Detailed Blockchain Simulation Flow

0. Startup

At the start of the application, a series of processes detailed below must be executed:

- Create wallets for the currencies with the following format: ‘LP-COINNAME’. These wallets do not have an initial balance of fiat money, they only have the related assets.
- As a liquidity pool, they are initially allocated a certain number of assets. BTC: 20, ETH: 40, USDT: 1000000, NCOIN: 100000, CCOIN: 20000.
- A genesis block will have to be created as the first block in the chain with no associated transactions.

1. Block Structure

Each block in the blockchain typically contains the following fields:

- **Block Index:**
  A sequential number representing the position of the block in the chain. The genesis block (the first block) usually has an index of 0.
- **Transactions List:**
  A collection of transactions that are pending confirmation. Each transaction has its own details (asset, amount, type, timestamp, etc.) and initially is marked as "PENDING".
- **Previous Block Hash:**
  A string representing the hash of the previous block in the chain. For the genesis block, this might be a fixed value (e.g., "0").
- **Timestamp:**
  The time at which the block is created. This helps in tracking the order of blocks and may be used in the hash calculation.
- **Nonce:**
  A number that starts at 0 and is incremented during the mining process. Its value is used to produce a hash that meets the difficulty criteria. The nonce is a critical part of the proof-of-work algorithm.
- **Current Block Hash:**
  The resulting hash value of the block after including all of its content (index, previous hash, timestamp, and nonce). This hash must satisfy the required difficulty level(4 for this specific case).

2. Proof-of-Work and Mining Process

Mining a block means finding a valid hash that meets a specific requirement defined by a "difficulty" parameter. Here's how it works:

a. Defining Difficulty

- **Difficulty Level:**
  A predetermined number that specifies how many leading zeros the block’s hash must have. For example, if the difficulty is set to 4, the hash must start with "0000".
- **Impact on Mining:**
  Higher difficulty levels mean that it takes more iterations (increasing the nonce) to find a valid hash, thus making the mining process computationally more intensive.

b. Hash Calculation

- **Data to Include:**
  When computing the hash for a block, the following data is typically concatenated into a single string:
  - The block index
  - The previous block's hash
  - The nonce value
  - The timestamp
- **Hash Function:**
  A secure cryptographic hash function (such as SHA-256) is applied to this concatenated string. The output is a fixed-length alphanumeric hash.

c. Mining Loop

- **Initializing the Nonce:**
  Start with a nonce value of 0. This value is incremented on each iteration if the generated hash does not meet the difficulty requirement.
- **Iterative Process:**
  1. Concatenate the block’s data along with the current nonce.
  2. Calculate the hash using SHA-256.
  3. Check if the resulting hash meets the difficulty criteria (i.e., it has the required number of leading zeros).
  4. If the hash does not meet the criteria, increment the nonce and repeat the process.
  5. Once a valid hash is found, the mining process is complete.
- **Storing the Result:**

- The found nonce is stored as part of the block.
- The valid hash becomes the block's current hash.
- The block is now considered "mined" and ready to be appended to the blockchain.

---

3. Creating a New Block

When there are pending transactions, the process to create a new block is as follows:

1. **Collect Pending Transactions:**
   - Gather all transactions that are still marked as "PENDING". These transactions will be included in the block.
2. **Determine Block Index and Previous Hash:**
   - The new block's index is determined by the current length of the blockchain.
   - The previous hash is taken from the most recent block in the chain.
3. **Initialize the Block Data:**
   - Set the index, transactions list, previous hash, and a current timestamp.
   - Initialize the nonce to 0 before starting the mining process.
4. **Mine the Block:**
   - Execute the mining loop as described above.
   - Once a valid hash is found, update the block's hash field with the valid hash.
5. **Confirm Transactions:**
   - After the block is mined, update the status of all included transactions from "PENDING" to "MINED".
   - Optionally, store a reference to the block in each transaction record for traceability.
6. **Append the Block to the Chain:**
   - Save the newly mined block.
   - The blockchain now grows by one block, with each block referencing the previous one through its hash.

---

4. Blockchain Integrity and Validation

Maintaining the integrity of the blockchain is crucial. Here's how validation works:

- **Recalculate Block Hashes:**
  - For each block in the chain (except the genesis block), recalculate the hash based on

the block's data (index, transactions, previous hash, timestamp, nonce).
    - Compare the recalculated hash with the stored hash in the block.
- **Check Previous Hash Links:**
    - For every block, verify that its stored previous hash matches the actual hash of the preceding block.
- **Validation Outcome:**
    - If every block’s hash is valid and every previous hash link is intact, the blockchain is considered valid.
    - Any discrepancy indicates tampering or data corruption.

