Smart Contracts of Investopolis

Smart contracts play a pivotal role in developing the business logic of decentralized applications (DApps). A smart contract is a self-executing contract with the terms of the agreement directly written into code. These contracts are deployed on blockchain platforms, such as Ethereum, and they enable the automation, transparency, and trustworthiness of business processes in a decentralized environment.

The purpose of using smart contracts in developing the business logic of a DApp is to eliminate the need for intermediaries or trusted third parties. Traditional business applications often require central authorities, such as banks or governments, to validate and enforce agreements. Smart contracts, on the other hand, utilize blockchain technology to replace these intermediaries by providing a decentralized and immutable system of agreement execution.

Here are some key purposes of smart contracts in developing business logic within a DApp:

1. Automation

2. Trust and Transparency

3. Decentralization

4. Cost and Efficiency

5. Customizability and Flexibility

Overall, smart contracts serve as the backbone of decentralized applications, enabling the development of trustless, automated, and transparent business logic. By leveraging the power of blockchain technology, businesses can create decentralized systems that are efficient, secure, and resistant to censorship or manipulation.

There have been several smart contracts written to initiate and execute Blockchain Transactions and store the business logic of Investopolis. They are as follows:

1. context.sol

The purpose of this smart contract is to provide context to any transaction. Context indicates the details about the wallet address which initiated the transaction and the message that is encased withing the transaction. Therefore, context.sol consists of two functions:

- _msgSender: Returns the wallet address of the transaction initiator
- _msgData: Returns the data sent through the transaction

This smart contract will be inherited in whichever smart contract necessary.

2. users.sol

The purpose of this smart contract basically is to perform user management. Registration of new users, sign in operations, retrieval of user information, verification of new users via administrators, etc., are all implemented in this smart contract. The functions written in this contract are as follows:

- login: This function performs the login operation for users. There are two ways in which a user can login, both of which are implemented in this function
- registerPossible: This function returns a boolean of whether the supplied credentials can be used to register a new user on the platform. This function is employed in order to conserve space dedicated through IPFS by minimizing the necessity to perform data purging.
- register: This function performs the new user registration operation. All the credentials furnished shall be saved in the contract for further user requests.
- userid: This further translates a given wallet address to the User ID it is associated with amongst the registered users.
- metamask: This further translates a given User ID to the wallet address it is associated with amongst the registered users.
- verify: This function is used to verify the registration of new users. As verification is only possible through administrators, only they are authorized to use this function.
- unverified: This function returns the user details of all the currently unverified users. This will be the case when an administrator wants to retrieve data of unverified users in order to assess their credibility and verify them. Only administrators are authorized to use this function.
- usersdb: This function returns the user details of all the verified users on the platform. This will be the case when an administrator wants to view all the participants' details. Only administrators are authorized to use this function.
- addAdmin: Investopolis begins with 1 administrator at the beginning. However, in order to make the process of admission of new users into the platform and their verification more decentralized, multiple number of administrators can partake. New administrators can be added only by existing administrators and thus, only administrators are authorized to use this function.
- removableFiles: In order to remove redundant files from IPFS i.e., files which are associated with rejected users, this function can be called to know their CIDs. This function can only be used by administrators:
- getInfo: In order to mention the ideators' address and contact details on the share documents, there must be a way to return only necessary information without returning extra information. This function is implemented to do the same.

- userids: The User IDs of all the verified participants of the platform are returned by this function. The same will be used for the search results in the search bar placed on messages and ideators features.

In addition to these functions, they are some helper functions employed to perform various tasks within the contract. They are:

- _areEqual: This function is used to compare two strings as there is no in-built function in solidity to perform such operation.
- _isAdmin: This function checks whether the transaction initiator is an administrator to assess whether he/she is authorized to perform administrator-level operations.

A mutex lock is implemented in the contract to avoid any sorts of race conditions in several function implementations.

3. messages.sol

The purpose of this smart contract basically is to perform communication management. Retrieval and Transmission of messages are implemented in this smart contract. The functions written in this contract are as follows:

- retrive: This function retrieves personal as well as broadcast messages that are sent to a particular user. This function is also responsible for data purging i.e., clearing out messages that are stored in the contract from 7 or more days.
- send: This function appends a new message and its metadata into the smart contract storage.

Helper function _areEqual is implemented in this contract too.

4. shares.sol

The purpose of this smart contract basically is to perform the management of a decentralized stock-exchange for unlisted shares. Handling ideator related information, operations such as new share investment, transfer of existing shares, verification of the validity of share documents, etc., are all implemented in this smart contract. The functions written in this contract are as follows:

- addIdeator: This function appends the information about a new ideator such as the evaluation of his start-up and the face value he set for his shares into the smart contract storage.
- updateIdeator: This function updates the company evaluation data stored on the smart contract.
- getIdeator: This function retrieves the information about a specific ideator.
- retrieve: This function retrieves information about all the ideators currently registered on the platform who had updated their company evaluation at least once.

