08-Ideas

- Youtube (https://youtu.be/teqBRphH38s)
- Github - PDF (https://github.com/FrankBevr/100-Ideas/blob/main/practice/08/08-Ideas.pdf)
- Github - Code (https://github.com/FrankBevr/100-Ideas/blob/main/practice/08/daffodils/lib.rs)

Overview

🧠 Think

100 Words

Theme	Words			
Februrary	grey	cold	karneval	daffodils
Home	Bred	Dog	Sister	Couch
Series	Sherlock	One Piece	Scrubs	Lie
Street	yew gum			
Drinking	meters			

10 Sentence

Word	Name	Phrase	Rating
daffodils	Flowerbombing	Spread your seed	7
Sherlock	Analyso	Fraud Decetion	5
Dog	Disabilties	Your Assitent on your feet	4

Rate & Select

`Daffodils` - Flowerbombing - Spread your seed

📝 Sketch

Idea

Problem: Too much grey in City
Idea: Flowerbombing - Spread your seed
Solution: `daffodils.xyz`

Requirements

Functional Requirements:

- The App must do seed the city
- The User must able to select seeding area
- The User must be able to propose seed
- The Smart Contract must handle the funding part

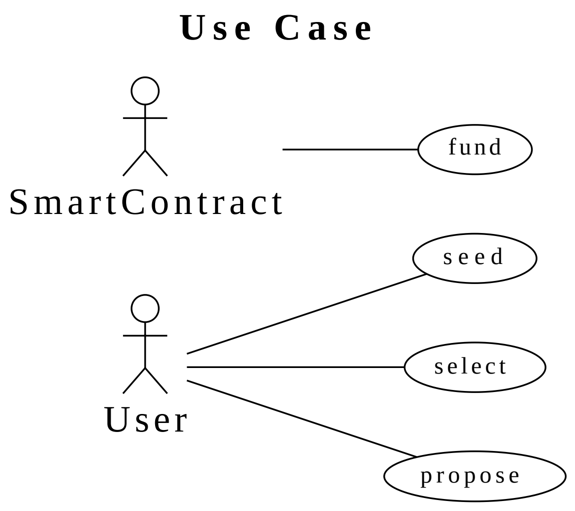
NonFunctional Requirements:

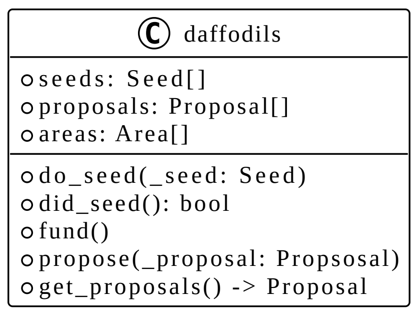
- The App should be in align with the England Law
- The App should be easy to use

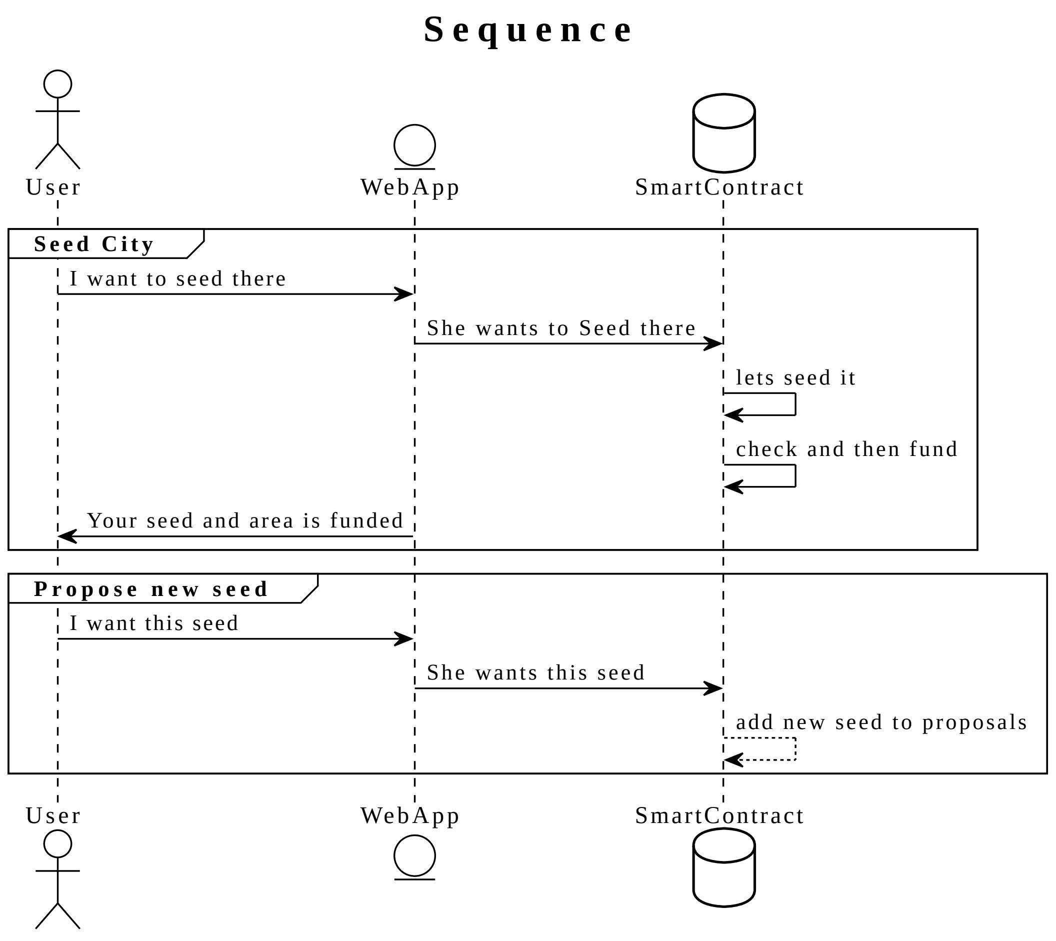
User Stories

- As a User I want to seed my city
- As a User I want to select where to seed
- As a User I want to propose seed
- As a Smart Contract I want to fund it nicely

Diagrams


Use Case
SmartContract
fund
User
seed
select
propose


daffodils
seeds: Seed[]
proposals: Proposal[]
areas: Area[]
do_seed(_seed: Seed)
did_seed(): bool
fund()
propose(_proposal: Propsosal)
get_proposals() -> Proposal


Sequence
User
WebApp
SmartContract
Seed City
I want to seed there
She wants to Seed there
lets seed it
check and then fund
Your seed and area is funded
Propose new seed
I want this seed
She wants this seed
add new seed to proposals
User
WebApp
SmartContract

🎨 Paint

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

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

    #[ink(storage)]
    pub struct Daffodils {
        seed: String,
        area: String,
        proposed_seed: String,
        value: bool,
    }

    impl Daffodils {
        #[ink(constructor)]
        pub fn new() -> Self {
            Self {
                seed: String::from("Daffodils"),
                area: String::from("DistrictA, "),
                proposed_seed: String::from(""),
                value: true,
            }
        }

        #[ink(message)]
        pub fn add_seed(&mut self, _seed: String) {
            self.seed.push_str(&_seed);
        }

        #[ink(message)]
        pub fn add_area(&mut self) {
            self.area.push_str("DistrictX, ");
        }

        #[ink(message)]
        pub fn propose_seed(&mut self) {
            self.proposed_seed = String::from("Carlos");
        }

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

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

        #[ink(message)]
        pub fn get_area(&mut self) -> String {
            self.area.clone()
        }

        #[ink(message)]
        pub fn fund(&self) {}

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

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