---

5. Example Walkthrough

Imagine the following scenario:

1. **Pending Transactions:**
    - Several transactions (e.g., asset purchases and sales) are queued with a status of "PENDING".
2. **Block Initialization:**
    - The system determines the new block should have an index of 5 (assuming there are already 5 blocks in the chain).
    - The previous hash is taken from block 4.
    - The transactions are collected and included in the new block.
    - A timestamp is recorded, and the nonce is set to 0.
3. **Mining Process:**
    - The block data is concatenated with the current nonce value.
    - The SHA-256 hash is computed.
    - The hash is checked: if it does not start with "0000" (assuming a difficulty of 4), the nonce is incremented.
    - This loop continues until, after thousands of iterations, a nonce produces a hash that starts with "0000".
    - The successful nonce and hash are recorded.
4. **Block Finalization:**
    - The block’s hash field is updated with the valid hash.
    - Each transaction in the block is updated from "PENDING" to "MINED".
    - The block is added to the blockchain.
5. **Chain Validation:**

- At any time, the system can run a validation routine that recalculates hashes for each block and checks the chain links to ensure the integrity of the blockchain.

---

6. Key Concepts Recap

- **Nonce:**
  A number that is iteratively changed during mining until a valid hash is found.
- **Difficulty:**
  The target condition for a valid hash (e.g., a certain number of leading zeros). This defines how challenging the mining process is.
- **Hash Calculation:**
  The process of combining block data (index, transactions, previous hash, timestamp, nonce) and applying a cryptographic hash function (SHA-256) to produce a fixed-length hash.
- **Block Confirmation:**
  Once a valid hash is found, the block is considered mined. The pending transactions are then confirmed and the block is added to the blockchain.
- **Chain Integrity:**
  The immutability of the blockchain is maintained by linking each block to its predecessor via the previous hash. Validation ensures that this linkage and all computed hashes remain intact.

Resources based on app functionality

Startup

- The application must generate a **genesis block** once the app is started:

```
mysql> select * from block;
+----+-------------+------------------------------------------------------------------+----------------------+-------+--------------+---------------+
| id | block_index | hash                                                             | is_genesis           | nonce | previous_hash | timestamp     |
+----+-------------+------------------------------------------------------------------+----------------------+-------+--------------+---------------+
|  1 |           0 | 000054ba7a15300dcc48ccc86d6402f1086717bb0ff828a050892524fff8412a | 0x01                 | 77403 | 0             | 1739921419757 |
+----+-------------+------------------------------------------------------------------+----------------------+-------+--------------+---------------+
```

- The application must generate wallets for all the coins listed (app must contain more coins, this is just an example):

```
mysql> select * from wallet;
+----+----------------+---------+---------+---------------------+---------+
| id | account_status | address | balance | net_worth           | user_id |
+----+----------------+---------+---------+---------------------+---------+
|  1 | NULL           | LP-USDT |       0 |              998900 |    NULL |
|  2 | NULL           | LP-ETH  |       0 |            34253500 |    NULL |
|  3 | NULL           | LP-BTC  |       0 | 125360400.00000001 |    NULL |
+----+----------------+---------+---------+---------------------+---------+
```

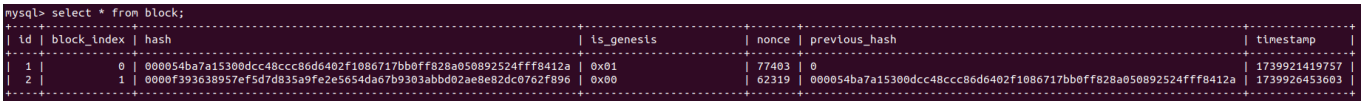
- The application must generate the liquidity pools for all the coins listed (app must contain more coins, this is just an example):

```
mysql> select * from asset;
+----+----------------+----------+--------+-----------+
| id | purchase_price | quantity | symbol | wallet_id |
+----+----------------+----------+--------+-----------+
|  1 |              0 |  1000000 | USDT   |         1 |
|  2 |              0 |    50000 | ETH    |         2 |
|  3 |              0 |    10000 | BTC    |         3 |
+----+----------------+----------+--------+-----------+
```

All other tables must be empty.

