Micro frontend with Webpack Module Federation

**Objective**

To implement data communication across different micro frontends and render multiple micro frontends along with routing structure in the host micro frontend.

**Components**

1. **React** – Javascript library used in all micro frontends in this implementation

2. **Zustand** – state management library supports with react

2. **Webpack** – Javascript bundler which supports module federation

**Execution Overview**

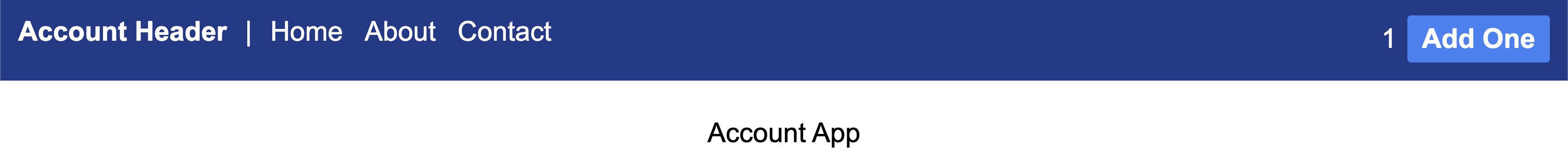
Four micro frontends containing one host and three remotes all using react library are implemented in this example. The three remotes are product, account, assessment. Zustand is integrated to implement data communication across micro frontends. Global state management is handled in host application. Global state and actions to manipulate it are created and exposed from host using module federation for the remotes to use them. The remote apps are properly routed using react-router-dom and their router is exposed from each of them using module federation. The host app utilizes the remote routers to render them in self.

**Data communication**

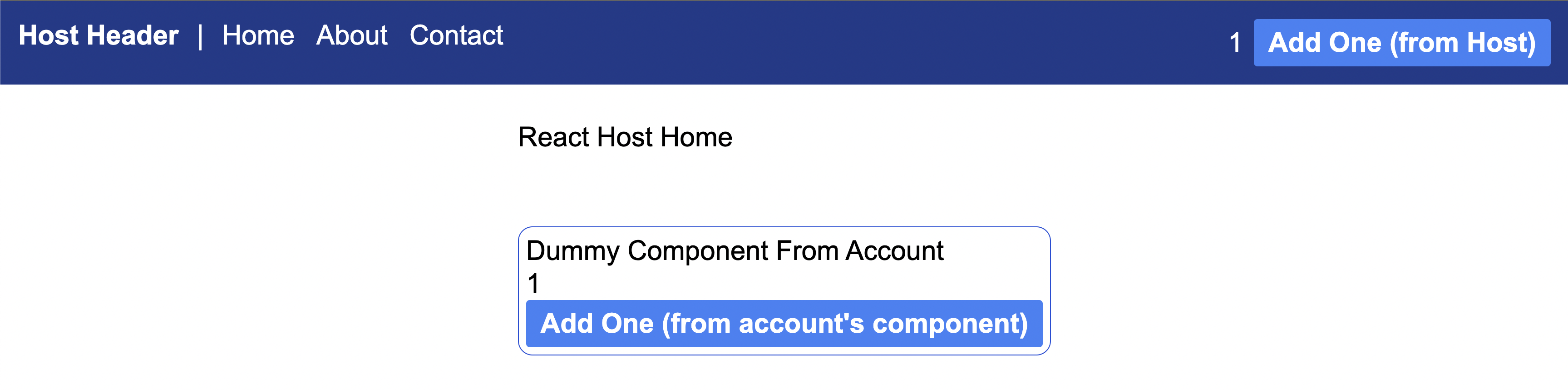
A basic counter store is implemented in the host app. Then exposed in the webpack config.



The counter state and a button to increment the count are rendered in the host header. The same is done in the Account Header using the exposed state and functions from host 



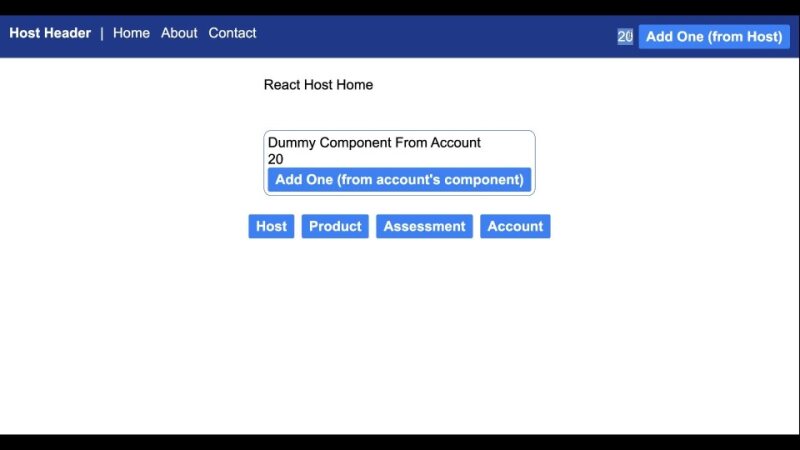
A Dummy component is exposed from account remote app and rendered in host app’s home page



The Dummy component gets the count state and increment function through props and utilizes inside.

**Result**

When either of the "Add One” button is clicked in the Host home page, the state changes and will be synced in both host app and the remote Dummy component. But the state change doesn’t affect the Account Header which is rendered in the `/account` route

[](https://iresponsivesolutions-my.sharepoint.com/:v:/p/dikshitkumar_n/ESd1QuQkrvBMo3oKa9Oc7FkB604YYzANM0mMEcm-VS8j4A?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&e=VCTJ3Y)

**Routing structure**

All the micro frontends have their own routing structure defined in the respective projects. The router is exposed without wrapping it in the remote browser router. Instead, those routers are imported and wrapped inside the host browser router.

**Result**

All the remote micro frontends are structured based on the routing defined in respective routers

**Host Home page at `/`**

**Remote Home page at `/${remote}`**

**Remote sub routes at `/${remote}/...`**