Cookie Jar UI MVP Project Brief

Project Overview

The Cookie Jar application is a virtual petty cash drawer that allows users to create, fund, and manage digital "jars" from which authorized individuals can withdraw funds. This brief outlines the UI components needed for integration into a scaffold-eth2 project.

Technical Requirements

- **Language**: TypeScript for all components
- **Framework**: Next.js with App Router
- **UI Library**: DaisyUI for Tailwind CSS
- **Design Approach**: Mobile-first
- **Integration**: Components built to work with scaffold-eth2 project
- **Blockchain Interaction**: Use placeholders for blockchain functionality
- **Token Support**: ETH and ERC20 tokens
- **Network Support**: Optimism, Base, Gnosis, Cello, Unichain, Arbitrum

Design Guidelines

- Consistent design patterns across all components
- Text truncation (1 line with expand/collapse) for descriptions unless specified otherwise
- Use scaffold-eth2 components where applicable (e.g., AddressInput)
- All pages include navigation (primarily back arrows)
- 1% mandatory fee represented in UI for all fund transfers
- Claim window functionality clearly explained via tooltips

- Button states tied to field changes (e.g., Save buttons only enabled when field is modified)

Page Structure

1. Landing Page

- Two primary buttons: "Search for a jar" and "Create a jar"
- Background image featuring cookies and cookie jars
- GitHub link and attribution links (RG & partners, Allo) at bottom

2. Jar Search Page

Components:

- **SearchField** (only visible when wallet not connected)
- **SearchResults** (with two tabs: "Claim from Jars" and "Manage Jars")
  - Each tab contains **JarOverviewCard** components

3. Claimable Jar Detail Page

Components:

- Back button to return to search results
- **JarHeaderCard** containing:
  - **WithdrawCookies**
  - **ClaimHistory**

4. Adminable Jar Detail Page

Components (in order):

- **JarHeaderCard**
- **AddFunds**

- **WithdrawalSettings**

- **AccessControl**

- **ClaimHistoryCard**

- **JarOwnerCard**

- "Danger zone" area with jar draining functionality

5. Jar Creation Page

Components:

- **JarOwnerCard** (auto-populated with creator's address)

- Jar title input

- Jar description input

- Claim window settings (part of **WithdrawalSettings**)

- Network selection dropdown

- Jar token selection

- Initial funding amount (**AddFunds** component modified for creation)

- Maximum withdrawal amount (part of **WithdrawalSettings**)

- **AccessControl** for whitelist and blacklist

- "Create cookie jar" button

Component Specifications

SearchField

```

- Input field for EVM-compatible addresses

- Integration with scaffold-eth2's AddressInput component

- Search button and Enter key functionality
```

- Queries all networks for jars the address is whitelisted for

```

### SearchResults

```

- Two tabs: "Claim from Jars" and "Manage Jars"

- Default tab selection based on user access

- Empty state message with jar creation button

- Contains JarOverviewCard components for each available jar

```

### JarOverviewCard

```

- Clickable card displaying:

  - Jar title/name

  - Description (truncated to 1 line, expandable)

  - Network icon/badge

  - Total jar balance

  - Max withdrawal amount

  - Claimable timer

  - Truncated contract address with copy button

- Routes to appropriate detail page when clicked

```

### JarHeaderCard

```

- Core presentation component for jar identity and status
- Displays:
  - Jar title
  - Description (truncated, expandable)
  - Current balance with token information
  - Network badge/icon
  - Contract address with copy and explorer functions
- In claimable view, contains:
  - ClaimHistory
  - WithdrawCookies
- In adminable view, contains:
  - AddFunds
  - WithdrawalSettings
  - AccessControl
  - ClaimHistoryCard
  - JarOwnerCard

```

### ClaimHistoryCard

```
- Collapsible section showing all jar transactions
- Persistent total claims count displayed at top
- Contains scrollable list of ClaimTransactionCard components
- Empty state handling for no claims
```
```

ClaimTransactionCard

```
- Displays for each transaction:
  - Claim amount with token icon
  - Timestamp with formatted date
  - Claim reason as quoted text
  - Truncated claimer wallet address
  - Block explorer transaction link
```

WithdrawCookies

```
- Withdrawal interface with:
  - Availability status indicators (checkmark/timer)
  - Tooltips explaining claim window functionality
  - Maximum withdrawal amount display
  - Amount selection (slider and direct input)
  - Reason documentation field (minimum 20 characters)
  - Submit button with conditional enabling
- Real-time validation and feedback
```

AddFunds

```
- Collapsible section for adding funds
- Amount input denominated in jar's token
```

- Fee calculation with adjustable percentage (minimum 1%)
- Tooltip explaining jar token limitations
- Dynamic calculation showing jar funding vs. fee amounts
```

### WithdrawalSettings
```
- Collapsible section for withdrawal configuration
- Maximum withdrawal limit input (denominated in jar's token)
- Cooldown period management with time unit selection
- Individual save buttons for each setting (enabled only when changed)
- Tooltips explaining functionality
```

### AccessControl
```
- Collapsible section containing:
  - UserListCard for whitelisted addresses
  - UserListCard for blacklisted addresses
- Tooltips explaining whitelist/blacklist functionality
```

### UserListCard
```
- Configurable for whitelist or blacklist via props
- Address list management:

  - Manual address entry with validation
  - CSV import/export
  - Address formatting and deduplication
- Individual save button for list changes
```

### JarOwnerCard

```
- Displays current jar owner address
- Ownership transfer functionality
- Warning dialogs for ownership changes
- Address validation for new owner
- Confirmation flow similar to password change patterns
```

Integration Notes

- All components should use scaffold-eth2's existing functionality where possible
- All blockchain transactions should be represented with placeholders
- The UI should clearly indicate the 1% mandatory fee on all transactions
- Token functionality limited to a single token per jar (with appropriate explanations)
- The claim window mechanism should be clearly explained via tooltips
- All deployments target the specified L2 networks
- Save/submit buttons only enabled when corresponding fields have been changed

Special Considerations
```

- The jar creation page should clearly indicate that token selection is permanent
- Jar administration requires wallet connection matching the jar owner
- Non-administrative viewing of jars is possible without ownership
- Mobile-first design is essential for all components

Would you like me to clarify or add more detail to any specific part of this brief?