

10-Ideas

- Youtube (https://drive.google.com/file/d/1hgev9DiSWYy0rvLfXvK_AEgjQbJMf6Cf/view?usp=sharing).
- Github - PDF (<https://github.com/FrankBevr/100-Ideas/blob/main/practice/11/11-Ideas.pdf>).
- Github - Code (<https://github.com/FrankBevr/100-Ideas/blob/main/practice/11/styleo/lib.rs>).

Overview

- Intro
- Think
 - ☒ 100 Words
 - ☒ 10 Sentences
 - ☒ Rate & Select
- Sketch
 - ☒ Idea
 - ☒ Requirements
 - ☒ Stories
 - ☒ Diagrams
- Paint
 - ☒ Write
 - ☒ Build
 - ☒ Call
- Outro

100 Words

- Note: Create Rich Pictures. That's it.

Words				
Earrings	Chair	Technician	Suits	Merch
Mandala	Fight Club	After Life	Board	Monopoly
Keyboard	Coffee	Thailand	Mess	Outlook
Blender	Ask	Games	Hydra	Acala

10 Sentences

Theme	Word	Phrase	Rating
Delivery	Hydra	Sharing is caring	7
Perception	Suits	Your Look on chain	5
Organisation	Mess	Your room in order	4
Games	Monopoly	Earn by play	3

Rate & Select

The winner is `percepto.xyz`



Sketch

Idea

Common is Problem Solution

Problem: Monetize your fashion style

Solution: Your Style your Brand

Solution: `percepto.xyz`

Requirements

Functional Requirements

- The app must collect your style choices
- The app must be able to sell things

Non Functional Requirements

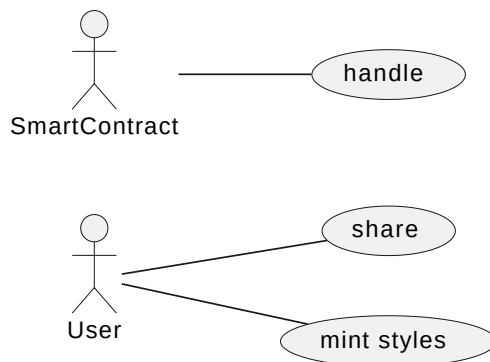
- The app should align with Thai law
- The User must have a happy feeling

Stories

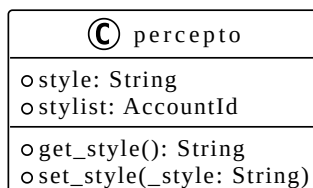
- As a User, I want to share my style to the world
- As a User, I want to mint unique styles
- As a SmartContract, I want to handle it

Diagrams

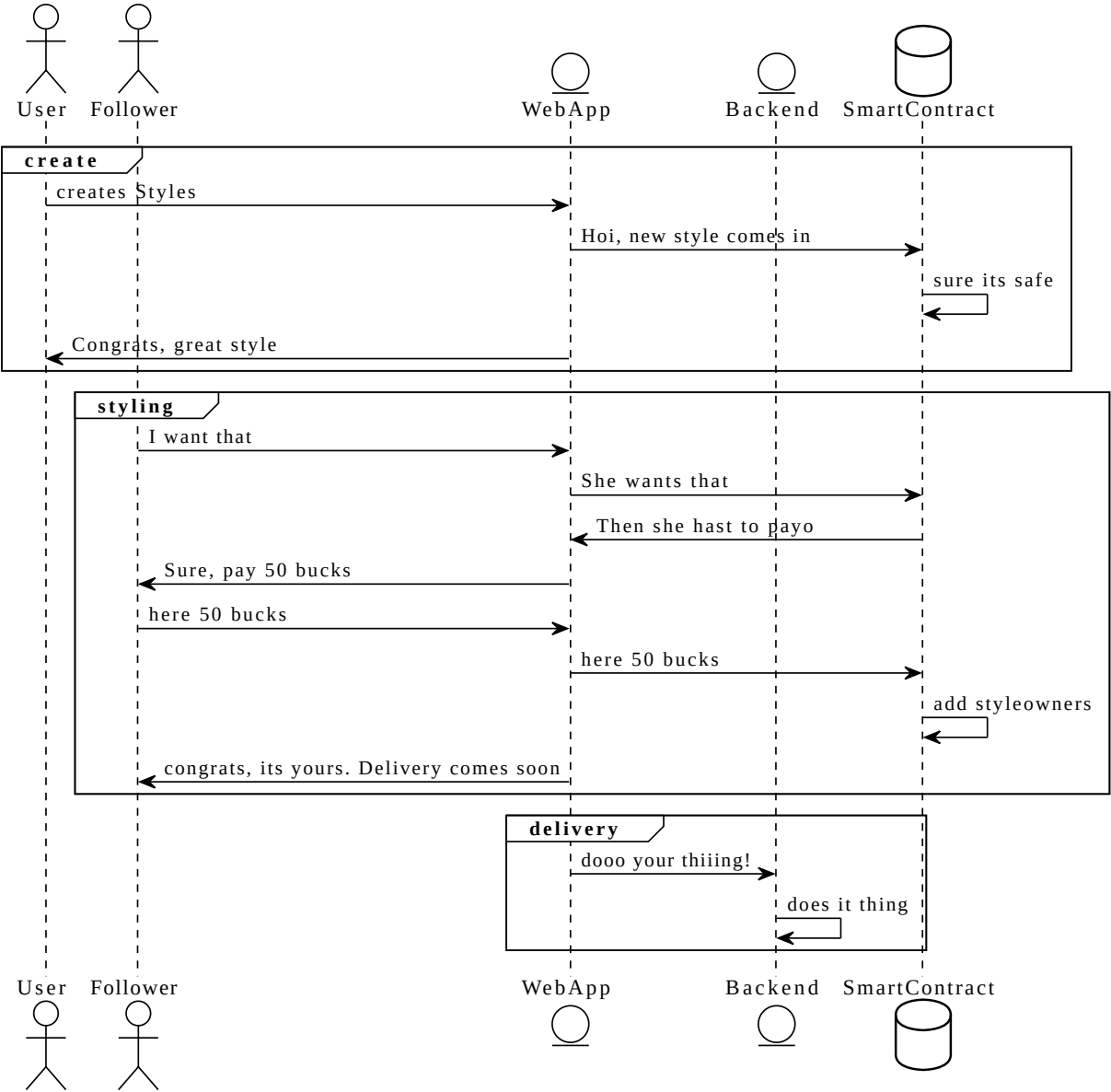
Use Case Diagram



ClassDiagram



SequenceDiagram



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

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

    #[ink(storage)]
    pub struct Stylo {
        value: bool,
        style: String,
        stylist: AccountId,
    }

    impl Stylo {
        #[ink(constructor)]
        pub fn new() -> Self {
            Self {
                value: true,
                style: String::from("{style:Garbani Hat 400, color: blue, accesoir:
                stylist: AccountId::from([0xff; 32]),
            }
        }

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

        #[ink(message)]
        pub fn set_style(&mut self, _another_style: String) {
            self.style = String::from(_another_style);
        }

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

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