

## Part 3: Building a simple Web Application

In this part of the tutorial, we will learn how to build a simple web application that will interact with the contract from the first part of this tutorial series. You need to have Neutron up and running, and the contract to be instantiated. You can learn how to do all of that by following the first part of this tutorial.

To build the UI, we will be using:

- [Next.js](#)
- as a framework
- [shadcn-ui](#)
- for components
- [CosmosKit](#)
- for wallets

The final result of this tutorial can be found at [https://github.com/neutron-org/onboarding/tree/main/minimal\\_ui](https://github.com/neutron-org/onboarding/tree/main/minimal_ui).

### Setup Next.js app

1. Install [Node.js](#)
2. v18 or higher.
3. Create Next.js app:
4. `npx create-next-app@latest minimal_ui`
5. **- Would you like to use TypeScript: Yes**
6. **- Would you like to use ESLint: Yes**
7. **- Would you like to use Tailwind CSS: Yes**
8. **- Would you like to use src/ directory: Yes**
9. **- Would you like to use App Router: Yes**
10. **- Would you like to customize the default import alias (@/\*)?: No**
11. `cd`
12. `minimal_ui`
13. Install shadcn-ui and add components:
14. `npx shadcn@latest init --defaults`
15. `npx shadcn@latest`
16. `add`
17. `button card input label`

### Add CosmosKit

1. Install packages:
2. `npm`
3. `install`
4. `--save @cosmos-kit/react @interchain-ui/react @cosmos-kit/core @cosmos-kit/keplr-extension @cosmjs/amino @cosmjs/proto-signing chain-registry @chain-registry/types`
5. Create `contextsrc/context/CosmosKit.tsx`
6. with the following content:
7. `"use client"`
8. `;`
9. `import`
10. `{`
11. `AssetList`

```

12. ,
13. Chain
14. }
15. from
16. "@chain-registry/types"
17. ;
18. import
19. {
20. GasPrice
21. }
22. from
23. "@cosmjs/stargate"
24. ;
25. // You can add more wallets here
26. import
27. {
28. wallets
29. as
30. keplrExtension
31. }
32. from
33. "@cosmos-kit/keplr-extension"
34. ;
35. import
36. {
37. ChainProvider
38. }
39. from
40. "@cosmos-kit/react"
41. ;
42. import
43. assert
44. from
45. "assert"
46. ;
47. import
48. {
49. assets
50. ,
51. chains
52. }
53. from
54. "chain-registry"
55. ;
56. import
57. React
58. from
59. "react"
60. ;
61. import
62. "@interchain-ui/react/styles"
63. ;
64. // This is a Neutron Localnet chain, we need to add it manually because it's not in the chain registry.
65. // This new chain can be based on Neutron Testnet, we just need to adjust some parameters.
66. const
67. localnetChain
68. :
69. Chain
70. =
71. (
72. (
73. )
74. =>
75. {
76. const
77. chain
78. =
79. chains

```

```

80. .
81. find
82. (
83. (
84. chain
85. )
86. =>
87. chain
88. .
89. chain_name
90. ===
91. "neutrontestnet"
92. )
93. ;
94. assert
95. (
96. chain
97. )
98. ;
99. return
100. {
101. ...
102. chain
103. ,
104. // Chain ID is a unique identifier for the chain. You can find one in localnet_config.json.
105. chain_id
106. :
107. "ntrntest"
108. ,
109. // Chain name is another unique identifier for the chain that is used in CosmosKit.
110. chain_name
111. :
112. "neutronlocalnet"
113. ,
114. // Pretty name is a human readable name for the chain.
115. pretty_name
116. :
117. "Neutron Localnet"
118. ,
119. apis
120. :
121. {
122. ...
123. chain
124. .
125. apis
126. ,
127. // RPC and REST endpoints are used to communicate with the chain.
128. // We provide proxy endpoints here, which can be found in the "First launch" section.
129. rpc
130. :
131. [
132. {
133. address
134. :
135. "http://localhost:3001/proxy"
136. }
137. ]
138. ,
139. rest
140. :
141. [
142. {
143. address
144. :
145. "http://localhost:3002/proxy"
146. }
147. ]

