**SOFTWARE REQUIREMENTS SPECIFICATION**

**FunixPricingChain**

**1. Product overview**

* Product name: FunixPricingChain – an online product pricing application.
* Product description:
  + Provide a platform where participants can price a product and at the end of each session the admin set a price for that product and those participants who joined the pricing will have a new accumulated deviation which is a weight for the next session that they join
* Users – who will use the application:

|  |  |  |
| --- | --- | --- |
| No. | User | Description |
| 1. | admin | Who in charge of managing the sessions and determine the final price |
| 2. | participants | Who price the product in the session and contribute to the final price of the product |

* The benefit to users:

The Dapp is beneficial to people who do not know the price of a product that they want to buy or a product that is going to be sold but does not have a specific price on the market. Briefly, the FunixPricingChain Dapp addresses the problem of pricing of an unknown price product.

* Functional Diagram:

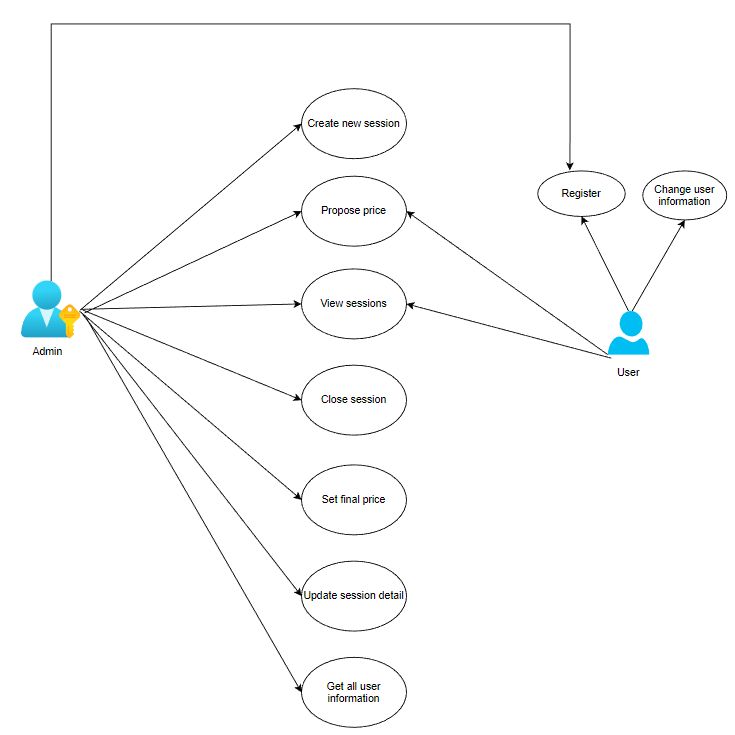


Figure 1 Functional diagram

**2. Functional Requirements**

2.1. Create new session

* Function name: createNewSession
* Function description: create new pricing session
* User to use: Only admin can use the function
* Function Inputs:

|  |  |
| --- | --- |
| **Input** | **Requirement** |
| Product name | String, required. |
| Product description | String, required. |
| Product images | Array of string, optional. |
| Session duration | Uint256, default to 0 |

* Function rule: the image’ string is 46-character-length
* Function result: a new product with valid information is added and a pricing session is initiated.

