Frontend structure:

my-app/

├── public/

│ ├── index.html

│ └── favicon.ico

├── src/

│ ├── components/

│ │ ├── App/

│ │ │ ├── App.jsx

│ │ │ └── App.css

│ │ ├── Header/

│ │ │ ├── Header.jsx

│ │ │ └── Header.css

│ │ ├── Customer/

│ │ │ ├── BuyCarbonCredits/

│ │ │ │ ├── BuyCarbonCredits.jsx

│ │ │ │ └── BuyCarbonCredits.css

│ │ │ ├── SellCarbonCredits/

│ │ │ │ ├── SellCarbonCredits.jsx

│ │ │ │ └── SellCarbonCredits.css

│ │ │ └── OffsetCarbonFootprint/

│ │ │ ├── OffsetCarbonFootprint.jsx

│ │ │ └── OffsetCarbonFootprint.css

│ │ └── Treasury/

│ │ ├── WrapCarbonCredits/

│ │ │ ├── WrapCarbonCredits.jsx

│ │ │ └── WrapCarbonCredits.css

│ │ └── BridgeCarbonCredits/

│ │ ├── BridgeCarbonCredits.jsx

│ │ └── BridgeCarbonCredits.css

│ ├── utils/

│ │ ├── contracts.js

│ │ └── tronWeb.js

│ ├── App.test.js

│ ├── index.js

│ └── index.css

├── .env

├── .gitignore

├── package.json

└── README.md

Here's a step by step implementation and integration plan for each file:

1. Set up the React project:
   * Run **npx create-react-app my-app** to create the basic structure of the React app.
   * Install required dependencies: **npm install @tronprotocol/tron-web tronlink-lib**
2. Configure TronWeb:
   * Create a **src/utils/tronWeb.js** file to set up and export the TronWeb instance.
3. Create and set up the contract instances:
   * In **src/utils/contracts.js**, import the TronWeb instance from **tronWeb.js**.
   * Add the ABI and contract addresses for the Marketplace, Bridge, ERC1155Token, and WrappedBCT contracts.
   * Create and export the contract instances using TronWeb.
4. Implement the Header component:
   * Create a **Header.jsx** file in **src/components/Header/** with a functional component that renders the header, including the app title and wallet connection status.
   * Add styles for the header in **Header.css**.
5. Implement the App component:
   * In **src/components/App/App.jsx**, import the Header component.
   * Conditionally render the Customer and Treasury components based on the connected wallet's status (customer or treasury).
   * Add the necessary imports and state management logic for wallet connection and user type determination.
   * Add styles for the App component in **App.css**.
6. Implement the Customer components:
   * BuyCarbonCredits:
     + Create a **BuyCarbonCredits.jsx** file in **src/components/Customer/BuyCarbonCredits/** with a functional component that includes a form to input the token ID, amount, and payment method.
     + Add the necessary event handlers and logic to interact with the Marketplace contract to buy carbon credits.
     + Add styles for the BuyCarbonCredits component in **BuyCarbonCredits.css**.
   * SellCarbonCredits:
     + Create a **SellCarbonCredits.jsx** file in **src/components/Customer/SellCarbonCredits/** with a functional component that includes a form to input the token ID, amount, and price per token.
     + Add the necessary event handlers and logic to interact with the Marketplace contract to sell carbon credits.
     + Add styles for the SellCarbonCredits component in **SellCarbonCredits.css**.
   * OffsetCarbonFootprint:
     + Create an **OffsetCarbonFootprint.jsx** file in **src/components/Customer/OffsetCarbonFootprint/** with a functional component that includes a form to input the token ID and amount of carbon credits to offset.
     + Add the necessary event handlers and logic to interact with the Offset contract to offset carbon credits.
     + Add styles for the OffsetCarbonFootprint component in **OffsetCarbonFootprint.css**.
7. Implement the Treasury components:
   * WrapCarbonCredits:
     + Create a **WrapCarbonCredits.jsx** file in **src/components/Treasury/WrapCarbonCredits/** with a functional component that includes a form to input the amount of carbon credits to wrap.
     + Add the necessary event handlers and logic to interact with the WrappedBCT contract to wrap carbon credits.
     + Add styles for the WrapCarbonCredits component in **WrapCarbonCredits.css**.
   * BridgeCarbonCredits:
     + Create a **BridgeCarbonCredits.jsx** file in **src/components/Treasury/BridgeCarbonCredits/** with a functional component that includes a form to input the wrapper address and the amount of WrappedBCT tokens to bridge.
     + Add the necessary event handlers and logic to interact with the Bridge contract to bridge carbon credits.
     + Add styles for the BridgeCarbonCredits component in **BridgeCarbonCredits.css**.
8. Update src/index.js and src/index.css:

* Import the necessary dependencies and components in **src/index.js**.
* Wrap the App component with the necessary providers (e.g., TronWebProvider) if required.
* Update the **src/index.css** file with any global styles for the application.

1. Update the **.env** file:
   * Add any necessary environment variables such as API keys, contract addresses, and wallet private keys for local development and testing.
2. Write tests:
   * Create test files for the components and utilities, such as **App.test.js**, **Header.test.js**, **BuyCarbonCredits.test.js**, etc.
   * Write test cases for the component rendering, state management, event handling, and contract interactions.
   * Run the tests using the built-in testing framework (**npm test**).
3. Optimize the application for production:
   * Check the performance of the application using the built-in React developer tools.
   * Optimize the code and styles for better performance if necessary.
4. Build the application:
   * Run **npm run build** to generate an optimized build for production.
5. Deploy the application:
   * Choose a hosting provider (e.g., Netlify, Vercel, Firebase, etc.) and follow their deployment process to make the application publicly accessible.

This step-by-step plan should guide you through the implementation and integration of each file in your frontend project, helping you create a well-structured and functional application that meets your requirements.