:: Develop a Transaction Validation System ::

Develop a system where the database insert will be done by maintaining some algorithm/logic. Basically, you will develop a validation system before inserting a value into the database. The first task is to identify the set of information or values which needs to verify before writing to the database.

For example, you are working with the admin role. Now in your system admin can update or insert product information. So consider here Product is an asset. Now, whenever a product gets updated or added, it needs to be validated.

So in this way, try to identify your asset info or sensitive information that needs to validated before doing any changes. Please fillup the following link after identifying your asset/entity that you want to validate before adding it to the database.

Now implement the following validation system. A general system flow has been given below (for `Product` as asset):

:: Application 1 [For CRU] ::

1. User will fill up an add product page. (From this page you will get the product info that needs verification before storing it to DB)

2. Now call a contract function that will take this information as parameters. Somehow make this function immutable (think about it). No one can change this function logic. You can store this code in a JSON file & read this JSON file (contract_code.json) when you need to execute any particular function of that code.

4. This function will generate additional information with this product information like create time, created by, creators identity (could be a hash value), etc.

5. This information/transaction will be stored in a DB table named pending transaction log & will create a HTTP request to send this data to another application (validation app).

:: Application 2 [Validation App] ::

6. After receiving data from the contract function, this data will be stored in a `pending_transaction.json` file.

7. Transaction validators will see this pending list on their home page after login (read the JSON file & display it).

8. Validator will select one pending transaction and will approve it. (Click approve button after checking to create time, timestamp & his own valid transaction list. Each validator will maintain separate JSON.).

9. After getting this approval system will add one vote & save the transaction in that `pending_transaction.json` file again.

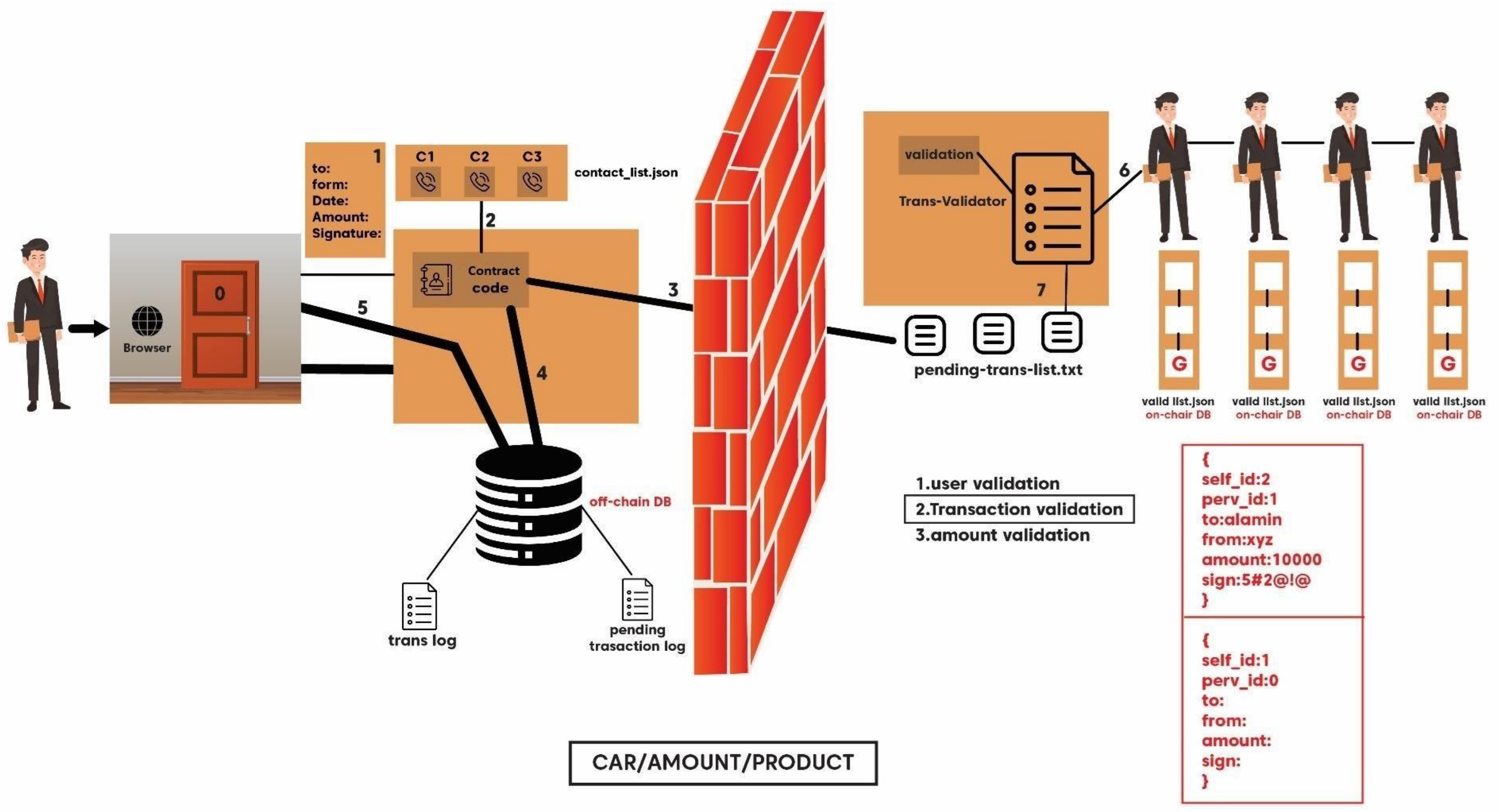
10. Another validator will login & can see who pending transaction list with vote count so he can choose any transaction & do the validation by the same process.

11. System will update the pending transaction JSON file with the new vote count & save it back to the JSON file.

12. System will check if any pending transaction gets more than 50% approval of total validators, then it will automatically be removed from the pending transaction list.

13. After removing it from the pending transactions list all the validator's own JSON file will be updated. (All the validators maintain a valid transaction list as JSON, so new valid transaction will be added here with a unique id).

14. This valid information/transaction will be sent back to the client app again. Client app will receive this transaction information & it will remove the transaction from the pending transaction log table & will insert it to the valid transaction log table (insert value to the product table).


Browser
0
1
to:
form:
Date:
Amount:
Signature:
C1
C2
C3
contact_list.json
2
Contract code
3
4
5
off-chain DB
trans log
pending trasaction log
CAR/AMOUNT/PRODUCT
validation
Trans-Validator
6
7
pending-trans-list.txt
valid list.json on-chair DB
valid list.json on-chair DB
valid list.json on-chair DB
valid list.json on-chair DB
G
1.user validation
2.Transaction validation
3.amount validation
{
self_id:2
perv_id:1
to:alamin
from:xyz
amount:10000
sign:5#2@!@
}
{
self_id:1
perv_id:0
to:
from:
amount:
sign:
}