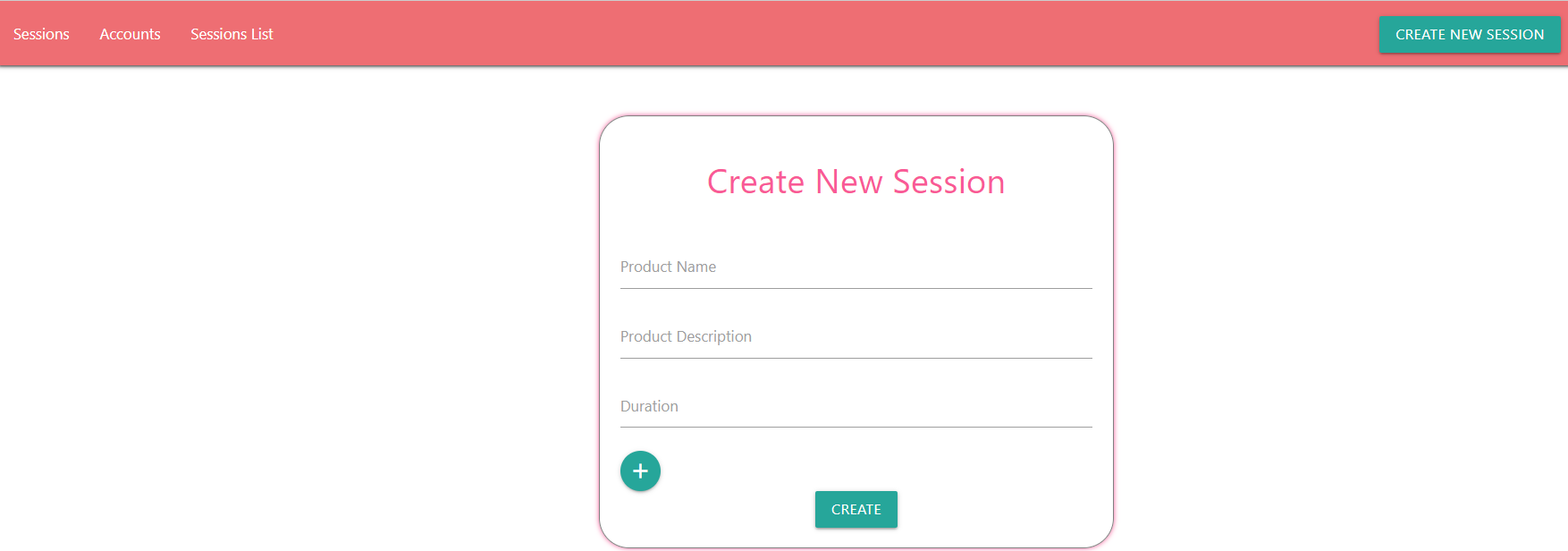


Figure 2 mockup screen for create new session

2.2 Register

* Function name: register
* Function description: sign up for new user
* User to use: new user
* Function Inputs:

|  |  |
| --- | --- |
| **Input** | **Requirement** |
| Full name | String, required. |
| Email | String, required. |

* Function rule: Have not registered before and must fill in both name and email. If a user has not signed up before then he/she will be redirected to the register page.
* Function result: new participant is created.

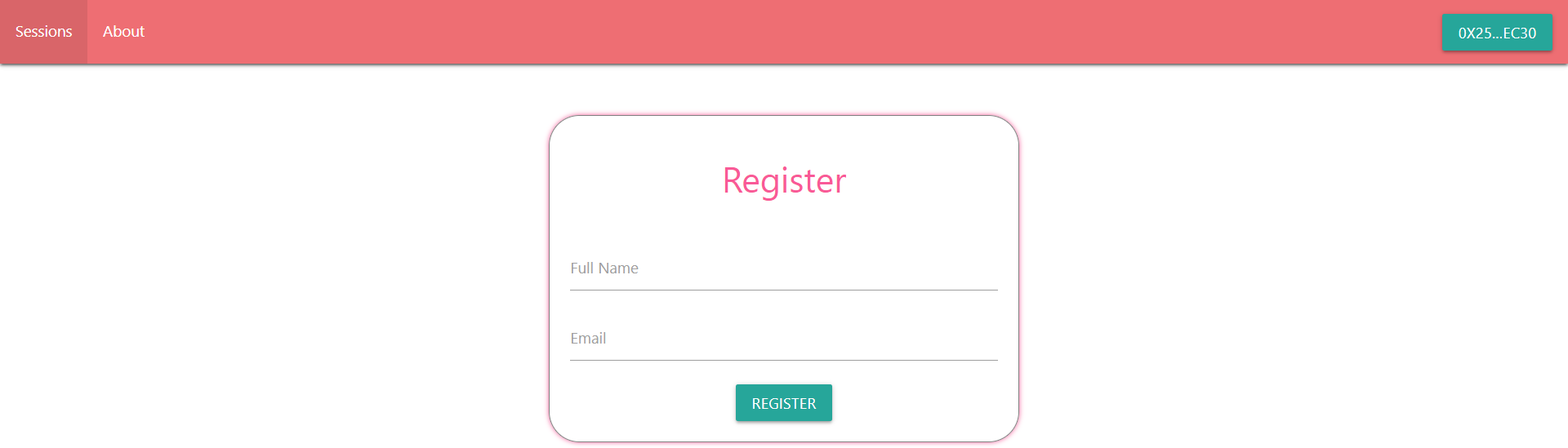


Figure 3 mockup screen for register

2.3 Change user information

* Function name: updateParticipantDetail
* Function description: allow user to change his/her full name and email
* User to use: user who has registered
* Function Inputs:

|  |  |
| --- | --- |
| **Input** | **Requirement** |
| Full name | String, required. |
| Email | String, required. |

* Function rule: user can change full name or email or both.
* Function result: update user new information.

