Design Document

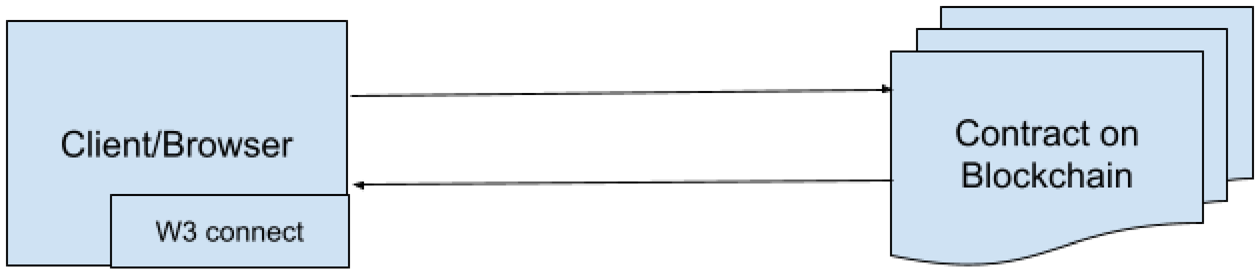
Application - Dutch Auction

Team - Deepak Kumar, Piyush Madan, Prateek Jain

Requirements

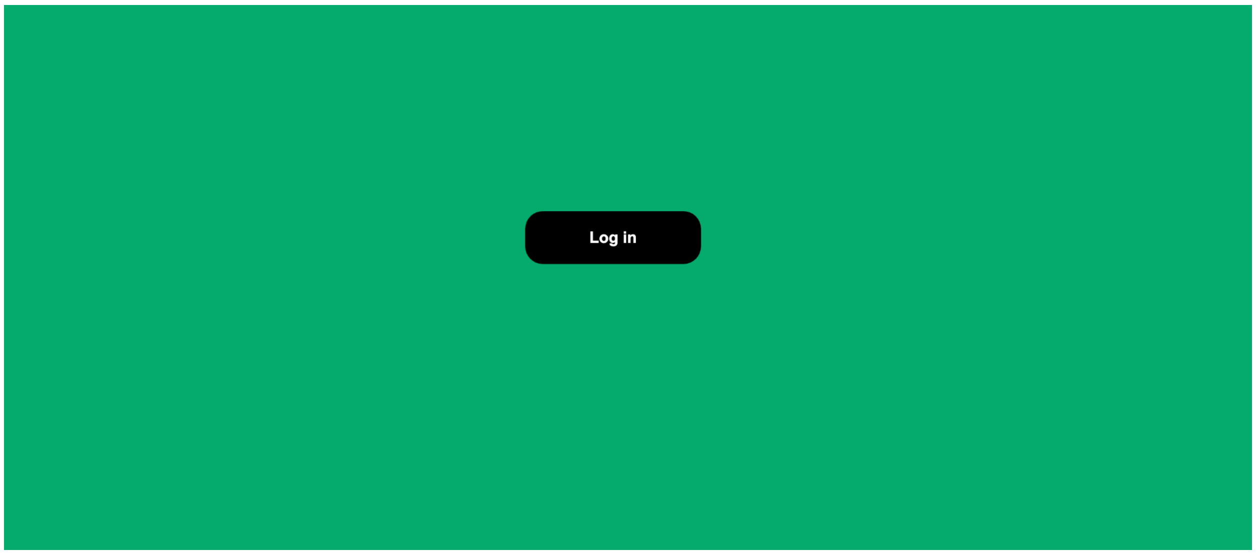
To design and develop a DAPP on Dutch Auction which fulfills the requirements mentioned in this linked document.

