FinTech Bootcamp Capstone Project

Blockchain Real Estate Marketplace

Executive summary:
Three entrepreneurs with experience in government contracting, finance, and Saas want to build a decentralized marketplace for tokenizing real estate.

Real Estate has made more millionaires than any other industry. However, it is not accessible to everyone. This is why we built a decentralized application to tokenize and invest in real estate.

Decentralization provides more investment opportunities to more people at a lower cost. As blockchain is rapidly spreading across industries, we believe decentralizing real estate is both valuable for long term decentralization.

Being one of the biggest industries in the world, we are confident that creating a DApp for real estate would be a great investment long term.

Concept:
We are creating a decentralized marketplace application for users to invest in and tokenize real estate property listings.

Software Version Control

GitHub
Solidity
Python

Approach:
This application will be built using Python to run the front end and Solidity will be used to deploy smart contracts to run the backend.

Development

Contracts
A Solidity contract is written for the underlying smart contract functionality.

Front-end Development

A decentralized

Timeline:
Monday 1/31 - Wednesday 2/2: Code Solidity contracts
Saturday 2/5: Compile & Test contracts

Monday 2/7: Practice Presentation
Wednesday 2/9: Present & Demo App

Deliverables:

1. Smart Contracts to Deploy
2. Python Script to run application
3. Marketplace End Product

Links:
EVERYONE READ THIS - https://moralis.io/how-to-create-an-nft-marketplace/ - they used JS to build it
https://docs.streamlit.io/
https://docs.openzeppelin.com/
https://medium.com/chainlink-community/building-my-first-chainlink-adapter-996ab51dfe9
https://github.com/thodges-gh/CL-EA-Python-Template

Capstone Project Guidelines

This week, you will begin work on your final project of the boot camp! You can choose any topic from any section of the course for this capstone project.

Project examples include:

- **Tokenize an asset using the proper ERC standards**
- **Create a decentralized auction in Solidity**
- **Build a digital marketplace powered by smart contracts**
- **Create an automated algorithmic trading system**
- **Design an in-depth dashboard for investors to quickly visualize their portfolios**
- **Use machine learning and NLP to predict trends in stock prices**

Technical Requirements

Note that your assignment will be graded according to the following rubric:

- **Proficiency (≥ 90% of the points)**
- **Approaching proficiency (≥ 80% of the points)**
- **Progressing (≥ 70% of the points)**

- **Emerging (< 70% of the points)**

The following subsections list the technical requirements for the final Capstone project.

Software Version Control (10 points)

- **Repository created on GitHub. (2 points)**
- **Files frequently committed to repository. (3 points)**
- **Commit messages with appropriate level of detail included. (2 points)**
- **Repository organized, and relevant information and project files included. (3 points)**

Data Collection and Preparation (10 points)

- **Data collected, cleaned, and prepared for the application or analysis. (10 points)**

Development (40 points)

- **Jupyter notebook, Google Colab notebook, Amazon SageMaker Studio notebook, or Streamlit application created. (10 points)**
- **One or more Python modules, machine learning models, or Solidity smart contracts created. (10 points)**
- **Calculations, metrics, visualizations, or video needed to demonstrate the application included. (10 points)**
- **One new technology or library used that the class hasn't covered. (10 points)**

Documentation (15 points)

- **Code is well commented with concise, relevant notes. (5 points)**
- **GitHub README.md file includes a concise project overview. (2 points)**
- **GitHub README.md file includes detailed usage and installation instructions. (3 points)**

- **GitHub README.md file includes either examples of the application or the results and summary of the analysis. (5 points)**

Presentation (25 points)

Each project group will prepare a formal 10-minute presentation that includes the following:

- **An executive summary of the project and project goals. (5 points)**
    - **Explain how this project relates to fintech.**
- **The approach that your group took to achieve the project goals. (10 points)**
    - **Include any relevant code or demonstrations of the analysis or application.**
    - **Describe the techniques that you used to test or evaluate the code.**
    - **Discuss any unanticipated insights or problems that arose and how you resolved them.**
- **The results and conclusions from the analysis or application. (5 points)**
    - **Include relevant images or examples to support your work.**
    - **If the project goal wasn't achieved, share the issues and what the group tried for resolving them.**
- **Next steps. (5 points)**
    - **Take a moment to discuss the potential next steps for the project.**
    - **Discuss any additional questions that you'd explore if you had more time. Specifically, if you had additional weeks to work on your project, what would you research next?**

House Account Wallet

+ Revenue

- Property tax

- Water bill

- Electric bill

- Insurance bill

- overhead bill

- X% for improvement fund

Income

Distributed by % ownership

Income distributed to individual owners' wallets

Links from RJ
https://uniswap-python.com/getting-started.html
https://solidity-by-example.org/app/dutch-auction/
https://github.com/FrankielsLost/smart-batched-auction
https://medium.com/coinmonks/dissecting-the-fractional-protocol-dc3867584bdb

Set up to trade – bid swap

```
pragma solidity ^0.5.0;

import
"https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contra
cts/token/ERC721/ERC721Full.sol";
import
"https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contra
cts/ownership/Ownable.sol";
import './EstateAuction.sol';

contract Estate is ERC721, Ownable {
    constructor() ERC721("Estate", "ACS") public {}
```

```
    // cast a payable address for the Martian Development Foundation to be the
beneficiary in the auction
    // this contract is designed to have the owner of this contract (foundation)
to pay for most of the function calls
    // (all but bid and withdraw)


    address payable foundation_address = address(uint160(owner()));

    mapping(uint => EstateAuction) public auctions;



    function registerproperty(string memory TokenURI) public payable onlyOwner {
        uint _id = totalSupply();
        _mint(msg.sender, _id);
        _setTokenURI(_id, TokenURI);
        createAuction(_id);
    }

    function createAuction(uint token_id) public onlyOwner {
       auctions[token_id] = new EstateAuction(foundation_address);
    }


    function endAuction(uint token_id) public onlyOwner {
        require(_exists(token_id), "Blue Print not available");
        EstateAuction auction = auctions[token_id];
        auction.auctionEnd();
        safeTransferFrom(owner(), auction.highestBidder(), token_id);
    }

    function getAuction(uint token_id) public view returns(EstateAuction auction)
{
       EstateAuction auction = auctions[token_id];
        return auction.getAuction();

    }

    function auctionEnded(uint token_id) public view returns(bool) {
         EstateAuction auction = auctions[token_id];
          return auction.ended();
    }
```

```
    function highestBid(uint token_id) public view returns(uint) {
       EstateAuction auction = auctions[token_id];
        return auction.highestBid();
    }

    function pendingReturn(uint token_id, address sender) public view
returns(uint) {
      EstateAuction auction = auctions[token_id];
        return auction.pendingReturn(sender);
    }

    function bid(uint token_id) public payable {
        EstateAuction auction = auctions[token_id];
        auction.bid.value(msg.value)(msg.sender);
    }

}
```