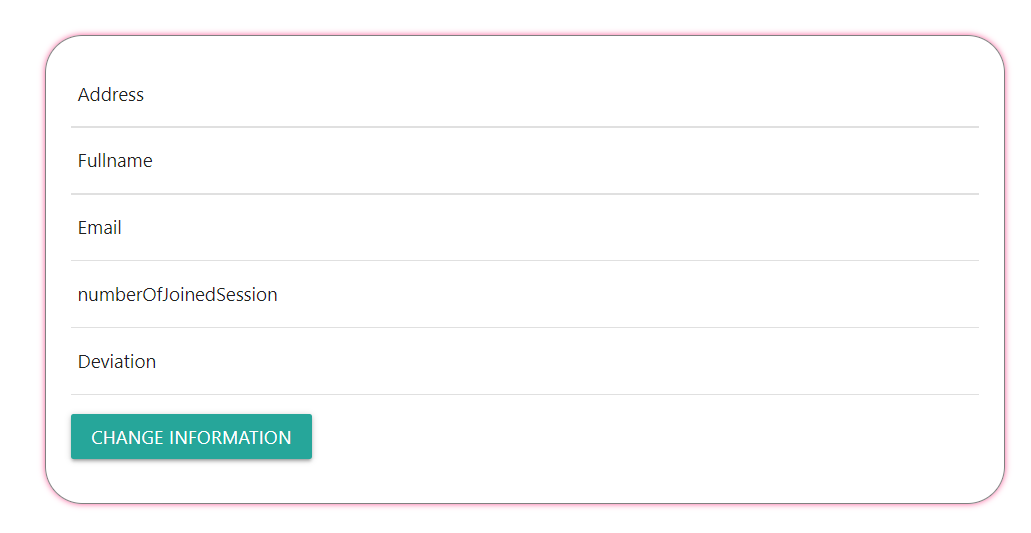


Figure 4 mockup screen for change user information

2.4 View sessions

* Function name: getSessions
* Function description: allow user and admin to get all the sessions
* User to use: user and admin
* Function Inputs: none
* Function rule: user who has already registered and connected to the Dapp can view all the sessions
* Function result: show all available sessions on screen for user and admin