```

```
148. ,
149. }
150. ,
151. }
152. ;
153. }
154. )
155. (
156. )
157. ;
158. // The same for assets
159. const
160. localnetAssets
161. :
162. AssetList
163. =
164. (
165. (
166. )
167. =>
168. {
169. const
170. asset
171. =
172. assets
173. .
174. find
175. (
176. (
177. asset
178. )
179. =>
180. asset
181. .
182. chain_name
183. ===
184. "neutrontestnet"
185. )
186. ;
187. assert
188. (
189. asset
190. )
191. ;
192. return
193. {
194. ...
195. asset
196. ,
197. chain_name
198. :
199. "neutronlocalnet"
200. ,
201. }
202. ;
203. }
204. )
205. (
206. )
207. ;
208. export
209. const
210. CosmosKitProvider
211. =
212. (
213. {
214. children
215. ,
```

```
216. }
217. :
218. {
219. children
220. :
221. React
222. .
223. ReactNode
224. ;
225. }
226. )
227. =>
228. (
229. <
230. ChainProvider
231. chains
232. =
233. {
234. [
235. ...
236. chains
237. ,
238. localnetChain
239. ]
240. }
241. assetLists
242. =
243. {
244. [
245. ...
246. assets
247. ,
248. localnetAssets
249. ]
250. }
251. // We need to specify gas price to be able to sign transactions.
252. // The provided value works just fine for the localnet.
253. signerOptions
254. =
255. {
256. {
257. signingCosmwasm
258. :
259. (
260. )
261. =>
262. (
263. {
264. gasPrice
265. :
266. GasPrice
267. .
268. fromString
269. (
270. "0.01untrn"
271. )
272. ,
273. }
274. )
275. ,
276. signingStargate
277. :
278. (
279. )
280. =>
281. (
282. {
283. gasPrice
```

```
284. :
285. GasPrice
286. .
287. fromString
288. (
289. "0.01untrn"
290. )
291. ,
292. }
293. )
294. ,
295. }
296. }
297. wallets
298. =
299. {
300. [
301. ...
302. keplrExtension
303. ]
304. }
305.

306. {
307. children
308. }
309. <!--
310. ChainProvider
311. -->
312. )
313. ;
314. Add context by replacing the content of src/app/layout.tsx
315. with the following:
316. import
317. {
318. CosmosKitProvider
319. }
320. from
321. "@contexts/CosmosKit"
322. ;
323. import
324. type
325. {
326. Metadata
327. }
328. from
329. "next"
330. ;
331. import
332. "../globals.css"
333. ;
334. export
335. const
336. metadata
337. :
338. Metadata
339. =
340. {
341. title
342. :
343. "Minimal UI"
344. ,
345. }
346. ;
347. export
348. default
349. function
350. RootLayout
351. (
```

```

352. {
353. children
354. ,
355. }
356. :
357. Readonly
358. <
359. {
360. children
361. :
362. React
363. .
364. ReactNode
365. ;
366. }
367.

368. )
369. {
370. return
371. (
372. <
373. html
374. lang
375. =
376. "
377. en
378. "
379.

380. <
381. body
382.

383. <
384. CosmosKitProvider
385.

386. {
387. children
388. }
389. <!--
390. CosmosKitProvider
391. -->
392. <!--
393. body
394. -->
395. <!--
396. html
397. -->
398. )
399. ;
400. }

```

## Brief overview of CosmosKit

CosmosKit is a library that allows you to easily communicate with different Cosmos chains and wallets. To use it you setup a context with a list of chains and wallets you want to support.

Then you can use the `useChain` hook to connect to communicate with a specific chain like this:

tip The code below is an example of how to use CosmosKit. Don't copy it to the project! `const`

```
{ address , connect , disconnect , getCosmWasmClient , getSigningCosmWasmClient , }
```

```
=
```

```
useChain ( "neutronlocalnet" ,
```

```

true ) ;

// address is the address of the connected wallet. // It is undefined if no wallet is connected. console . log ( "Your address:" ,
address ??

"Not connected" ) ;

// connect is a function that opens a modal to connect to a wallet. await

connect ( ) ;

// disconnect is a function to disconnect from a wallet. await

disconnect ( ) ;

// getCosmWasmClient is a function that returns a CosmWasmClient for the connected chain. // This client is used to make
queries. const client =

await

getCosmWasmClient ( ) ; const balance =

await client . getBalance ( address ,

"untrn" ) ; console . log ( "Your balance:" , balance ) ;

// getSigningCosmWasmClient is a function that returns a SigningCosmWasmClient for the connected chain. // This client is used
to sign transactions. const client =

await

getSigningCosmWasmClient ( ) ; const

{ transactionHash }

=

await client . sendTokens ( address , receiver , [ { amount :

"1000000" , denom :

"untrn" , } , ] , "auto" ) ; console . log ( "TX hash:" , transactionHash ) ;

```

## Create a wallet button component

This component will be used to open CosmosKit wallets modal and perform connection to a wallet.

