# Vite Micro‑Frontend Starter (Host + Remote) with Routing, Layout, Zustand, Axios)

A production‑ready scaffold for a **Host** app and a **Remote** app using **Vite + vite-plugin-federation**, with:

* React 19 (works fine with this setup)
* **Routing** (react-router-dom)
* **Layout** (Header + Sidebar + Main content area)
* **State** with **Zustand** (auth store)
* **Axios** client with interceptors (shared across host & remotes)
* Shared package for common code (@shared/\*)
* Dev scripts for running both apps

Folder layout assumes a monorepo at ./packages like you already have.

root/  
 package.json # workspaces config (pnpm or npm workspaces)  
 pnpm-workspace.yaml # (if using pnpm)  
 packages/  
 shared/  
 package.json  
 src/  
 api/client.js  
 stores/auth.js  
 index.js  
 host-app/  
 package.json  
 vite.config.js  
 src/  
 main.jsx  
 App.jsx  
 router.jsx  
 layout/Header.jsx  
 layout/Sidebar.jsx  
 layout/Layout.jsx  
 pages/Home.jsx  
 remote-app/  
 package.json  
 vite.config.js  
 src/  
 main.jsx  
 App.jsx  
 pages/DashboardHome.jsx  
 pages/Stats.jsx  
 components/RemoteButton.jsx  
 .env # optional: VITE\_API\_URL, etc.

## Root (monorepo) package.json

{  
 "name": "mfe-monorepo",  
 "private": true,  
 "workspaces": [  
 "packages/\*"  
 ],  
 "devDependencies": {  
 "concurrently": "^9.0.1"  
 },  
 "scripts": {  
 "dev": "concurrently \"npm:dev:remote\" \"npm:dev:host\"",  
 "dev:host": "npm --workspace host-app run dev",  
 "dev:remote": "npm --workspace remote-app run dev"  
 }  
}

If you prefer **pnpm**, add pnpm-workspace.yaml:

packages:  
 - 'packages/\*'

## packages/shared/package.json

{  
 "name": "@shared/core",  
 "version": "0.1.0",  
 "main": "src/index.js",  
 "type": "module"  
}

### packages/shared/src/stores/auth.js

import { create } from 'zustand'  
  
export const useAuthStore = create((set, get) => ({  
 user: null,  
 token: null,  
 login: (user, token) => set({ user, token }),  
 logout: () => set({ user: null, token: null })  
}))

### packages/shared/src/api/client.js

import axios from 'axios'  
import { useAuthStore } from '../stores/auth.js'  
  
export const api = axios.create({  
 baseURL: import.meta.env.VITE\_API\_URL || 'http://localhost:3000',  
 withCredentials: true  
})  
  
// Request interceptor: attach token if present  
api.interceptors.request.use((config) => {  
 const token = useAuthStore.getState().token  
 if (token) {  
 config.headers.Authorization = `Bearer ${token}`  
 }  
 return config  
})  
  
// Response interceptor: simple auth handling  
api.interceptors.response.use(  
 (res) => res,  
 (err) => {  
 if (err?.response?.status === 401) {  
 // Optionally logout and/or trigger refresh flow  
 useAuthStore.getState().logout()  
 }  
 return Promise.reject(err)  
 }  
)

### packages/shared/src/index.js

export \* from './api/client.js'  
export \* from './stores/auth.js'

## Host App

### packages/host-app/package.json

{  
 "name": "host-app",  
 "private": true,  
 "version": "0.0.0",  
 "type": "module",  
 "scripts": {  
 "dev": "vite --port 5000 --strictPort",  
 "build": "vite build",  
 "preview": "vite preview --port 5000 --strictPort"  
 },  
 "dependencies": {  
 "@originjs/vite-plugin-federation": "^1.3.6",  
 "@shared/core": "workspace:\*",  
 "react": "^19.1.1",  
 "react-dom": "^19.1.1",  
 "react-router-dom": "^6.28.0",  
 "zustand": "^5.0.0",  
 "axios": "^1.7.7"  
 },  
 "devDependencies": {  
 "@vitejs/plugin-react-swc": "^4.0.0",  
 "vite": "^7.1.2"  
 }  
}