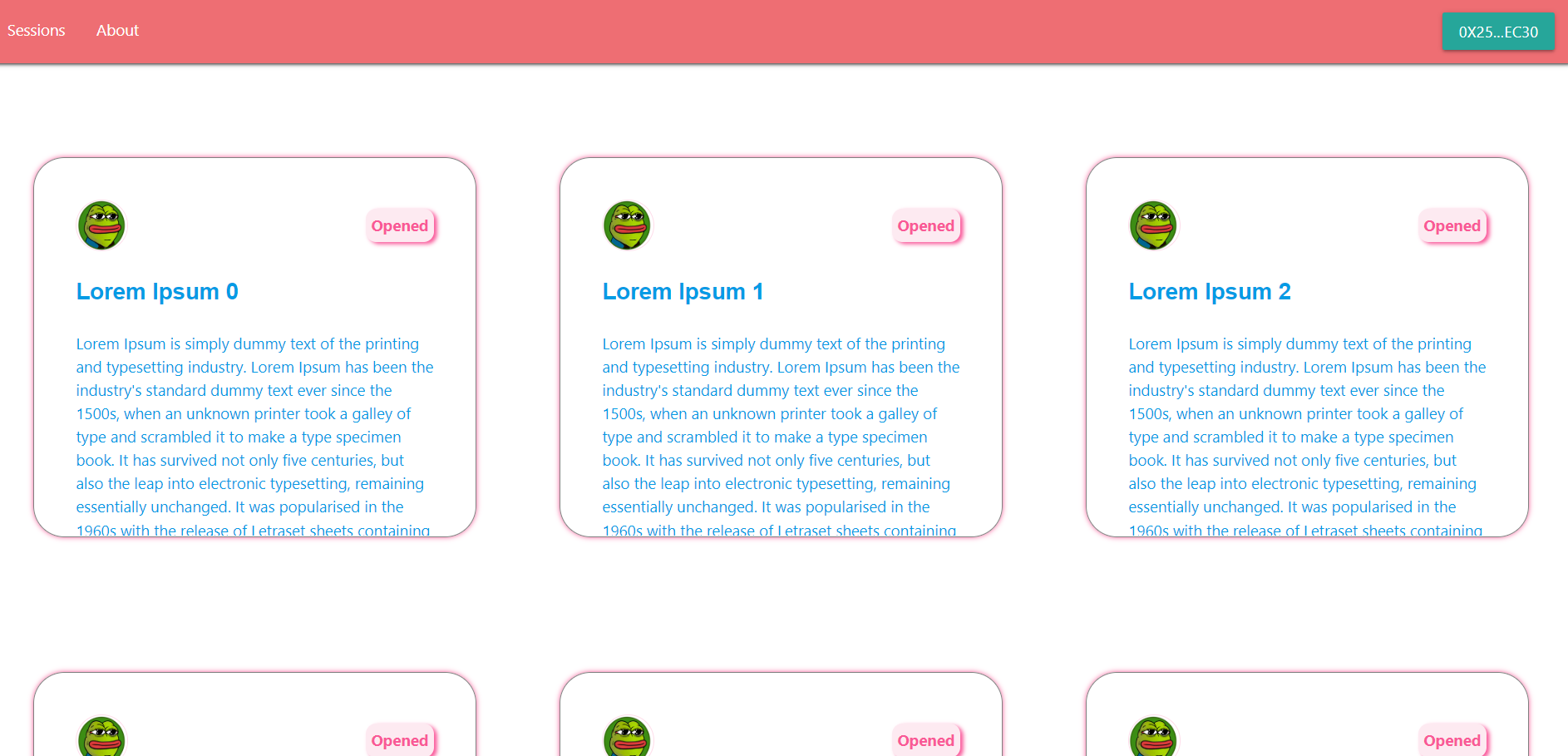


Figure 5 mockup screen for view sessions

2.5 Propose price

* Function name: propose
* Function description: allow participant to propose price
* User to use: participant
* Function Inputs:

|  |  |
| --- | --- |
| **Input** | **Requirement** |
| price | Uint256, required. |

* Function rule: Only participant who has already registered and connected can propose
* Function result: propose price of participant is stored and update the number of session that participant has joined by 1

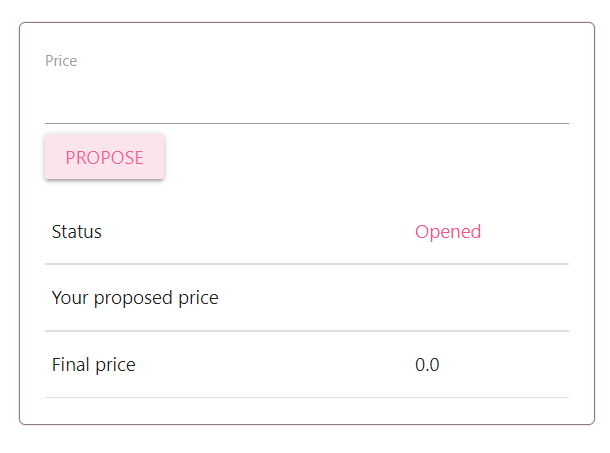


Figure 6 mockup screen for propose

2.7 Set final price

* Function name: afterClosingSession
* Function description: set the final price for the product and update deviation and number of joined session of participants joining the session.
* User to use: admin
* Function Inputs:

|  |  |
| --- | --- |
| **Input** | **Requirement** |
| finalPrice | Uint256, required. |

* Function rule: Only admin can set the final price and can only be set when the session is in “OPENED” state.
* Function result: set the final price for the product and update joined participants deviation and number of joined session. Finally, change the session’s state to “CLOSED”