Blockchain

Blockchain with more than a Genesis Block should appear like this in the DB:



```
mysql> select * from block;
+----+-------------+------------------------------------------------------------------+------------+-------+------------------------------------------------------------------+---------------+
| id | block_index | hash                                                             | is_genesis | nonce | previous_hash                                                    | timestamp     |
+----+-------------+------------------------------------------------------------------+------------+-------+------------------------------------------------------------------+---------------+
|  1 |           0 | 000054ba7a15300dcc48ccc86d6402f1086717bb0ff828a050892524fff8412a | 0x01       | 77403 | 0                                                                | 1739921419757 |
|  2 |           1 | 0000f393638957ef5d7d835a9fe2e5654da67b9303abbd02ae8e82dc0762f896 | 0x00       | 62319 | 000054ba7a15300dcc48ccc86d6402f1086717bb0ff828a050892524fff8412a | 1739926453603 |
+----+-------------+------------------------------------------------------------------+------------+-------+------------------------------------------------------------------+---------------+
```

Transactions

```
mysql> select * from transaction;
+----+--------+--------------+-----+---------------+--------+-------------------------+------+----------+------------------+------------------+
| id | amount | asset_symbol | fee | price_per_unit | status | timestamp               | type | block_id | receiver_wallet_id | sender_wallet_id |
+----+--------+--------------+-----+---------------+--------+-------------------------+------+----------+------------------+------------------+
|  1 |   1000 | USDT         |   0 |        1.0039 | MINED  | 2025-02-19 00:53:58.735000 | BUY  |        2 |                4 |                1 |
|  2 |    0.1 | ETH          |   0 |        241.45 | MINED  | 2025-02-19 00:57:21.014000 | BUY  |        3 |                4 |                2 |
+----+--------+--------------+-----+---------------+--------+-------------------------+------+----------+------------------+------------------+
```

With User registered and operating

```
mysql> select * from users;
+----+----------------------+--------------------------------------------------------------+----------+
| id | email                | password                                                     | username |
+----+----------------------+--------------------------------------------------------------+----------+
|  1 | testuser@example.com | $2a$10$RIoTNQsjS1MgMrwU1FCU8u06y5RZLt2J/Qw2d7EVJDLzoaYcJwbf6 | testuser |
+----+----------------------+--------------------------------------------------------------+----------+
```

```
mysql> select * from wallet;
+----+----------------+------------------------------------------------------------------+---------+----------------+---------+
| id | account_status | address                                                          | balance | net_worth      | user_id |
+----+----------------+------------------------------------------------------------------+---------+----------------+---------+
|  1 | NULL           | LP-USDT                                                          |       0 | 1004718.5826265 |    NULL |
|  2 | NULL           | LP-ETH                                                           |       0 |    21198957.602 |    NULL |
|  3 | NULL           | LP-BTC                                                           |       0 |       111652800 |    NULL |
|  4 | ACTIVE         | a55f630ee5067b339585b06c4c9d589430af98ed8e14a1a3449663f06480b009 |  8996.1 |    10020.224617 |       1 |
+----+----------------+------------------------------------------------------------------+---------+----------------+---------+
```

```
mysql> select * from asset;
+----+----------------+------------+--------+-----------+
| id | purchase_price | quantity   | symbol | wallet_id |
+----+----------------+------------+--------+-----------+
|  1 |              0 | 999024.145 | USDT   |         1 |
|  2 |              0 |    49999.9 | ETH    |         2 |
|  3 |              0 |      10000 | BTC    |         3 |
|  4 |              0 |    975.855 | USDT   |         4 |
|  5 |              0 |        0.1 | ETH    |         4 |
+----+----------------+------------+--------+-----------+
```

Smart Contracts

```
mysql> describe smart_contract;
+----------------------+--------------+------+-----+---------+----------------+
| Field                | Type         | Null | Key | Default | Extra          |
+----------------------+--------------+------+-----+---------+----------------+
| id                   | bigint       | NO   | PRI | NULL    | auto_increment |
| action               | varchar(255) | YES  |     | NULL    |                |
| action_value         | double       | NO   |     | NULL    |                |
| condition_expression | text         | NO   |     | NULL    |                |
| digital_signature    | text         | NO   |     | NULL    |                |
| issuer_wallet_id     | bigint       | YES  |     | NULL    |                |
| name                 | varchar(255) | YES  |     | NULL    |                |
| status               | varchar(255) | YES  |     | NULL    |                |
+----------------------+--------------+------+-----+---------+----------------+
8 rows in set (0,00 sec)

mysql> describe wallet_keys;
+-------------+--------+------+-----+---------+----------------+
| Field       | Type   | Null | Key | Default | Extra          |
+-------------+--------+------+-----+---------+----------------+
| id          | bigint | NO   | PRI | NULL    | auto_increment |
| private_key | text   | NO   |     | NULL    |                |
| public_key  | text   | NO   |     | NULL    |                |
| wallet_id   | bigint | YES  | UNI | NULL    |                |
+-------------+--------+------+-----+---------+----------------+
```