12-Ideas

Overview

- Intro
- Think
  - 100 Words
  - 10 Sentences
  - Select & Rate
- Sketch
  - Idea
  - Requirments
  - Stories
  - Diagrams
- Paint
  - Write
  - Build
  - Call
- Outro
  - How to generate Ideas(Think)
  - How to structure an Idea (Sketch)
  - How to make the first Step (Paint)

🧠 Think

Words

- rich pictures

Monocole, Chair, Technican, Microphon, Smile
Mandala, Ehrling, Drum&Bass, Hitch, Pub,
Reading Night, Nighshift, Romania, Leuven, Aachen,
Thailand, Train, Taxi, Energy, Chai

Sentences

Theme	Words	Sentence	Rating
Friends	Pub	Beer Master right now	5
Orchester	Technican	My Performance, my execlence	4
Dating	Romania	One Beer, One Date	3
Standards	Energy	Standardization on its best	2

Rate & Select

The winner is `talentOfProof.xyz`

📝 Sketch

Idea

Problem: Sub talents are hard to credit
Idea: A mechanism to valide your subtalents
Solution: `talentOfProof.xyz`

Requirments

Functional Requirments

- The WebApp must do add Certification
- The WebApp must display great Achiever
- The WebApp must reward best programs
- The SmartContract must handle it

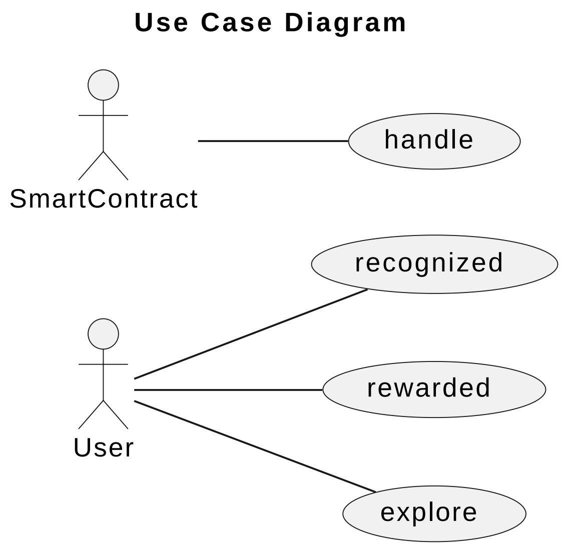
Non Functional Requirments

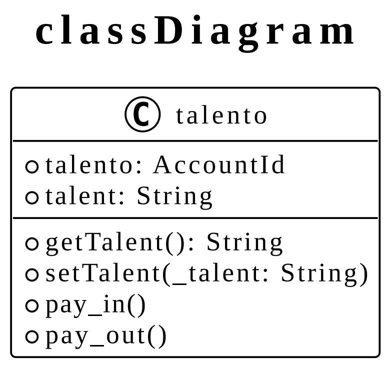
- The Application must be legal

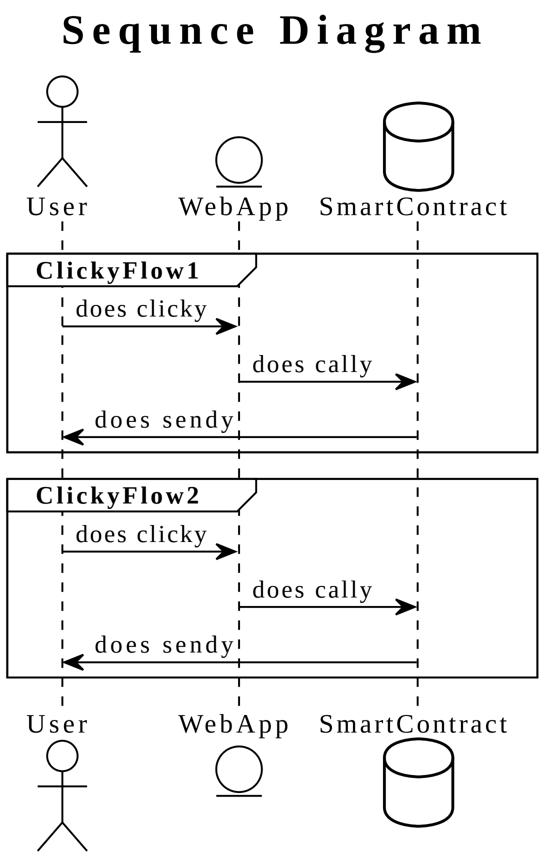
Stories

- As a User I want to get recognize for my subtalent
- As a User I want explore Inspiration
- As a User I want to get rewarded
- As a User I want to create my specialty program
- As a SmartContract I want to handle it

Diagrams


Use Case Diagram
SmartContract
handle
User
recognized
rewarded
explore


classDiagram
talento
talento: AccountId
talent: String
getTalent(): String
setTalent(_talent: String)
pay_in()
pay_out()


Sequnce Diagram
User
WebApp
SmartContract
ClickyFlow1
does clicky
does cally
does sendy
ClickyFlow2
does clicky
does cally
does sendy

🎨 Paint

```
#![cfg_attr(not(feature = "std"), no_std, no_main)]

#[ink::contract]
mod talento {
    use ink::prelude::string::String;

    #[ink(storage)]
    pub struct Talento {
        value: bool,
        talento: AccountId,
        talent: String,
    }

    impl Talento {
        #[ink(constructor)]
        pub fn new() -> Self {
            Self {
                value: true,
                talento: AccountId::from([0xff; 32]),
                talent: String::from("Turn thumb 180 degrees"),
            }
        }

        #[ink(message)]
        pub fn get_talent(&self) -> String {
            self.talent.clone()
        }

        #[ink(message)]
        pub fn set_talent(&mut self, _talent: String) {
            self.talent = _talent;
        }

        #[ink(message)]
        pub fn pay_in(&self) {
            //for the junior dev
        }

        #[ink(message)]
        pub fn pay_out(&self) {
            //for the junior dev
        }

        #[ink(message)]
        pub fn flip(&mut self) {
            self.value = !self.value;
        }

        #[ink(message)]
        pub fn get(&self) -> bool {
            self.value
        }
    }
}
```