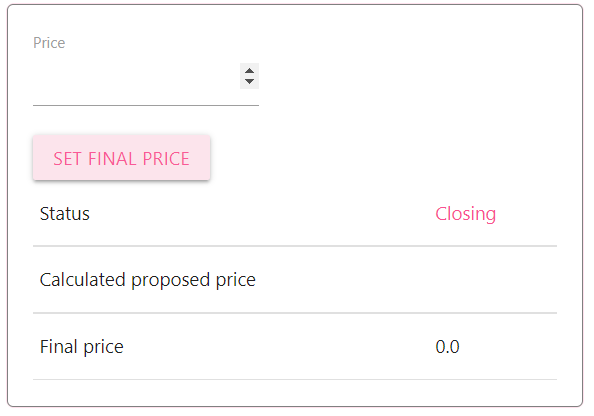


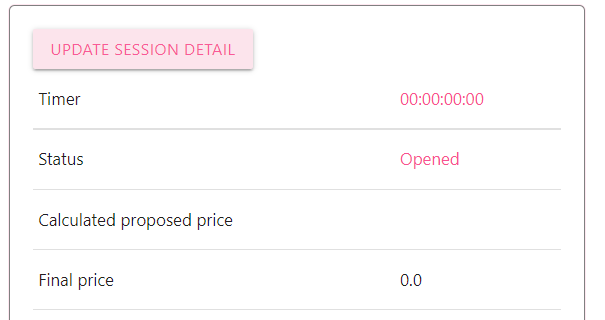
Figure 8 mockup screen for set final price

2.8 Update session detail

* Function name: updateSessionDetail
* Function description: change the session detail (name, description, images)
* Function Inputs:

|  |  |
| --- | --- |
| **Input** | **Requirement** |
| productName | String, required. |
| productDescription | String, required. |
| productImages | List of String, required. |

* Function rule: Only admin can change the session detail and can only be done in OPENED state.
* Function result: update the session detail



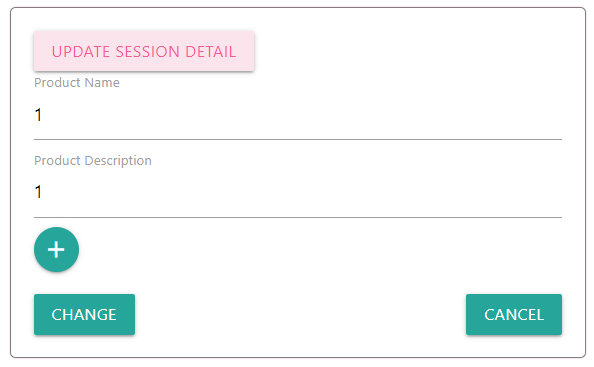


Figure 9 mockup screen for the update session detail

2.9 Get all user information

* Function name: getParticipants
* Function description: get all users information
* Function Inputs: none
* Function rule: Only admin can get the information
* Function result: get all users and their information

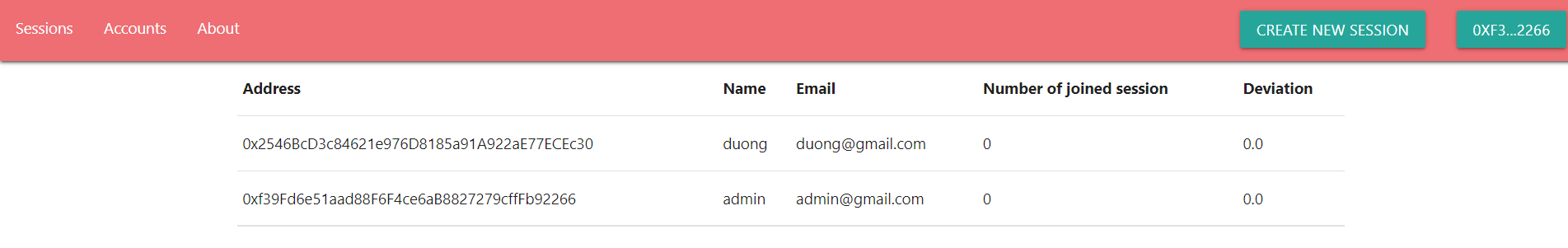


Figure 10 mockup screen for get users information

**3. Non-Functional Requirements**

- Web browser coverage: chrome, brave, firefox, Microsoft edge, opera, browsers that support metamask extension

**4. Product Upgrades**

* Make the website more responsive
* Mobile compatibility
* Score system for active participants
* Make the Dapp more decentralized (e.g: governance-based DAO)