

Cheesewheel.xyz



Benjamin Boule, Eugene Njinkeu,
Ian Melhorn, Will Pape





Overview

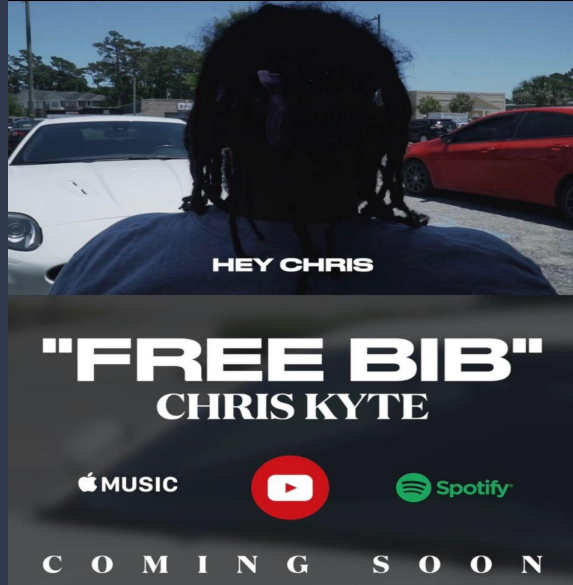
Cheesewheel.xyz is where Web3 meets music, social tokens, and NFTs.

A launchpad for emerging artists to get discovered and build a community using Web3 tools, powered by the CWD governance token (CheesewheelDAO).

An intuitive user interface aggregating and enabling the use cases below:

- **Music:**
 - Stream artist music on Audius.
- **Social tokens:**
 - Launch ERC-20 social tokens for artists.
- **NFTs:**
 - Launch “access” NFTs for artists.
 - Sell artist merchandise in the form of NFTs.
 - Physical delivery of items after NFT purchase.

Onboard First Artist



- **Name:** Chris Kyte
- **Location:** Myrtle Beach, SC
- **Sponsors:**
 - Wise Dreamers (clothing)
 - The Private Shop (apparel, sneakers)
 - AwalWandering (jewelry)
- **Label affiliations:**
 - BackGood / 10k Projects / Dreamville
 - Trippie Redd
 - Internet Money
 - 6IX9INE
 - J.Cole
- **Audius music:**
<https://audius.co/cheesewheelCWR/free-bibby-prod-by-king-kevin>

Project Deliverables

Project Tasks

Click **+ New** to add a new goal to your board.
Click on an existing goal to add notes, research, whatever you want.
Watch as the **Done** column fills up! 🍌

Click **By Status** to change how goals are grouped and displayed.

By Status ▾

The screenshot shows a Notion project board with three columns: 'To Do' (3 items), 'Doing' (3 items), and 'Done' (2 items). Each item is a task card with a title, assignee, and comment count.

To Do (3)	Doing (3)	Done (2)
Code an ERC-721 NFT (Streamlit/Pinata) Eugene	Create a Discord server Ian (1)	Code an ERC-20 token (CWD) Will/Ian
Build full front end Benjamin	Set up a Notion profile Ian (1)	Code an ERC-20 social token (KYTE) Will/Ian
Cheesewheel.xyz domain link Benjamin/Ian (1)	ENS/Snapshot/Mirror Ian (1)	

At the bottom of each column is a '+ New' button.

