UCT ECO5073S- Fintech & Cryptocurrencies Tutorial Practicals

Francisco Rutayebesibwa

Deadline: Monday 9th September 2024

Overview

The aim of these practicals is to prepare you for the exam and class project. You are allowed to collaborate with fellow classmates, however everyone should submit their **own** solutions in PDF format (see submission notes for more information). Attempt both practicals, and for each practical, make sure you have met the **minimum requirement**. The **challenge** task for each practical is optional (however highly recommended to attempt as it will help you prepare for the class project).

Prac 1: Blockchain enabled Voting system

The university is about to hold elections for the new student body council.The university wants a system in place that voters can use to submit their votes and view results in real time.

Minimum requirement: Your task is to build a blockchain based voting system using Algorand smart contracts. Make sure the smart contract is deployed to the Algorand Test Network. Here are several guidelines to follow:

- Assume there are only five candidates
- You can use PyTeal to build the smart contract.
    - Alternatively, you are allowed to use other libraries or programming languages to build your smart contracts. Here are examples of libraries you can use:
        - Beaker (Documentation & YouTube series)
        - Tealish (Documentation & YouTube series)
- The smart contract should enable users to 1) View Results in real-time and 2) Vote for one of the five candidates.
- The smart contract can only handle a maximum of 20 votes.
- Make sure that an Algorand address can only be used once to vote.

Challenge: Create a web based blockchain voting application. You can use tools covered in the tutorials, or any other tools you prefer.

Helpful Resources:

- Master Pyteal YouTube Series

- PyTeal Documentation
- Dappflow

Prac 2: Creating Accounts, transfer funds, and creating NFTs

Minimum requirement: Create a Command-Line Interface (CLI) programme using Python that does the following:

- Generates Algorand account (i.e. public & private keys, and mnemonic phrase) for the user if the user.
- Transfer ALGOs from one account to another,
    - Make sure you print the Transaction ID so that the user can verify the transaction on the Algorand Test Net.
- Creates an NFT for the user.
    - The user should enter the following arguments: sender address, unit name, asset name, and a URL where more information about the asset can be retrieved.
    - Set the: manager, reserve, freeze, and callback parameters to the sender's address
    - The output should contain the Transaction ID.

Make sure the accounts are generated on the Algorand Test Network, and all transactions can be viewed on Algorand's Test Network.

Challenge: Instead of a python CLI, create a web based application. You can use tools covered in the tutorials, or any other tools you prefer.

Helpful Resources:

- Fintech Tutorials Github Repository
- Algorand Standard Assets (ASA)
- Build Command Line Interface with Python's argparse

Submission notes

- Make use of comments and functions in your source code.
- For each Practical:
    - Include a README.md which describes both applications, how it works, how to run it and sequential screenshots of the output.
    - Push your code up to a public GitHub repository and include the Github link in your PDF submission. In addition, in your PDF submission, include screenshots of the output of your code as well.

- For Prac 1, screenshots should include: your deployed smart contract on Dappflow with the methods, example of successful transaction calls made to the smart contract.
- For Prac 2, screenshots of the output of the different functionalities of your CLI.