### packages/host-app/vite.config.js

import { defineConfig } from 'vite'  
import react from '@vitejs/plugin-react-swc'  
import federation from '@originjs/vite-plugin-federation'  
  
function reloadEndpointPlugin() {  
 return {  
 name: 'vite-plugin-reload-endpoint',  
 configureServer(server) {  
 server.middlewares.use((req, res, next) => {  
 if (req.url === '/\_\_fullReload') {  
 server.hot.send({ type: 'full-reload' })  
 res.end('Full reload triggered')  
 } else {  
 next()  
 }  
 })  
 }  
 }  
}  
  
export default defineConfig({  
 plugins: [  
 react(),  
 federation({  
 name: 'host\_app',  
 remotes: {  
 // MUST match the remote name used in remote-app federation `name`  
 remote\_app: 'http://localhost:5001/assets/remoteEntry.js'  
 },  
 shared: {  
 react: { singleton: true, eager: true },  
 'react-dom': { singleton: true, eager: true },  
 'react-router-dom': { singleton: true },  
 zustand: { singleton: true },  
 axios: { singleton: true }  
 }  
 }),  
 reloadEndpointPlugin()  
 ],  
 server: {  
 host: 'localhost',  
 port: 5000,  
 strictPort: true  
 },  
 build: {  
 target: 'esnext',  
 minify: false,  
 cssCodeSplit: false  
 }  
})

### packages/host-app/src/main.jsx

import React from 'react'  
import { createRoot } from 'react-dom/client'  
import { BrowserRouter } from 'react-router-dom'  
import App from './App.jsx'  
  
createRoot(document.getElementById('root')).render(  
 <React.StrictMode>  
 <BrowserRouter>  
 <App />  
 </BrowserRouter>  
 </React.StrictMode>  
)

### packages/host-app/src/App.jsx

import React, { Suspense } from 'react'  
import { useAuthStore } from '@shared/core'  
import { useNavigate, useRoutes } from 'react-router-dom'  
import Layout from './layout/Layout.jsx'  
import Home from './pages/Home.jsx'  
  
// Lazy import remote root app (will handle nested routes under /dashboard)  
const RemoteDashboardApp = React.lazy(() => import('remote\_app/App'))  
  
function RoutesElement() {  
 const routes = useRoutes([  
 { path: '/', element: <Home /> },  
 { path: '/dashboard/\*', element: (  
 <Suspense fallback={<div>Loading Remote…</div>}>  
 <RemoteDashboardApp />  
 </Suspense>  
 )  
 }  
 ])  
 return routes  
}  
  
export default function App() {  
 const { user, login, logout } = useAuthStore()  
 const nav = useNavigate()  
  
 return (  
 <Layout  
 user={user}  
 onLogin={() => { login({ name: 'Safi' }, 'fake-token'); nav('/dashboard') }}  
 onLogout={() => { logout(); nav('/') }}  
 >  
 <RoutesElement />  
 </Layout>  
 )  
}

### packages/host-app/src/layout/Header.jsx

import React from 'react'  
  
export default function Header({ user, onLogin, onLogout }) {  
 return (  
 <header style={{  
 height: 56,  
 display: 'flex',  
 alignItems: 'center',  
 justifyContent: 'space-between',  
 padding: '0 16px',  
 borderBottom: '1px solid #e5e7eb'  
 }}>  
 <strong>Host App</strong>  
 <div>  
 {user ? (  
 <>  
 <span style={{ marginRight: 12 }}>Hello, {user.name}</span>  
 <button onClick={onLogout}>Logout</button>  
 </>  
 ) : (  
 <button onClick={onLogin}>Login</button>  
 )}  
 </div>  
 </header>  
 )  
}

### packages/host-app/src/layout/Sidebar.jsx

import React from 'react'  
import { NavLink } from 'react-router-dom'  
  
const linkStyle = ({ isActive }) => ({  
 display: 'block', padding: '8px 12px', textDecoration: 'none',  
 color: isActive ? '#111827' : '#374151',  
 background: isActive ? '#e5e7eb' : 'transparent',  
 borderRadius: 8  
})  
  
export default function Sidebar() {  
 return (  
 <aside style={{ width: 220, padding: 12, borderRight: '1px solid #e5e7eb' }}>  
 <nav style={{ display: 'grid', gap: 6 }}>  
 <NavLink to="/" style={linkStyle}>Home</NavLink>  
 <NavLink to="/dashboard" style={linkStyle}>Dashboard (Remote)</NavLink>  
 <NavLink to="/dashboard/stats" style={linkStyle}>Dashboard → Stats</NavLink>  
 </nav>  
 </aside>  
 )  
}

