Decentralized Algorithm Trading Websites

Yunqiu Xu (z5096489), Xingshi Zhang (z5101900),
Yichen Zhu (z5098663), Qihai Shuai (z5119437)

COMP9900 Playing with Mud

Content

- Introduction
- Project overview
- Details and demos
- Implementation
- Summary

Introduction

Background

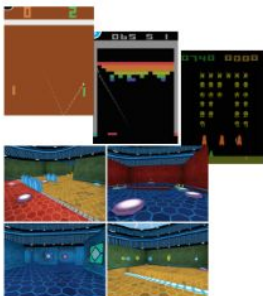- Machine learning and deep learning

Kohl and Stone, 2004

Ng et al, 2004

Tedrake et al, 2005

Kober and Peters, 2009

Mnih et al 2013 (DQN)
Mnih et al, 2015 (A3C)

Silver et al, 2014 (DPG)
Lillicrap et al, 2015 (DDPG)
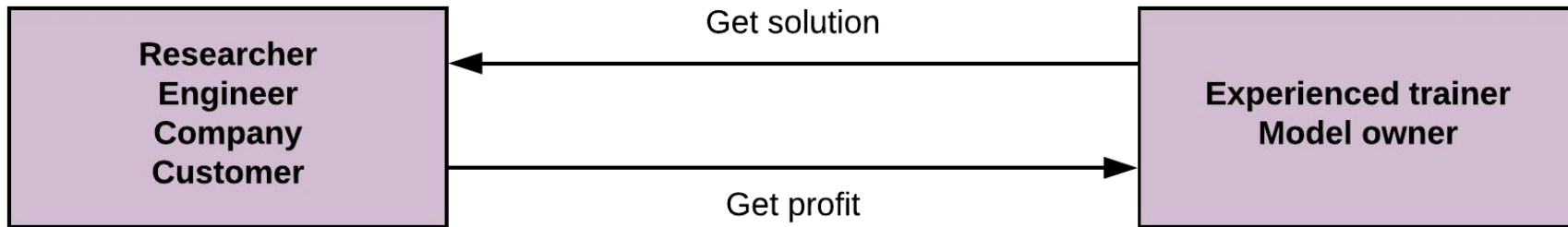
Schulman et al,
2016 (TRPO + GAE)

Levine*, Finn*, et
al, 2016
(GPS)

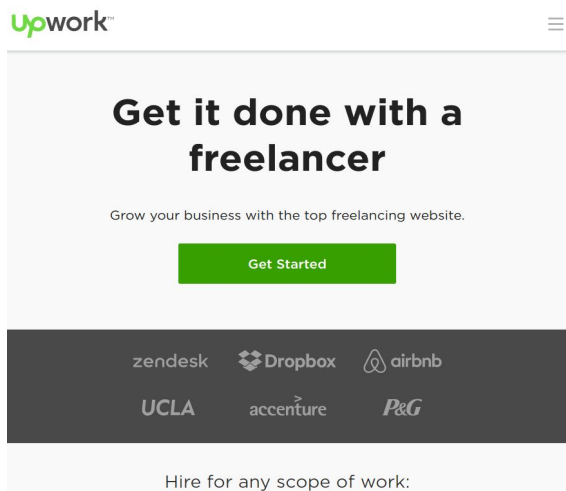Silver*, Huang*, et
al, 2016
(AlphaGo)

Background

- Machine learning and deep learning: hard to train
    - Requires long time and large memory
    - Hard to find suitable hyper-parameter set
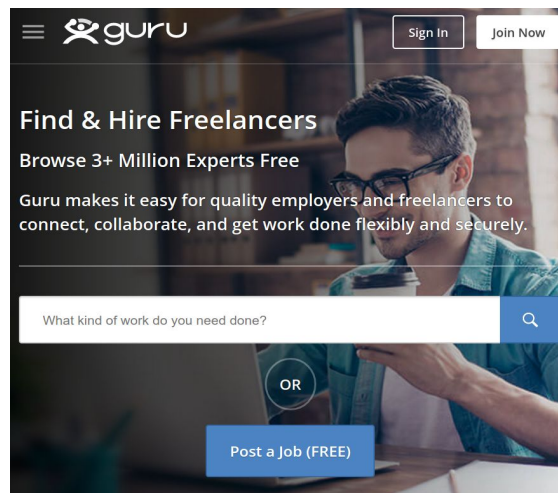   - Expensive to build GPU cluster

Existing Work

- Outsourcing websites
  - Complex → not machine learning specific
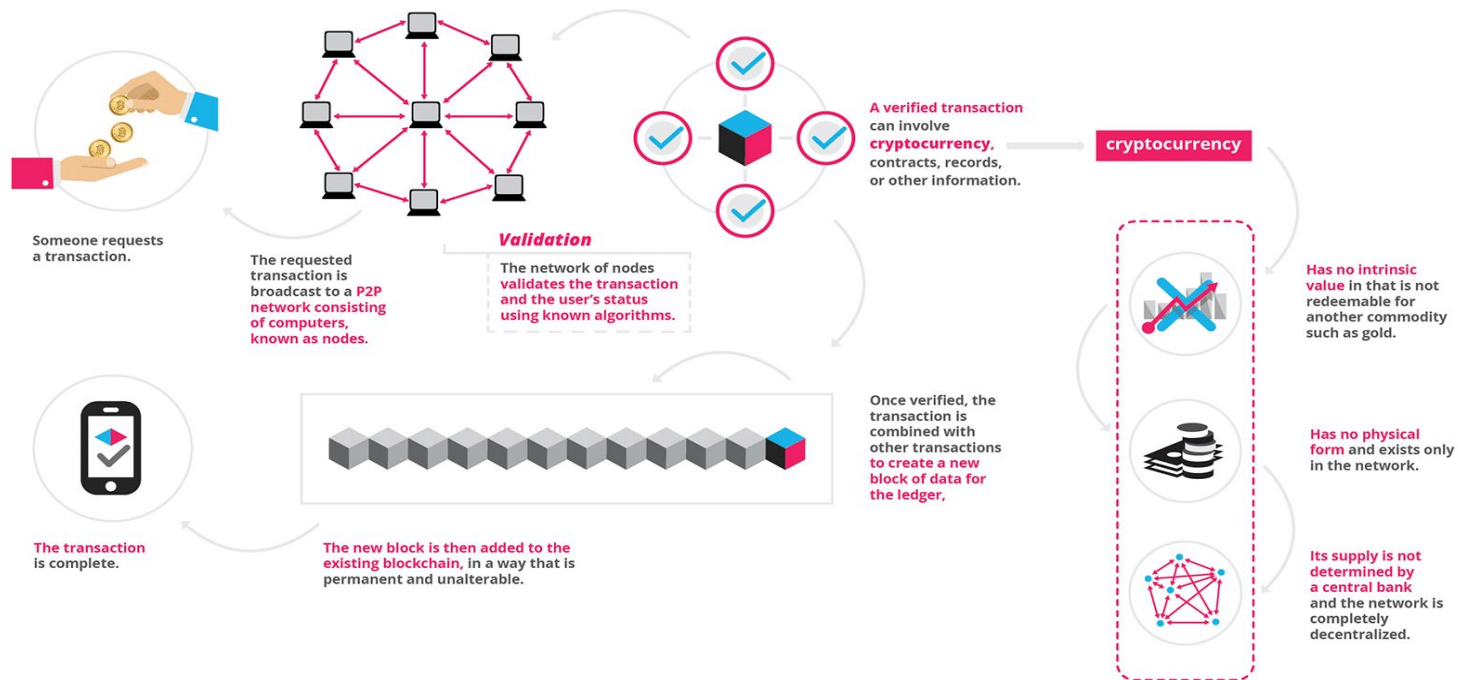 - One-to-one
  - Can not exchange goods directly

Upwork

Guru

Existing Work

- Blockchain and DAPPs

Someone requests a transaction.

The requested transaction is broadcast to a **P2P network consisting of computers, known as nodes.**

Validation
The network of nodes **validates the transaction and the user's status using known algorithms.**

A verified transaction can involve cryptocurrency, contracts, records, or other information.

cryptocurrency

Once verified, the transaction is combined with other transactions **to create a new block of data for the ledger,**

The new block is then added to the existing blockchain, in a way that is permanent and unalterable.

The transaction is complete.

Has no intrinsic value in that is not redeemable for another commodity such as gold.

Has no physical form and exists only in the network.

Its supply is not determined by a central bank and the network is completely decentralized.

Existing Work

- ## Blockchain and DAPPs
  - Decentralized → no third party intervention
  - Anonymous → no need of identities

