SDK Integration

Remove unnecasary code

Let's get started with working on theindex.tsx file.

In the jsx of the file let's remove all elements of the page except for the Head tags and Main tags. Your Component should look like this:

```
export

default

function

Home ( )

{ return

( <

    < Head

    < title

    Based Account Abstraction < / title

    < meta name = "description" content = "Based Account Abstraction"

/

    < / Head

    < main className = { styles . main }

< / main

    < /

    ) }
```

Notice I changed my title and description, feel free to do that. On the main tags I added a className forstyles.main to get all my contect the centered look.

Set up Particle Auth

Now we're going to leverage Particle Auth through the SDK to set up social logins. Remember we're focusing here on users who have never onboarded onto web3 via a wallet and just want o experience minting their first NFT.

Let's import the Particle Auth Package

```
import

{ ParticleAuthModule , ParticleProvider }

from

"@biconomy/particle-auth" ;
```

In your React component define an instance of the Particle Auth Module. The module will require api keys which you can get from theParticle Dashboard .

```
const particle =

new

ParticleAuthModule . ParticleNetwork ( { projectId :

"" , clientKey :

"" , appId :

"" , wallet :

{ displayWalletEntry :
```

true , defaultWalletEntryPosition : ParticleAuthModule . WalletEntryPosition . BR , } , }) ; I removed a couple extra items that might exist in their docs but those are all optional parameters, the paramaters listed above are the minimum ones you need to start the engine.

Next lets get going with a connect function. This will contain the logic for logging in with the SDK.

const

connect

=

async

( )

=>

{ try

{ const userInfo =

await particle . auth . login ( ) ; console . log ("Logged in user:" , userInfo) ; const particleProvider =

new

ParticleProvider (particle . auth) ; console . log ({ particleProvider }) ; const web3Provider =

new

ethers . providers . Web3Provider (particleProvider , "any" ,) ; }

catch

(error)

{ console . error (error) ; } } ; Let's create a button in our component that will execute the above function on click:

< main className = { styles . main }

    < button className = { styles . connect } onClick = { connect }

    Connect < / button

    < / main

    We pass the connect function to theonClick handler and pass some more styles from our styles object. Try this out you now log in with several different social providers or via email with a one time password. General user information will be logged into the console upon succesful login.

Create a Smart Account

Now that we have our login method enabled, lets use the Particle Network integration to now build our own smart account.

Add the following imports to yourindex.tsx :

import

{ useState }

from

"react" ; import

{ IBundler , Bundler }

from

"@biconomy/bundler" ; import

{ BiconomySmartAccount , BiconomySmartAccountConfig , DEFAULT_ENTRYPOINT_ADDRESS , }

from

"@biconomy/account" ; import

{ ethers }

from

"ethers" ; import

{ ChainId }

from

"@biconomy/core-types" ; import

{ IPaymaster , BiconomyPaymaster }

from

"@biconomy/paymaster" ; Now in the React component we're going to define the instance of our Bundler and Paymaster:

const bundler : IBundler =

new

Bundler ({ bundlerUrl :

// bundler URL from dashboard use 84531 as chain id if you are following this on base goerli, chainId : ChainId .
BASE_GOERLI_TESTNET , entryPointAddress :

DEFAULT_ENTRYPOINT_ADDRESS , })

const paymaster : IPaymaster =

new

BiconomyPaymaster ({ paymasterUrl :

// paymaster url from dashboard }) We're also going to add some state variables to the component along with their typings:

const

[address , setAddress]

=

useState < string

    ("") ; const

[loading , setLoading]

=

useState < boolean

    (false) ; const

[smartAccount , setSmartAccount]

=

useState < BiconomySmartAccount |

null

    (null ,) ; const

[provider , setProvider]

=

useState < ethers . providers . Provider |

null

(null ,) ; Now we'll update the Connect function to create a smart account using the Biconomy Smart Account package:

const

connect

=

async

( )

=>

{ try

{ setLoading (true) ; const userInfo =

await particle . auth . login ( ) ; console . log ("Logged in user:" , userInfo) ; const particleProvider =

new

ParticleProvider (particle . auth) ; const web3Provider =

new

ethers . providers . Web3Provider (particleProvider , "any" ,) ; setProvider (web3Provider) ; const biconomySmartAccountConfig : BiconomySmartAccountConfig =

{ signer : web3Provider . getSigner ( ) , chainId : ChainId . BASE_GOERLI_TESTNET , bundler : bundler , paymaster : paymaster , } ; let biconomySmartAccount =

new

BiconomySmartAccount (biconomySmartAccountConfig ,) ; biconomySmartAccount =

await biconomySmartAccount . init ( ) ; setAddress (await biconomySmartAccount . getSmartAccountAddress ( )) ; setSmartAccount (biconomySmartAccount) ; setLoading (false) ; }

catch

(error)

{ console . error (error) ; } } ; Now upon login we're also in the background creating a Smart Account for our users. Let's update the JSX in the component as well:

< main className = { styles . main }

< h1

Based Account Abstraction < / h1

< h2

Connect and Mint your AA powered NFT now < / h2

{ ! loading &&

! address &&

< button onClick = { connect } className = { styles . connect }

Connect to Based Web3 < / button

} { loading &&

< p

Loading Smart Account ... < / p

} { address &&

< h2

Smart Account :

{ address } < / h2

} < / main

Now when we login our Smart account address will be displayed for us on the screen. You'll notice that the main thing we need for interaction between Particle Auth and the Biconomy Smart Account is an ethers provider object. Keep this in mind if you want to use any other auth provider, as long as you can pass the ethers provider object your auth tool of choice will be compatible with the Biconomy SDK.

You now have integrated the SDK along with Particle Auth for Social Logins. Lets now execute our transaction and allow the user of our dApp to mint an NFT completely for free. Previous Initialize Frontend Next Execute a Gasless Transaction