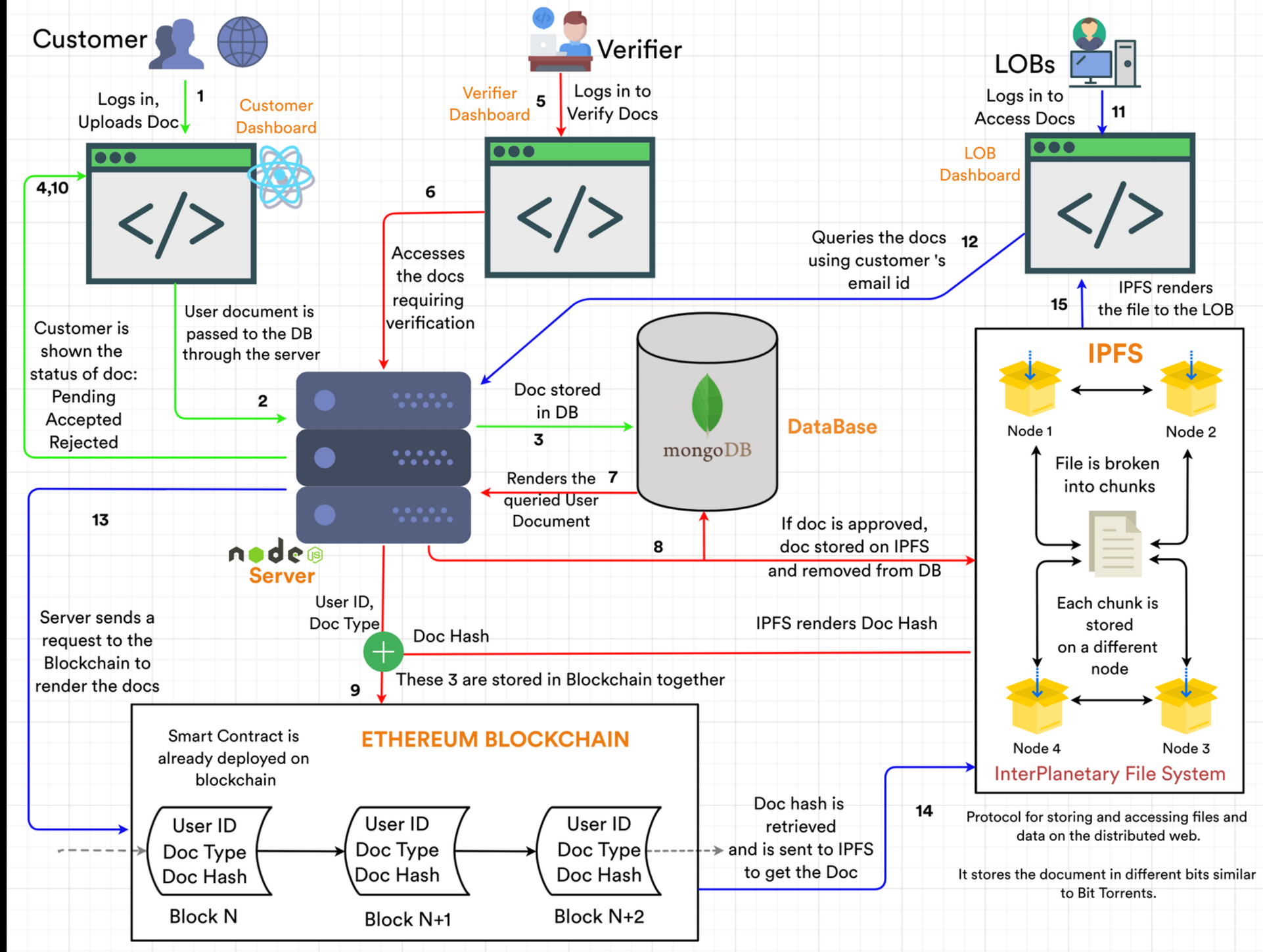
There can be multiple verifiers in the verifying team, but to verify a doc, no need for multiple admins to approve.

Only one organization (Wells Fargo) is involved in the KYC process. However, multiple LOBs within the organization can access a customer’s documents.

There can be multiple admins under a department (LOB), and if an LOB is granted permission by the customer, all admins in that LOB have access to the docs

Admins (Verifiers, LOBs) have access to all customer's emails, can search using email to see their document records. (LOBs see only documents specific to their requirement)

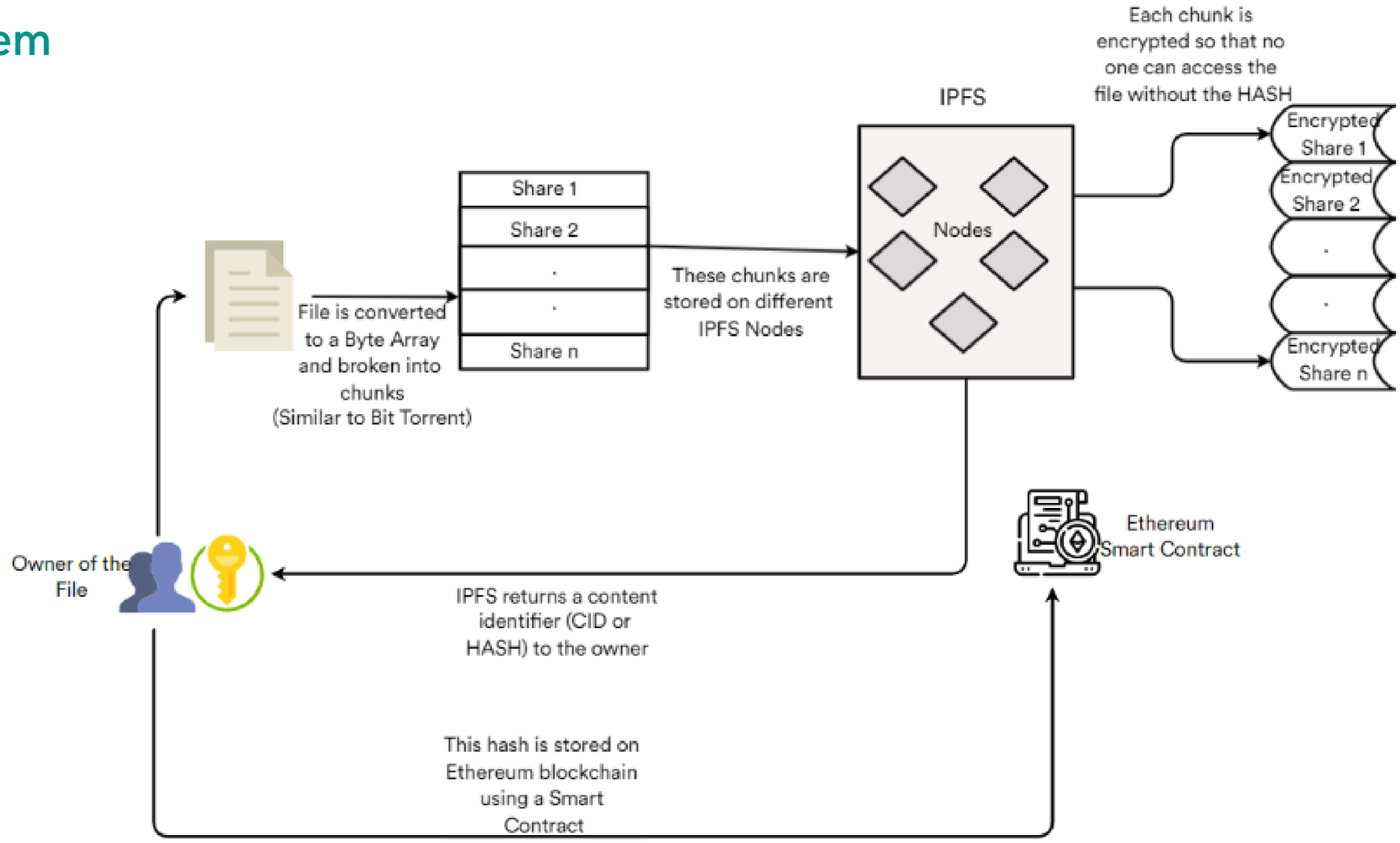
System Architecture Diagram



What is IPFS?

InterPlanetary File System

- A peer-to-peer hypermedia protocol for storing and accessing files and data on the decentralized web.
- It works as a hard drive for blockchain and Web3
- Analogous to Bit-Torrent
- Follows a principle of sharding the file
- Each file will have a unique identifier (or hash)
- If the file is the same, then the hash will be the same

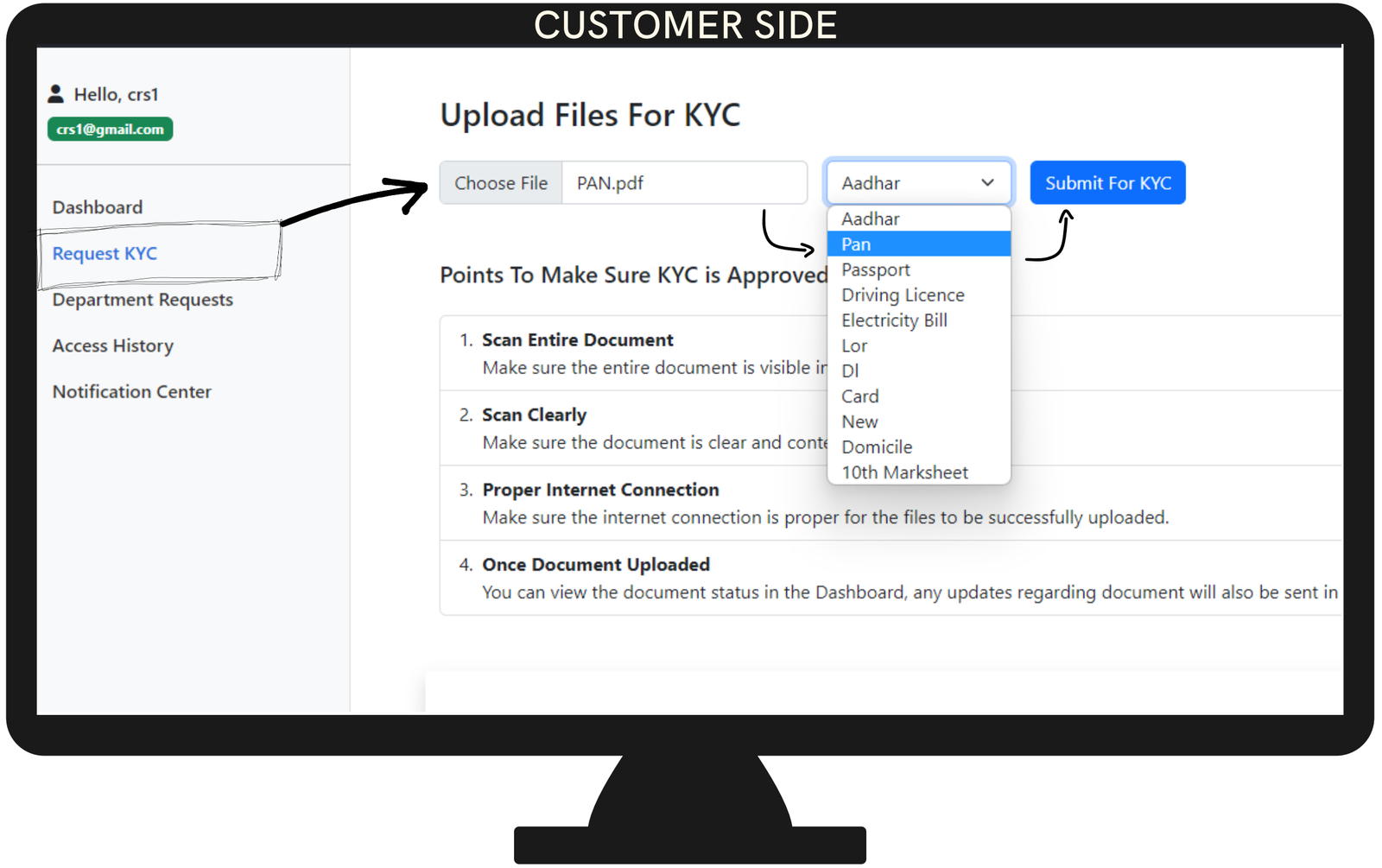


App Flow : Documents sent for KYC

1. Customer Uploads Docs

Customer selects the type of document to be uploaded, and it is then stored in mongoDB database via GridFS.

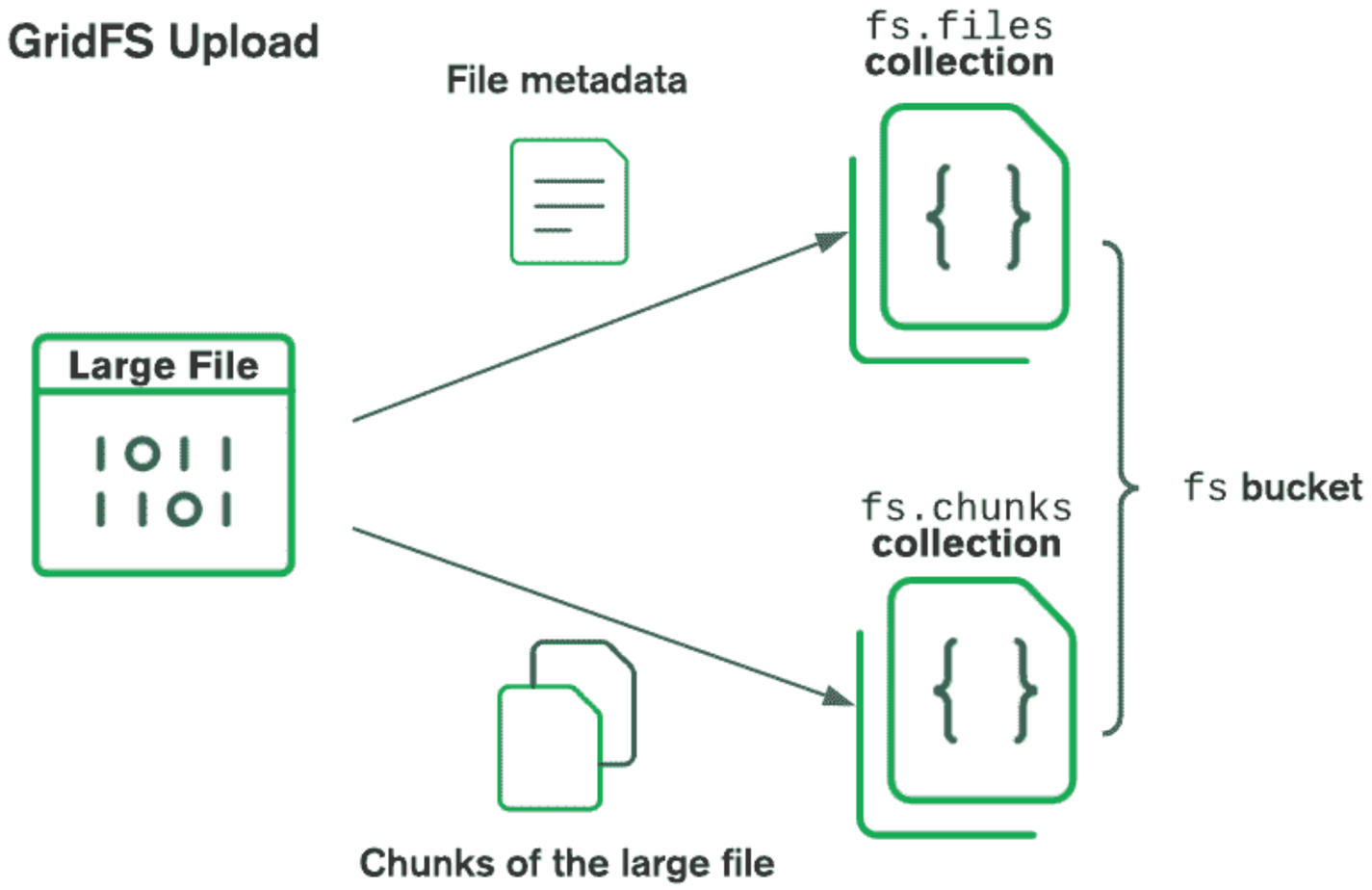
Assumption: Customer cannot re-upload a document already verified for KYC



Database

GridFS

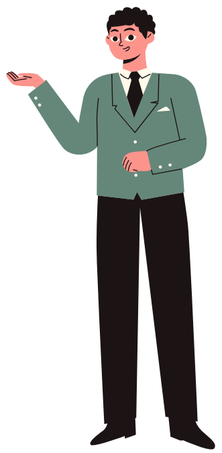
- Specification for mongoDB to store large files.
- GridFS divides files into small chunks.
- Each chunk is stored separately.