### packages/host-app/src/layout/Layout.jsx

import React from 'react'  
import Header from './Header.jsx'  
import Sidebar from './Sidebar.jsx'  
  
export default function Layout({ children, user, onLogin, onLogout }) {  
 return (  
 <div style={{ display: 'grid', gridTemplateRows: '56px 1fr', height: '100vh' }}>  
 <Header user={user} onLogin={onLogin} onLogout={onLogout} />  
 <div style={{ display: 'grid', gridTemplateColumns: '220px 1fr' }}>  
 <Sidebar />  
 <main style={{ padding: 16 }}>{children}</main>  
 </div>  
 </div>  
 )  
}

### packages/host-app/src/pages/Home.jsx

import React from 'react'  
import { api, useAuthStore } from '@shared/core'  
  
export default function Home() {  
 const { user } = useAuthStore()  
 const [message, setMessage] = React.useState('')  
  
 async function ping() {  
 try {  
 const res = await api.get('/ping')  
 setMessage(res.data?.message || 'pong')  
 } catch (e) {  
 setMessage('Request failed')  
 }  
 }  
  
 return (  
 <div>  
 <h1>Home (Host)</h1>  
 <p>User: {user ? user.name : 'Guest'}</p>  
 <button onClick={ping}>Ping API</button>  
 {message && <p style={{ marginTop: 8 }}>API says: {message}</p>}  
 </div>  
 )  
}

## Remote App

### packages/remote-app/package.json

{  
 "name": "remote-app",  
 "private": true,  
 "version": "0.0.0",  
 "type": "module",  
 "scripts": {  
 "dev": "vite --port 5001 --strictPort",  
 "build": "concurrently \"vite build --watch\" \"vite preview --port 5001 --strictPort\"",  
 "preview": "vite preview --port 5001 --strictPort"  
 },  
 "dependencies": {  
 "@originjs/vite-plugin-federation": "^1.3.6",  
 "@shared/core": "workspace:\*",  
 "react": "^19.1.1",  
 "react-dom": "^19.1.1",  
 "react-router-dom": "^6.28.0",  
 "zustand": "^5.0.0",  
 "axios": "^1.7.7"  
 },  
 "devDependencies": {  
 "@vitejs/plugin-react-swc": "^4.0.0",  
 "concurrently": "^9.0.1",  
 "vite": "^7.1.2"  
 }  
}

### packages/remote-app/vite.config.js

import { defineConfig } from 'vite'  
import react from '@vitejs/plugin-react-swc'  
import federation from '@originjs/vite-plugin-federation'  
  
function notifyHostOnRebuild(hostUrl) {  
 return {  
 name: 'vite-plugin-notify-host-on-rebuild',  
 apply(config, { command }) {  
 return Boolean(command === 'build' && config.build?.watch)  
 },  
 async buildEnd(error) {  
 if (!error) {  
 try {  
 await fetch(`${hostUrl}/\_\_fullReload`)  
 } catch (e) {}  
 }  
 }  
 }  
}  
  
export default defineConfig({  
 plugins: [  
 react(),  
 federation({  
 name: 'remote\_app', // IMPORTANT: this is what host imports  
 filename: 'remoteEntry.js',  
 exposes: {  
 './App': './src/App.jsx',  
 './RemoteButton': './src/components/RemoteButton.jsx'  
 },  
 shared: {  
 react: { singleton: true, eager: true },  
 'react-dom': { singleton: true, eager: true },  
 'react-router-dom': { singleton: true },  
 zustand: { singleton: true },  
 axios: { singleton: true }  
 }  
 }),  
 notifyHostOnRebuild('http://localhost:5000')  
 ],  
 server: {  
 host: 'localhost',  
 port: 5001,  
 strictPort: true  
 },  
 build: {  
 target: 'esnext',  
 minify: false,  
 cssCodeSplit: false,  
 watch: {}  
 }  
})

### packages/remote-app/src/main.jsx

import React from 'react'  
import { createRoot } from 'react-dom/client'  
import App from './App.jsx'  
  
createRoot(document.getElementById('root')).render(  
 <React.StrictMode>  
 <App />  
 </React.StrictMode>  
)

### packages/remote-app/src/App.jsx

import React from 'react'  
import { Routes, Route, Link } from 'react-router-dom'  
import { useAuthStore, api } from '@shared/core'  
import DashboardHome from './pages/DashboardHome.jsx'  
import Stats from './pages/Stats.jsx'  
  
export default function RemoteRoot() {  
 const { user } = useAuthStore()  
  
 return (  
 <div>  
 <h2>Remote: Dashboard</h2>  
 <p>User from shared store: {user ? user.name : 'Guest'}</p>  
 <nav style={{ display: 'flex', gap: 12, marginBottom: 12 }}>  
 <Link to="/dashboard">Home</Link>  
 <Link to="/dashboard/stats">Stats</Link>  
 </nav>  
 <Routes>  
 <Route path="/dashboard" element={<DashboardHome />} />  
 <Route path="/dashboard/stats" element={<Stats />} />  
 </Routes>  
 </div>  
 )  
}