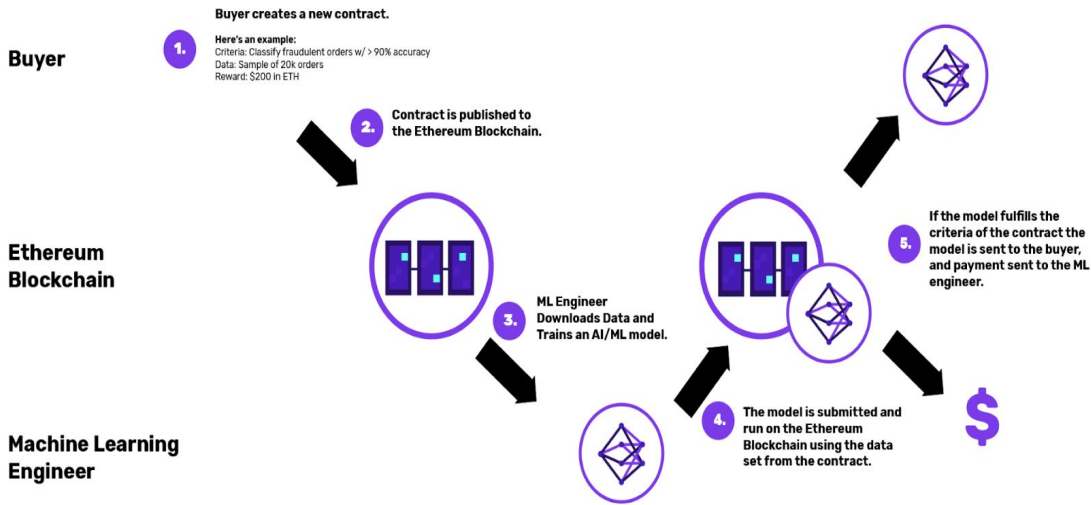Ethergarden

DANKU

Existing Work

- Blockchain and DAPPs
  - Decentralized → no third party intervention
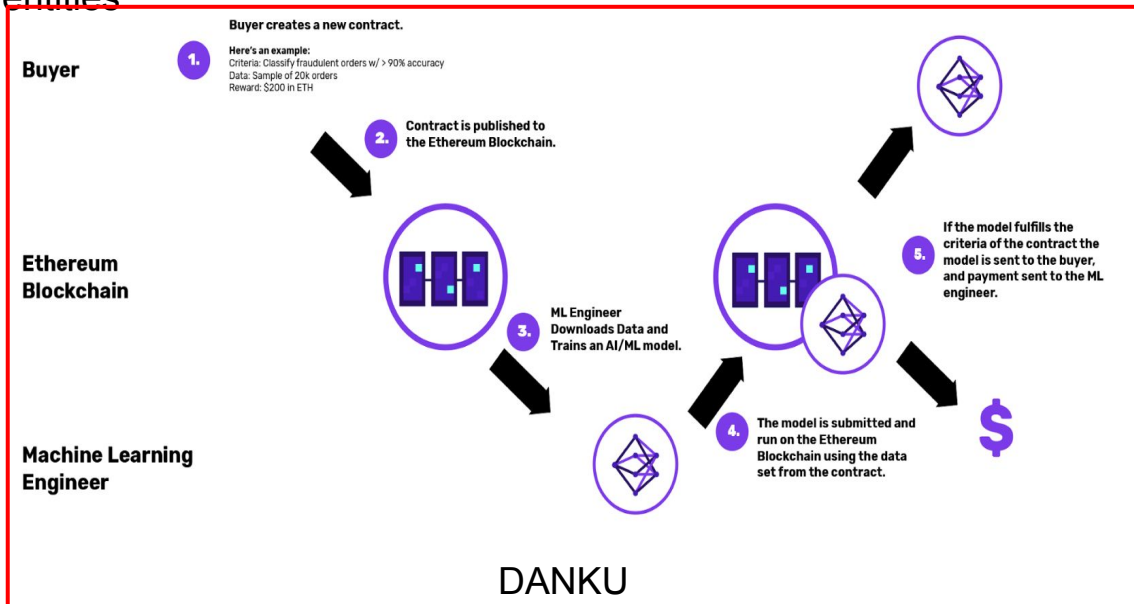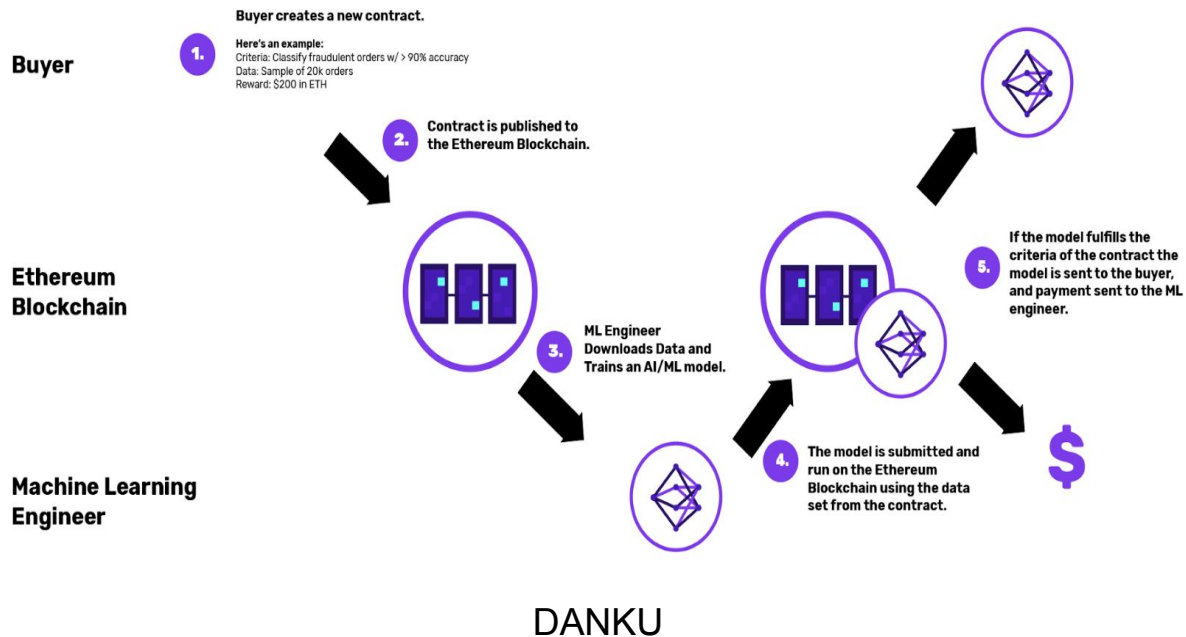 - Anonymous → no need of identities

Ethergarden

DANKU

Existing Work

- DANKU
  - Evaluating on blockchain
  - Outsourcing only
- Our work
  - Independent OJ module
  - Trading + outsourcing

Buyer creates a new contract.

Buyer
1. Here's an example:
   Criteria: Classify fraudulent orders w/ > 90% accuracy
   Data: Sample of 20k orders
   Reward: $200 in ETH

2. Contract is published to the Ethereum Blockchain.

Ethereum Blockchain

3. ML Engineer Downloads Data and Trains an AI/ML model.

4. The model is submitted and run on the Ethereum Blockchain using the data set from the contract.

5. If the model fulfills the criteria of the contract the model is sent to the buyer, and payment sent to the ML engineer.

Machine Learning Engineer

DANKU

Project Overview

Decentralized Algorithm Trading Websites

- Online shop for exchanging algorithm models
- Outsourcing platform for posting and accepting requests
- Decentralized and anonymous
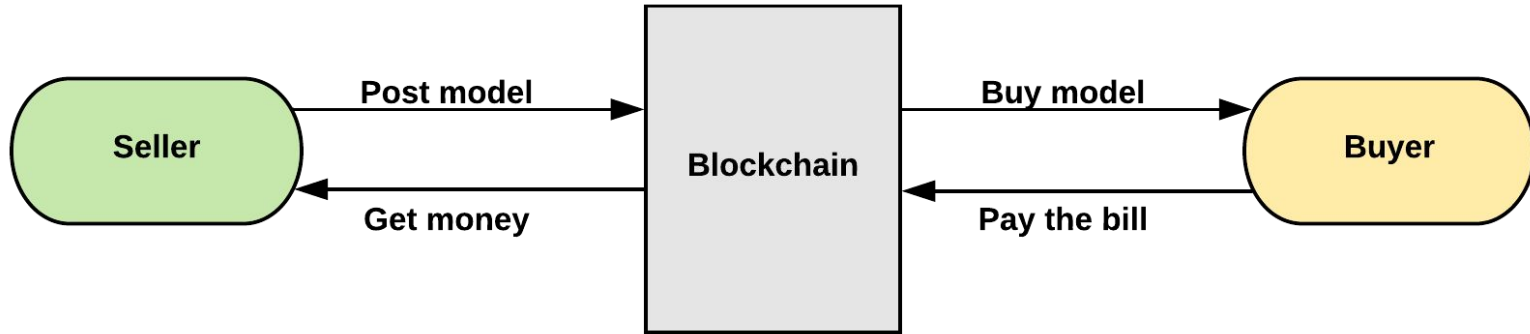
Online shop : short term transacton
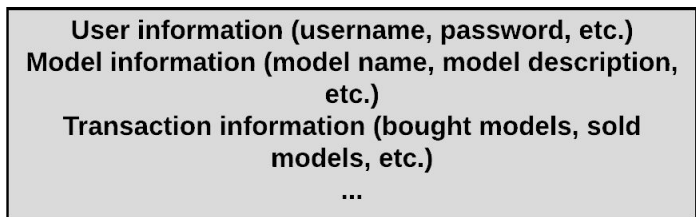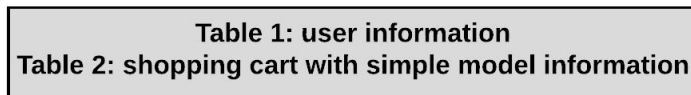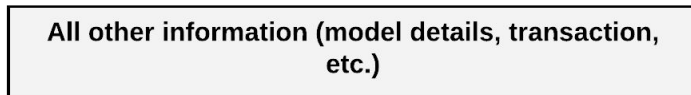
Decentralized Algorithm Trading Websites

- Online shop for exchanging algorithm models
- Outsourcing platform for posting and accepting requests
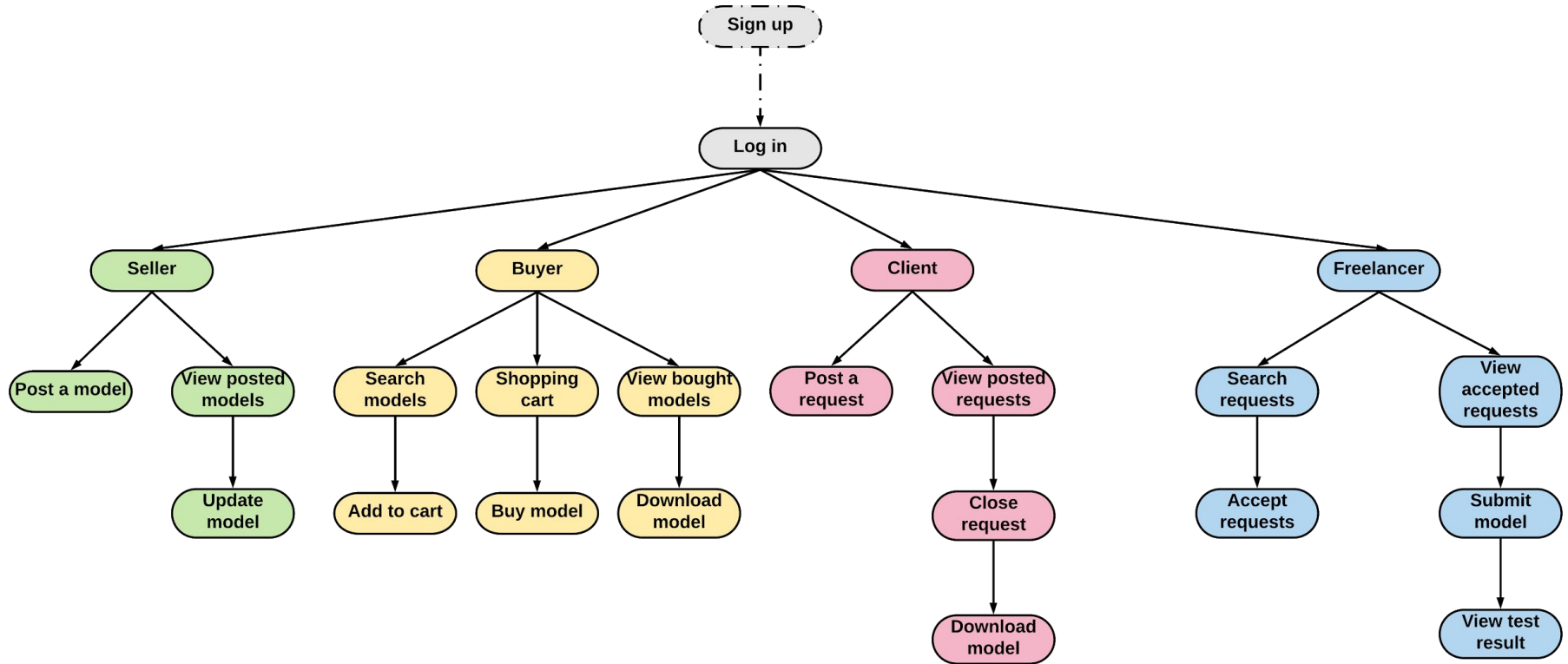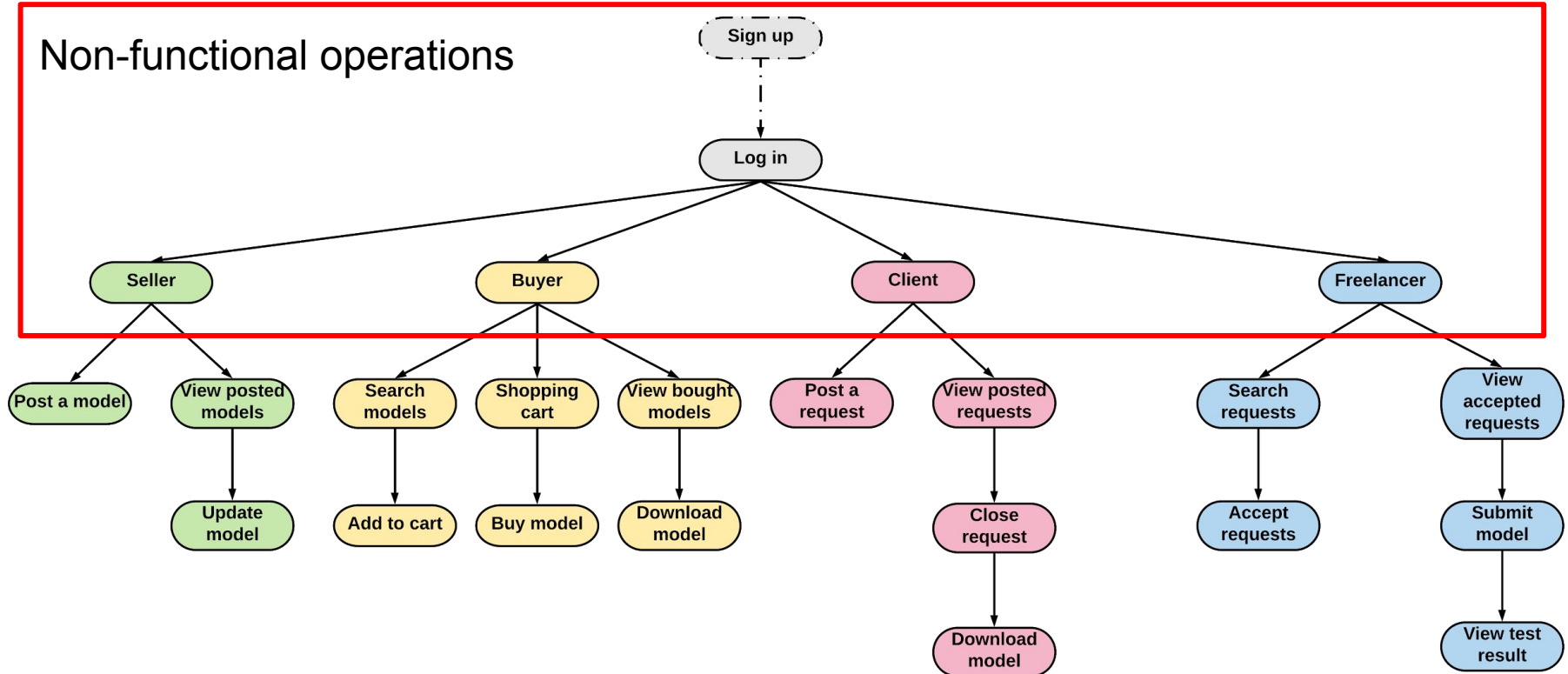- Decentralized and anonymous

Outsourcing platform : long term transacton

Decentralized Algorithm Trading Websites

- Online shop for exchanging algorithm models
- Outsourcing platform for posting and accepting requests
- **Decentralized and anonymous**

User information (username, password, etc.)
Model information (model name, model description, etc.)
Transaction information (bought models, sold models, etc.)
...