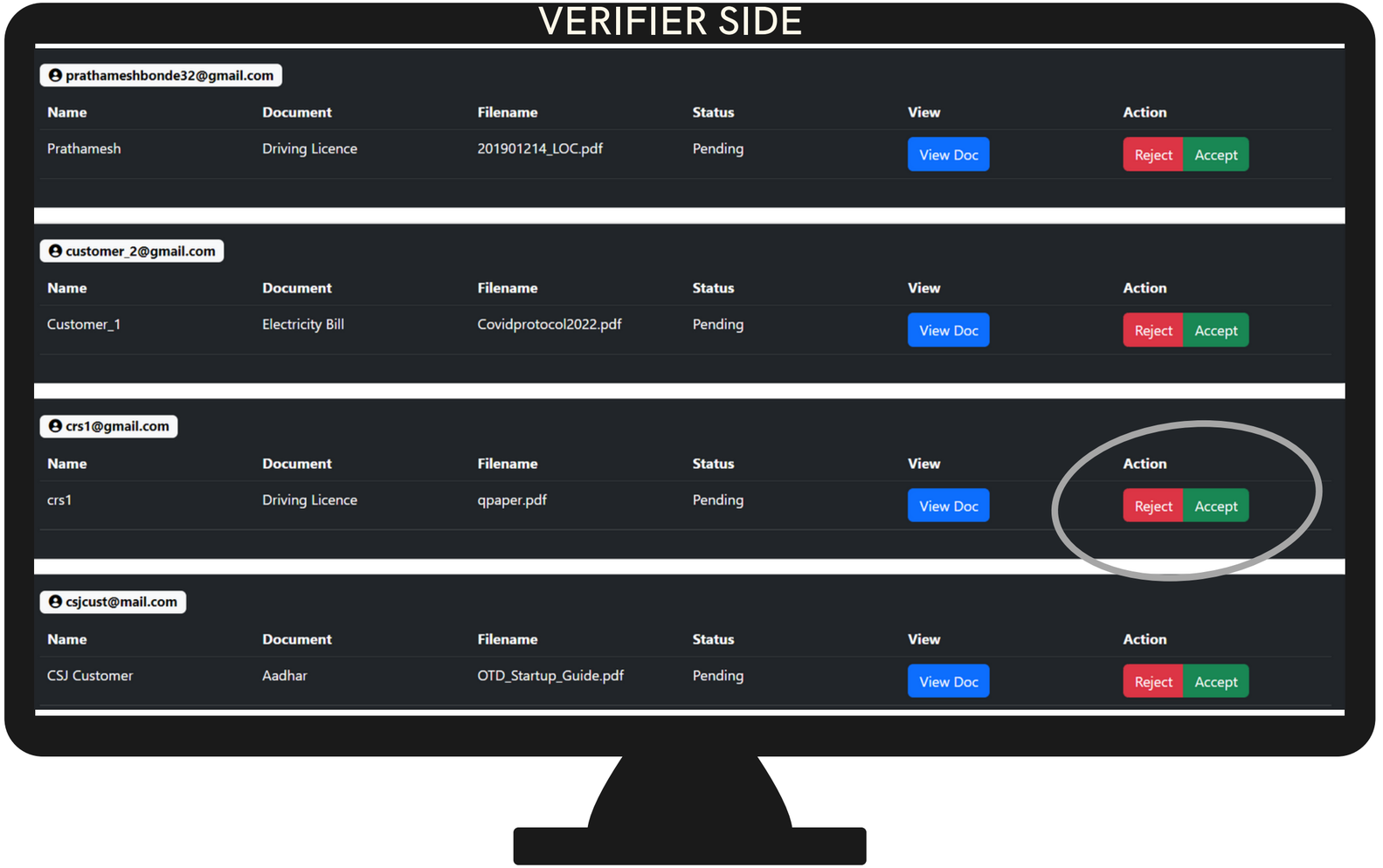


App Flow : Documents sent for KYC

2. Verifier gets the KYC Requests

The verifier sees all the pending requests put by customers, for KYC approval.





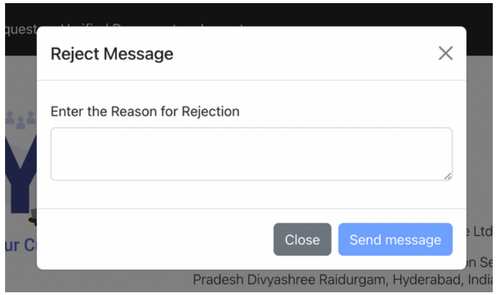
App Flow : Documents sent for KYC

Verifier Checks the Docs

- Document retrieved from GridFS
- Verification is manual
- Can ACCEPT or REJECT it

- Document -stored-> IPFS
- IPFS -returns-> Hash
- Hash -stored-> BLOCKCHAIN

- Enters reason for rejection



In both the cases,

- Document is deleted from database (gridFS), but not the doc record
- Doc status is changed to Accepted/Rejected [reflected in "Customer Dashboard"]
- Customer is notified [reflected in "Notification Center"]
- Email sent to Customer

KYC

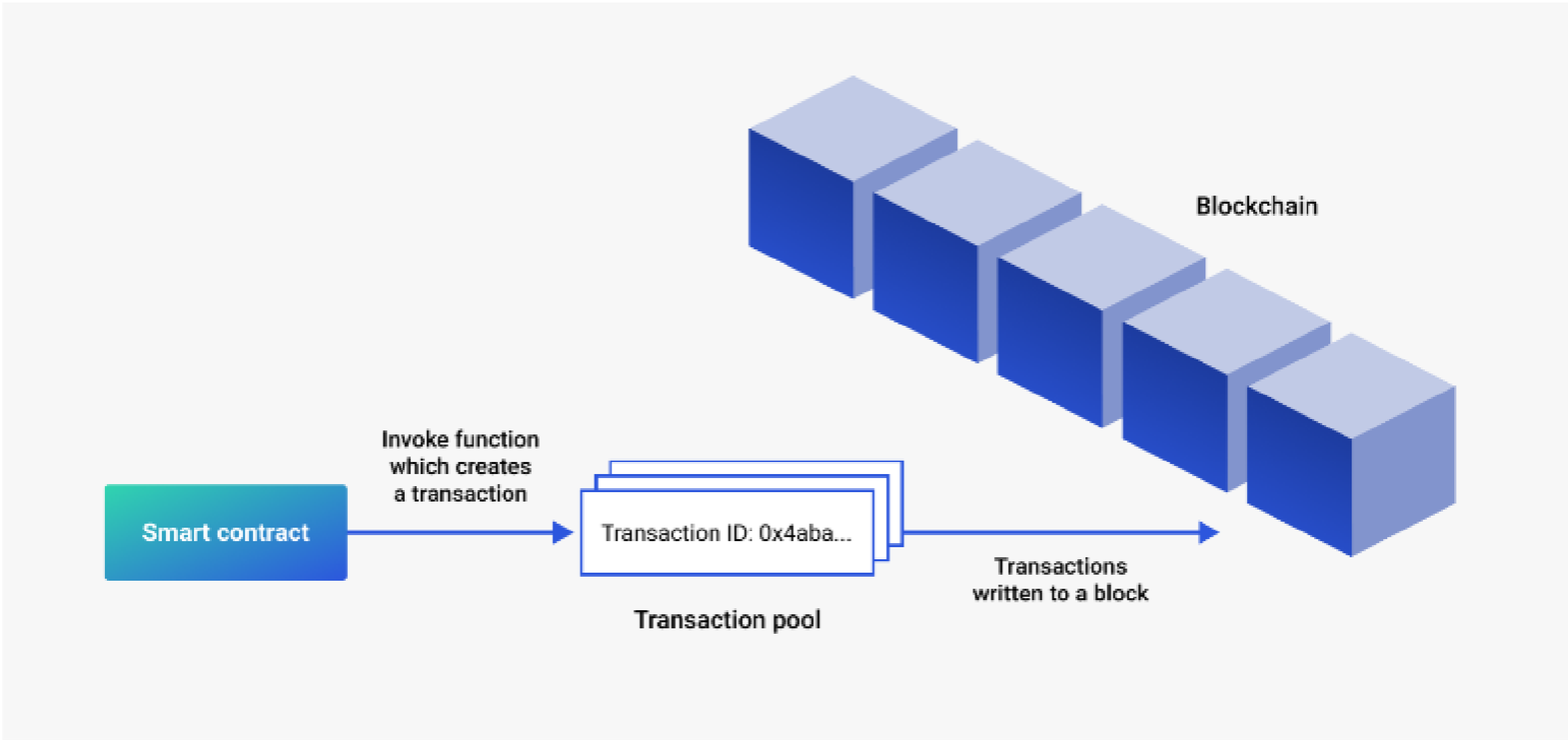
KYC Notification - Document (Passport) has been approved by the Verifying Team!

Blockchain

Smart contract (written in Solidity) is used to send the function calls as transactions to the blockchain.

```
struct Document{
    bytes32 email;
    bytes32 docType;
    string docHash;
}
```

The email and docType are hashed using the keccak256 hash function




```
function addDocument
```

arguments (email, docType, docHash); Sent as a transaction to the blockchain.

```
function viewDocument
```

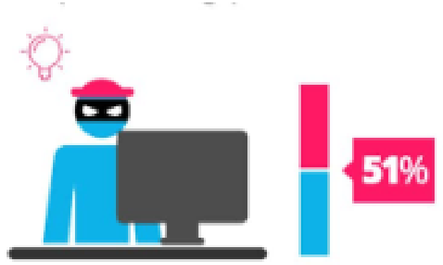
arguments (email, docType); Used to query the blockchain for the docHash.

Blockchain

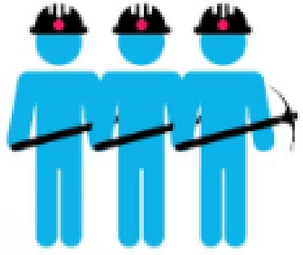
Validation using Proof-of-Work Consensus

- "Miner" nodes validate all transactions.
- Upon majority validation, they are packaged into a block.
- The hash for the block is calculated by a miner before its addition to the blockchain and they are rewarded Ether for it.
- This can then be used to account for the costs of sending transactions to the blockchain.

To add each block to the chain, miners must compete to solve a difficult puzzle using their computers processing power.



In order to add a malicious block, you'd have to have a computer more powerful than 51% of the network.



The first miner to solve the puzzle is given a reward for their work.

App Flow : LOB accesses Docs

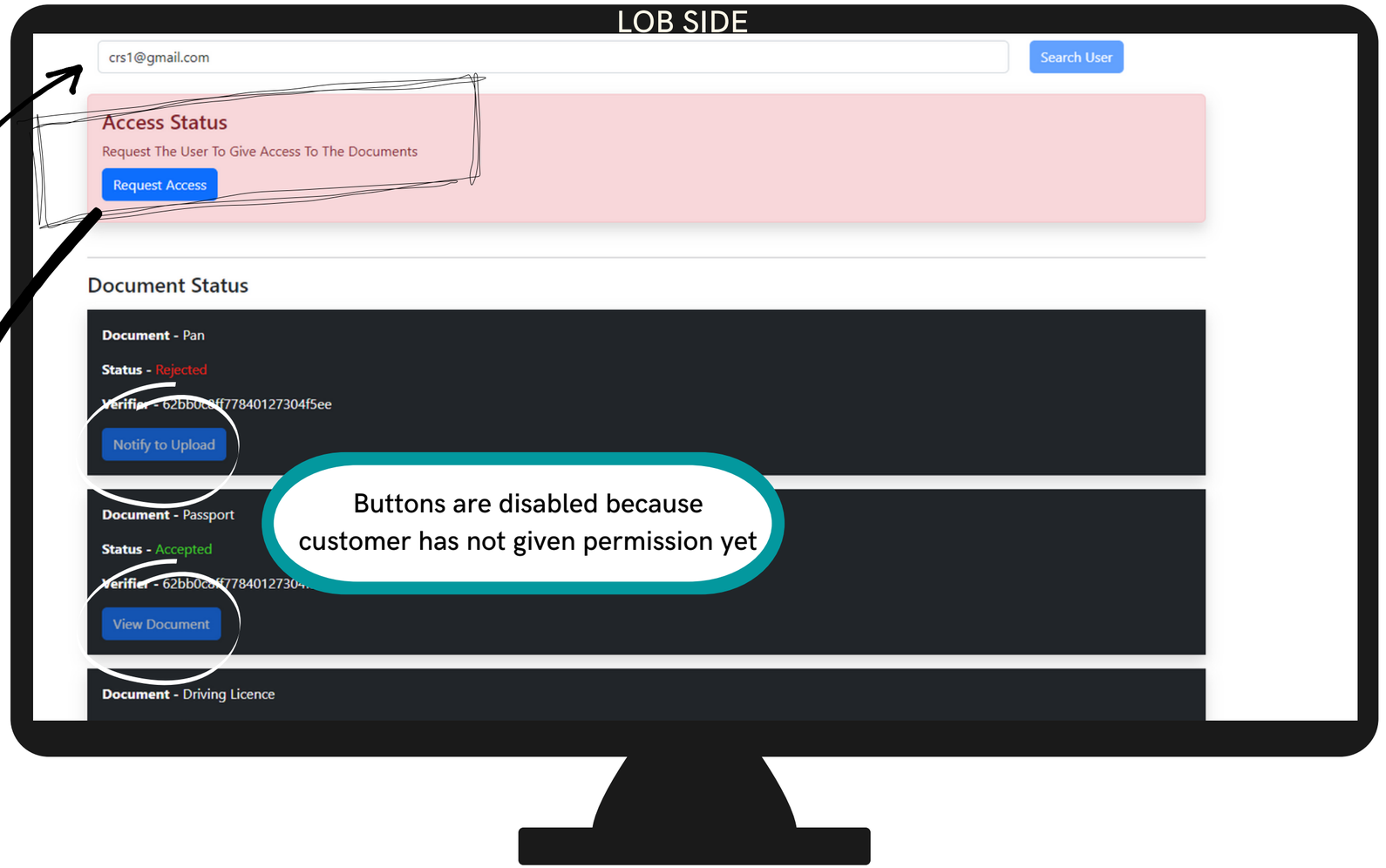
LOB Searches Customer

- LOB searches using email, but can see only those documents specific to that LOB

eg. Home Loan LOB sees only Aadhar and Pan

- LOB needs customer permission before viewing the actual document

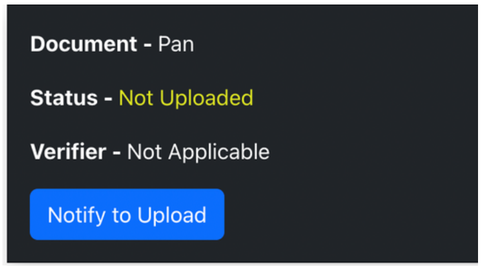
- Document status can be 4 types:
  - Not Uploaded
  - Pending Approval
  - Rejected
  - Accepted