1. Create a component with wallet button in src/components/WalletButton.tsx
2. :
3. import
4. {
5. Button
6. }
7. from
8. "@components/ui/button"
9. ;
10. import
11. {
12. useChain
13. }
14. from
15. "@cosmos-kit/react"
16. ;
17. const
18. formatAddress
19. =
20. (
21. address
22. :
23. string
24. )



```
25. =>
26. {
27. return
28. address
29. .
30. slice
31. (
32. 0
33. ,
34. 11
35. )
36. +
37. "..."
38. +
39. address
40. .
41. slice
42. (
43. -
44. 3
45. )
46. ;
47. }
48. ;
49. export
50. const
51. WalletButton
52. =
53. (
54. )
55. =>
56. {
57. const
58. {
59. address
60. ,
61. connect
62. }
63. =
64. useChain
65. (
66. "neutronlocalnet"
67. ,
68. true
69. )
70. ;
71. return
72. (
73. <
74. Button
75. variant
76. =
77. {
78. address
79. ?
80. "outline"
81. :
82. "default"
83. }
84. type
85. =
86. "
87. button
88. "
89. onClick
90. =
91. {
92. connect
```

```

93. }
94.

95. {
96.   address
97.   ?
98.   formatAddress
99.   (
100.    address
101.  )
102.  :
103.  "Connect Wallet"
104. }
105. <!--
106. Button
107. -->
108. )
109. ;
110. }
111. ;
112. Replace thesrc/app/page.tsx
113. with the following content:
114. "use client"
115. ;
116. import
117. {
118.   WalletButton
119. }
120. from
121. "@/components/WalletButton"
122. ;
123. export
124. default
125. function
126. Home
127. (
128. )
129. {
130.   return
131.   (
132.     <
133.       main
134.       className
135.       =
136.       "
137.       flex flex-col gap-8 items-center justify-center h-screen
138.       "
139.
140.     <
141.       WalletButton
142.     />
143.     <!--
144.     main
145.     -->
146.   )
147.   ;
148. }

```

## First launch

Now we have a simple UI that can connect to a wallet. Let's try it out.

### Prepare your browser

1. Install [Keplr Wallet Extension](#)
2. .
3. When prompted to create a wallet, click **Import an existing wallet**

4. and then
5. clickUse recovery phrase or private key
6. .
7. Enter mnemonic from [Part 1](#)
8. :
9. kiwi valid tiger wish shop time exile client metal view spatial ahead

## Run CORS Proxy

Before we can do anything with the local chain, we need to launch [Local CORS Proxy](#) for the localnet. Run the following commands in a separate terminals and keep them in the background:

```
npx local-cors-proxy@latest --proxyUrl http://0.0.0.0:26657 --port 3001 npx local-cors-proxy@latest --proxyUrl http://0.0.0.0:1317 --port 3002
```

## Launch the app

npm run dev And open it in the browser: <http://localhost:3000>

Now you should see a wallet button and should be able to connect to a wallet.

## Interact with the contract

In this UI we provide an example of interacting with the contract created in [Part 1](#).

Contract address The address of your contract might be different from what you see in this tutorial. Make sure that you are replacing the addresses from the commands below with the address of your contract! To interact with the contract we'll be using `CosmWasmClient` and `SigningCosmWasmClient` from [CosmJS](#).

Querying value from the contract:

```
const client =
  await
    getCosmWasmClient ( ) ;
const
  { current_value }
  =
  ( await client . queryContractSmart ( "neutron1nyuryl5u5z04dx4zsqqvsuw7fe8gl2f77yufynauuhklmnmnjncqcls0tj" , {
    current_value :
      { } } ) )
  as
  Promise < { current_value :
    string
  }
  ;
```

