- General
    - If a component has functionality that could use a scaffold-eth2 component add it as a placeholder
        - EG if the searchfield component allows a user to manually enter an address use scaffold-eth2's addressinput component instead of creating our own input field with the same functionality

  always use consistent design patterns

  eg if description is truncated to 1 line in one component make sure all other descriptions are also truncated unless I specify otherwise

    - anytime a component will interact w/ the blockchain we will use a placeholder in the component that will be replaced w/ functionality from the scaffold-eth2 project
    - components are built using daisy ui for tailwind
    - we are building all components in daisy ui so they can easily be integated into an existing scaffold-eth2 project
    - all pages include navigation
        - mainly back arrows to go back to previous pages
    - 1% mandatory fee any time funds are sent to jar
        - Handled at SC level but need to be represented in FE
    - claim window is global for whitelist control mechanism
        - EG If anyone makes a claim, everyone needs to wait until after claim window has passed in order to claim
        - Tooltips should make this functionality clear
    - claimable Jars visible to anyone, even if they aren't eligible to claim from them
    - L2s to launch on
        - Create a script that enables us to easily launch on all semirelevant L2s
        - Optimism
        - Base
        - Gnosis
        - Cello
        - Unichain
        - Arbitrum
    - Accepted tokens
        - All ERC20s
        - ETH
    - ensure consistency use claim window every where instead of cool down period

  in the jar admin page the save changes/submit transaction button is only enabled if the corresponding field has been changed

  in the jar creation page there should be some indicator to let the user know that the token they fund the jar with is the only token that the jar will ever be able to accept. Maybe include a note that if there is demand the next version of cookie jar may allow for jars w/ multiple tokens

mobile 1st design

- we will need the following components
  - General component info

    Each component interfaces with the parent CookieJar data structure. Components respond to changes in the underlying jar data and provide appropriate validation and user feedback.

  - SearchField

    input field for user to enter an evm compatible address

    checks all cookiejar whitelists for the user's address

    uses an api provider like infura to query the smart contract to find deployed cookie jars on all networks and looks through all those cookie jar whitelists to find jars the address is whitelisted on

    has search button and also searches when the user presses enter after entering the address

  - SearchResults

    This is where the user will see all jars that the provided evm address has access to. if the user doesn't have access to any jars then we will display this message "looks like the provided address doesn't have access to any cookie jars :crying_face" then it will have a button they can click to create their own cookie jar - This will bring them to the jar creation page.

    this component has 2 separate tabs (need help on labeling the tabs in a user friendly way)

    tab 1 - Claim from Jars - lists jars users are allowed to claim from

    tab 2 - Manage Jars - lists jars users are owners of

    each tab in the search results component will display the jar overview card component for each available jar

    - if user only has access to claim or admin that tab will open by default
      - if they have access to both, claimable will open by default.
  - JarOverviewCard

will display the following info

CJ title/name

CJ description (truncated to 1 line and expandable (read more/show less)

this is a subheading of the cj title heading

Network the jar is deployed on - network icon/badge in upper right corner of card

total jar balance denominated in jar’s token

Max withdrawal amount denominated in jars token

Claimable in ___ days/hrs/minutes/now timer

Truncated address of the deployed cookiejar with the ability to copy the address using a copy button

component is clickable and will take users to the appropriate next page (claimable jar details, adminable jar details (adminable jars only take users to the admin jar details page if they are connected w/ a wallet that owns the jar)).

- JarHeaderCard

  This is the general card that all jar specific information is displayed w/in

  The JarHeader component displays essential information about a Cookie Jar in a card format, serving as the primary presentation layer for the jar's identity and status.

  separate token display w/in the header isn’t necessary in the card header because it’s used as the denominator for the balance in the jarheader component & the withdrawCookies component

  - it displays the following information
    - Dynamic jar title display with prominent typography
    - Jar description
      - Automatically truncates after 1 line and has the ability to be expanded/collapsed via read more/show less
    - Current balance denominated in jar’s specific token
      - EG USDC, ETH, DAI etc
      - Shows the token’s Icon if it is available
    - Network badge/icon to represent network jar is deployed on
    - Contract address with copy and block explorer link (clickable to open in relevant block explorer)