Database

VS

Table 1: user information
Table 2: shopping cart with simple model information

Database

All other information (model details, transaction, etc.)

Blockchain

Traditional platform

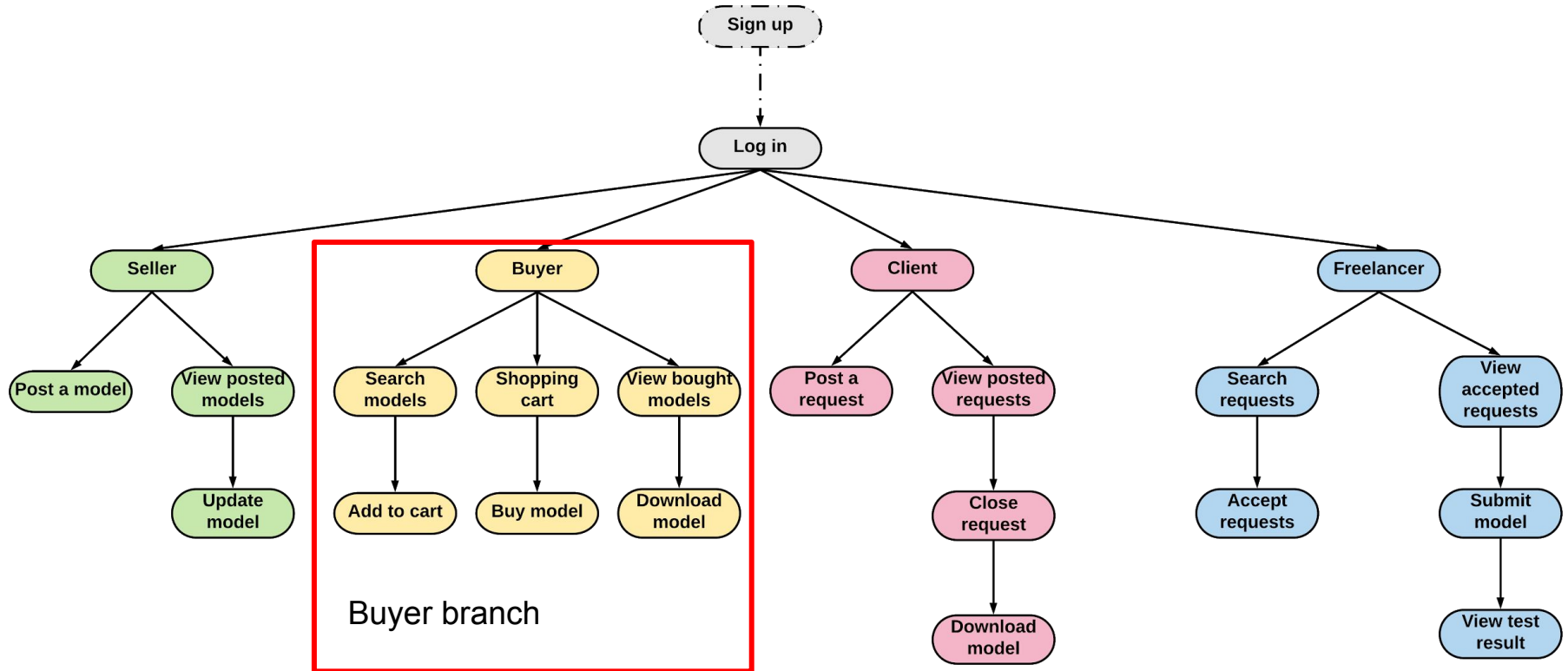Our DAPP

Architecture

Architecture
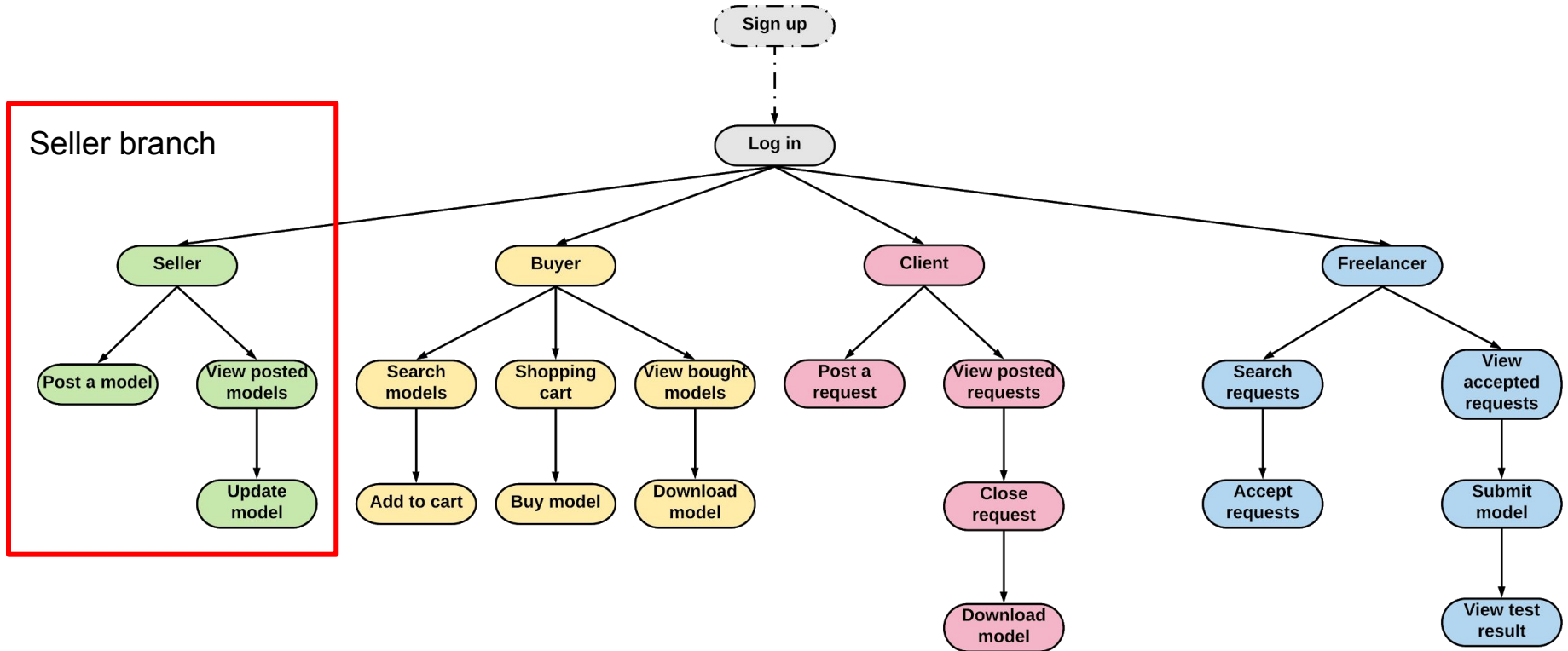
Architecture

Architecture
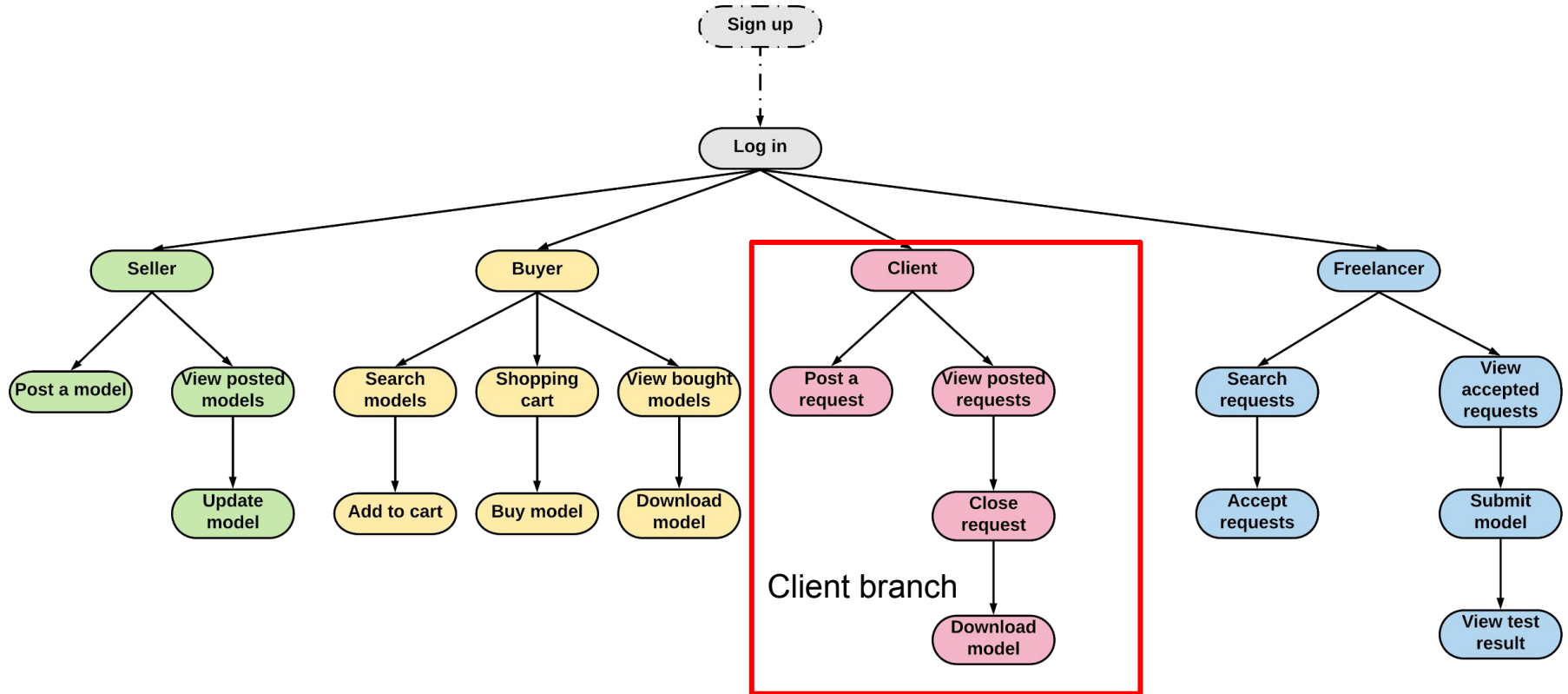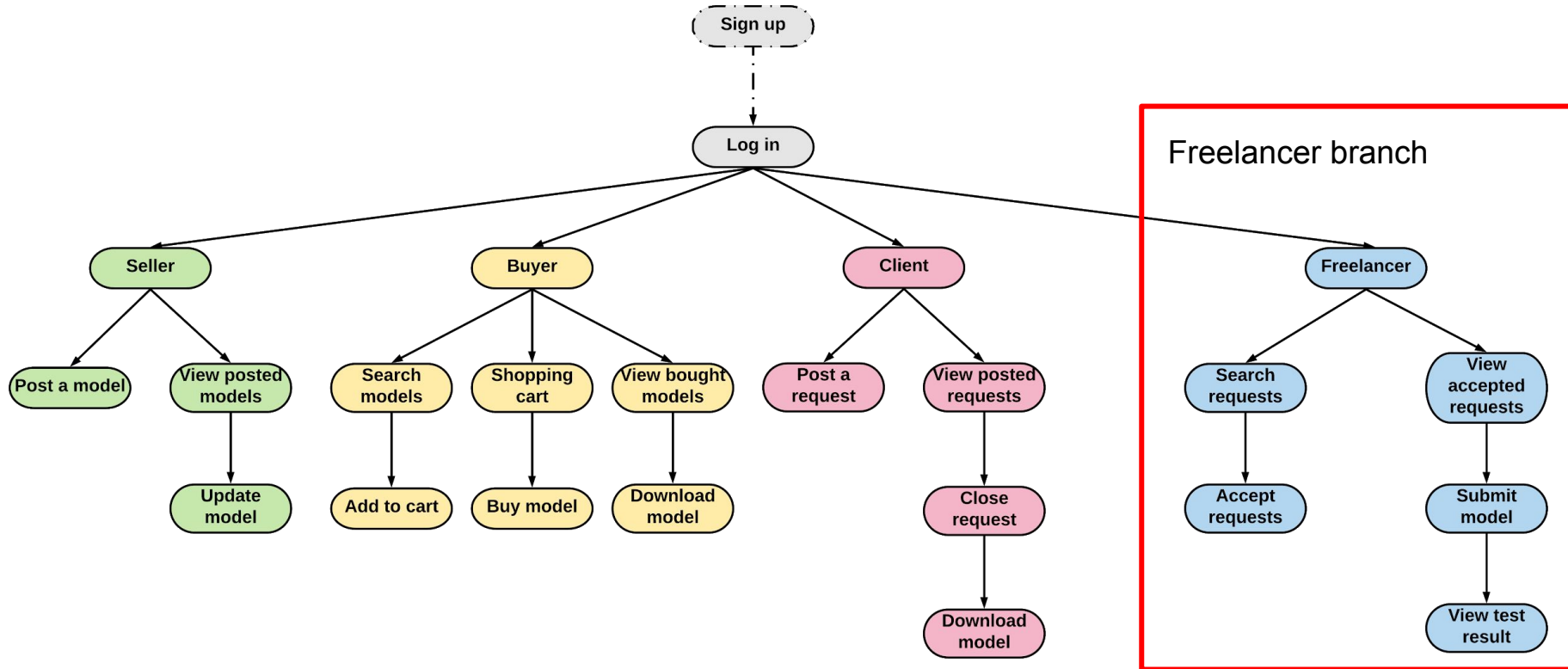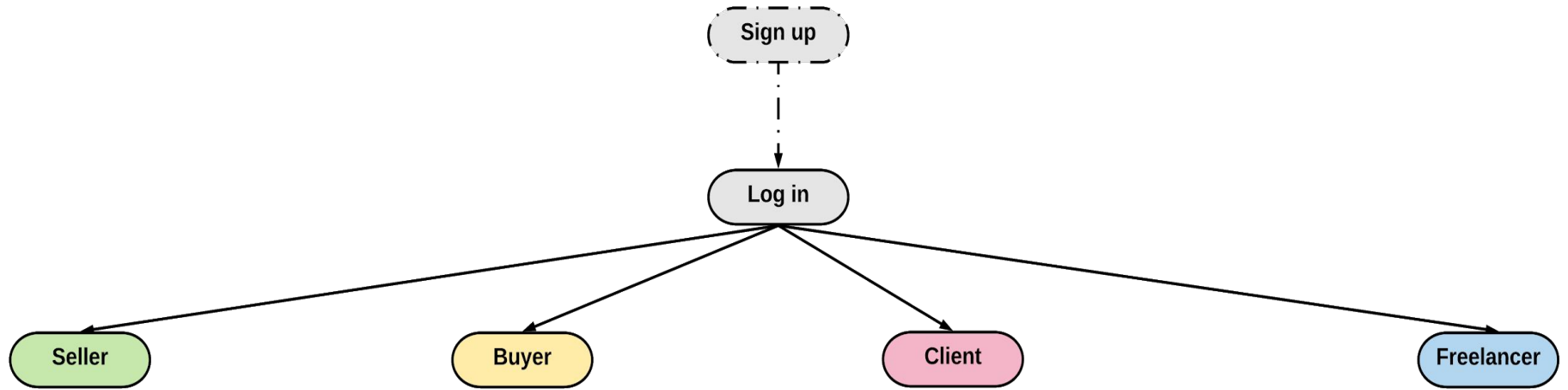
Architecture