Flow Diagram

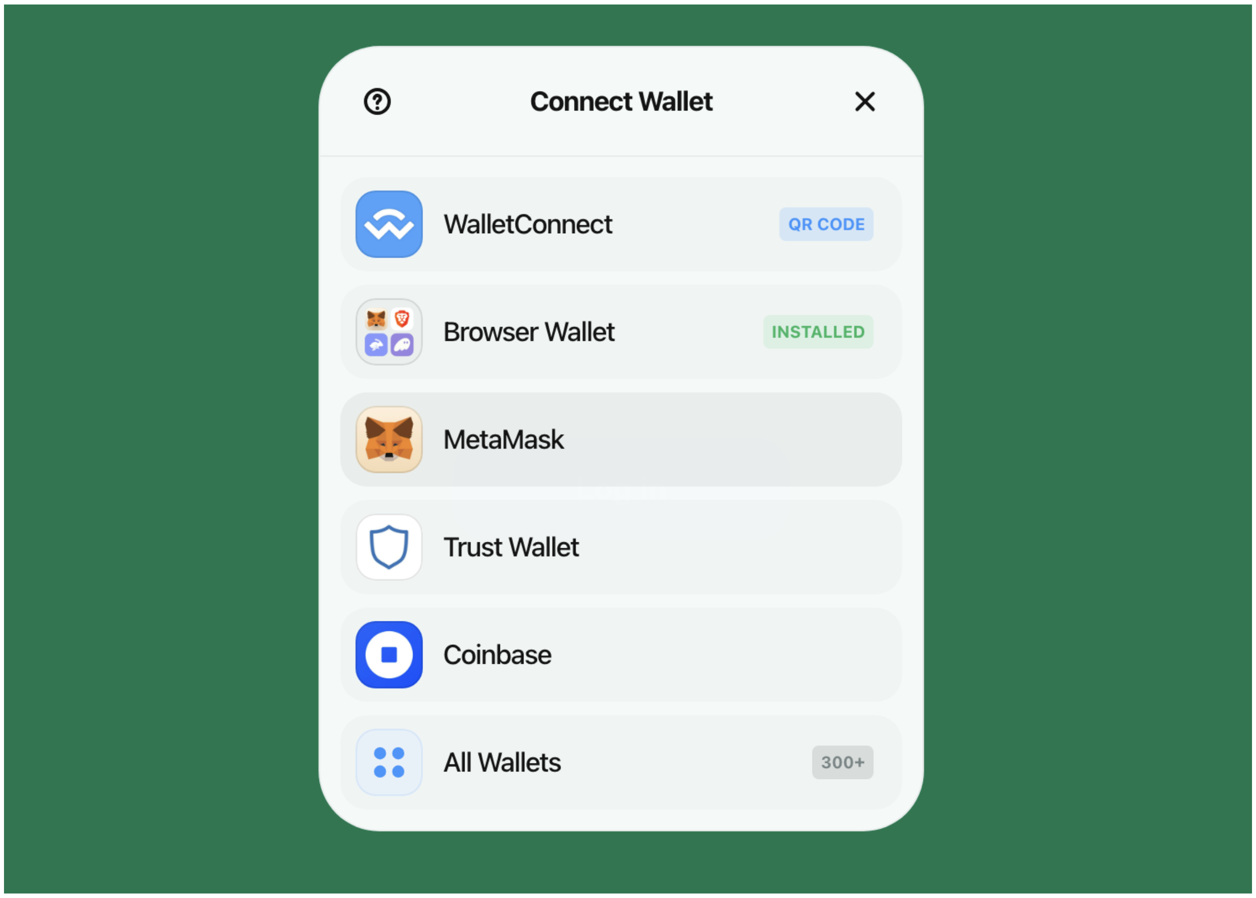


UI Flow

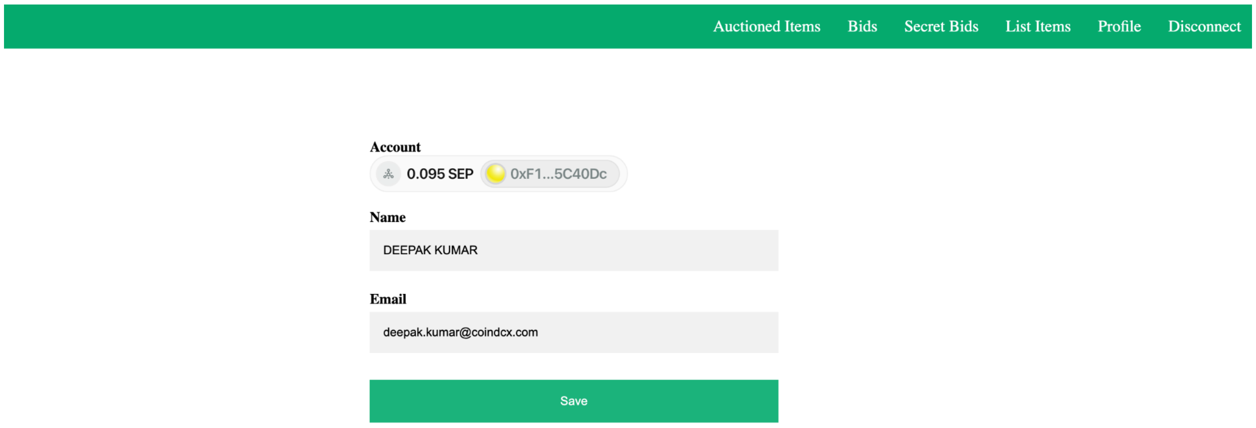
1. User open the application and asked to log in using wallet connect