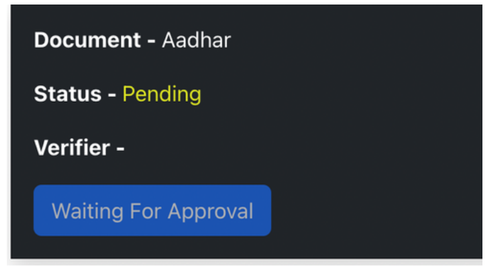
App Flow : LOB accesses Docs

Document Status

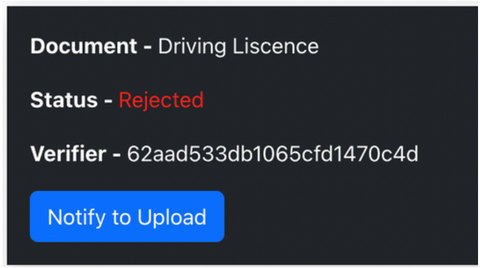
LOB SIDE



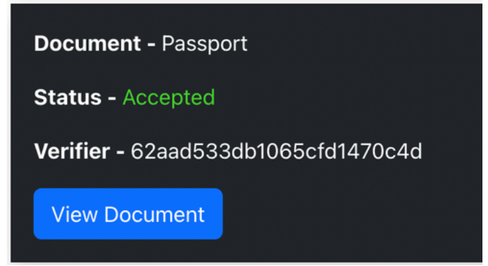
LOB can notify customer to Upload the doc



LOB needs to wait for verifier's approval



LOB can notify customer to re-Upload the doc



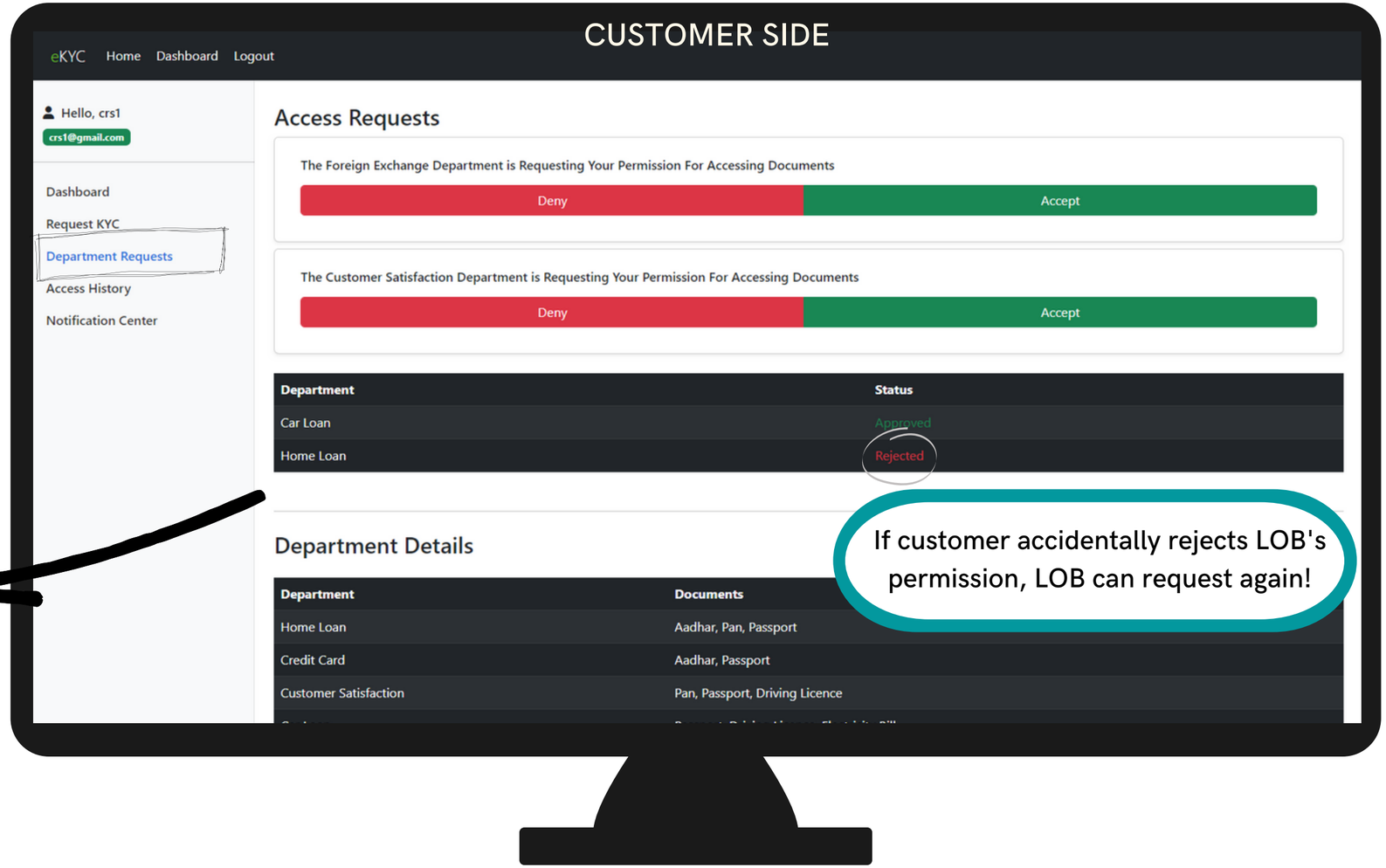
LOB can now view the document via the magic of Blockchain and IPFS!

App Flow : LOB accesses Docs

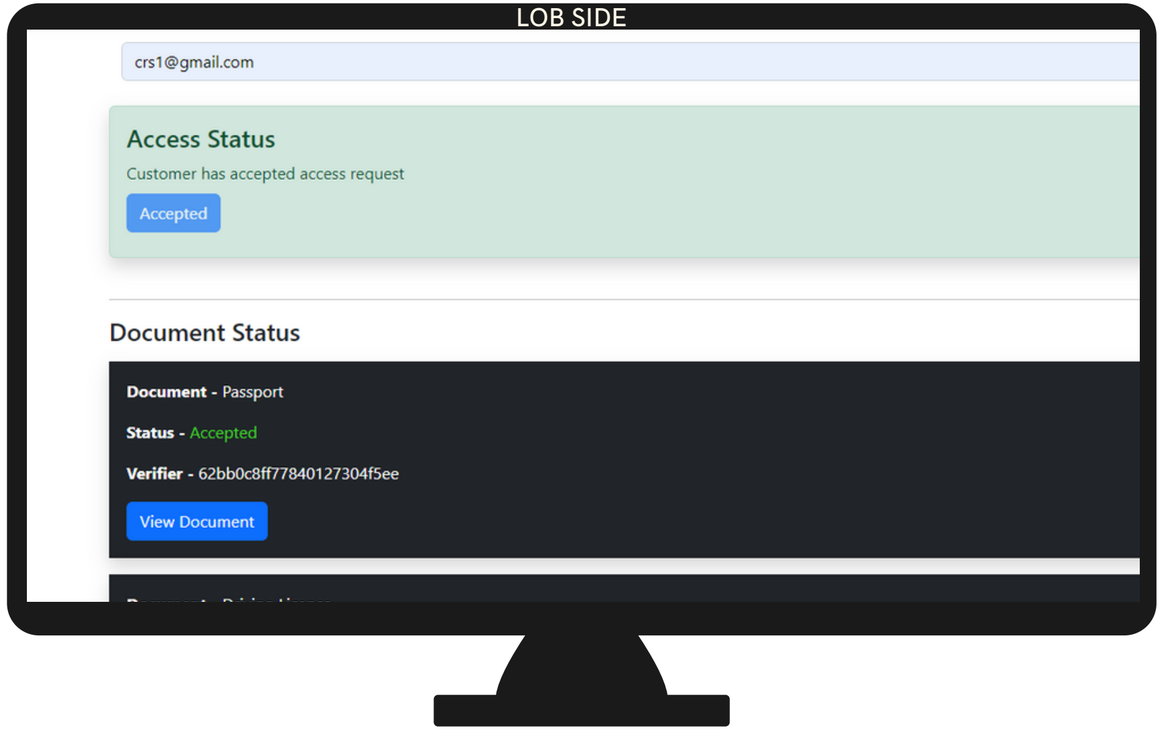
BUT, LOB needs customer permission before viewing the document

Department Requests

- Customer can Accept/Reject LOB's permission for doc access
- Customer can also see list of LOBs who have access to their documents



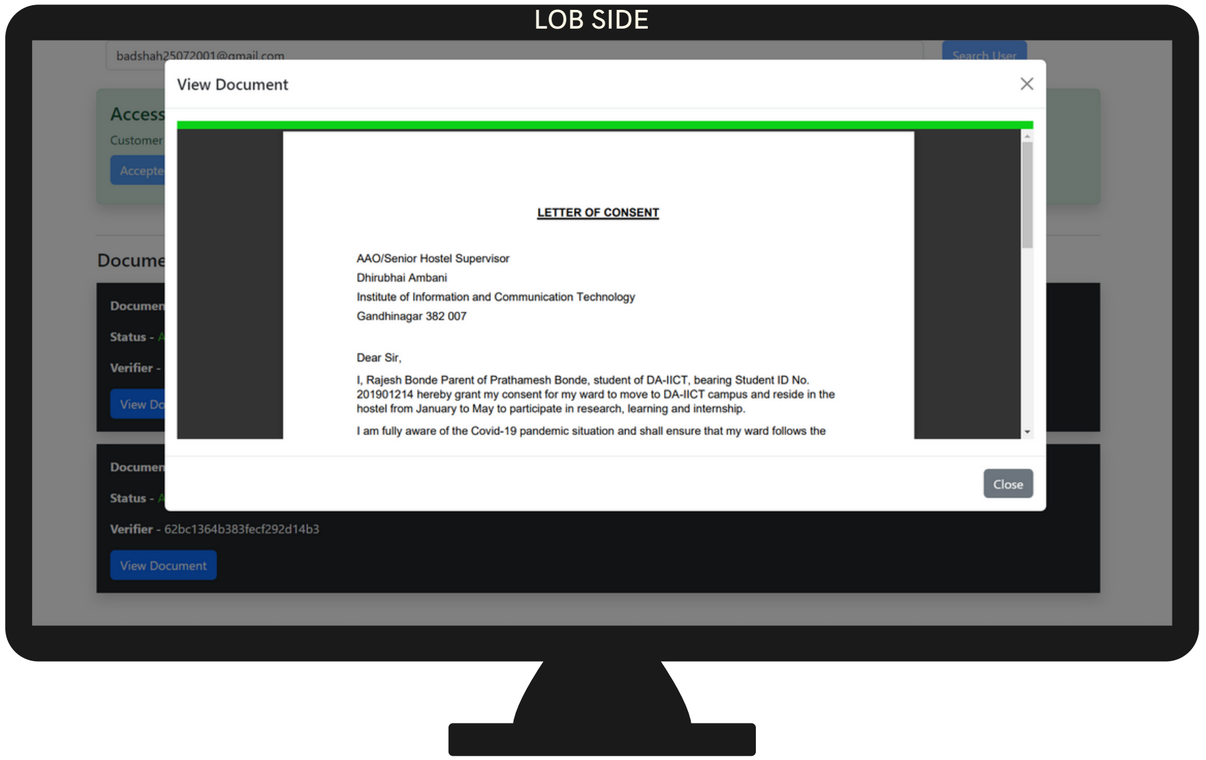
App Flow : LOB accesses Docs



STEPS in viewing the doc:

- First, the hash is retrieved from the Ethereum blockchain
- Using the Hash, different IPFS nodes retrieve chunks of the file, and then it's stitched together and shown as the pdf file

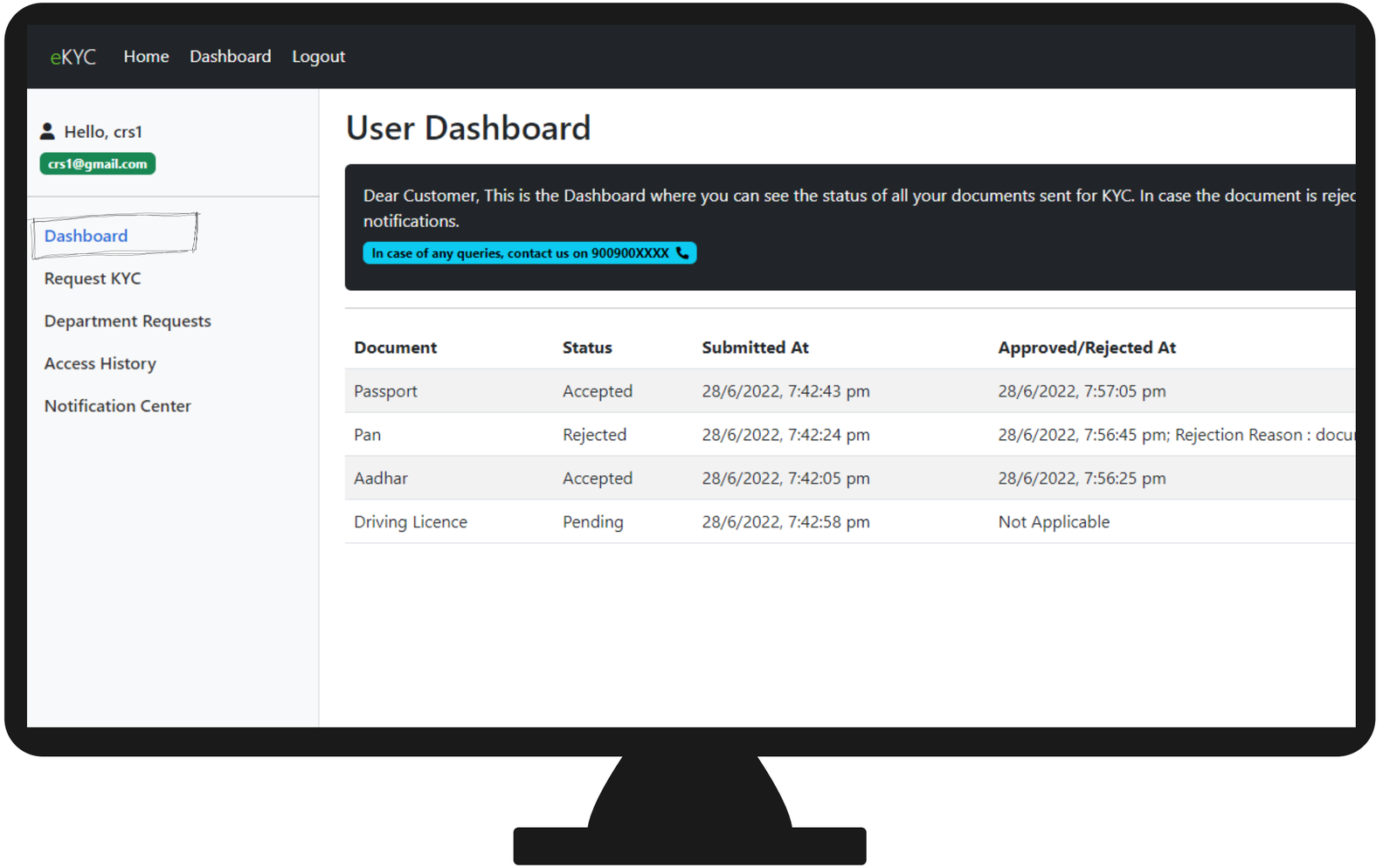
Now that LOB has permission, he can view the document!