Architecture

Details and Demos

Project website: https://sites.google.com/view/playingwithmud

Non-functional Operations
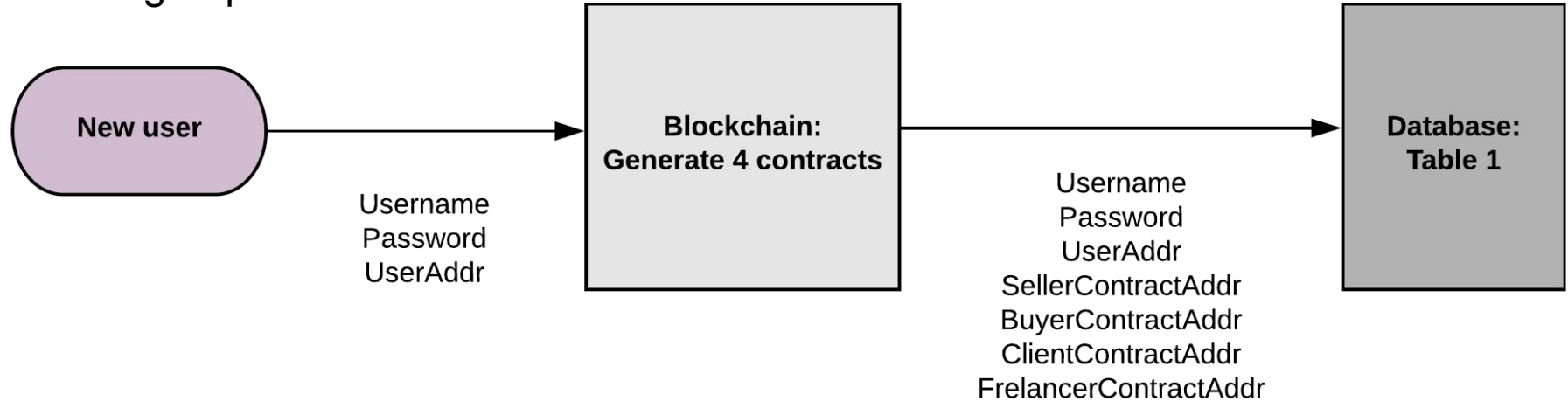
Non-functional Operations
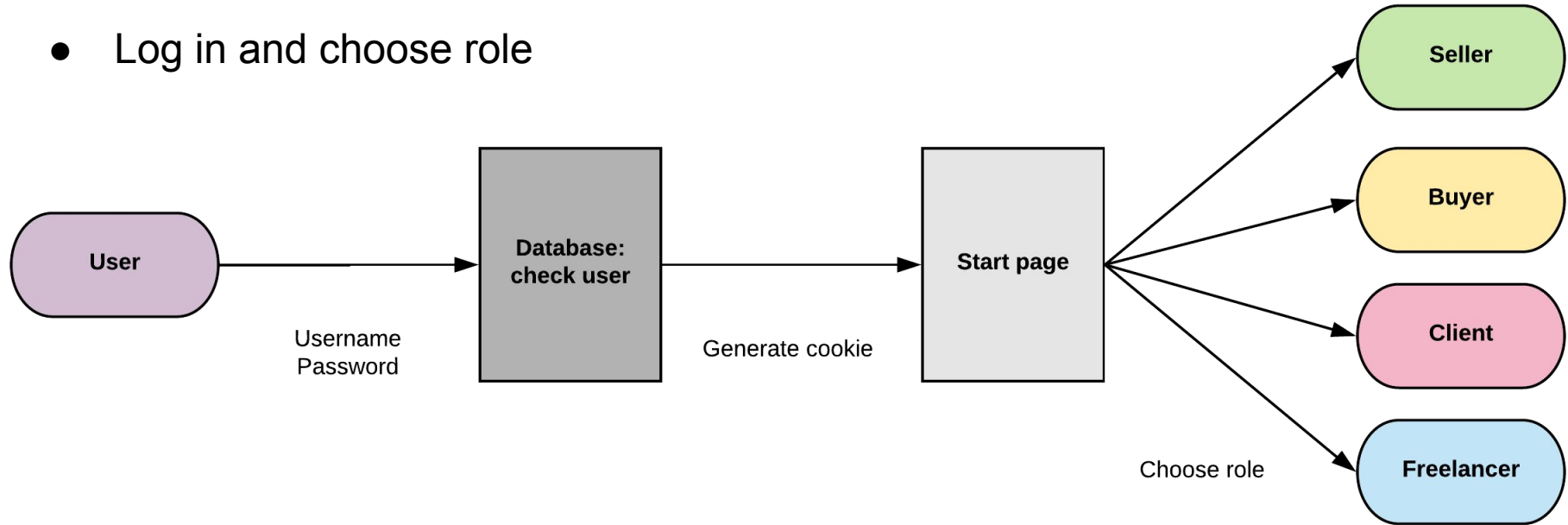
- Sign up

Table 1

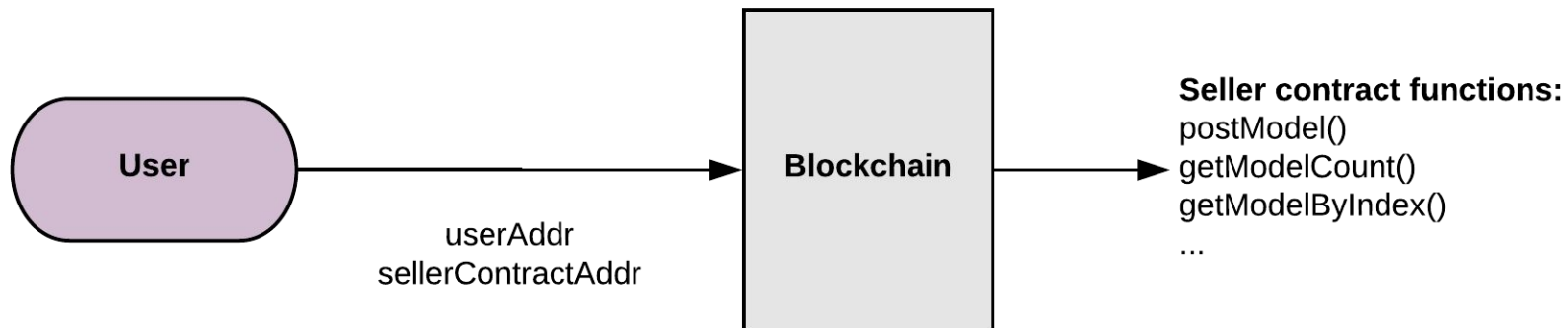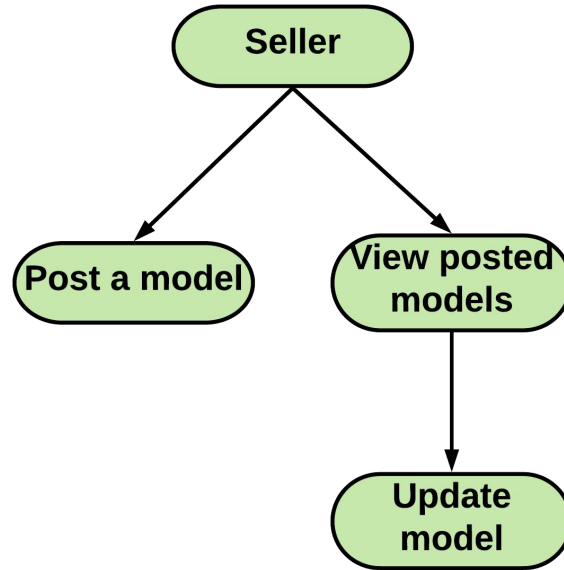| UserID	UserName	Password	UserAddr	SellerCont	BuyerCont	ClientCont	Freelancer Cont

Non-functional Operations

- Log in and choose role

Non-functional Operations

- How to get access to a contract

User → Blockchain → Seller contract functions:

User

Blockchain

userAddr
sellerContractAddr

Seller contract functions:
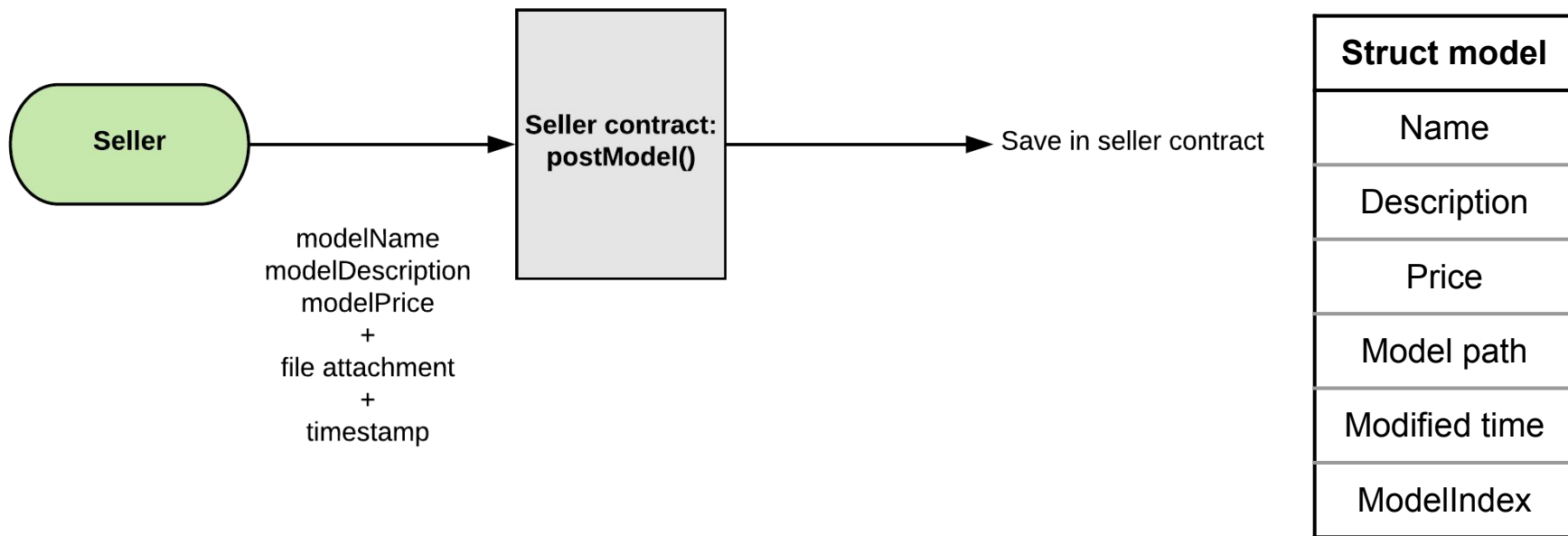postModel()
getModelCount()
getModelByIndex()
...

- Similar to buyer, client and freelancer

Seller Branch

Seller Branch

- Post a model


```
Seller  →  modelName
           modelDescription
           modelPrice
           +
           file attachment
           +
           timestamp
```

Seller contract: postModel() → Save in seller contract

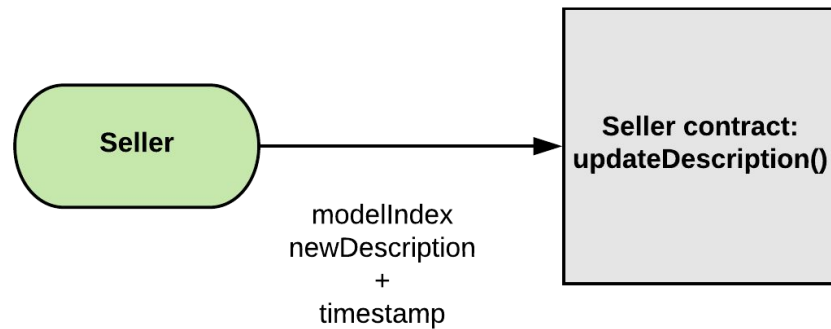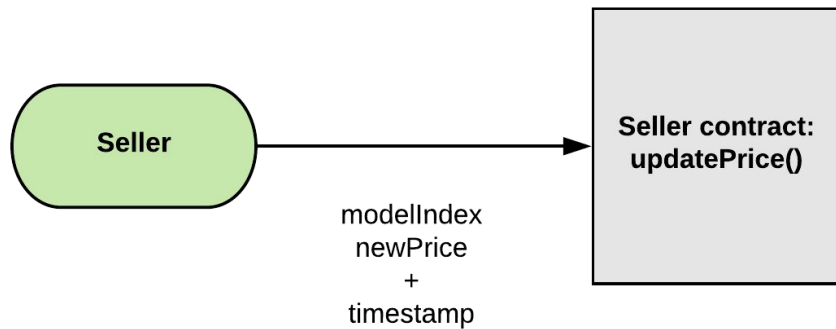| Struct model |
|---|
| Name |
| Description |
| Price |
| Model path |
| Modified time |
| ModelIndex |

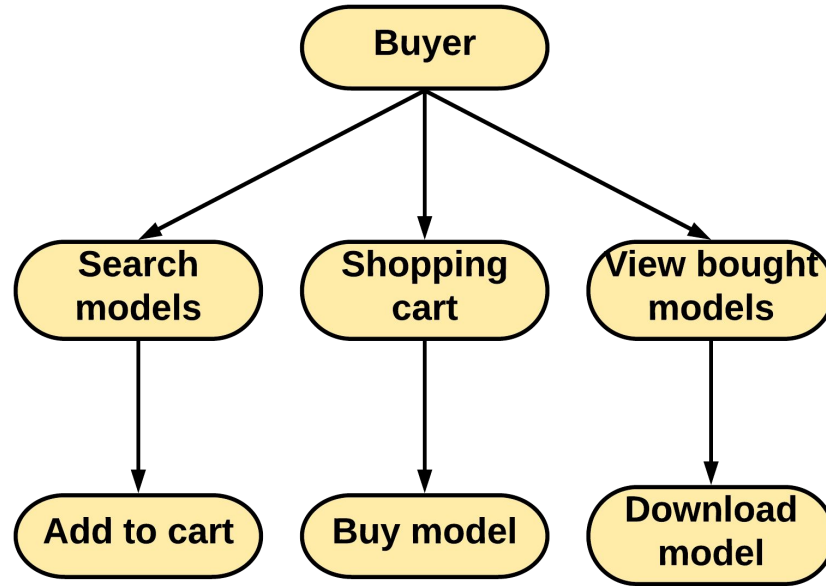# Seller Branch

- View posted models
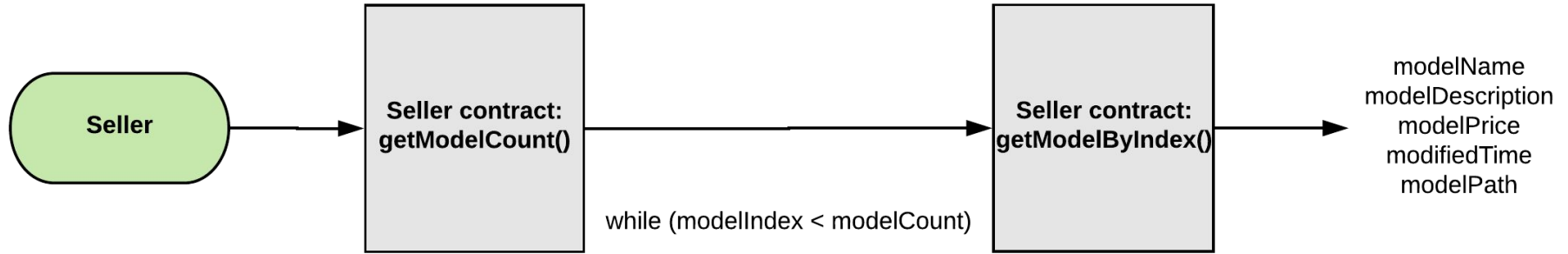
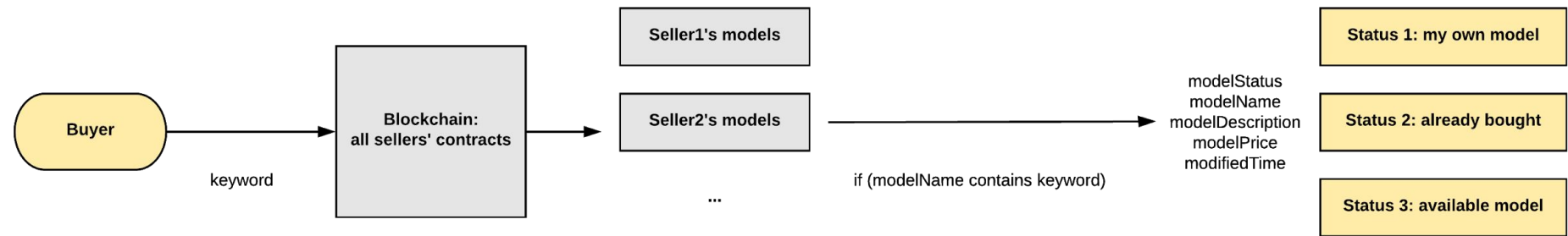# Seller Branch

- Update model

Demo 1

- User acts as a seller
- Post a model
- View posted models
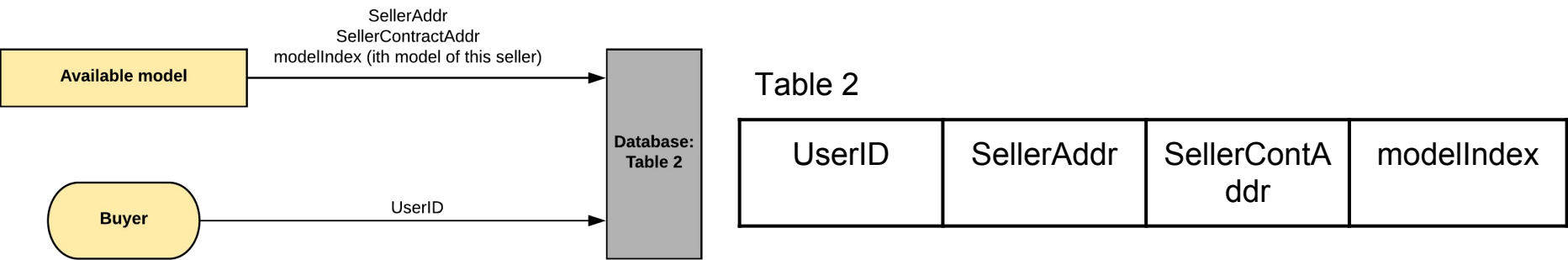- Update model

Buyer Branch

Buyer Branch

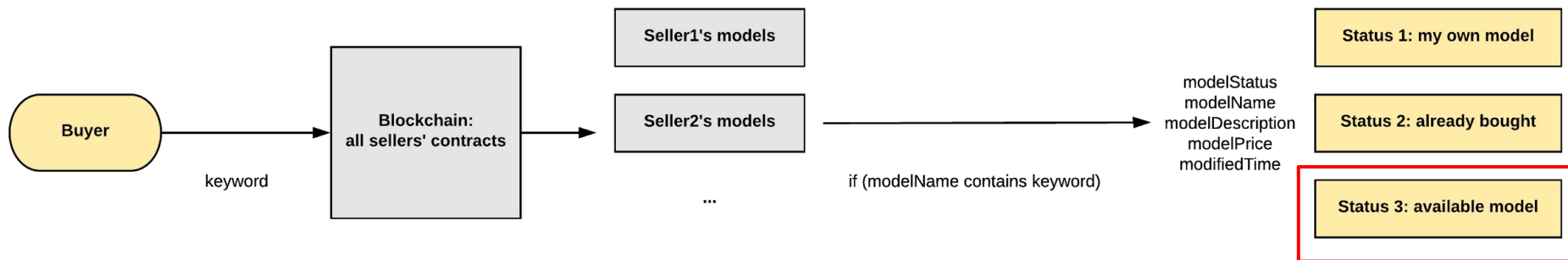- Search models: if no keyword, show all models

- Add to shopping cart

Table 2

UserID	SellerAddr	SellerContAddr	modelIndex

Buyer Branch

- Search models: if no keyword, show all models

| Buyer | → keyword → | Blockchain: all sellers' contracts | → | Seller1's models |
| Seller2's models |
| ... |

if (modelName contains keyword)

modelStatus
modelName
modelDescription
modelPrice
modifiedTime

Status 1: my own model

Status 2: already bought

Status 3: available model

- Add to shopping cart

| Available model | → SellerAddr / SellerContractAddr / modelIndex (ith model of this seller) → | Database: Table 2 |
| Buyer | → UserID → |

