AI Generated NFT Video Game Console Collection / Fullstack NFT Marketplace dApp

Christopher De Leon
Mike West
Daniyar Mussin

Objectives & Introduction

- Determine the feasibility of developing a local NFT marketplace based on knowledge acquired in class.
- Identify methods to improve functionality and visual appeal beyond Streamlit for the proposed NFT marketplace.
- Investigate the possibility of integrating Chat GPT's generative image AI for the creation of an art collection within the marketplace.

Key Research Questions

Our research (feasibility) questions are as follows:

- Can we build a functional & dynamic NFT marketplace from what we learned in class?
- Can we utilize a platform that enhances functionality and visual aesthetic in a greater capacity than Streamlit?
- Can we implement DALL-E 3 generative image API into creating an art collection for us?

Breakdown of tasks and roles

- System Design & Architecture: To be led by Christopher, outlining the main components of the NFT marketplace system and how they will interact.

- Art Development: Daniyar will design the NFT's using ChatGPT's DALL-E 3 API function to create collections of art using prompt engineering.

- Interface Design: Mike will take the helm in designing a sleek, intuitive, and interactive interface, ensuring the system is user-friendly and efficient.

- Integration & Testing: The team will collaborate in this final phase to integrate all system components to ensure full functionality.

Resources Used

- https://docs.alchemy.com/docs/nft-project-code-templates
  - https://docs.alchemy.com/docs/how-to-build-an-nft-marketplace-from-scratch
  - Project template, code snippets and javascript layout template for decentralized app

- https://docs.openzeppelin.com/
  - https://docs.openzeppelin.com/contracts/4.x/erc721 (Constructing an ERC721 Token Contract/URI Storage)
  - https://docs.openzeppelin.com/contracts/4.x/api/utils (Counters: a simple way to get a counter that can only be incremented, decremented or reset. Very useful for ID generation, counting contract activity, among others.)

Libraries & Tools Used

- Remix IDE (to launch smart contract)
- React (Javascript from project template)
- Hardhat (configuration to connect to Ganache local blockchain network)
- Alchemy (project template)
- OpenAI library (DALL-E 3 API)
- Piñata (to upload data to IPFS)
- MetaMask (software cryptocurrency wallet used to interact with the Ethereum blockchain)
- Ganache (local blockchain network)

Libraries & Tools Used Cont’d (Frameworks)

- React for frontend development.
- Hardhat for Ethereum smart contract development.
- React because it lets us easily reuse different parts of our website, like building blocks. Plus, it makes everything run smoother and faster for users.
- Hardhat provides a comprehensive development environment with Ethereum integration and extensibility.
- Advantages:
- React: Modular, efficient, scalability and supported by a strong ecosystem.
- Hardhat: Streamlined development lifecycle, real-world Ethereum integration, and extensible architecture.

Creating Stunning Visuals with the DALL-E-3 API

Point of Using DALL-E 3 API for Generating NFTs

In the realm of Non-Fungible Tokens (NFTs), the demand for unique and compelling visual content is paramount.

In our project, we explore how AI, specifically the DALL-E 3 model, can revolutionize the process of generating NFT artwork.

- Diverse Content Creation: DALL-E 3's ability to generate images from textual prompts enables the creation of a wide range of visual content, fostering artistic exploration and innovation.
- Enhanced Creativity: AI-generated imagery pushes the boundaries of creativity, offering novel and unconventional artistic possibilities for NFTs.
- Efficiency and Scalability: By harnessing AI technology, the code facilitates a more efficient and scalable approach to NFT art generation, potentially reducing time and resource investments. Instead of manually creating each piece, artists can use the API to quickly generate a large number of unique images, which can then be tokenized as NFTs.

Future Directions and Opportunities for Growth

DALL·E 3

Let’s talk about the potential and variety of growth for this program in terms of NFT Platform creation

- Real-World Applications in NFT Markets: our solution can be applied in real-world scenarios, particularly in the context of NFT markets. It has the potential to streamline the process of creating NFT artwork by automating image generation and facilitating integration with blockchain technologies and NFT platforms.

- Opportunities for Development: it is noteworthy to highlight the enhancing prompt customization, integrating advanced image manipulation features, and optimizing performance for scalability.

Key Smart Contract Global Variables

- **_tokenIds:** This is the latest token ID that corresponds to an NFT minted with this smart contract. tokenIDs map to tokenURI which is the URL that contains the metadata of the corresponding NFT
- **_itemsSold:** Is a count of the number of items sold on the marketplace
- **owner:** This is the owner of the smart contract. The only address that can issue a withdrawal request.
- **listPrice:** The price (in ETH) any user needs to pay to list their NFT on the marketplace
- **ListedToken:** A solidity struct (similar to Javascript object) dictating the format an NFT's data is stored in
- **TokenListedSuccess:** Event emitted when a token is successfully listed
- **idToListedToken:** It is the mapping of all existing tokenId's to the corresponding NFT token

Key Smart Contract Functions

- **createToken and createListedToken**
  This function turns a tokenURI (URL with metadata) into an actual NFT on-chain, with details stored in the smart contract. This is useful for the List your NFT page
- **getAllNFTs**
  This function returns all the "active" NFTs (currently on sale) in the marketplace. This is useful for the marketplace home page.
- **getMyNFTs**
  This function returns all the "active" NFTs (currently on sale) in the marketplace, that the current logged in user owns. This is useful for the profile page.
- **executeSale**
  When a user clicks "Buy this NFT" on the profile page, the executeSale function is triggered.
- If the user has paid enough ETH equal to the price of the NFT, the NFT gets transferred to the new address and the proceeds of the sale are sent to the seller.

Backend & Smart Contract Integration

- Upload the image along with metadata to IPFS
- Send the metadata tokenURI and price to the smart contract
- Smart contract records and manages NFT ownership, transfers, and sales in a decentralized manner.
- The dApp is deployed on testnet.opensea.io, allowing users to interact with and test the platform before deploying to the main Ethereum network.

Front End Integration (React Components)

For the platform to work seamlessly, the frontend must integrate functions from the smart contract.

- Has a function that creates a provider, signer, and a contract object
- Fetches relevant data from the smart contract
- Fetches relevant data from IPFS via Axios
- has a return where it returns the JSX/HTML for the page

Demo

Full Stack deployment
Live OpenSea testnet page: https://testnets.opensea.io/collection/ai-nft-collection-5

Future Functionality & Questions

Potential developments & extensions could be:

- Use Alchemy's getNFTs and getNFTsForCollection endpoints to fetch NFTs for the marketplace and profile page
- Add functionality to let users list pre-existing NFTs to the marketplace
- Adding Royalties such that the original NFT creator gets 10% of the proceeds every time that NFT gets sold
- Adding OpenSea API/endpoints for creating a customized interface/a local NFT marketplace for a private art show/community

Links

- https://github.com/Chrisdeleon91/AI-NFT-Art-Collection-Project-3
- https://testnets.opensea.io/collection/ai-nft-collection-5