        - contract address goes at the top of the card w/ the other network info
        - Copy-to-clipboard functionality
        - External block explorer link
        - Visual feedback for copy action
    - it contains the following components when it is in the claimable view

      ClaimHistory

      WithdrawCookies

    - it contains the following components in this order when it is in the adminable view

      AddFunds

      WithdrawalSettings

      AccessControl

      ClaimHistoryCard

      JarOwnerCard

  Technical Details:

    - Responsive grid layout that adapts to different screen sizes
    - Text truncation with progressive disclosure for longer descriptions
    - Address formatting utilities for improved readability
    - Clipboard API integration for copying contract address
    - External linking to blockchain explorers for transaction verification
- ClaimHistoryCard

  The ClaimHistory component contains a collapsable section that displays all transactions that have been made for a specific jar.

    - there should be a persistent total claims count at the top of the component before the scrollable transaction history

  has total # of claims made at the top of the card (maybe next to the heading / title of the card?)

  Shown above the scrollable section of claimtransactioncards

  Features:

    - Claim history section showing:
    - Chronological list of all claims made against the jar
    - each transaction is contained in a ClaimTransactionCard
    - Scrollable interface for viewing many transactions
    - Empty state handling when no claims exist

  Technical Details:

    - Virtualized scrolling for performance with many transactions
    - Address truncation for better UI presentation
    - External linking to blockchain explorers for transaction verification
    - Timestamp formatting with relative time display
    - Scrollable claim history with overflow containment
  - ClaimTransactionCard
    - For each transaction, displays:
      - Claim amount denominated in jar's token w/ token icon (if available)
      - Claim amount and token type
      - Timestamp with formatted date
      - Claim reason/purpose as quoted text
      - Truncated wallet address of the claimer
      - Link to transaction on the appropriate block explorer
  - WithdrawCookies

    The WithdrawCookies component provides a dedicated interface for users to withdraw funds from a Cookie Jar.

    Features:

    - Dynamic withdrawal availability status with visual indicators:
      - Green checkmark for available withdrawals
        - available now
      - Clock icon with countdown timer for cooldown periods
        - ensure user friendly unit display
          - eg don't list it in seconds if it is days away
      - tooltip that shows the total cooldown period for that jar and explains how it works
    - Maximum withdrawal amount displayed prominently denominated in jar's token
    - Amount selection system with multiple input methods:
      - Slider control for intuitive amount selection
      - Direct numerical input with validation
      - token type is always listed next to amounts as denominator for best ux
      - Real-time input validation and constraints
    - Reason documentation field:

        - Character count tracking (minimum 20 characters)
        - Validation error messaging
        - Visual feedback for validation errors
    - Submit action (eg withdraw cookies) button with:
        - Conditional enabling based on withdrawal eligibility
        - Loading state during transaction processing
        - Cookie icon for thematic consistency
        - Success notification upon completion

Technical Details:

- Multi-input synchronization between slider and numeric input
- Real-time validation of input values with bounds checking
    - Prevents withdrawals above maximum limit
    - Ensures reason requirements are met
- Integration with withdrawal cooldown system
- Visual feedback system for transaction success
    - happens after transaction is successfully submitted
- Responsive design that works across device sizes

max withdrawal info is only needed in the withdrawal component

cool down info will be contained in the withdrawal component

timer until available or available now

maybe hover to list total cooldown time

- AddFunds

The AddFunds component contains a collapsable section that facilitates adding funds to the jar with options for donating additional funds to the cookiejar team on top of the mandatory 1% fee. all donations use the jar's token

- Amount input field is denominated in Jar's token
    - tooltip explicitly state that jars can only be funded in their original token
    - Displays what amount of funds go to jar and what amount of funds are collected as a fee
    - Flexible input allows the user to specify
        - how much in total is added (eg adding 100 tokens but only 99 go to jar and 1 goes to CJ team)
        - How much goes to the jar (if they want 99 to go to the jar they need to add 100 total)
        - Adjustable fee amount

        - they can adjust the fee % up or down but it can never be below 1%
        - explicitly state that 1% fee is the minimum

  Features:

    - Numerical input for fund amount with token-specific context
    - Configurable fee percentage with real-time calculation
    - Dynamic total calculation showing both jar funding and fee amounts

  Technical Details:

    - Form validation for numeric inputs
    - Real-time calculation of fee amounts
    - Token-specific UI elements showing the jar's native token
    - Percentage-based fee calculations with bounds checking (0-100%)
- WithdrawalSettings

  The JarWithdrawalSettings component contains a collapsable section that manages all withdrawal-related settings for a specific jar.

  each configurable field has it's own save changes button because each change requires an onchain transaction.

  the save changes button isn't clickable until the corresponding setting has actually been changed

  Features:

    - Maximum withdrawal limit configuration with token-specific settings
        - denominated in jar's token
    - Cooldown period management with flexible time unit selection (minutes, hours, days, months
        - tooltip for cooldown period describing its functionality. The amount of time that needs to pass between withdrawals

  Technical Details:

    - Form validation for numeric inputs
    - Error handling for invalid entries
- AccessControl

  The AccessControl component contains a collapsable section that manages all access-related settings for a specific jar.

  contains the following components

UserListCard (whitelisted addresses)

UserListCar (blacklisted addresses)

the whitelist and blacklist variations of this component have tooltips describing their functionality

- UserListCard
    - The UserListCard component contains a collapsable section that manages all list-related settings for a specific jar.
    - it can accept props that have it serve as a blacklist or as a whitelist for accessing a jar
    - each configurable field has it's own save changes button because each change requires an onchain transaction
        - EG if whitelist or blacklist is changed it requires the user to execute an onchain transaction
    - Automatic formatting and deduplication of addresses
    - Supports multiple address input formats (comma-separated, newline-separated)
    - Error handling for invalid entries
    - Form validation for numeric inputs and address formatting
    - Comprehensive list management system:
        - Manual address entry with validation for EVM addresses
        - CSV import/export functionality for bulk address management
        - Real-time validation of address formats
        - list persistence across sessions
            - data will be fetched from / submitted to the appropriate registry smart contract
- JarOwnerCard

    This component displays the address of the jar's current owner

    It displays the title "Jar Ownership Settings"

    It provides the user the ability to transfer ownership of the jar to a different address

    it has address validation checking (checks if entered address is evm compatible address)

    has an edit button next to where current owner's address is displayed

    when user hits edit button it provides a pop up warning that says:

    Caution! you are about to transfer ownership of this jar. Once you execute this transaction you will no longer be able to modify this jar's settings.

Button options:

Change Owner Address

Go Back

If user hits “change owner address” they will see a pop up that lists the current owner address and 2 empty fields for the new owner address. the 2 new owner address fields have validation checking to ensure the values are identical and that it is an EVM compatible addresses. this flow is similar to the flow of many apps for changing passwords

They will also see a Submit button at the bottom of the pop up that only turns green once the 2 address validation check passes. Hitting this button prompts an alert that says;

Caution: You are about to execute a transaction that will immediately kick you off this page. please ensure you have submitted all other changes before proceeding.

button options: submit changes, go back

- pages
    - landing page

      the landing page of the app has 2 buttons.

      1 - search for a jar

      2 - create a jar

      add a background image of cookies and cookie jars

      link to GH at bottom of page

      attribution links

      RG & partners, allo

    - jar search page

      the page will check if a user has connected their wallet. if they haven’t connected their wallet it will provide the connect wallet button so they can see jars they have access to. it will also provide them with the search field component where they can enter an evm compatible address to see which jars it has access to if their wallet isn’t connected connect your wallet to see jars you have access to or use the search field to check if an address has jar access

if their wallet is connected they will no longer be able to see the search field component

if their wallet is not connected the search field component will be persistent so they can continuously search for more jars an address may have access to.

when there are results to display the user will see them displayed in the appropriate tabs of the search results component

when a user clicks on a JarOverViewCard component it takes them to the appropriate jar details page (admin vs claim)

- claimable jar detail page

  has a back button at the top of the page to navigate back to the jar search page that displays the results from the search that got the user to the jar details page.

  eg if the user searched for wallet 0xabc… and clicked on one of the claimable jars displayed, when they navigate back to the jarsearch page they should see the same search results for 0xabc…

  if they scrolled down the page ideally they land at the same level of scroll they were at before they clicked on the jar

  Expanded view shown when user clicks on a JarOverviewCard from the jar search page featuring:

  This page will display all the relevant information for a selected claimable jar.