Table 2

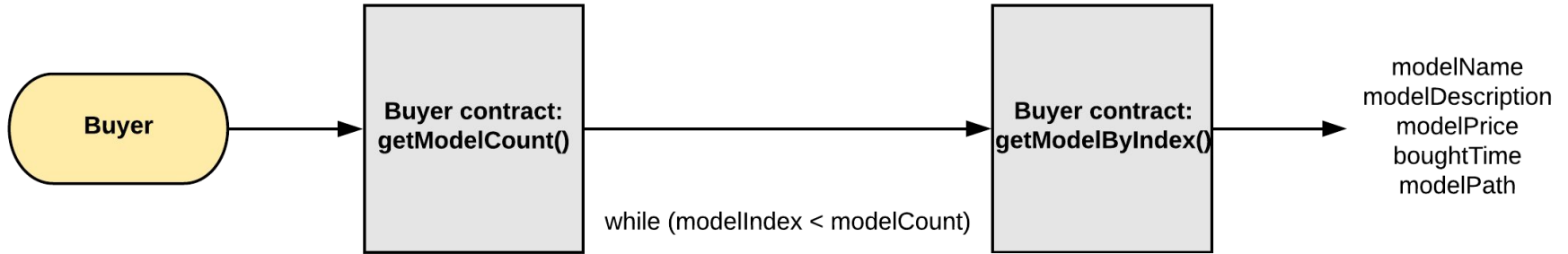| UserID	SellerAddr	SellerContAddr	modelIndex

Buyer Branch

- ## View shopping cart
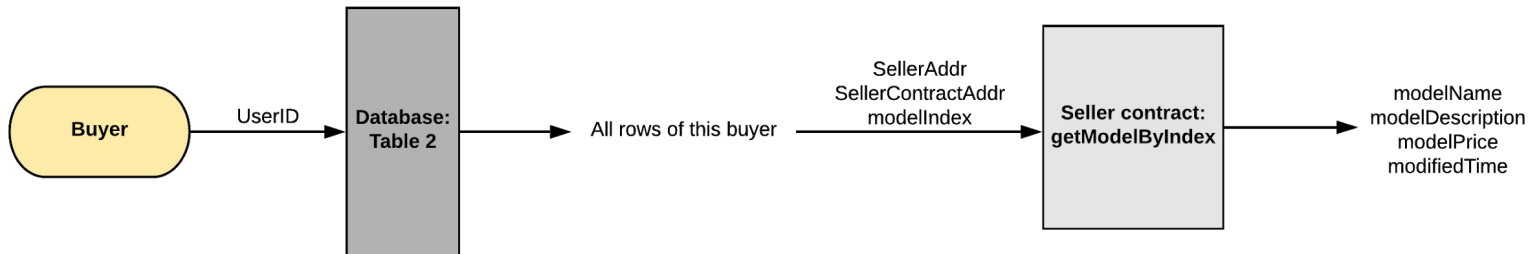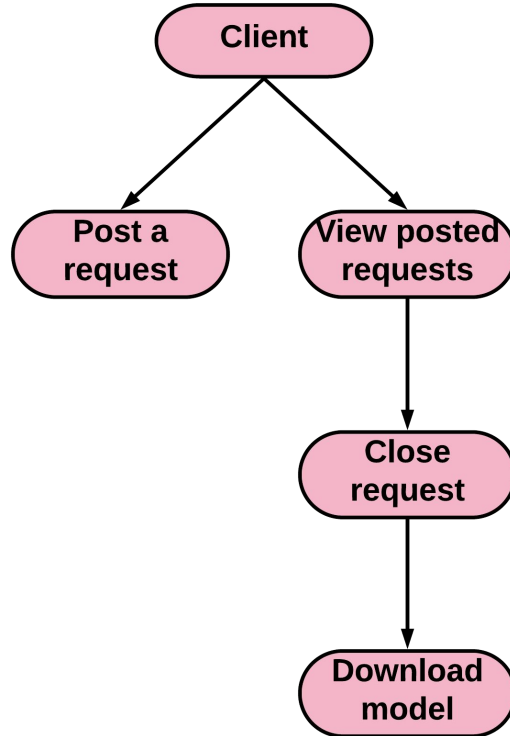

- ## Buy model
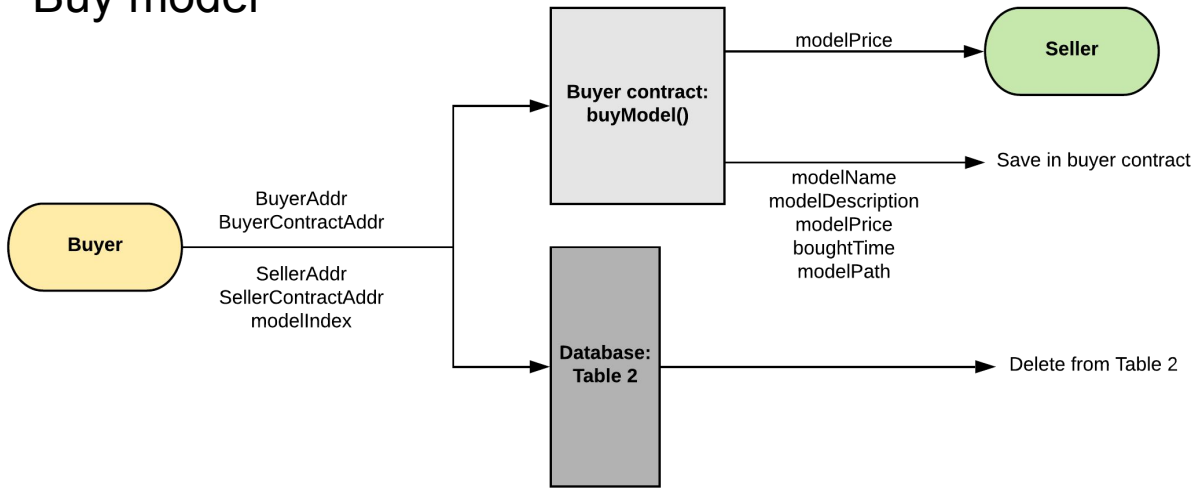
Buyer Branch
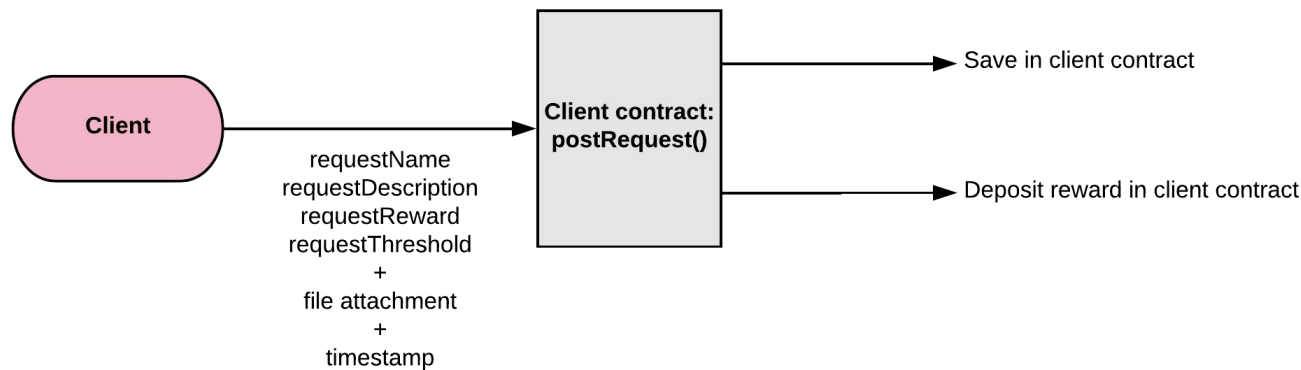
- View bought models

Demo 2

- Act as a buyer
- Search models by name
- Add model to shopping cart
- Buy model
- View bought models

Client Branch

Client Branch

- Post a request

Client → requestName
requestDescription
requestReward
requestThreshold
+
file attachment
+
timestamp

Client contract:
postRequest()

→ Save in client contract

→ Deposit reward in client contract

Struct request
Name
Description
Reward
Threshold
Testcode path
Posted time
Request index
Whether closed (init : false)
Final model (init : null)
Test result (init : 0)
Model author (init : this client)

Client Branch
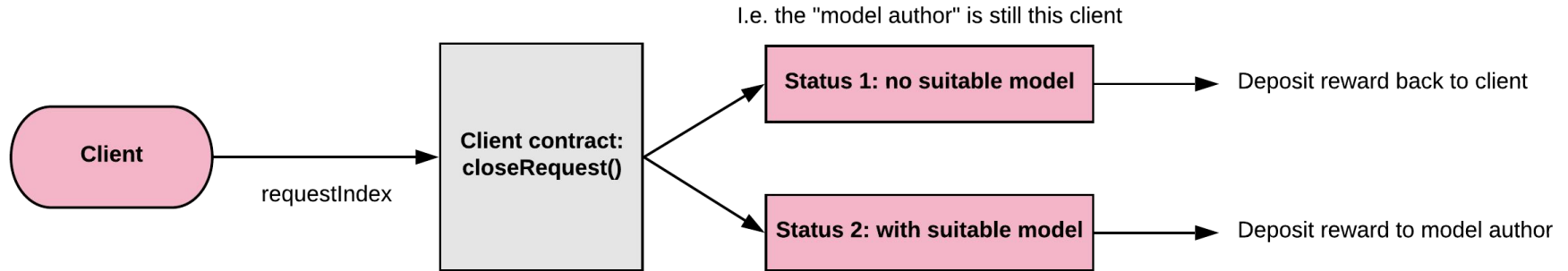
- View posted requests
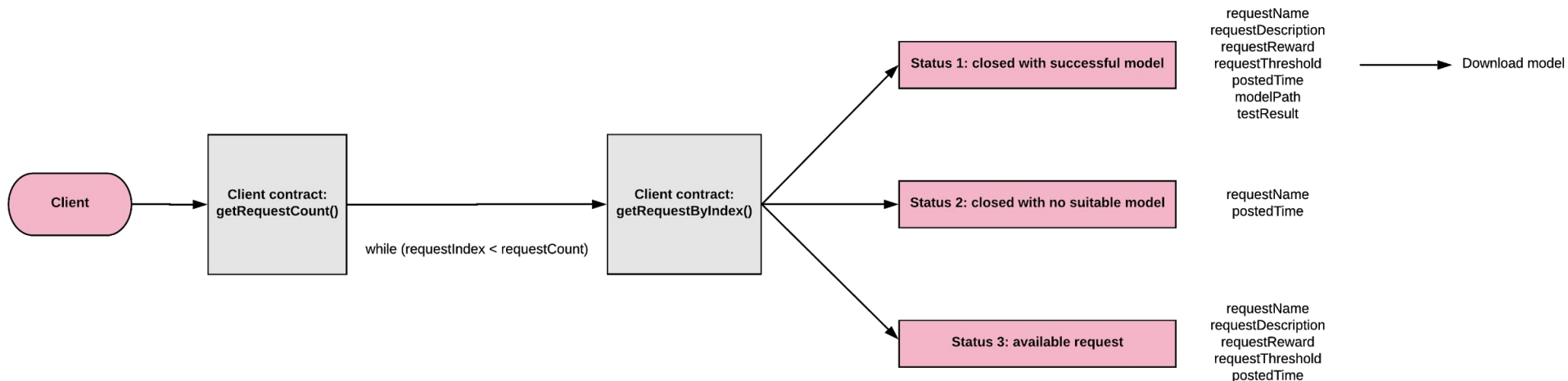
- Before closing, the client will not know whether there is suitable model
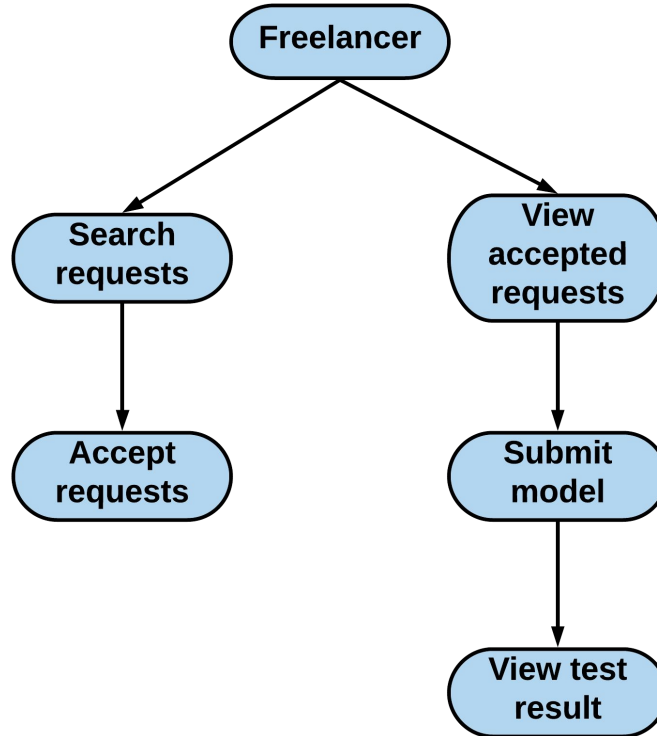
Client Branch