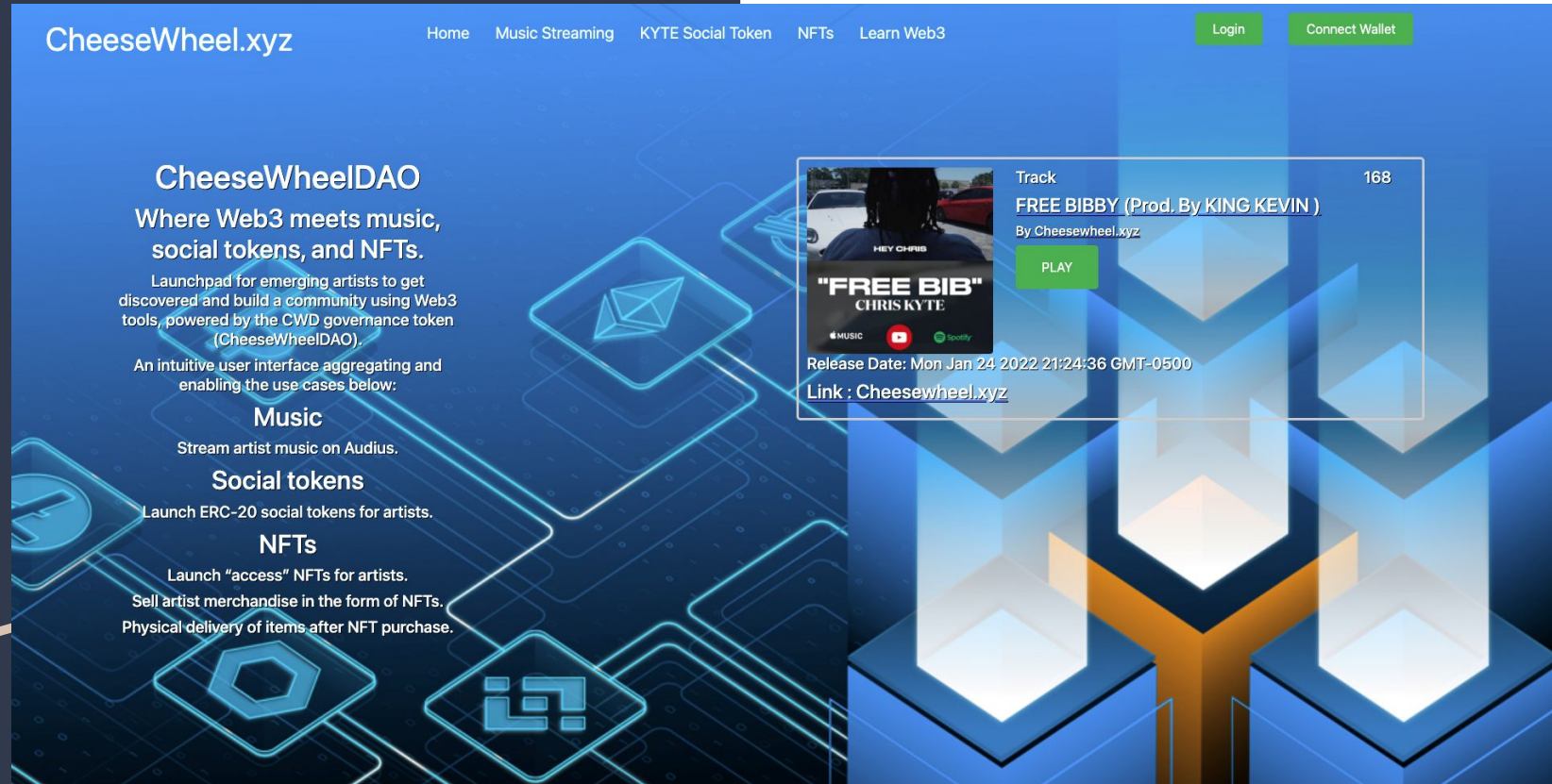
Notion

@cheesewheelDAO and cheesewheel.xyz secured

1. Create the CheesewheelDAO in Discord.
2. Set up an ENS and Snapshot profile for member governance/voting.
3. Link a Mirror blog account for project updates.
4. **Code an ERC-20 token (CWD) for platform governance.**
5. **Code an ERC-20 social token (KYTE) smart contract with crowdfunding functions for artist.**
6. **Code an ERC-721 NFT contract with the CheesewheelDAO logo to showcase on Streamlit.**
7. **Build a front-end dApp interface.**

dApp Front End

- Front end website



dApp Back End

- CWD ERC-20 governance token contract

The screenshot displays the Etherscan web interface for deploying a smart contract. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar is visible, showing the environment set to 'Injected Web3' on a 'Custom (5777) network'. The account address is '0x5bF...b129A (99.9744811)' and the gas limit is '3000000'. The contract being deployed is 'CheesewheelDAO - contracts/CWDto'. The 'DEPLOY' section shows an 'INITIAL_SUPPLY' of '21000000' and a 'transact' button. Below this, there are options to 'Publish to IPFS' or 'At Address'. The bottom of the sidebar shows 'Transactions recorded' and 'Deployed Contracts'.

