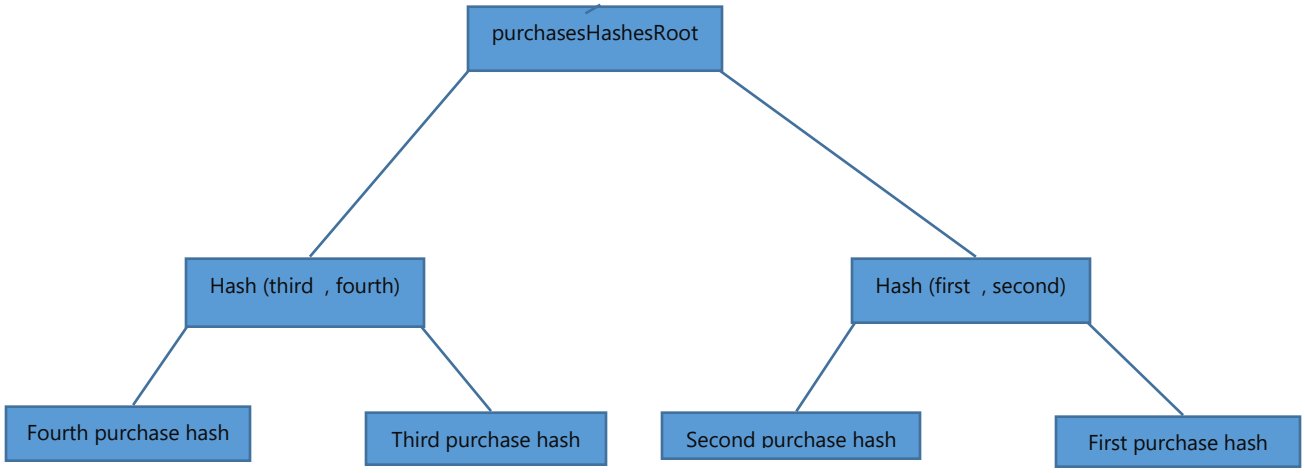
A Secure Blockchain Solution: Empowering Sellers with Trust Management and Safe Withdrawals

This project is a smart contract for managing online purchases, which addresses the need for buyers to trust sellers regarding whether they will receive the purchased item on time. With the advancement of technology and the expansion of the online space, online shopping in modern societies has made significant progress. As technology advances, it brings prosperity and comfort to people, but on the other hand, it also provides many ways for opportunistic individuals and scammers to carry out various fraudulent activities. One of the areas where fraud has increased in recent years is online shopping. Fraudsters attract the trust of their customers through extensive advertising, and after the customer makes their purchase from their online store, they do not deliver the purchased item. This project provides a blockchain-based smart contract solution to this problem. The logic of this contract is as follows: The user's purchase is made using cryptocurrency (preferably stable coins like Tether) on the blockchain network, but the money for this purchase is not deposited into the seller's account until two days after the delivery time of the goods. These two days provide an opportunity for the user to complain about not receiving their goods. If the seller does not respond to the user's complaint within these two days, the user can withdraw their money from the network.

When making a purchase, the user can use the **`createPurchase`** function, The inputs of the function are explained below:

- **`purchaseID`**: The ID of the item that the seller has considered for their item.
- **`price`**: The price of the purchased item (in Tether).
- **`deliverTime`**: The time to deliver the item to the user (in Unix timestamp).
- **`seller`**: The seller's wallet address.
- **`hashRandomNumber`**: The hash of a random number that the user uses when making the purchase transaction.
- **`hashes`**: An array of hashes used to calculate the Merkle tree root for each user, which is ultimately stored in the **`purchaseHashesRoot`** mapping.

When executing this function, it is first checked that the maximum time for delivering the item (in days) is less than the **`Period`** value. Then, it is checked whether the buyer is not among the blocked addresses (**`BlockedAddresses`**). Then, the transfer of Tether currency to the contract is performed. After that, the root hash of the Merkle tree is updated. Calculate the hash of each purchase as follows: **`Hash(PurchaseID, Price, DeliverTime, Seller, HashRandomNumber)`**.



After calculating the hash for each purchase, the obtained value is inserted as a branch into this tree, and finally, the root of the Merkle tree (`purchaseHashesRoot`) is recalculated and placed in the `purchaseHashesRoot` mapping.

The `conditionalTokens` mapping is used to store the balance of sellers (in Tether) deposited to the contract by buyers. This mapping contains 8 accounts, each representing the amount of purchases made on each day. At the end of this function, it first calculates to which account the user's purchase balance should be added, and then performs this balance addition.

When the purchased item is not delivered to the user at the specified time, the user can submit a complaint using the `SubmitComplaint` function, Initially, the function checks whether the delivery time has arrived and whether it has been less than 12 hours since the delivery time. This means that if the user's item has not been delivered, they must file a complaint within less than 12 hours from the delivery time. Additionally, the user must not be among the blocked network addresses.

The inputs of this function include:

- `purchaseID`: The ID of the item that the seller has considered for their item.
- `deliverTime`: The time to deliver the item to the user (in Unix timestamp).
- `seller`: The seller's wallet address.
- `hashRandomNumber`: The hash of a random number that the user uses when making the purchase transaction.

Finally, the complaint is recorded in the `complaints` mapping.

A Secure Blockchain Solution: Empowering Sellers with Trust Management and Safe Withdrawals

When the user's complaint is registered on the network, the seller has 12 hours to reject the complaint in case they have delivered the item to the user and have the user's random number. Otherwise, after the expiration of this 12-hour period for the seller, the user can withdraw their money from the smart contract.

Rejecting the user's complaint is done using the **`BlockedInvalidComplaints`** function, The inputs of this function include:

- **`plaintiff`**: The complainant's address.
- **`seller`**: The seller's address.
- **`randomNumber`**: The random number that the user participates in the purchase transaction with and is obligated to provide to the seller when receiving the item.

First, this function checks whether more than 12 hours have passed since the complaint was registered, and also whether the user's address is not among the blocked addresses. Then, the hash value of **`randomNumber`** is calculated, and it is compared with the value registered by the user in the **`complaints`** mapping. If these values are equal, meaning that the seller has delivered the item and has the user's random number, the complaint is rejected, and the complainant's address is recorded in the **`BlockedAddresses`** mapping.

When the user has submitted their complaint and it has not been rejected by the seller, the user can withdraw their paid money from the smart contract by calling the **`ResolveComplaint`** function, The inputs of this function are:

- **`price`**: The price of the purchased item.
- **`seller`**: The seller's address.
- **`hashes`**: An array of hashes used to calculate the Merkle tree root for the **`purchaseHashesRoot`**.

First, this function checks whether more than 12 hours have passed since the complaint was registered, and also whether the user's address is not among the blocked addresses. Then, the Merkle tree root **`purchaseHashesRoot`** is calculated using the details of the complaint recorded in the **`complaints`** mapping and the **`hashes`** array. If the calculated value matches the value registered in the **`purchaseHashesRoot`** mapping, the user's purchase money is withdrawn from the contract.

After the **`period`** has elapsed, sellers can withdraw their money from accounts where the time has come for withdrawal. To withdraw money, sellers must call the **`withdrawFunds`** function,This function does not take any inputs.

First, it calculates using `block.timestamp` to determine the time of withdrawal from which account. Then, the balance is deducted from the desired account in the `conditionalTokens` mapping, and the transfer of Tether to the seller's account is carried out from the `MainContract` contract.