- Close request

I.e. the "model author" is still this client

```
Client  --requestIndex-->  Client contract:
                           closeRequest()
```

Status 1: no suitable model ----> Deposit reward back to client

Status 2: with suitable model ----> Deposit reward to model author

- **Note that the client is not allowed to update description, reward or threshold**
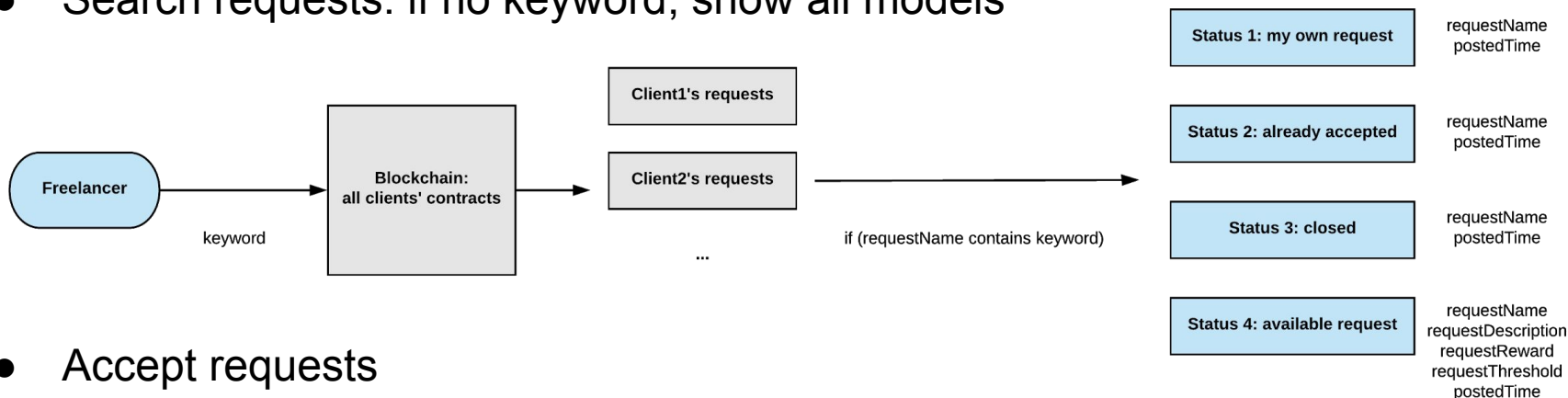
Demo 3

- Act as a client
- Post a request
- View posted requests

Freelancer Branch

Freelancer Branch

- Search requests: if no keyword, show all models

Status 1: my own request	requestName postedTime
Status 2: already accepted	requestName postedTime
Status 3: closed	requestName postedTime
Status 4: available request	requestName requestDescription requestReward requestThreshold postedTime

Freelancer → **Blockchain: all clients' contracts** → **Client1's requests** / **Client2's requests** / ...

keyword

if (requestName contains keyword)

- Accept requests

Available request →

ClientAddr
ClientContractAddr
requestIndex (ith request of this client)

→ **Freelancer contract: acceptRequest()** → Save in freelancer contract

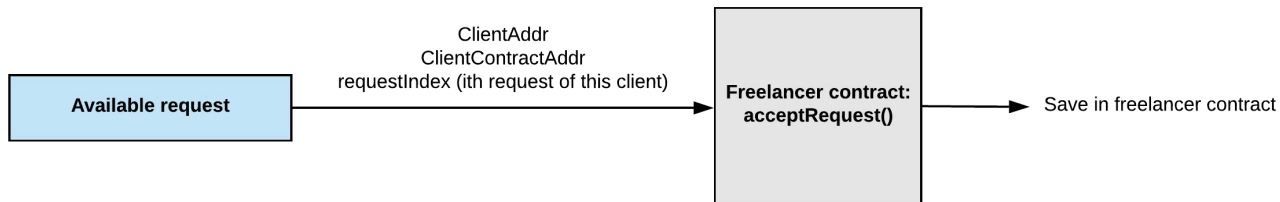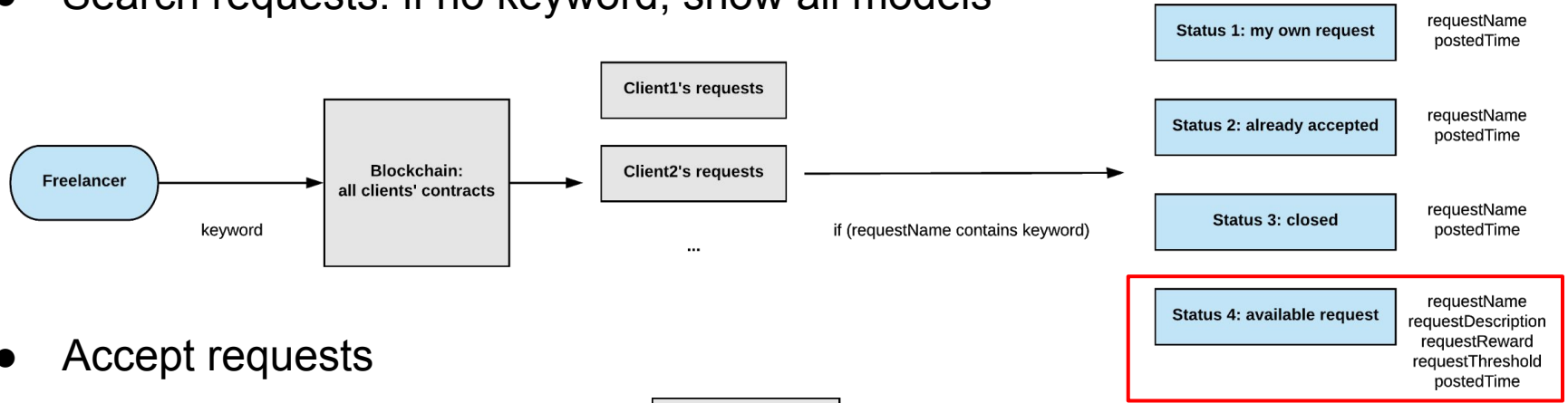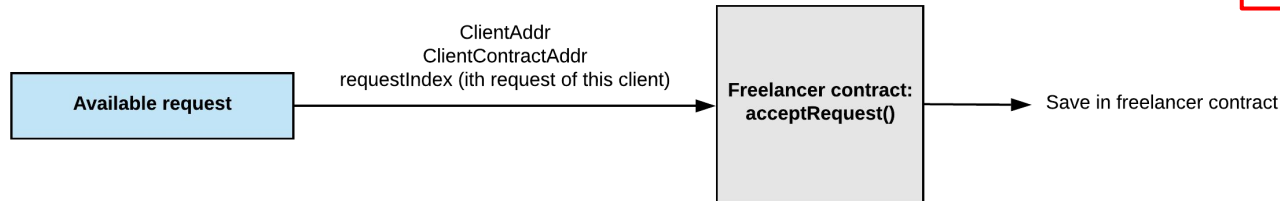- For long term transaction, it's not necessary to use shopping cart

Freelancer Branch

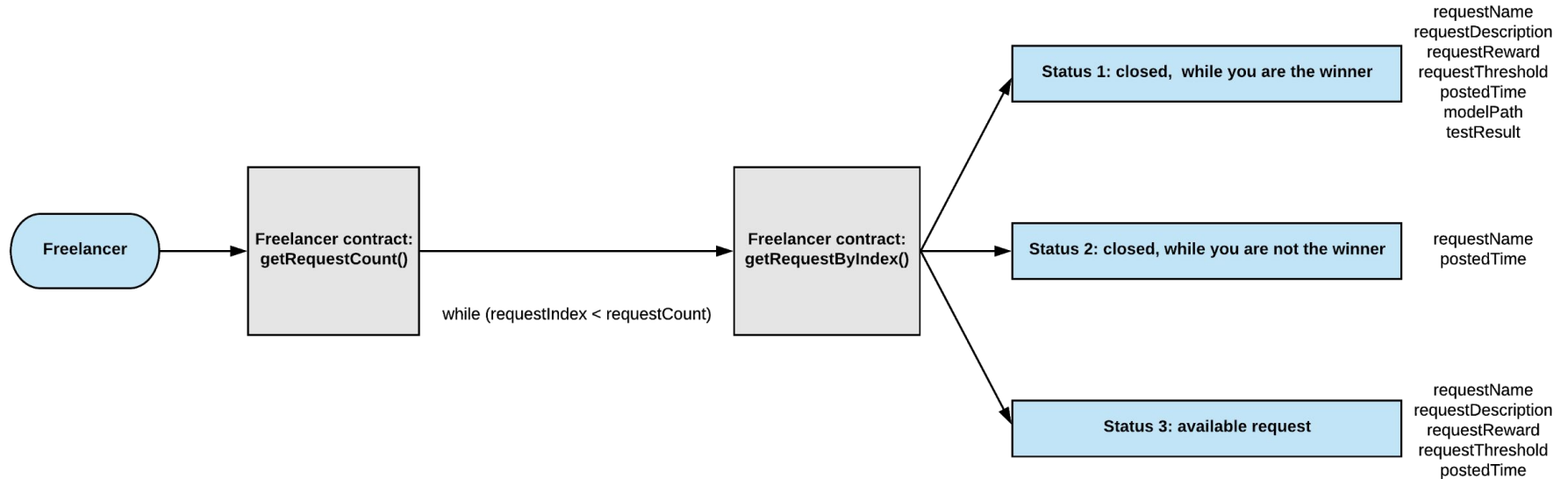- Search requests: if no keyword, show all models