  it will contain the following components

  jar header component

  WithdrawCookies

  ClaimHistory

  Technical Details:

  - Multi-input synchronization between slider and numeric input
  - Real-time validation of input values with bounds checking
    - Prevents withdrawals above maximum limit
    - Ensures reason requirements are met

        - Integration with withdrawal cooldown system
        - Transaction simulation with state updates
        - Visual feedback system for transaction success
        - Responsive design that works across device sizes
    - adminable jar detail pages

      in order to click on any of the JarOverviewCards in the adminable tab of the SearchResults component the user needs to be connected with a wallet that is the owner of that jar.

      when they mouse over the card it should visually indicate they can't click on the card unless they are connected with the address that is the jar owner

      Each jar can only have one owner at a time.

      Jar Ownership can be transferred

      There is a button at the bottom of the page that allows the owner to "empty the jar" which would transfer all funds from the jar to their address. This is contained in a "danger zone" area similar to the danger zone area on github. the JarOwnerCard component is also in this area. clikcing the empty the jar button triggers an alert that says

      DANGER: you are about to drain this jar. Doing so will transfer all funds to your address.

      button options: go back, drain funds!

      This page contains the following components (in this order):

      JarHeaderCard

      AddFunds

      WithdrawalSettings

      AccessControl

      ClaimHistoryCard

      JarOwnerCard

    - Jar creation page

      This is the page the user will create and deploy cookie jars from the user needs to have their wallet connected in order to create a jar

if they click on the create jar button they will get prompted to connect their wallet in order to proceed if it's not already connected

When jars are created the following information needs to be specified by the jar creator:

Jar owner - uses JarOwnerCard component

jar title

jar description.

claim window

(how long users must wait between fund withdrawals. Eg if the claim window is 60 min then funds can only be withdrawn every 60 min)

should have a tooltip describing this setting

- network selection

  tooltip that explains that this will always be the network for the jar and can't be changed after the jar is created

  this will be a dropdown to select the network the cookie jar is deployed on

  only one network may be selected

  it will be populated with the most popluar EVM compatiable chains where the cookiejar factory contracts are deployed

- Jar token selection

  tooltip has info that explains that this will always be the token for the jar and can't be changed after the jar is created

  user will specify the jar token w/ a dropdown that has a preloaded list of erc20 tokens and eth. when the user selects the token it displays the token contract address and token symbol & token decimals. this is so the user can verify they are actually using the correct contract when using the preloaded tokens

  the dropdown will also have an option to use a custom erc20 token. if the user selects custom erc20 it will present the token contract address as an input field w/ a tooltip that describes how to find the address/what it is. it will also show fields that auto populate the token symbol and token decimals by making a call that fetches this info from the contract.

  the ui will use this information to adjust the display of the initial funding amount to be in a user friendly format (eg it would use the token decimals to show the input in ETH instead of WEI). The ui uses the same logic to adjust the value of the initial funding amount field if the user uses a preloaded token from the dropdown

- Initial funding amount - uses the AddFunds Component but adjusted for the following info:

  Will be denominated using the token symbol entered in the jar token selection input

  it will have an option to see more or tooltip that shows the user how it is using the token decimal to display the token units in a more user friendly way.

  this field can have 0 tokens entered at creation but will always be in the token selected from jar token selection setting

maximum withdrawal amount (this sets the maximum amount of funds a user can withdraw per claim window period.

Not sure if we should use the withdrawal settings component here

if we do it will not have save buttons for each individual setting since everything will all be submitted in one transaction when the jar is created.

access control

uses AccessControl component configured for whitelist and blacklist

Create cookie jar button at the bottom of the page

This initiates a transaction that the user must execute in order to deploy the cookie jar w/ all their selected settings. If they declared an amount in the intial funding amount field the jar will be funded when it is deployed

funded the jar it will also fund the jar at the same time

- this page will contain the following components:

  AccessControl

AddFunds

- WithdrawalSettings

  (won't have buttons for each individual setting like it does on the jar admin page.

- JarOwnerCard

  (needs to be an evm compatiable wallet address),

  By default this field will be auto populated with the address of the jar creator

  Will have an edit button that enables the jar creator to provide a different address for the jar owner

  when they hit the edit button it will warn them that only the new address will be able to modify jar settings after the jar is created