App Features : Customer

Customer Dashboard

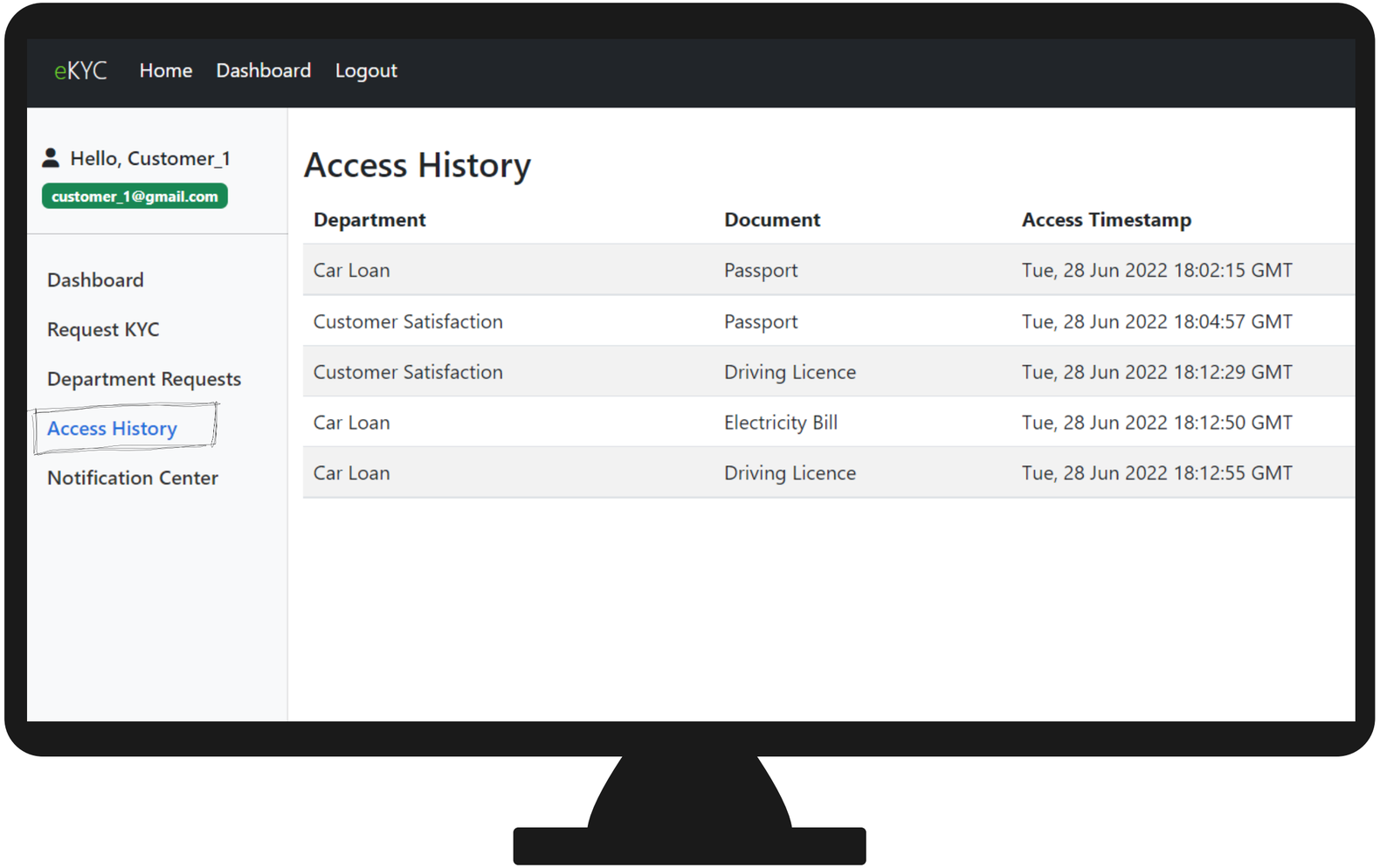
Customer can view all their documents and their verification status and timestamps.



App Features : Customer

Access History

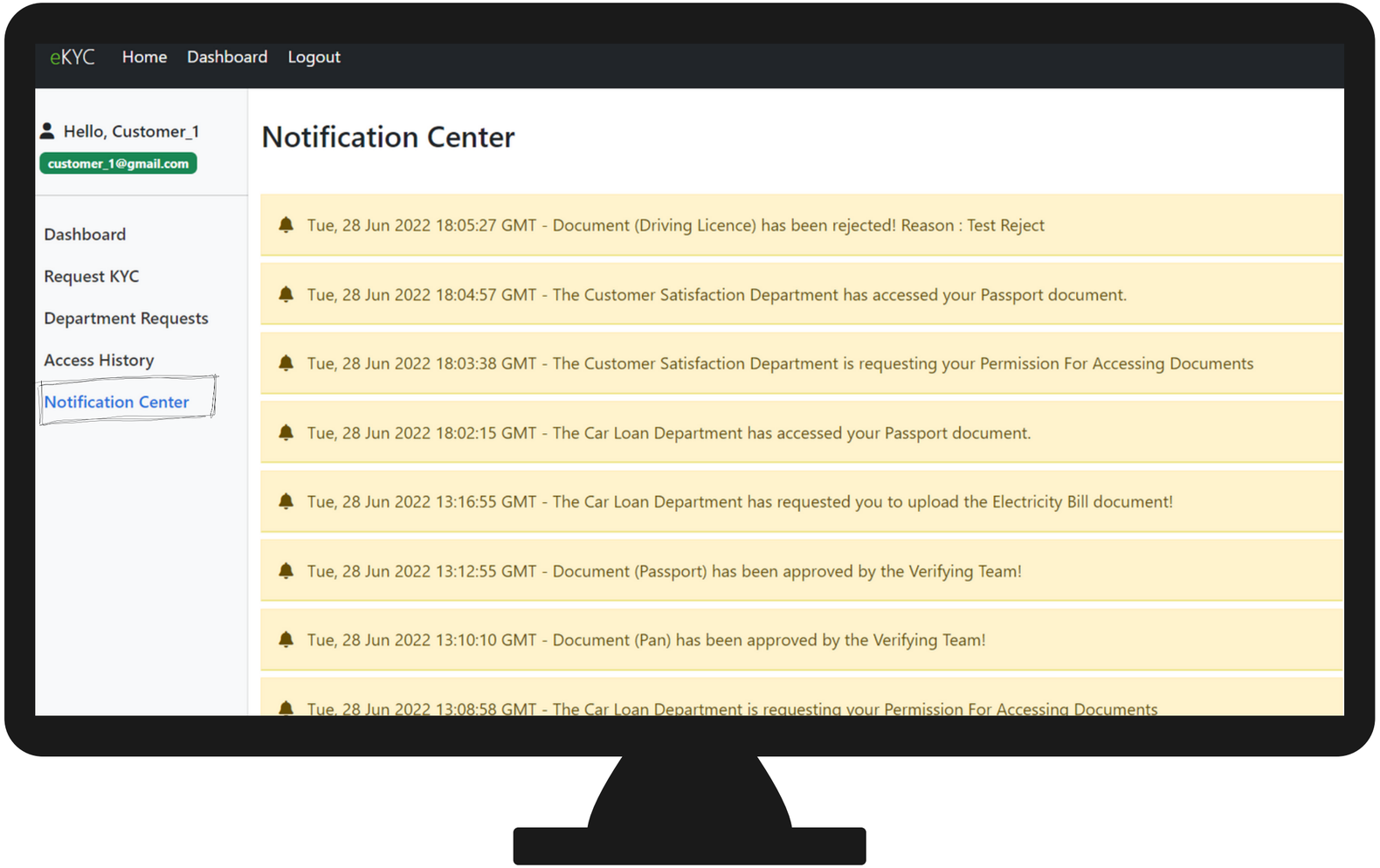
Customers can view list of all departments that have accessed their documents and at what time



App Features : Customer

Notification Center

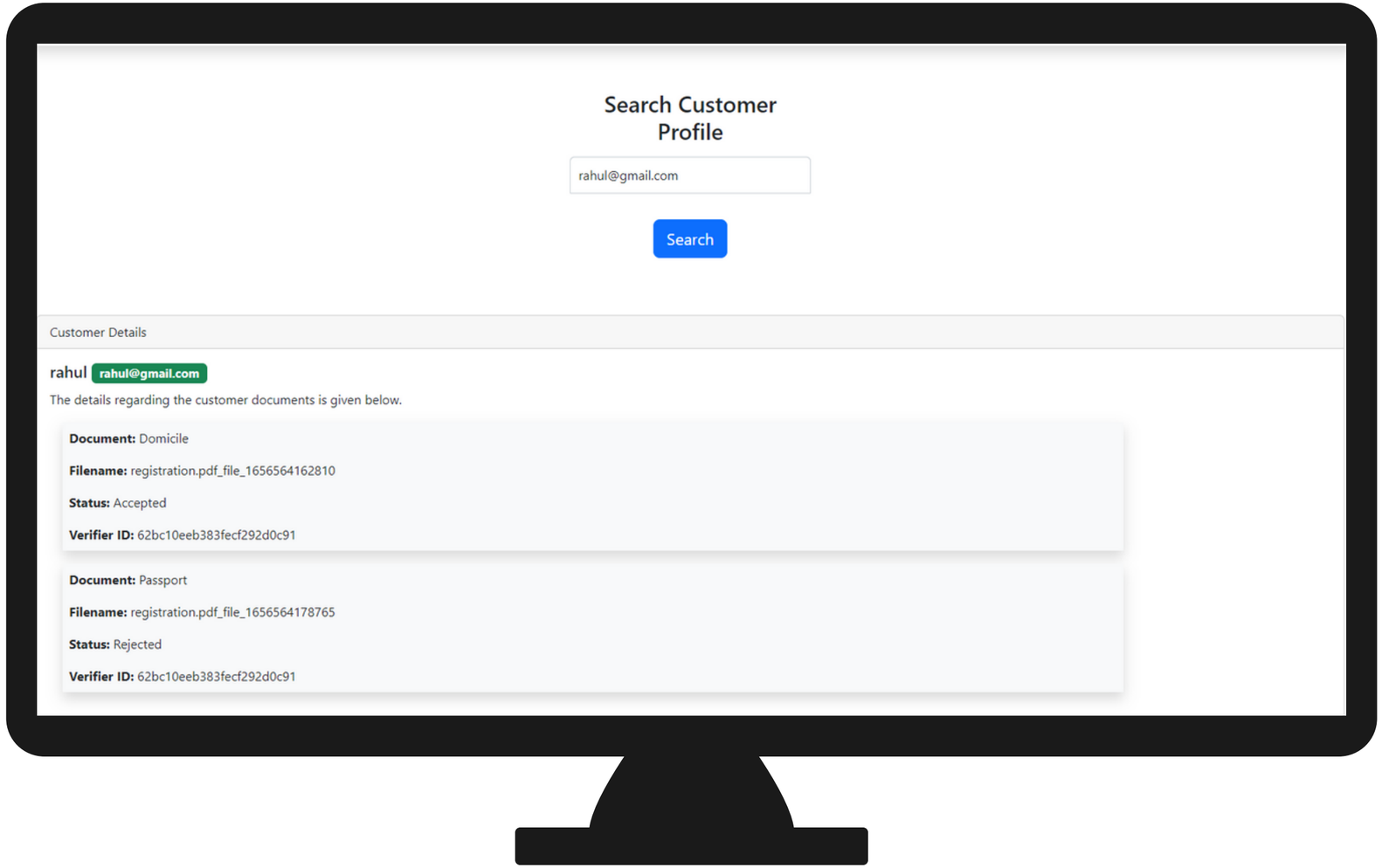
The customer can view all notifications from Verifier and LOBs here (newest on top)