2. User choose which wallet they want to use to connect.



3. First time user go to profile section to complete their profile

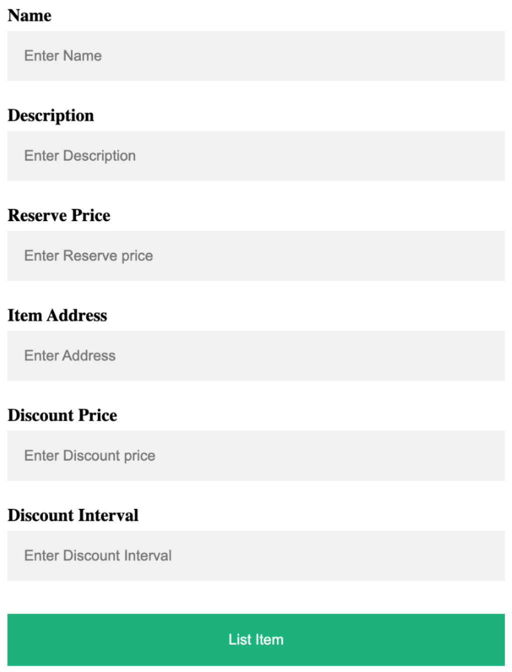


Account
0.095 SEP 0xF1...5C40Dc

Name
DEEPAK KUMAR

Email
deepak.kumar@coindcx.com

4. User can go to List Item section to add new Item for bid



Name

Enter Name

Description

Enter Description

Reserve Price

Enter Reserve price

Item Address

Enter Address

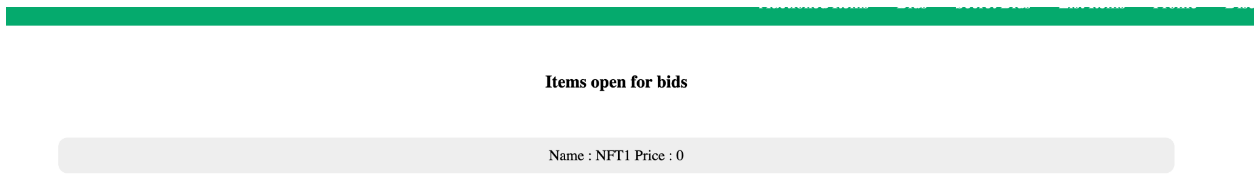
Discount Price

Enter Discount price

Discount Interval

Enter Discount Interval

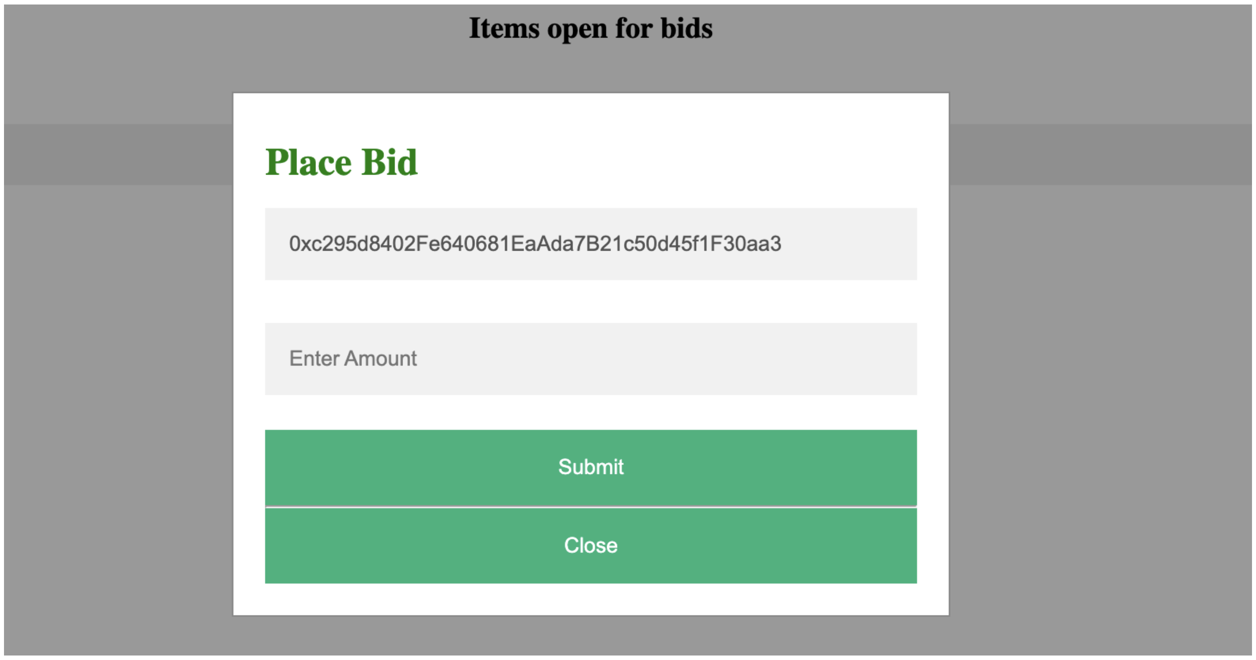
5. User can bid on listed items



Items open for bids

Name : NFT1 Price : 0

6. When user click on item a pop up will open to enter price details and submit



7. User will have option to see all auctioned items
8. User can log out using Disconnect button

BE Flow

- There is no BE service for this app
- Contract act as BE for read/write calls
- Contract is stored in blockchain in distributed manner along with all the data
- Contact will have following interface

```JavaScript
contract AuctionContract {
  struct User {
    string name;
    string email;
    Item[] items;
  }

  struct Item {
   string name;
   string description;
   uint reservePrice;
   address addr;
```

```
    uint startsAt;
    uint expiresAt;
    uint discountPrice;
    uint discountInterval;
    address owner;
  }
  //status 0 - not started, 1 - secred bid open, 2- secret bid close and normal
bid opens
  // 3 - bid close
  struct Auction {
    Item item;
    address owner;
    uint status;
    uint startPrice;
    mapping(address => uint) secretBids;
  }

  mapping(address => User) public users;
  address[] public userList;
  mapping(address => Auction) public auctionItems;
  address[] public allAuctionItems;

  function upsertUser(string memory name, string memory email) public {
  }

  function getUser(address adr) public view returns (User memory) {
  }

  function listItem(Item memory item, uint expireAfter) public {
  }

  function placeSecretBid(uint amount, address addr) public {
  }

  function getItemsForSecretBids() public view returns (Item[] memory) {
  }