The main area displays the Solidity code for the 'CWDToken.sol' contract. The code is as follows:

```
1 pragma solidity ^0.5.0;
2
3 import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/token/ERC20/ERC20.sol";
4 import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/token/ERC20/ERC20Detailed.sol";
5
6 contract CheesewheelDAO is ERC20, ERC20Detailed {
7     address payable owner;
8
9     modifier onlyOwner {
10         require(msg.sender == owner, "You do not have permission to mint these tokens!");
11         _;
12     }
13
14     constructor(uint initial_supply) ERC20Detailed("CheesewheelDAO", "CWD", 18) public {
15         owner = msg.sender;
16         _mint(owner, initial_supply);
17     }
18
19     function mint(address recipient, uint amount) public onlyOwner {
20         _mint(recipient, amount);
21     }
22 }
```

The bottom of the interface shows a transaction log with the following details:

- Block: 1
- Transaction Index: 0
- From: 0x5bF...b129A
- To: CheesewheelDAO.(constructor)
- Value: 0 wei
- Data: 0x608...06f40
- Logs: 1
- Hash: 0xb6e...a6f19

dApp Back End

- KYTE ERC-20 social token crowdfunding contract

The screenshot displays a web application interface for managing transactions and contracts. On the left, a sidebar contains navigation icons. The main panel is titled "DEPLOY & RUN TRANSACTIONS". It features a "CONTRACT" dropdown menu set to "KYTE - contracts/KYTE.sol". Below this, there are buttons for "Deploy" (with a "string name, string symbol" input), "Publish to IPFS", and "At Address" (with the address "0x58408A4F962adB5F7B0e0"). A section labeled "Transactions recorded" shows a list of transactions, including "CHEESEWHEELDAO AT 0X18B...3110" and "KYTECROWDSALEDEPLOYER AT 0XF...". Below this, a section titled "Deployed Contracts" lists "kyte_crowdsal_" and "kyte_token_ad_". A "Low level interactions" section includes a "CALLDATA" input field and a "Transact" button. The right side of the interface shows a code editor with two files: "KYTE.sol" and "KYTECrowdsale.sol". The "KYTECrowdsale.sol" file is open, displaying Solidity code for a crowdsale contract. The code includes imports for "KYTE.sol" and "MintedCrowdsale.sol", and defines a "KYTECrowdsale" contract that inherits from "Crowdsale" and "MintedCrowdsale". It also defines a "KYTECrowdsaleDeployer" contract that interacts with the "KYTECrowdsale" contract.

```
1 pragma solidity ^0.5.0;
2
3 import "./KYTE.sol";
4 import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/crowdsale/Crowdsale.sol";
5 import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/crowdsale/emission/MintedCrowdsale.sol";
6
7 contract KYTECrowdsale is Crowdsale, MintedCrowdsale {
8     constructor(
9         uint rate,
10        address payable wallet,
11        KYTE token
12    )
13        Crowdsale(rate, wallet, token)
14        public
15    {
16        // constructor body can stay empty
17    }
18 }
19
20 contract KYTECrowdsaleDeployer {
21     address public kyte_token_address;
22     address public kyte_crowdsale_address;
23
24     constructor(
25         string memory name,
26         string memory symbol,
27         address payable wallet
28     )
29     {
30         public
31
32         KYTE token = new KYTE(name, symbol, 0);
33         kyte_token_address = address(token);
34
35         KYTECrowdsale kyte_crowdsale = new KYTECrowdsale(1, wallet, token);
36         kyte_crowdsale_address = address(kyte_crowdsale);
37
38         token.addMinter(kyte_crowdsale_address);
39     }
40 }
```