App Features : Verifier

Search customer profiles

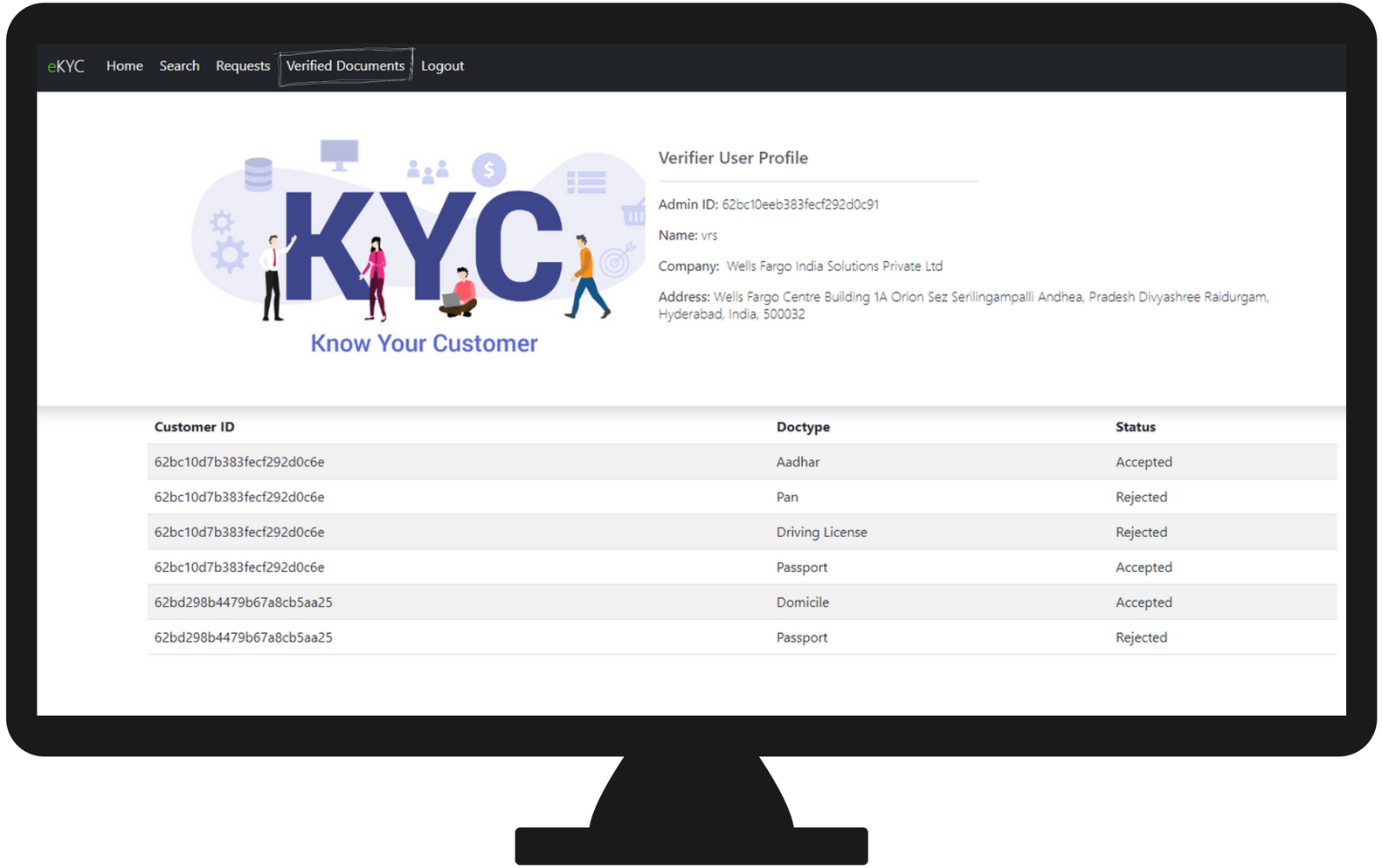
Verifier can also search a customer to see their documents' status, filename, who verified it etc...



App Features : Verifier

View verified documents

A verifier can view the list of all the documents he has verified, with status, doctype and customer details.

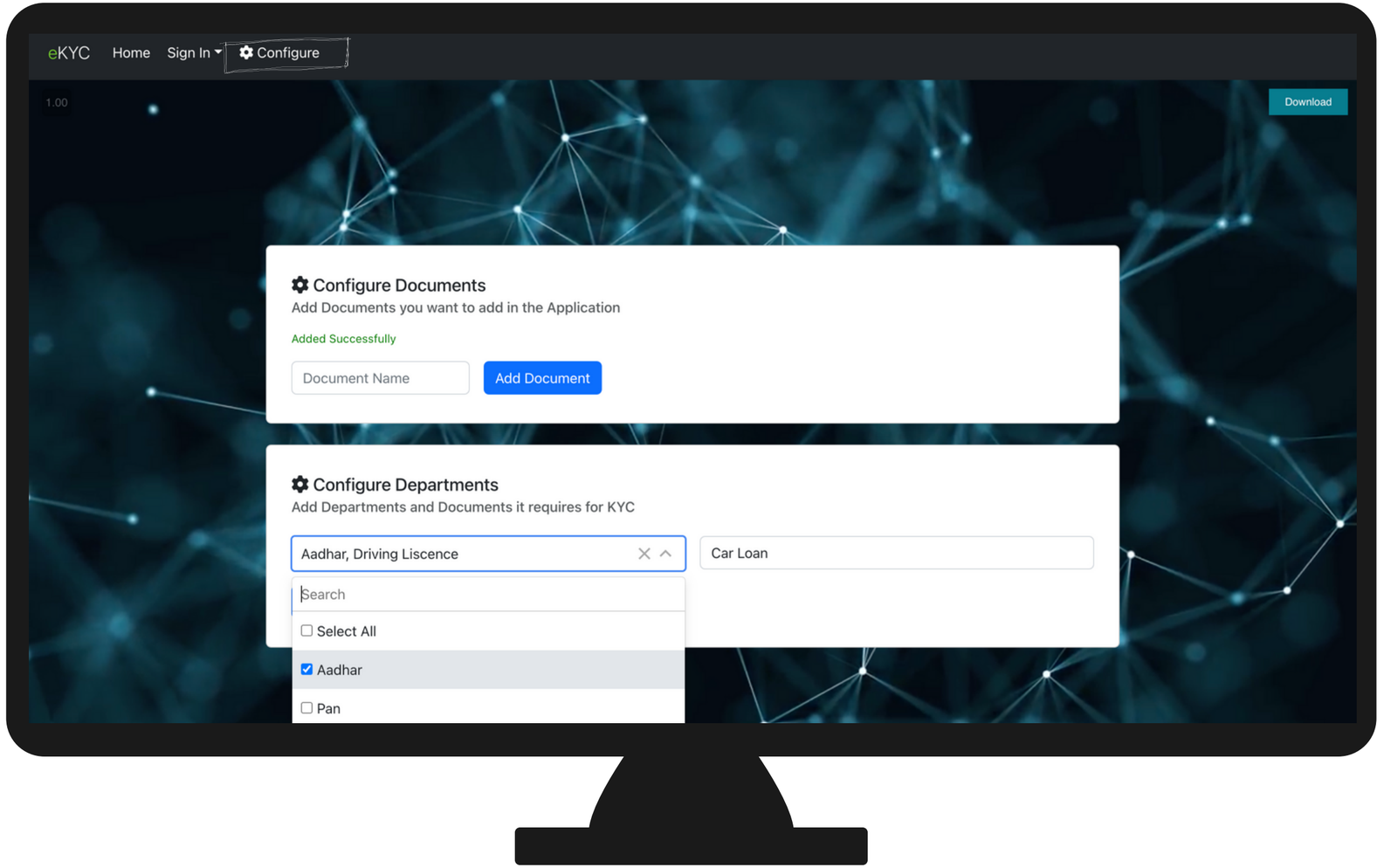


App Feature

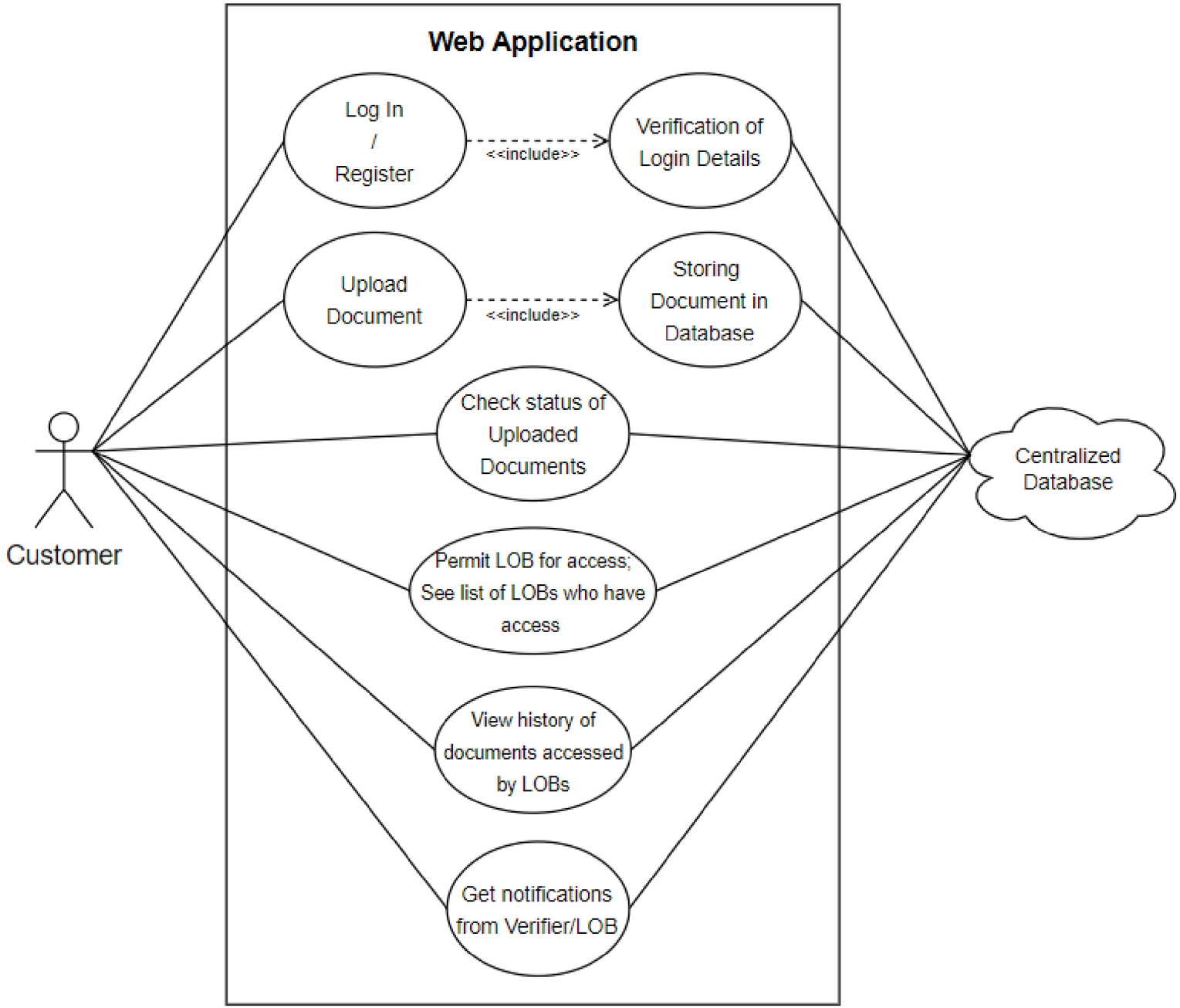
Dynamic LOBs and DocTypes

The number of types of documents in the webapp, along with LOBs and which documents they require, are not hard coded!

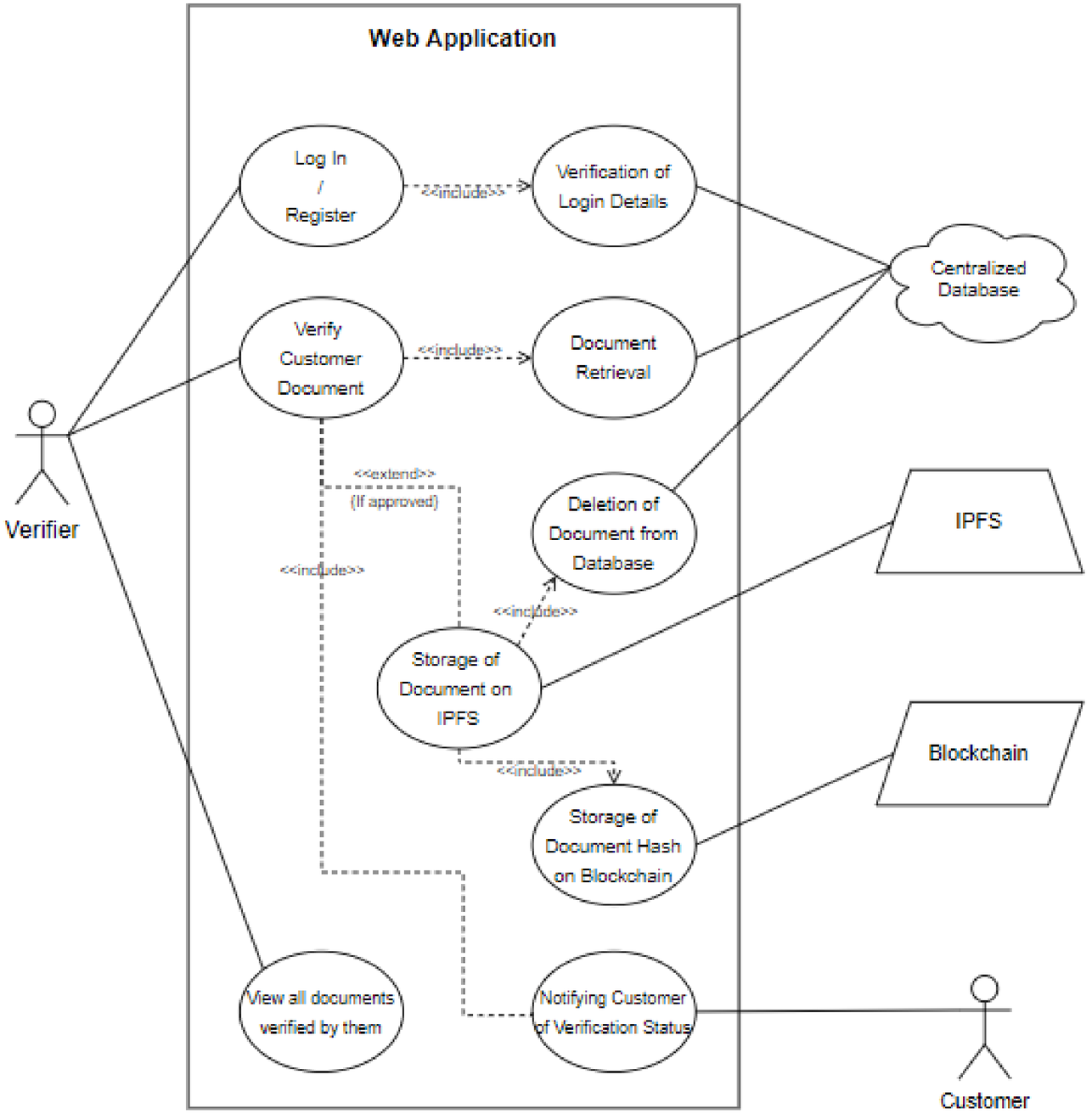
They can be changed in the Configure page in the App.