- Accept requests

- **For long term transaction, it's not necessary to use shopping cart**

Freelancer Branch

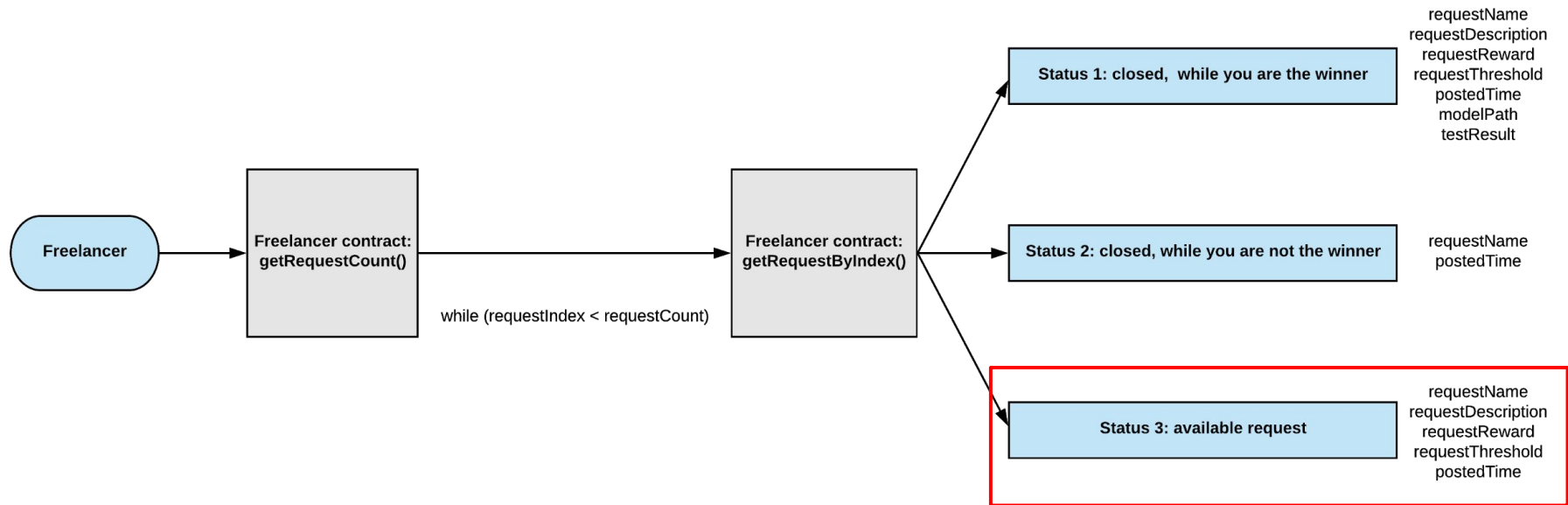- View accepted requests

Demo 4

- Act as a freelancer
- Search requests by name
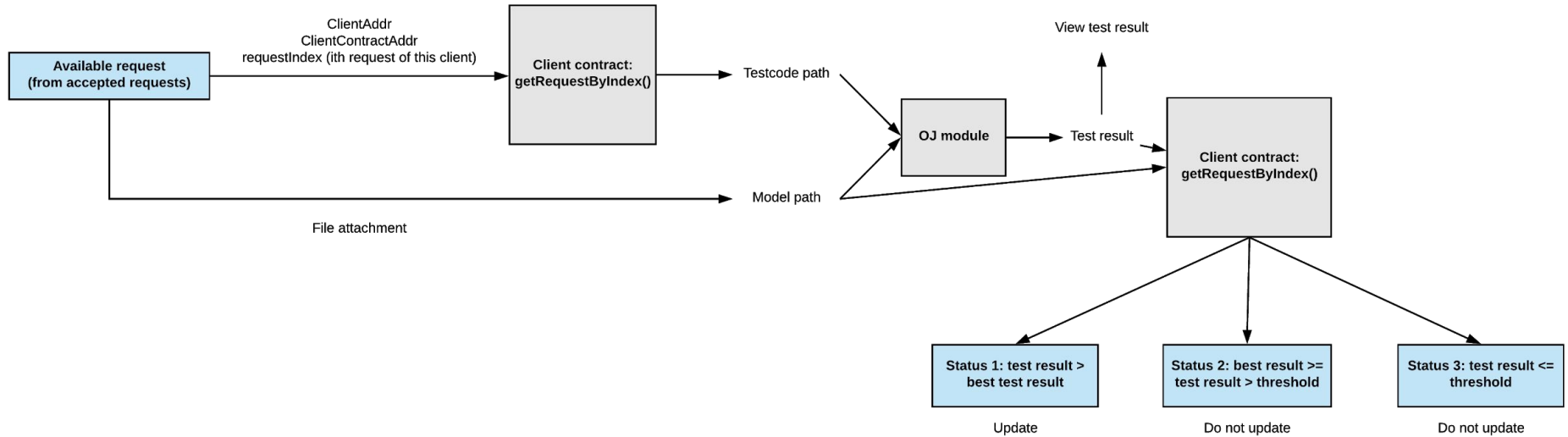- Accept requests
- View accepted requests

Freelancer Branch

- View accepted requests
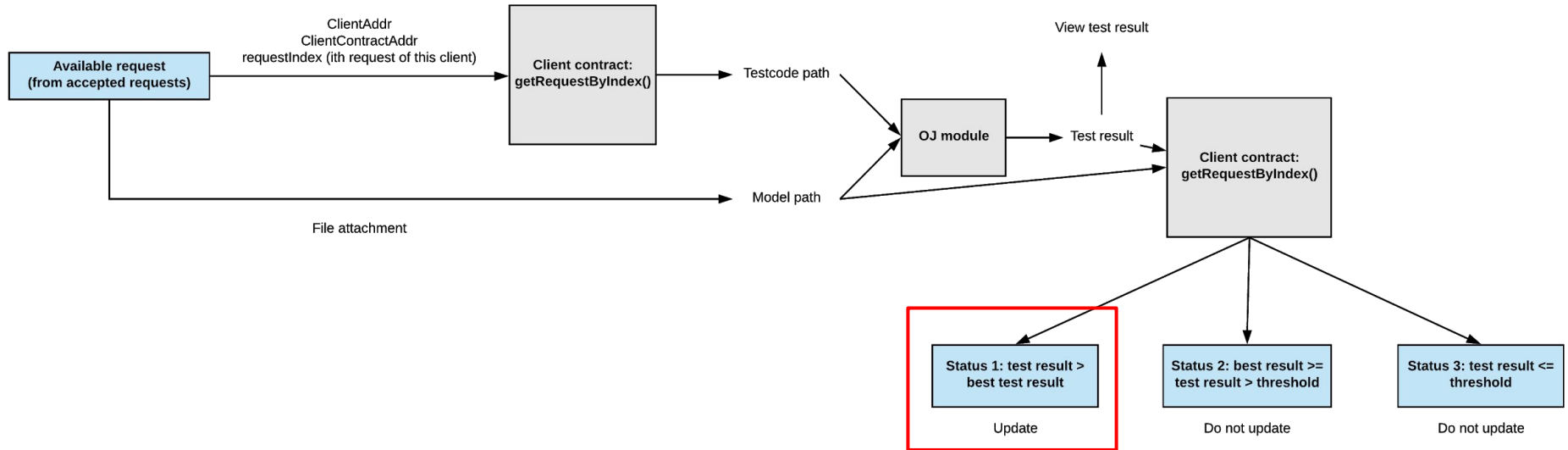
Freelancer Branch

- Submit model and view test result

Freelancer Branch

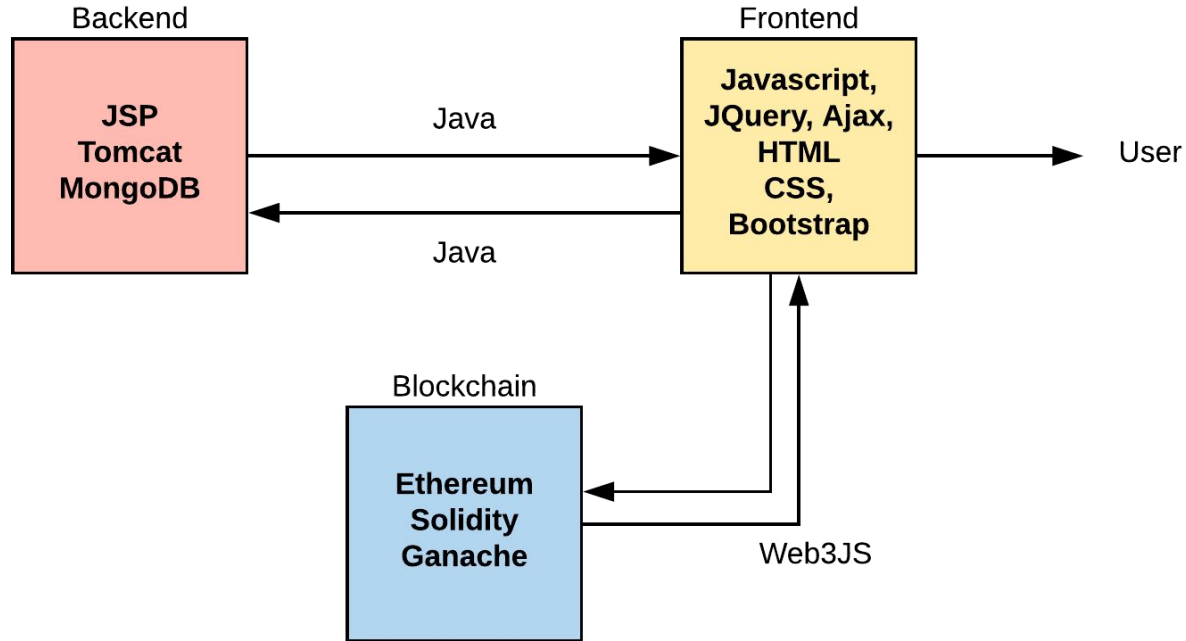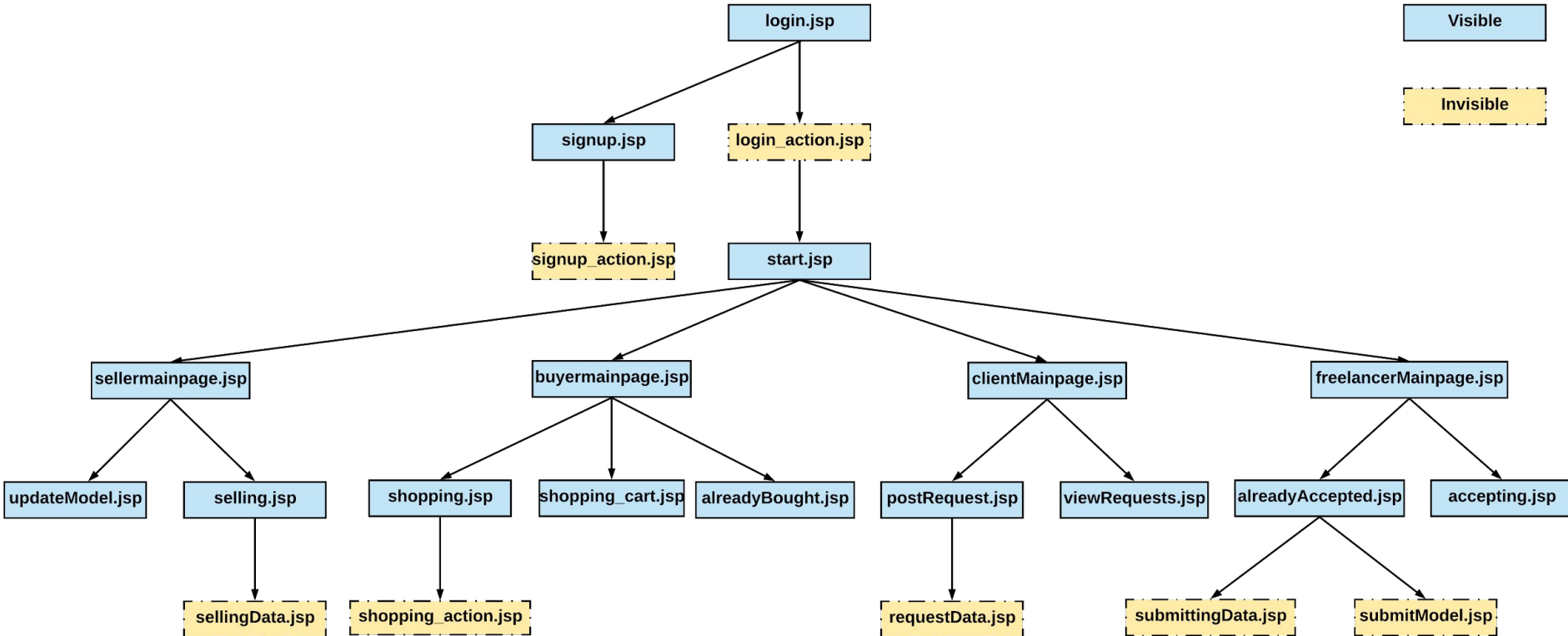● Submit model and view test result

Demo 5

- Interaction between freelancer and client
- Freelancer submits models and view result
- Client closes requests and view result
- Freelancer checks whether he wins the reward

Implementation

Technology Stacks

Page Architecture

Team Contribution

Name	Role	Work division						
		Architecture	**Page style**	**Page functionality**	**Blockchain**	**Server**	**DB**	**OJ**
Yunqiu Xu	PM & Dev	Y		Y	Y			Y
Xingshi Zhang	Dev		Y	Y		Y	Y	
Yichen Zhu	Dev			Y		Y	Y	Y
Qihai Shuai	Dev		Y				Y	Y

Summary

Summary

- Specific for machine learning models
- Online shop → short term transactions
- Outsourcing platform → long term transactions
- Based on blockchain → decentralized, anonymous, safer

Future Work

Our limitations:

- Local server
- Simple OJ module
- Simple DB operations
- Need more robust tests

Future work:

- Deploy on remote server
- More real models
- More thorough tests
- Try to improve performance
- Combine with other progress
- Explore other applications

Thanks for Your Attention!

Project website: https://sites.google.com/view/playingwithmud
Github: coming soon
Document: coming soon