console . log ( current\_value ) ; Executing a message:

```
const client =
  await
    getSigningCosmWasmClient ( ) ;
const
  { transactionHash }
  =
```

```
await client . execute ( address , "neutron1nyuryl5u5z04dx4zsqqvsw7fe8gl2f77yufynauuhklnnmnjncqcls0tj" , {
increase_count :
```

```
{ amount :
```

```
"1" , } , } , "auto" ) ;
```

```
console . log ( transactionHash ) ; Let's create a component that will do this for us.
```

```
1. Create a component insrc/components/Counter.tsx
```

```
2. :
```

```
3. import
```

```
4. {
```

```
5. Button
```

```
6. }
```

```
7. from
```

```
8. "@components/ui/button"
```

```
9. ;
```

```
10. import
```

```
11. {
```

```
12. Card
```

```
13. ,
```

```
14. CardContent
```

```
15. ,
```

```
16. CardFooter
```

```
17. ,
```

```
18. CardHeader
```

```
19. ,
```

```
20. CardTitle
```

```
21. ,
```

```
22. }
```

```
23. from
```

```
24. "@components/ui/card"
```

```
25. ;
```

```
26. import
```

```
27. {
```

```
28. Input
```

```
29. }
```

```
30. from
```

```
31. "@components/ui/input"
```

```
32. ;
```

```
33. import
```

```
34. {
```

```
35. Label
```

```
36. }
```

```
37. from
```

```
38. "@components/ui/label"
```

```
39. ;
```

```
40. import
```

```
41. {
```

```
42. useChain
```

```
43. }
```

```
44. from
```

```
45. "@cosmos-kit/react"
```

```
46. ;
```

```
47. import
```

```
48. assert
```

```
49. from
```

```
50. "assert"
```

```
51. ;
```

```
52. import
```

```
53. {
```

```
54. useCallback
```

```
55. ,
```

```
56. useEffect
```

```
57. ,
```

```
58. useState
```

```
59. }
```

```
60. from
```

```

61. "react"
62. ;
63. // Your contract address here
64. const
65. CONTRACT_ADDRESS
66. =
67. "neutron1nyuryl5u5z04dx4zsqgvsw7fe8gl2f77yufynauuhklnnmnjncqcls0tj"
68. ;
69. const
70. useCounter
71. =
72. (
73. )
74. =>
75. {
76. const
77. {
78. address
79. ,
80. getCosmWasmClient
81. ,
82. getSigningCosmWasmClient
83. }
84. =
85. useChain
86. (
87. "neutronlocalnet"
88. ,
89. true
90. )
91. ;
92. const
93. [
94. value
95. ,
96. setValue
97. ]
98. =
99. useState
100. <
101. string
102. |
103. undefined
104.
105. (
106. )
107. ;
108. const
109. fetchValue
110. =
111. useCallback
112. (
113. async
114. (
115. )
116. =>
117. {
118. const
119. client
120. =
121. await
122. getCosmWasmClient
123. (
124. )
125. ;
126. const
127. {
128. current_value