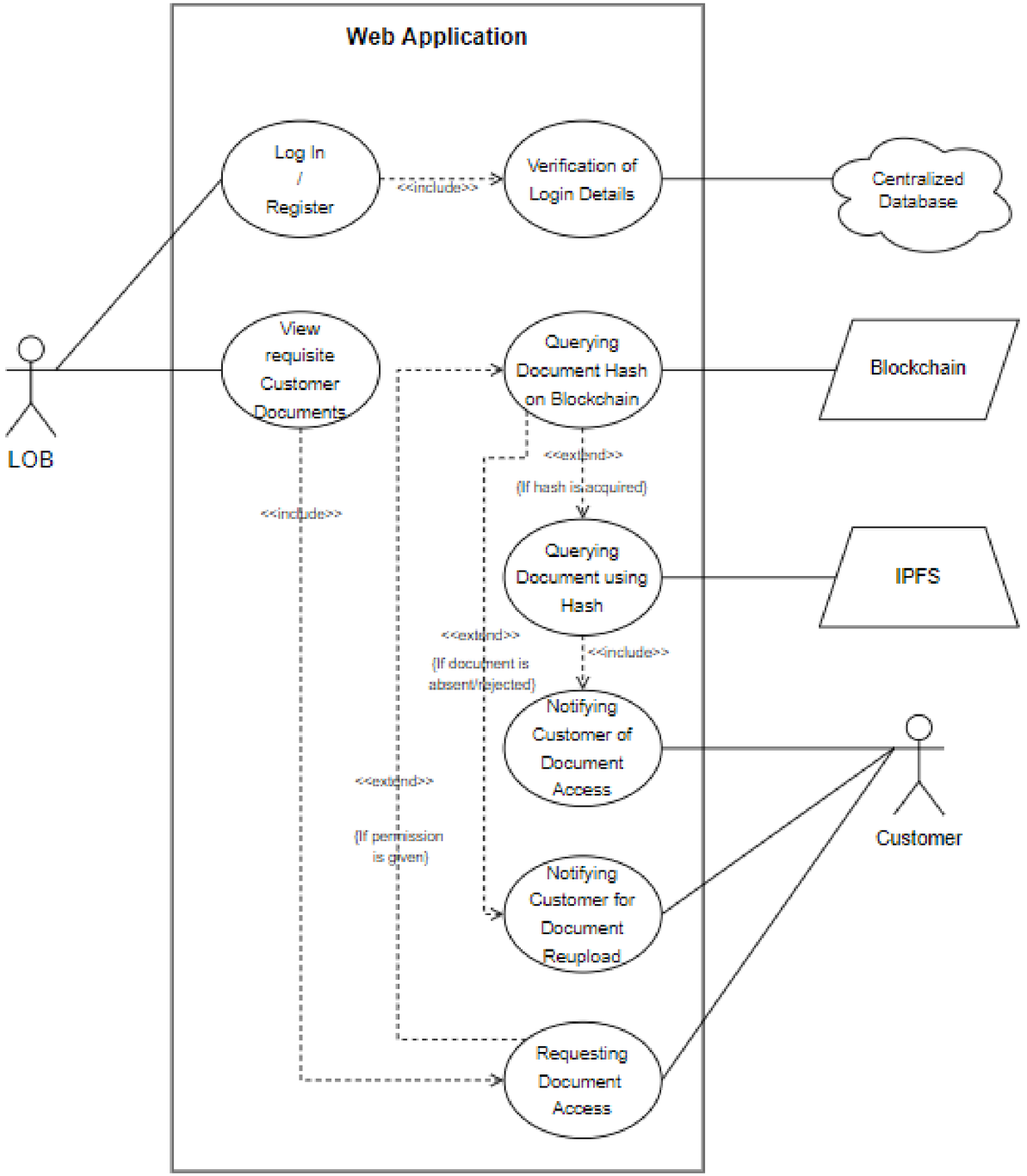
Customer Use Case Diagram



Verifier Use Case Diagram



LOB Use Case Diagram



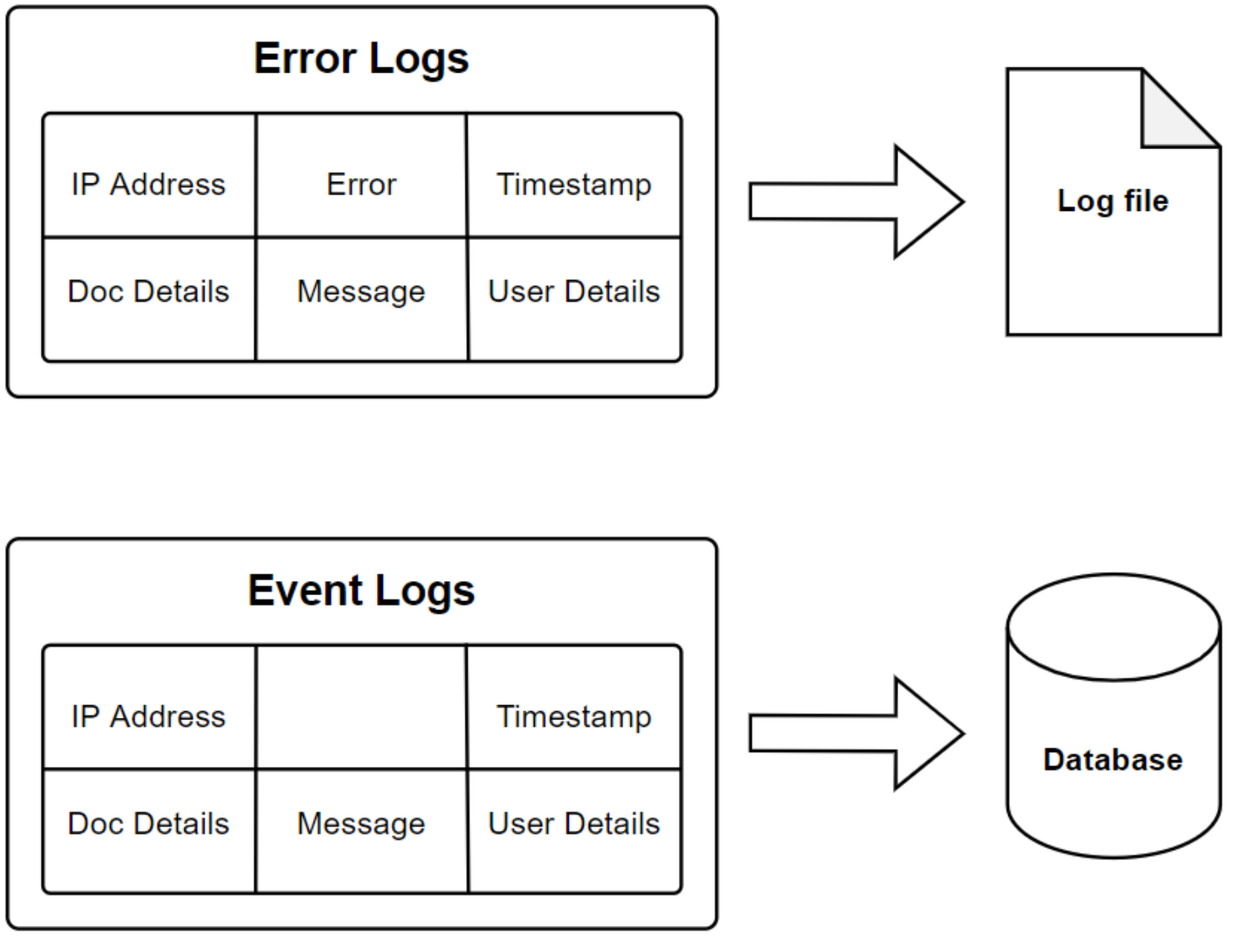
Logging

Winston Library

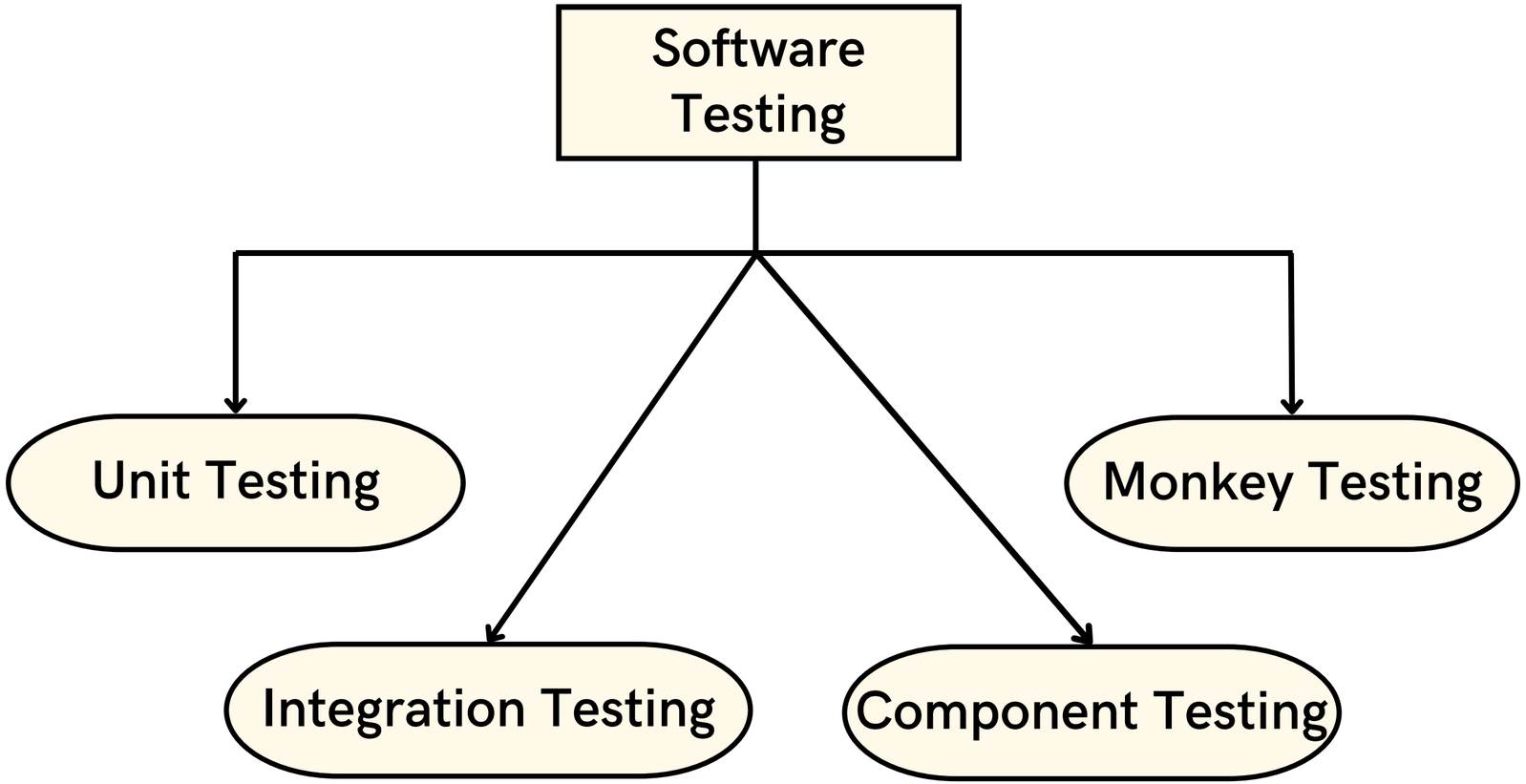
Winston library has been used for logging which is a widely used inbuilt npm package.

Advantages:

- Multiple transports
- Multiple logging levels
- Comprehensive documentation
- Good community support



Testing


Software Testing
Unit Testing
Monkey Testing
Integration Testing
Component Testing

Deployment

Ganache-CLI : Blockchain (local)

It is a private local network on your system to test the ethereum smart contracts with fake gas costs and dummy ether.

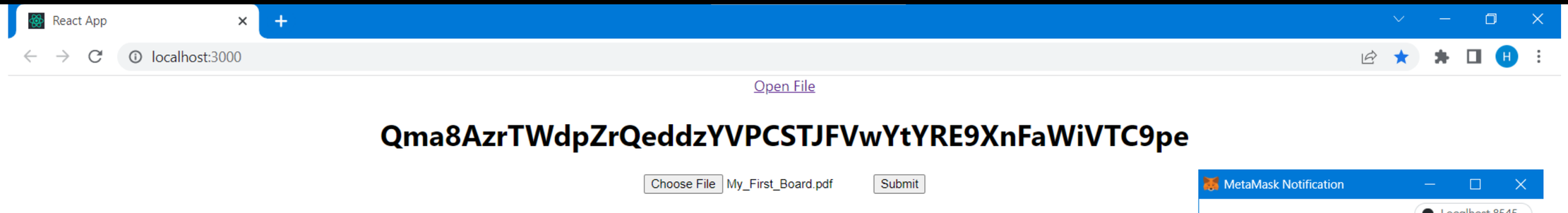
Ropsten Testnet : Blockchain online host

It permits testing of blockchain development prior to mainnet release. It is similar to the mainnet and identical to other testnets, but obtaining dummy currency on Ropsten was a bit easier than other other testnets.

Heroku : Webapp

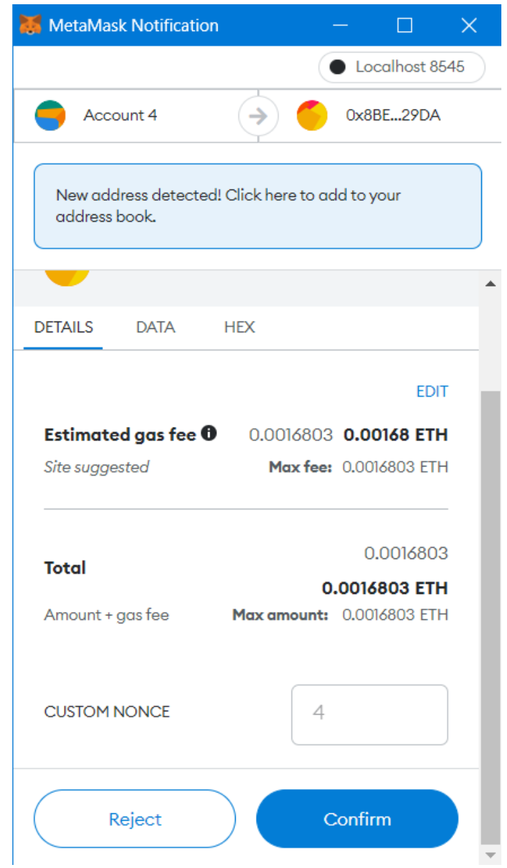
Heroku manages app deployments for free, using the Git Version Control System. The Heroku Command Line Interface creates a new application along with an empty git repo to move the project to that repository.