  function getItemsForBids() public view returns (Item[] memory) {
  }

  function getPrice(address addr) public returns (uint) {
  }

  function buy(uint amt, address addr) external returns (string memory, bool) {
  }
```

```
    function getAllItems() public view returns (Item[] memory) {
    }

    function closeSecretBid(address addr) internal {
    }

    function calculateStartPrice(address addr) internal view returns (uint) {
    }
}
```

Structs

There will be following structs in the contract for different entities:

1. User : To store user data
2. Item : To store item data
3. Auction: To store data related to auction of an item

Variables

There will be following variables to store state of contract:

1. users : It will store mapping of user address to user profile
2. userList : It will be an address array to store all the users signed up on platform to iterate over all the users
3. auctionItems: It will be mapping to store the auction item against their address
4. allAuctionItems: It will be an array to store the address of all auction items for iterate

Price Calculation

There will be 2 price calculation functions in above contract:

1. getPrice : This function will be used to get the current price of an auction item. It will use discountInterval and discountPrice to calculate the price since start of auction
2. calculateStartPrice: This function will be used to calculate the starting price of an item after users place a secret bid. The logic will be 1.5 times of maximum bid and if it is less than reserve price then price will be 1.5 times of reserve price.

Security Flaws

There are following security flaws of above design:

1. User profile is being managed on blockchain, so every time user initiate update request on their profile, gas will be utilized from user wallet
2. The read calls on contract are initiated from the browser, so there is no check on who is calling, what kind of data.
3. The address of item can be any valid address, there is no validation on the valid item being mapped to that item
4. There is no validation on list of item, whether user owns it or not
5. There are no tokens involved in the above design. Amount is just dummy value.