```

```
129. }
130. =
131. (
132. await
133. client
134. .
135. queryContractSmart
136. (
137. CONTRACT_ADDRESS
138. ,
139. {
140. current_value
141. :
142. {
143. }
144. }
145. )
146. )
147. as
148. {
149. current_value
150. :
151. string
152. }
153. ;
154. setValue
155. (
156. current_value
157. )
158. ;
159. }
160. ,
161. [
162. getCosmWasmClient
163. ]
164. )
165. ;
166. const
167. increaseValue
168. =
169. useCallback
170. (
171. async
172. (
173. amount
174. :
175. string
176. )
177. =>
178. {
179. assert
180. (
181. address
182. ,
183. "Address is required"
184. )
185. ;
186. const
187. client
188. =
189. await
190. getSigningCosmWasmClient
191. (
192. )
193. ;
194. const
195. {
196. transactionHash
```

```
197. }
198. =
199. await
200. client
201. .
202. execute
203. (
204. address
205. ,
206. CONTRACT_ADDRESS
207. ,
208. {
209. increase_count
210. :
211. {
212. amount
213. ,
214. }
215. ,
216. }
217. ,
218. "auto"
219. )
220. ;
221. void
222. fetchValue
223. (
224. )
225. ;
226. return
227. transactionHash
228. ;
229. }
230. ,
231. [
232. address
233. ,
234. getSigningCosmWasmClient
235. ,
236. fetchValue
237. ]
238. )
239. ;
240. useEffect
241. (
242. (
243. )
244. =>
245. {
246. void
247. fetchValue
248. (
249. )
250. ;
251. }
252. ,
253. [
254. fetchValue
255. ]
256. )
257. ;
258. return
259. {
260. value
261. ,
262. increaseValue
263. }
264. ;
```

```
265. }
266. ;
267. export
268. const
269. Counter
270. =
271. (
272. )
273. =>
274. {
275. const
276. {
277. address
278. }
279. =
280. useChain
281. (
282. "neutronlocalnet"
283. ,
284. true
285. )
286. ;
287. const
288. {
289. value
290. ,
291. increaseValue
292. }
293. =
294. useCounter
295. (
296. )
297. ;
298. const
299. [
300. amount
301. ,
302. setAmount
303. ]
304. =
305. useState
306. (
307. ""
308. )
309. ;
310. const
311. handleAmountChange
312. =
313. useCallback
314. (
315. (
316. event
317. :
318. React
319. .
320. ChangeEvent
321. <
322. HTMLInputElement
323.
324. )
325. =>
326. {
327. setAmount
328. (
329. event
330. .
331. currentTarget
332. .
```



```
333. value
334. )
335. ;
336. }
337. ,
338. [
339. ]
340. )
341. ;
342. const
343. handleIncreaseClick
344. =
345. useCallback
346. (
347. async
348. (
349. )
350. =>
351. {
352. if
353. (
354. !
355. address
356. ||
357. !
358. amount
359. )
360. return
361. ;
362. const
363. transactionHash
364. =
365. await
366. increaseValue
367. (
368. amount
369. )
370. ;
371. console
372. .
373. log
374. (
375. transactionHash
376. )
377. ;
378. setAmount
379. (
380. ""
381. )
382. ;
383. }
384. ,
385. [
386. address
387. ,
388. amount
389. ,
390. increaseValue
391. ]
392. )
393. ;
394. return
395. (
396. <
397. Card
398. className
399. =
400. "
```

```
401. w-[350px]
402. "
403.

404. <
405. CardHeader
406.

407. <
408. CardTitle
409.

410. Counter contract
411. <!--
412. CardTitle
413. -->
414. <!--
415. CardHeader
416. -->
417. <
418. CardContent
419.

420. <
421. div
422. className
423. =
424. "
425. grid w-full items-center gap-4
426. "
427.

428. <
429. div
430. className
431. =
432. "
433. flex flex-col space-y-1.5
434. "
435.

436. <
437. Label
438. htmlFor
439. =
440. "
441. value
442. "
443.

444. Current value
445. <!--
446. Label
447. -->
448. <
449. Input
450. id
451. =
452. "
453. value
454. "
455. value
456. =
457. {
458. value
459. ??
460. ""
461. }
462. disabled
463. />
```

```
464. <!--
465. div
466. -->
467. <
468. div
469. className
470. =
471. "
472. flex flex-col space-y-1.5
473. "
474.

475. <
476. Label
477. htmlFor
478. =
479. "
480. amount
481. "
482.

483. Amount to increase
484. <!--
485. Label
486. -->
487. <
488. Input
489. id
490. =
491. "
492. amount
493. "
494. type
495. =
496. "
497. number
498. "
499. min
500. =
501. "
502. 1
503. "
504. value
505. =
506. {
507. amount
508. }
509. onChange
510. =
511. {
512. handleAmountChange
513. }
514. />
515. <!--
516. div
517. -->
518. <!--
519. div
520. -->
521. <!--
522. CardContent
523. -->
524. <
525. CardFooter
526. className
527. =
528. "
529. flex justify-between
530. "
```

```
531.
532. <
533. Button
534. disabled
535. =
536. {
537. !
538. address
539. ||
540. !
541. amount
542. }
543. type
544. =
545. "
546. button
547. "
548. onClick
549. =
550. {
551. handleIncreaseClick
552. }
553.

554. Increase
555. <!--
556. Button
557. -->
558. <!--
559. CardFooter
560. -->
561. <!--
562. Card
563. -->
564. )
565. ;
566. }
567. ;
568. AddCounter
569. component tosrc/app/page.tsx
570. :
571. "use client"
572. ;
573. import
574. {
575. Counter
576. }
577. from
578. "@components/Counter"
579. ;
580. import
581. {
582. WalletButton
583. }
584. from
585. "@components/WalletButton"
586. ;
587. export
588. default
589. function
590. Home
591. (
592. )
593. {
594. return
595. (
596. <
597. main
```

```
598. className
599. =
600. "
601. flex flex-col gap-8 items-center justify-center h-screen
602. "
603.
604. <
605. WalletButton
606. />
607. <
608. Counter
609. />
610. <!--
611. main
612. -->
613. )
614. ;
615. }
```

Now you should be able to see the counter value in UI and be able to increase it [Previous Part 2: Calling Modules and Contracts](#) [Next Integration tests for chain](#)