- invest: This function is called whenever an investor initiates an investment towards an ideator's shares. A unique code is generated through the transaction metadata which will then be returned in order to be presented in the share document generated in the frontend after successful investment.
- verifydocument: This function verifies whether the unique code supplied is a valid one assigned to a share document. It also specified whether or not the share document currently has the shares or has been transferred prior.
- interest: This function marks the interest of an investor towards reselling the share(s) he bought previously.
- disinterest: This function eradicates the interest an investor marked before towards reselling the share(s) he bought.
- retrieveInterests: This function retrieves the entire data with reference to the interest marked by various users towards reselling the share(s) they bought previously.
- transfer: This function performs the transfer of shares from one investor to another.

As the functions invest and transfer involve transfer of ether in addition to the transaction gas fees, certain payment-associated functions are written in the contract. They are:

- receive
- fallback
- getBalance

Certain helper functions such as _areEqual and _unit256ToString were also written in the contract. A mutex lock is placed as a modifier within the contract to avoid race conditions in the context of investment or transfer of shares.

5. lands.sol

The purpose of this smart contract basically is to perform the management of real-estate through the perks of Blockchain Technology. Handling registrar related information, operations such as new land registration, transfer of ownership of lands, verification of property transactions, etc., are all implemented in this smart contract. The functions written in this contract are as follows:

- addRegistrar: This function appends the information about a new registrar such as his/her region coordinates, the stamp duty, transfer duty in his locality, etc., into the smart contract storage.
- updateRegistrar: This function updates registrar related information like the stamp and transfer duties.
- getRegistrar: Retrieves information related to a specific registrar

- retrieve: Retrieves information related to all the registrars registered on the platform.
- registerLand: Initiates the registration process of a new property on the platform.
- getOwnedLands: Retrieves information related to all the properties owned by a particular investor.
- transferLand: Initiates the transfer of ownership process for a property already registered on the platform.
- retrieveCoordinates: Retrieves coordinates that encompass a specific land registered on the platform.
- unverifiedLands: Retrieves the information about all the properties which have pending verifications with respect to registration on the platform or transfer of ownership requests.
- verifyRegistration: The function is called by a registrar whenever he intends to verify the authenticity of a particular property newly registered on the platform.
- VerifyTransfer: This function is called by a registrar whenever he intends to accept the transfer of ownership request of a particular property already registered on the platform.

As the function verifyTransfer involves transfer of ether in addition to the transaction gas fees, certain payment-associated functions are written in the contract. They are:

- receive
- fallback
- getBalance

Certain helper functions such as _areEqual were also written in the contract.

6. nfts.sol

The purpose of this smart contract basically is to perform the management of real-estate through the perks of Blockchain Technology. Handling registrar related information, operations such as new land registration, transfer of ownership of lands, verification of property transactions, etc., are all implemented in this smart contract. The functions written in this contract are as follows:

- mint: This function mints a new NFT on the blockchain network. The NFT Metadata shall include the IPFS CID wherein the image (artwork or photograph) is pushed.
- getMintCount: This function returns the count of number of NFTs minted by a vendor up until then.
- retrieve: This function retrieves the data related to the NFTs minted by a particular vendor.
- interest: This function marks the interest of the vendor to list it for sale.

- disinterest: This function eradicates the interest marked prior by the vendor to list it for sale.
- retrieveInterests: This function retrieves the interests marked by the vendors of the platform with regards to the NFTs they minted which they intend to sale.
- buy: This function performs the sale of NFT operation. The ownership of the NFT is transferred from the vendor to the investor who bought it.

As the function buy involves transfer of ether in addition to the transaction gas fees, certain payment-associated functions are written in the contract. They are:

- receive
- fallback
- getBalance

Certain helper functions such as _areEqual and _unit256ToString were also written in the contract. A mutex lock is placed as a modifier within the contract to avoid race conditions in the context of investment or transfer of shares.

7. tokens.sol

The purpose of this smart contract is to manage the reward system which plays a pivotal role in the Customer Loyalty Program (CLP) we wish to implement for the Decentralized Investment Platform – Investopolis. The functions written as part of that are:

- getInvestor: This function returns the token profile of a specific investor. The number of tokens the investor earned up until then and the reward he enabled at that instant of time (if he/she did so) shall be returned.
- invest: This function increments the count of tokens i.e., loyalty points earned by the investor by one
- activateToken: This function activates a particular reward based on the number of tokens the investor collected till that particular instant of time.
- transferTokens: This function transfers the tokens from one investor's token profile to another

Therefore, we have pondered over the purpose and usage of smart contracts in the context of decentralized applications i.e., dApps, listed out the smart contracts we employed to implement the business logic of Investopolis and briefly outlined the purpose of each function written in those contracts.