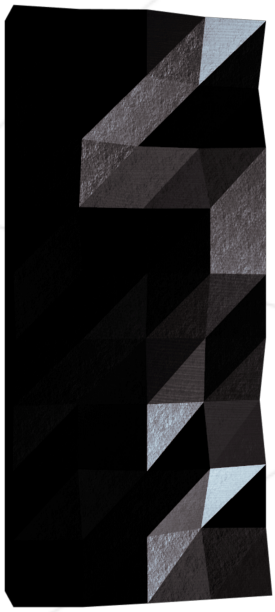


TokenZswap

Build your own AMM

Summary: This document is a Web3 and DeFi related exercise.

Version: 1

Contents

I Preamble **2**

II Introduction **3**

III Objectives **4**

IV Mandatory part **5**

IV.1 AMM Smart Contract 5

IV.2 Token creation and pool setup 5

IV.3 Frontend interaction 6

IV.4 README and Documentation 6

V Bonus part **7**

VI Submission and peer-evaluation **8**

Chapter I

Preamble

This subject is the production of a partnership between 42 and BNB Chain.

BNB Chain is a distributed blockchain network that enables developers to create decentralized applications (DApps) as part of the Web3 ecosystem.

In this project, you will build an Automated Market Maker (AMM), a core component of decentralized finance (DeFi). Your AMM will allow users to swap tokens, provide/remove liquidity, and optionally buy or sell NFTs using smart contracts on the BNB Chain testnet.

You can retrieve Tbnb for free and with no minimum coins on your wallet via this faucet: BNB Chain Faucet

Figure I.1: https://www.bnbchain.org/

Chapter II

Introduction

Welcome to the decentralized future of finance!

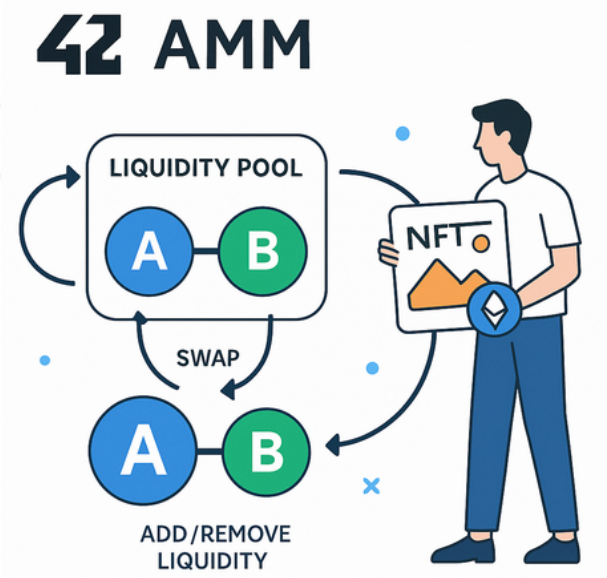
In this project, you'll learn how to create a smart contract that facilitates token swaps without a traditional order book.

AMMs use liquidity pools and pricing algorithms (e.g., constant product formulas) to execute trades.

Your task is to implement a basic AMM system similar to Uniswap or PancakeSwap, deployed on a testnet.

You must allow users to trade NFTs (buy/sell using tokens from the AMM) through the same interface.

Let's explore how DeFi is redefining finance.



Chapter III

Objectives

By working on this project, you will:

- Learn to write secure smart contracts for decentralized trading
- Understand liquidity pools and how swaps are executed
- Interact with token standards such as ERC-20 and ERC-721
- Create a basic frontend to interact with your contracts

This project is not just about coding; it's about understanding how decentralized systems operate and designing a reliable DeFi application.

Chapter IV

Mandatory part

IV.1 AMM Smart Contract

You must write a smart contract that:

- Creates and manages a liquidity pool (Token A / Token B)
- Allows adding/removing liquidity and issuing LP tokens
- Executes token swaps with a constant product pricing formula
- Applies a small transaction fee to swaps
- Supports buying and selling NFTs using tokens from the AMM

Use your ERC-20 compliant test token of your own creation. que tu as creer dans le projet tokenizer.

Make sure all state variables and functions are well-commented. Naming should be clear and explicit.

IV.2 Token creation and pool setup

Deploy at least two ERC-20 tokens for testing your AMM. You'll use these tokens to create your first liquidity pool.

You may use Remix, Hardhat, Truffle, or other tooling, but the project must be deployable with minimal setup.

IV.3 Frontend interaction

Build a minimal interface (no need for styling) that allows:

- Swapping tokens
- Adding/removing liquidity
- Viewing pool balances

`The UI should work with a wallet like MetaMask.`

IV.4 README and Documentation

You must write a clear `README.md` explaining:

- How to deploy and test your AMM
- How your smart contract works
- Screenshots of the DApp in action

You must also provide a `documentation` folder with:

- Diagrams of the system
- Contract explanations
- Known limitations

`Documentation will be evaluated. Write it so that someone else can reuse your AMM. Here is an example of real documentation:` https://docs.pancakeswap.finance/

Chapter V

Bonus part

Here are some ideas to enrich your project:

- On-chain price chart or history log
- Flash loan support (for very advanced users)
- Dynamic fee based on volatility or usage
- Advanced UI with charts, notifications, and wallet connect

`Bonus points are only considered if the mandatory part is PERFECT.`

Chapter VI

Submission and peer-evaluation

Submit your assignment in your `Git` repository with the following structure:

```
$> ls -al
README.md
code/
frontend/
deployment/
documentation/
```

Only the contents of your repository will be evaluated during the defense. Make sure everything is there and clearly organized.