At the bottom of the interface, there is a search bar labeled "Search with transaction hash or address" and a list of transactions. One transaction is highlighted, showing a call to "KYTECrowdsaleDeployer.kyte_token_address" with the following details:

```
call from: 0x5b77FC954ff974D3A2288a17e3c8E1fB118b129A to: KYTECrowdsaleDeployer.kyte_token_address() data: 0x847...3e62d [call]
```

dapp Back End

- Artist ERC-721 NFT Contract
- Streamlit registry

```
pragma solidity ^0.5.0;

import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/token/ERC721/ERC721Full.sol";

contract KYTENFT is ERC721Full {
    constructor() public ERC721Full("cheesewheel", "KYTE") {}

    struct Artwork {
        string name;
        string artist;
        uint256 appraisalValue;
    }

    mapping(uint256 => Artwork) public artCollection;

    event Appraisal(uint256 tokenId, uint256 appraisalValue, string reportURI);

    function registerArtwork(
        address owner, string memory name, string memory artist, uint256 initialAppraisalValue, string memory tokenURI )
        public returns (uint256) {
        uint256 tokenId = totalSupply();

        _mint(owner, tokenId);
        _setTokenURI(tokenId, tokenURI);

        artCollection[tokenId] = Artwork(name, artist, initialAppraisalValue);

        return tokenId;
    }

    function newAppraisal(
        uint256 tokenId,
        uint256 newAppraisalValue,
        string memory reportURI
    ) public returns (uint256) {
        artCollection[tokenId].appraisalValue = newAppraisalValue;

        emit Appraisal(tokenId, newAppraisalValue, reportURI);
    }
}
```


dapp Back End

- Artist ERC-721 NFT Contract
- Streamlit registry

Cheesewheel DAO Registry System

Choose an account to get started

Select Account

0x4E22947c217463a60125399ec56abAde25fF86e2

dapp Back End

- Artist ERC-721 NFT Contract
- Streamlit registry

Register New Artwork

Enter the name of the artwork

cheesewheelNFT

Enter the artist name

Chris KYTE

Enter the initial appraisal amount

500000

Upload Artwork



Drag and drop file here

Limit 200MB per file • JPG, JPEG, PNG

Browse files



Cheese-wheel.jpg 45.7KB



Register Artwork

dApp Back End

- Artist ERC-721 NFT Contract
- Streamlit registry

Transaction receipt mined:

```
{
  "transactionHash": "HexBytes('0x3f77efc2f8e2bc9031d9bca968920c7c6d59c77e2b232c1d4326edd83718e84f')",
  "transactionIndex": 0,
  "blockHash": "HexBytes('0xaa445f7f406fb701be44ad4853ef4172c865605722146fb58844c25b08e898bf')",
  "blockNumber": 26,
  "from": "0x0Ca9AD91A7526F02775150bB1F20bEd09E952864",
  "to": "0x547A065671cF416C054C1E432e2FF9AeD74fe85c",
  "gasUsed": 263588,
  "cumulativeGasUsed": 263586,
  "contractAddress": null,
  "logs": [
    {
      "AttributeDict": {
        "logIndex": 0,
        "transactionIndex": 0,
        "transactionHash":
          HexBytes('0x3f77efc2f8e2bc9031d9bca968920c7c6d59c77e2b232c1d4326edd83718e84f'),
        "blockHash":
          HexBytes('0xaa445f7f406fb701be44ad4853ef4172c865605722146fb58844c25b08e898bf'),
        "blockNumber":
          26,
        "address":
          '0x547A065671cF416C054C1E432e2FF9AeD74fe85c',
        "data":
          '0x',
        "topics":
          [
            HexBytes('0xddf252ad1be2c0989c3b66f3778d9d82b277183c4a11828f554d752333e3f'),
            HexBytes('0x0000000000000000000000000000000000000000000000000000000000000000'),
            HexBytes('0x0000000000000000000000000000000000000000000000000000000000000000'),
            HexBytes('0x0000000000000000000000000000000000000000000000000000000000000000')
          ],
        "type":
          'mined'
      }
    }
  ]
}
```

You can view the pinned metadata file with the following IPFS Gateway Link

[Artwork IPFS Gateway Link](#)

dApp Back End

- Artist ERC-721 NFT Contract
- Streamlit registry

Display an Art Token

This address owns 1 tokens

Choose an account to get started

select token_id

0

Display

The token is registered to 0x0Ca9AD91A7526F02775150bB1F20bEd09E952864

The tokenURI is ipfs://bafkreigovpqoe56t24wmj52naa4sw5257tj5i2zrr4u66e7szfmmcfpza



Questions?