MIN Token Project

Contents

Project Overview 2

1. Functional Requirements 2
   - 1.1 Roles 2
   - 1.2 Features 2
   - 1.3 Use Cases 3
2. Technical Requirements 3

Project Overview

On software level, MIN Token project is an ERC20 based vesting project. Given a tokenomics structure with Token Generation Events (TGE), cliff and vesting periods MIN Vesting contract will allow the vested release of MIN Token to defined addresses. These addresses will belong to project owners and will have different purposes. Two other contracts, MINPrivateSwap and MINStrategicSale will handle some amount of token sale defined in the tokenomics. Customers will be able to buy vested tokens from MINPrivateSwap at a set rate. When cliff ends and vesting period begins, customers should be able to release their tokens as they wish. MINStrategicSale will handle sales that project owners will make via legal agreements. These agreements will require the project owners to add the buyer's address and bought amount as beneficiary to the smart contract.

1. Functional Requirements

1.1 Roles

MIN Token project doesn't have any other privileged roles than **owner** which is who deployed Ownable contracts (all three vesting contracts).
Owner for MINVesting will be able to setup contract's vesting schedules with project owners' addresses. Owner for MINPrivateSwap will be able to withdraw swapped token incomes and any excess MIN Tokens from the contract. Owner for MINStrategicSale will be able to define buyers manually and withdraw any excess MIN Tokens from the contract.

1.2 Features

These contracts will allow following interactions;

- Set vesting schedules on MINVesting (Owners)
- Release vested amounts of MIN Token to wallets (Owners and buyers)
- Buy MIN Token with a set swappable token and rate (Buyers)
- Give up buying MIN Token through MINPrivateSwap and claim sent swappable token before sale ends (Buyers)
- Claim any unsold MIN Tokens and sale revenue in swappable token after sale ends (Owners)
- Calculate a beneficiary's releasable amount of vested MIN Tokens (Everyone)

1.3 Use Cases

1. Owners deploy the MIN Token and MINVesting. They setup vesting schedules on MINVesting according to their tokenomics.

2. Owners deploy the MINPrivateSwap with the swappable token of their choice and rate. They set the sale duration and vesting plan of this contract at deployment. Buyer's will be able to buy and give up buying a limited amount of MIN Token during the sale at set rate. Once sale ends owners will be able to claim any unsold MIN Tokens and their revenues in swappable token.

3. Owners will sell MIN Token through legal agreements with buyers of their choice and get buyers' choice of wallet address. They will then set the vesting schedule for a buyer's address and amount. Buyer's will be able to release their tokens according to the tokenomics.

2. Technical Requirements

This project has been developed with Solidity language, using Hardhat as a development environment. Typescript is the selected language for testing and scripting. In addition, OpenZeppelin's libraries are used in the project. All information about the contracts library and how to install it can be found in their GitHub.

In the project folder, the following structure is found:

```
├──contracts/
│   ├──test/
│   │   └──MockToken.sol
│   ├──token/
│   │   └──MINToken.sol
│   ├──utils/
│   │   ├──MINStructs.sol
│   │   └──MINVestingBase.sol
│   └──vesting/
│       ├──MINPrivateSwap.sol
│       ├──MINStrategicSale.sol
│       └──MINVesting.sol
├──docs/
```

```
│    └──.gitkeep
├──ignition/
│    └──modules/
│         └──MINToken.ts
├──test/
│    ├──MINPrivateSwap.test.ts
│    ├──MINStrategicSale.test.ts
│    ├──MINToken.test.ts
│    └──MINVesting.test.ts
├──tokenomics/
│    └──tokenomics.ts
├──.gitignore
├──.prettierignore
├──.prettierrc
├──.solhint.json
├──hardhat.config.ts
├──package-lock.json
├──package.json
├──README.md
└──tsconfig.json
```

Start with README.md to find all basic information about project structure and scripts that are required to test and deploy the contracts.

Inside the contracts folder, token/MINToken.sol is the ERC20 contract of this project. There is also test/MockToken.sol ERC20 contract that derives from token/MINToken.sol. MockToken.sol contract is used to mimic an external transfer fail in tests and not intended for deployment.

In contracts/utils folder there are 2 .sol files, which are shared between other contracts in the project. utils/MINStructs.sol contains the library MINStructs which holds VestingSchedule contract used to represent a vesting schedule in other contracts. utils/MINVestingBase.sol contains an abstract contract which implements base features of vesting used in other contracts as well.

In contracts/vesting folder, there are MINPrivateSwap.sol, MINStrategicSale.sol and MINVesting.sol, which are the main contracts of this project.

In the test folder, there are 4 test.ts files which’ll test every deployable contract in this project. In order to run them, you can compile the contracts with “npx hardhat compile”, and then run “npx hardhat coverage”.

The tokenomics folder contains only one file which holds the tokenomics information. It’s being used in tests and will be used at deployment as well.

Deployable contracts can be deployed using the scripts/deploy.ts file. You’ll need to set a swappable token address beforehand to be able to successfully perform a deployment.

The project configuration is found in hardhat.config.ts, where dependencies are indicated. More information about this file’s configuration can be found in the Hardhat Documentation.

Finally, this document, graphs about the project architecture and NatSpec generated documentation can be found under docs folder.