Hash Generation and Storage Prototype



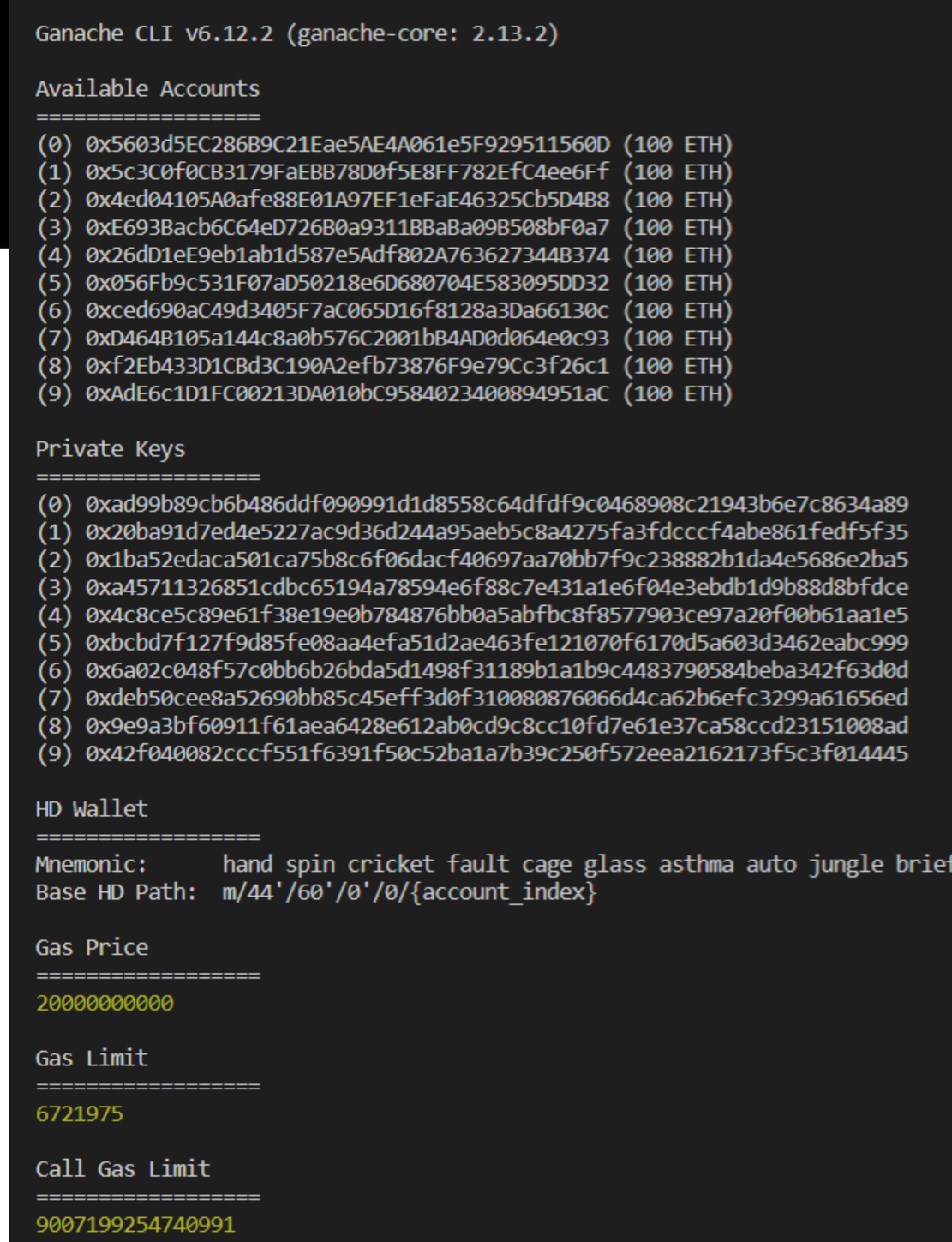
A prototype of Hash Generation and storage on Ethereum blockchain using the following technologies:

- Truffle Suite
- IPFS
- Ganache-CLI
- MetaMask Wallet
- React.js and WEB3.js



Ganache-CLI

- Provides 10 dummy Ethereum accounts
- 100 Ethers balance in each account
- Works only on LocalHost and is widely used for testing purposes.
- Their private keys can be used for paying the gas fees for **deploying the contract** and adding data to it.
- First, the smart contract is deployed here, and then the account address and contract address in our Webapp are changed



Blockchain Deplyment: Ropsten

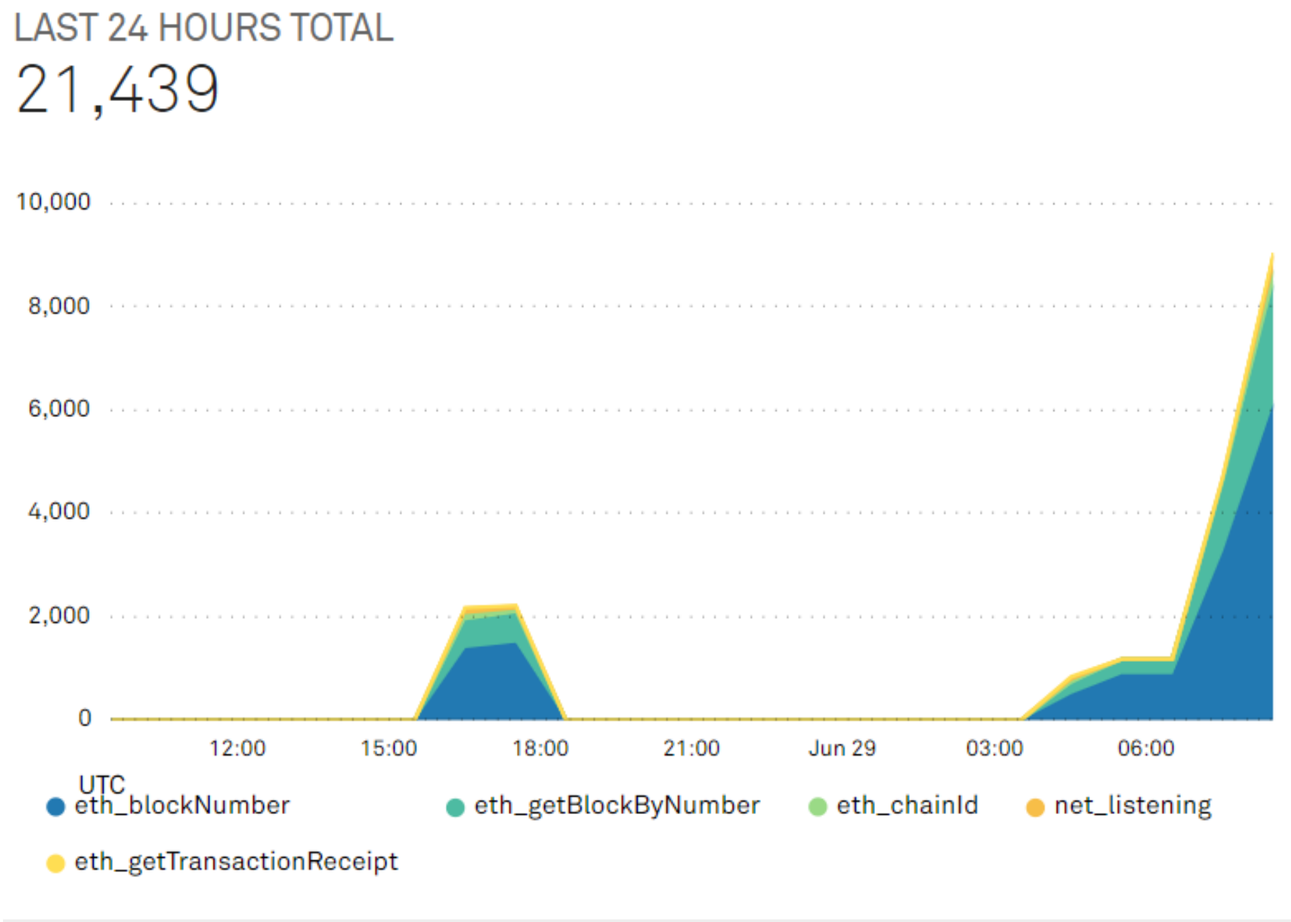
net_listening: Checks if Web3 is connected to the website

Add Document function calls:

- eth_blockNumber
- eth_getTransactionReceipt

View Document function calls:

- eth_chainId
- eth_getBlockByNumber



- eth_blockNumber: Returns the number of the current Ethereum block
- eth_getBlockByNumber: Returns the block of the requested number
- eth_chainId: Returns the chain id in which the requested block exists
- eth_getTransactionReceipt: Returns the receipt of the transaction on the blockchain

Ethereum Transaction Receipt

Transaction Hash:	0x31a9fcf8908889bc0cea240e627681e81e0805205313073a08867b5bdb36c701
Status:	Success
Block:	12521801 1290 Block Confirmations
Timestamp:	4 hrs 24 mins ago (Jul-04-2022 06:24:24 AM +UTC)
From:	0x595f4030575accd032dad62b9838911917a5d5fd
To:	Contract 0x9c8c95dda33b52c25227c11551a89f3c36874562
Value:	0 Ether ($0.00)
Transaction Fee:	0.000000016079733135 Ether ($0.00)
Gas Price:	0.000000000000065535 Ether (0.000065535 Gwei)
Gas Limit & Usage by Txn:	1,048,575 | 245,361 (23.4%)
Gas Fees:	Base: 0.000000007 Gwei
Others:	Txn Type: 0 (Legacy) Nonce: 72 Position: 34
Input Data:	


```
0x36a8915b000000000000000000000000000000000000000000000000000000000000006000000000000000000000
00000000a00000000000000000000000000000000000000000000e00000000000000000000000000000000000000
00000000001073616b73686940676d61696c2e636f6d000000000000000000000000000000000000000000000000
0000000000006416164686172000000000000000000000000000000000000000000000000000000000000000000
0000000000002e516d556848364b4c5542364456527a7033443572366836376b6e525a47314d56504a465758783
000000000000000
```


View Input As | Decode Input Data

This is our metamask account so we can decrypt the transaction

Data
jainsakshi9301@gmail.com
Passport
Qma8AzrTWdpZrQeddzYVPCSTJFVwYtYRE9XnFaWiVTC9pe

Ropsten : Request Analytics

METHOD	NETWORK	REQUESTS VOLUME	SUCCESSFUL REQUESTS (%)	FAILED REQUESTS (%)
eth_blockNumber	Ropsten	6,138	100.00%	0.00%
eth_blockNumber	Ropsten	3,279	100.00%	0.00%
eth_getBlockByNumber	Ropsten	2,304	100.00%	0.00%
eth_blockNumber	Ropsten	1,506	100.00%	0.00%
eth_blockNumber	Ropsten	1,403	100.00%	0.00%
eth_getBlockByNumber	Ropsten	1,329	100.00%	0.00%
eth_blockNumber	Ropsten	897	100.00%	0.00%
eth_blockNumber	Ropsten	896	100.00%	0.00%
eth_getBlockByNumber	Ropsten	566	100.00%	0.00%
eth_getBlockByNumber	Ropsten	535	100.00%	0.00%

Ropsten : Transaction History

Txn Hash	Method ⓘ	Block	Age	From		To	Value	Txn Fee
0x009be77a3f0ceb715a...	Add Document	12487820	41 mins ago	0x595f4030575accd032...	OUT	0x9c8c95dda33b52c252...	0 Ether	0.000000013515
0xb4ed04c209649dc38f...	Add Document	12487698	1 hr 6 mins ago	0x595f4030575accd032...	OUT	0x9c8c95dda33b52c252...	0 Ether	0.000000013664
0x1c2b32d82df6bcf8e84...	Add Document	12487686	1 hr 9 mins ago	0x595f4030575accd032...	OUT	0x9c8c95dda33b52c252...	0 Ether	0.000000013331
0x16129bb872a11d8e80...	Add Document	12487641	1 hr 19 mins ago	0x595f4030575accd032...	OUT	0x9c8c95dda33b52c252...	0 Ether	0.000000013011
0x8e7ff51ed7ade325ac9...	Add Document	12487347	2 hrs 19 mins ago	0x595f4030575accd032...	OUT	0x9c8c95dda33b52c252...	0 Ether	0.000000012831
0xdff012bce37e8846a01...	Add Document	12486488	5 hrs 16 mins ago	0x595f4030575accd032...	OUT	0x9c8c95dda33b52c252...	0 Ether	0.000000012964
0x73f76f7ba6bc812c099...	Add Document	12483096	17 hrs 2 mins ago	0x595f4030575accd032...	OUT	0x9c8c95dda33b52c252...	0 Ether	0.000000012635
0xd44f5f3870404342d2b...	Add Document	12483024	17 hrs 17 mins ago	0x595f4030575accd032...	OUT	0x9c8c95dda33b52c252...	0 Ether	0.000000012311
0x9da9e9ff55528d4336d...	Add Document	12445444	7 days 57 mins ago	0x595f4030575accd032...	OUT	0x9c8c95dda33b52c252...	0 Ether	0.000000012141
0x1d828c71e0bb9cc4e2...	Add Document	12445295	7 days 1 hr ago	0x595f4030575accd032...	OUT	0x9c8c95dda33b52c252...	0 Ether	0.00000001197
0x10da57bbe8cd91ec9f...	Add Document	12433146	8 days 22 hrs ago	0x595f4030575accd032...	OUT	0x9c8c95dda33b52c252...	0 Ether	0.000000011798

Our Journey

Week 1: May 16 - May 22

16 May

Problem statement selected - Blockchain

Created Notion mind maps and brainstormed on various aspects of the project.

Noted down the key deliverables

Thought from the user's, the admin's and the LOB's perspective.

Looked for efficient ways to store data

Researched about efficient and secure ways to store customer's files and crucial information.

High level design

Made the HLD for the project, wrote what goes in and what goes out, then looked for tech stack.

Documentation

BRD, functional requirements and ER model created.

22 May

Week 2: May 23 - May 29

23 May

Work distribution

Came to a common consensus on who is working on which part of the problem, and noted the reviews given by FMs from Demo 1.

Solidity and web-app

Created a basic smart contract based on Solidity and looked for tech for the web-app.

Document Storage Research

Researched about whether to store document on blockchain or its hash. Also then researched about how to generate a document's hash.

Blockchain research

Researched about public, private and consortium based blockchains, and decided to go with Ethereum.

MERN stack

MongoDB Atlas, Express.js, React.js and Node.js used for the web-app purpose

29 May

Week 3: May 30 - June 05

30 May

Some key features

Implemented basic functions like user login, user register, get user info, etc.

User Dashboards and APIs

Created dashboards for user, lob and verifier side, made forms for uploading files, and created basic backend APIs

Database Research

More in-depth research on which database to choose and how exactly the data is stored in the database, making MongoDB the final choice

Resolving ABI errors

Worked on the Application Binary Interface (ABI) and solved some errors it had.

GridFS Connection

Made Gridfs connections in the backend API to store files in the database and retrieve those files with a simple API call

05 June

Week 4: June 06 - June 12

06 June

Basic Integration

Integrating UI with backend: customer and admin dashboards, connected the frontend with IPFS with a successful generation of hash, extracted and dispayed data from database to the UI

Changes suggested by FMs

Hiding DOB and address in admin portal, keeping person name and email in same place, etc, hash gen and accept-view-reject calls.

Making diagrams

Worked on the sequence diagram, system architecture diagram, etc as told by FMs

End to end integration

Making sure the front end and back end functionalities and database insertions and retrievals work properly and also connected the backend with the blockchain

Interface for LOB

As FMs suggested, enhanced LOB dashboards, displayed data according to the documents associated with that LOB Department and modified Mongo Models accordingly

12 June

Week 5: June 13 - June 19

13 June

Enhancing logging and Webapp

Enhanced the logging functionality, i.e. keeping track of insertions, updations, new user requests, etc. Improved webapp features and resolved errors.

Hyperledger technology

Studied about peers and organizations, and about permissioned blockchain.

Hyperledger network

Based on what we learnt, we created one simple Hyperledger Fabric network and implemented some add/delete functionalities.

Completed unit tests

Did the testing of each unit and made sure all units of the project work fine and there are no errors coming up.

Testing Doc

Prepared the spreadsheet for testing of all the units and showed with test case descriptions, and faulty test cases, etc.

19 June

Week 6: June 20 - June 26

20 June

Deployment options

Explored various cloud apps for deployment and read through AWS, Azure, etc. Deployed the blockchain on Roptsten Network and the Webapp on Heroku

Notification Features

Some basic changes in the notification model and API additions for request-permission notifications.

Furthering web-app

Added extra features like sending emails to the customer, created configure options, added spinners, refined the application, and some features in the verifier and lob dashboards

Database Options

Explored some options for relational DB and alternate flow of document storage, through MySQL.

Presentation

Prepared the final presentation for our project and divided our tasks for the same.

26 June

Future Scopes

OTP feature in Login and Registration

Adding this as a module to Wells Fargo app

Expanding this Application over multiple organizations

Adding Video KYC for more trustful verification

Shifting to a Permissioned Blockchain

Verification of Documents done by Multiple Banks

Alternate Flows

A) Exploring different types of Blockchain

B) Storing documents long-term on database while not using IPFS

A) Types of Blockchain Networks

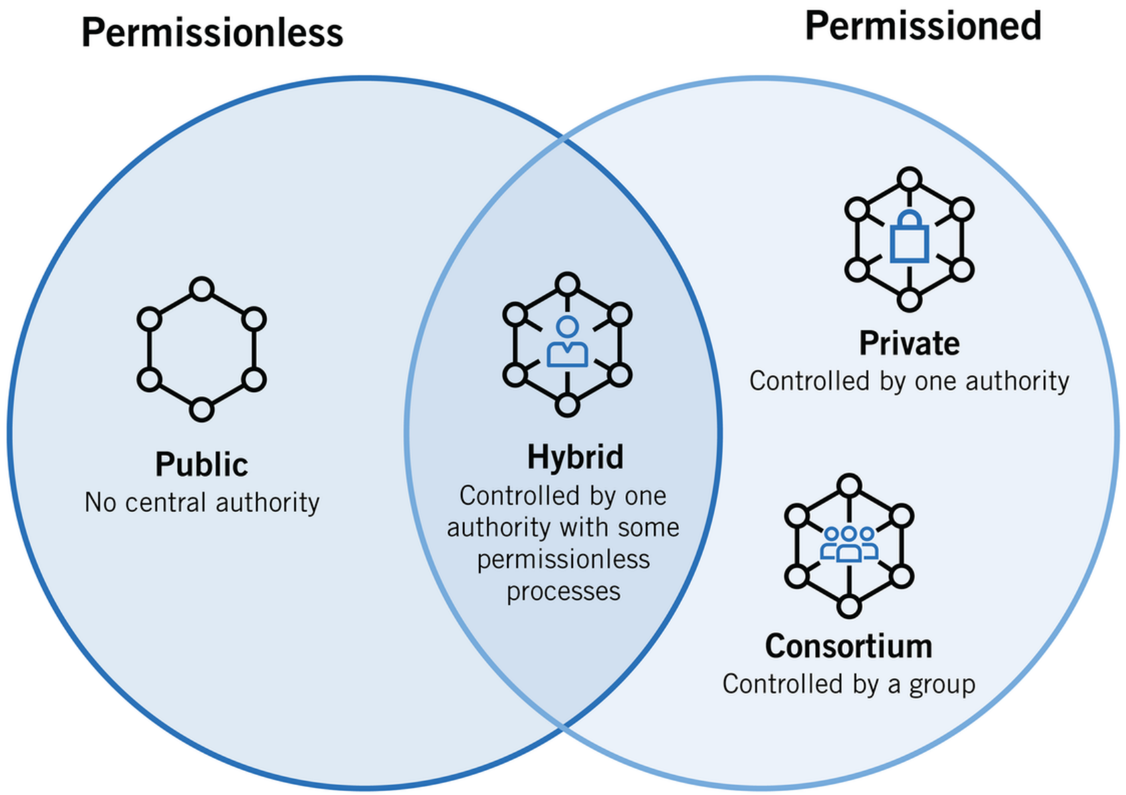
Permissionless blockchains allow any user to become a restriction-less node in the network

---

Permissioned blockchains restrict network access to certain nodes which might still have restricted rights.

Ethereum is a public (permissionless) blockchain that allows anyone to join and is completely decentralized

Hyperledger is a consortium (permissioned) blockchain wherein the level of access and rights of available to every participant node is preestablished.



A) Comparing Technologies

WHY ETHEREUM

Fully decentralized

Increased trust

Easy deployment of dApps

Incentivized consensus mechanism

WHY HYPERLEDGER

Predefined node access

Highly secure for B2B contexts

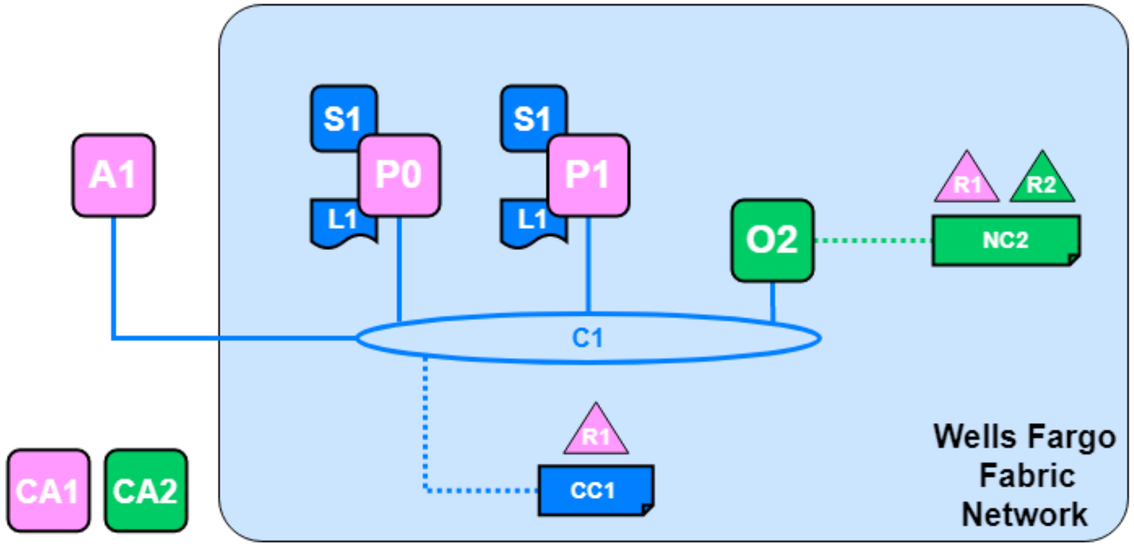
Better scalability

Non-volatile network

A) Custom Hyperledger Fabric Network

We created a custom Hyperledger Fabric network with two organizations, contributing one *Orderer* node and two *Peer* nodes to it.

```
CONTAINER ID   COMMAND                  STATUS          NAMES
725f79e0f6a8   "chaincode -peer.add…"   Up 23 minutes   dev-peer1.org1.example.com-kyc_1-1713584583ff99a2facae06ea99c57aa84e1d68bb6bd6e02090826837b903812
58902d9676d1   "chaincode -peer.add…"   Up 23 minutes   dev-peer0.org1.example.com-kyc_1-1713584583ff99a2facae06ea99c57aa84e1d68bb6bd6e02090826837b903812
5586d4687687   "peer node start"        Up 24 minutes   peer0.org1.example.com
0269aea10d20   "orderer"                Up 24 minutes   orderer.example.com
80eb5c56b566   "tini -- /docker-ent…"   Up 24 minutes   couchdb0
0aa4994b87b5   "sh -c 'fabric-ca-se…"   Up 24 minutes   ca.org1.example.com
58130b6b3e43   "peer node start"        Up 24 minutes   peer1.org1.example.com
20c5296b1c7d   "tini -- /docker-ent…"   Up 24 minutes   couchdb1
```



R1 : Peer Organization

R2 : Orderer Organization

A1 : e-KYC Application

P0, P1 : R1 Peer Nodes

CA : Certification Authority

O2 : R2 Orderer Node

C1 : Channel 1

S1 : Channel 1 Chaincode

L1 : Channel 1 Ledger and State

CC1 : Channel 1 Configuration

NC2 : Network Configuration

A) Custom Hyperledger Fabric Network

- We installed the requisite chaincode (written in Golang) on all peer nodes for implementing the KYC functionality.
- For updating the ledger, the chaincode is invoked and a three-step consensus mechanism is used.
- Querying the ledger doesn't require a consensus mechanism and no block creation is involved.

```
type Document struct {
    Email   string `json:"email"`
    Doctype string `json:"doctype"`
    Dochash string `json:"dochash"`
}
```

```
func (s *SmartContract) add_document
```

arguments (Email, Doctype, Dochash)

```
func (s *SmartContract) query_hash
```

arguments (Email, Doctype)

Successful invoke

```
2022-07-04 07:00:48.158 IST 0001 INFO [chaincodeCmd] chaincodeInvokeOrQuery -> Chaincode invoke successful. result: status:200 payload
:"{\"email\":\"nikhil.e@gmail.com\",\"doctype\":\"DL\",\"dochash\":\"EABF7028559CF2F44CF9B6CDFBA4615842E01AB403610FE8\"}"
```

Failed invoke

```
Error: endorsement failure during invoke. response: status:500 message:"EABF7028559CF2F44CF9B6CDFBA4615842E01AB403610FE8"
```

Successful query

```
{"dochash":"EABF7028559CF2F44CF9B6CDFBA4615842E01AB403610FE8","doctype":"DL","email":"nikhil.e@gmail.com"}
```

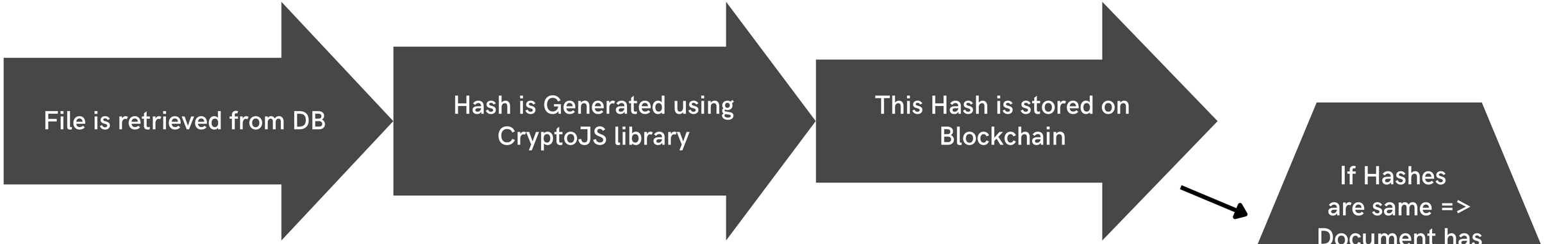
Failed query

```
Error: endorsement failure during query. response: status:500 message:"Document hash not found!"
```

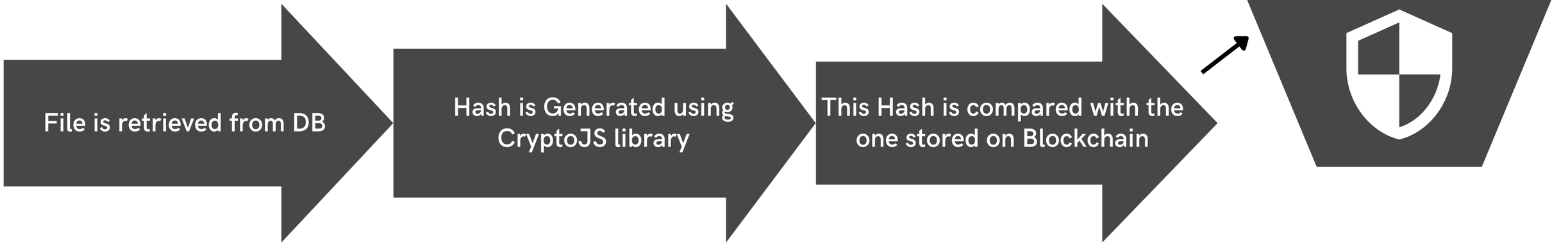
B) Hash Comparison

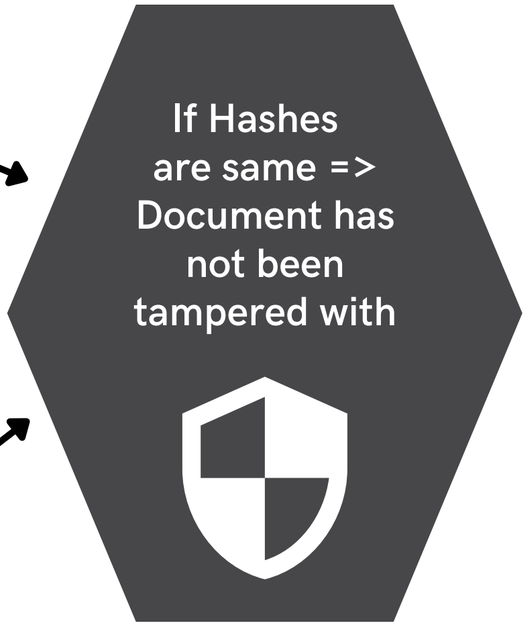
Instead of storing the documents on IPFS, we store in on a database (with **ACID compliance**)

When Admin verifies docs



When LOB views docs





Individual Contributions

Aryan Kharbanda

- Team Leader
  - Held daily meets
  - Managed work-distribution
  - Overlooked every aspect of project
  - Took critical decisions at every step
- Full stack webapp (database, backend, frontend)
- IPFS and Blockchain integration
- Presentation flow

Hariom Vyas

- BRD Document
- System Architecture Diagram
- Presentation
- Truffle + IPFS Hash Generation and Storage Prototype
- Connection scripts of Blockchain on webapp
- Web3 connections
- IPFS connections
- Webapp Frontend
- Deployment on Blockchain on test network
- Deployment of Web-App

Sakshi Jain

- BRD Document
- Database Models
- GridFS Connection
- Frontend Application
- Backend APIs
- Frontend-Backend Integration
- API Testing
- Game Plan Document

Nikhil E

- BRD Document
- Solidity smart contract
- Smart contract APIs
- Use-case Diagrams
- Hyperledger Fabric Alternate Flow - Research and Implementation
- Presentation

Individual Contributions

Rahul Singhadia

- BRD document and flow diagrams
- Database Models
- Few Backend APIs
- Logging
- Testing and test case documents
- Presentation

Prathamesh Bonde

- BRD Document
- Designing the product flow
- Solidity smart contract
- Presentation
- Testing and Test Case Documentation

Chinmay Joshi

- Solidity smart contract
- Hyperledger Fabric research
- Webapp CSS designing
- Presentation

Thank you!

Feel free to reach out to us if you have any questions.