### packages/remote-app/src/pages/DashboardHome.jsx

import React from 'react'  
import { api } from '@shared/core'  
  
export default function DashboardHome() {  
 const [data, setData] = React.useState(null)  
  
 React.useEffect(() => {  
 (async () => {  
 try {  
 const res = await api.get('/dashboard/summary')  
 setData(res.data || { cards: 3, alerts: 1 })  
 } catch {  
 setData({ cards: 3, alerts: 1 })  
 }  
 })()  
 }, [])  
  
 return (  
 <div>  
 <h3>Dashboard Home</h3>  
 <pre>{JSON.stringify(data, null, 2)}</pre>  
 </div>  
 )  
}

### packages/remote-app/src/pages/Stats.jsx

import React from 'react'  
  
export default function Stats() {  
 return (  
 <div>  
 <h3>Stats</h3>  
 <p>Some stats coming from the remote app.</p>  
 </div>  
 )  
}

### packages/remote-app/src/components/RemoteButton.jsx

import React from 'react'  
  
export default function RemoteButton() {  
 return <button>Remote Button</button>  
}

## .env (root or per app)

VITE\_API\_URL=http://localhost:3000

## Run it

From the **root**:

# install once at root (npm or pnpm)  
npm install  
  
# run both apps together  
npm run dev

Or start separately:

# Terminal A (Remote)  
npm --workspace remote-app run dev  
  
# Terminal B (Host)  
npm --workspace host-app run dev

Now open http://localhost:5000 and use the sidebar to navigate to **Dashboard**.

* Click **Login** in the header to populate auth store (simulates login). Remote sees the same user via shared store.
* Use **Home → Ping API** to test the Axios client (you can run a mock server at VITE\_API\_URL).

## Notes / Next Steps

* Replace the fake login() with your real auth flow (JWT/refresh) and persist token in localStorage if needed.
* Add ErrorBoundary around the remote mount to isolate failures.
* For production, host should point remotes to CDN URLs:
* remotes: {  
   remote\_app: 'https://cdn.example.com/remote-app/remoteEntry.js'  
  }
* Consider versioning remotes and a manifest loader to support **safe rollout**.
* Add ESLint + Prettier configs across workspace.

**Quick step-by-step (what to do, in order)**

1. **Create the repo root and enable workspaces**
   * mkdir mfe && cd mfe
   * npm init -y
   * Edit package.json to add "workspaces": ["packages/\*"] and the dev script to run both apps (or use the canvas root package.json).
2. **Create the packages folder structure**
   * mkdir -p packages/shared/src packages/host-app/src packages/remote-app/src
   * Inside packages/shared run npm init -y (or create package.json with name @shared/core and "type":"module").
   * Inside packages/host-app and packages/remote-app run npm init -y.
3. **Install dependencies**
   * From repo root (recommended): install root dev tools:
     + npm install --save-dev concurrently
   * For each package install its dependencies (you can run from root using npm --workspace <pkg> install <dep>):
     + Host & Remote common deps: react react-dom react-router-dom zustand axios @originjs/vite-plugin-federation
     + Dev deps (per app): vite @vitejs/plugin-react-swc
     + Example: npm --workspace host-app install react react-dom react-router-dom zustand axios @originjs/vite-plugin-federation and npm --workspace host-app install -D vite @vitejs/plugin-react-swc
   * (Or run npm install from root after you paste the package.json files from the canvas — they use workspace links like @shared/core: workspace:\*.)
4. **Create files**
   * Create the vite.config.js for host and remote (canvas has complete configs).
   * Add the src files:
     + Shared: stores/auth.js, api/client.js, index.js
     + Host: main.jsx, App.jsx, layout/Layout.jsx, layout/Header.jsx, layout/Sidebar.jsx, pages/Home.jsx
     + Remote: main.jsx, App.jsx, pages/DashboardHome.jsx, pages/Stats.jsx, components/RemoteButton.jsx
   * (All file contents are in the canvas document — copy/paste them into your editor.)
5. **Run locally (dev)**
   * From root: npm run dev (this runs both host and remote dev servers via concurrently — see canvas package.json).
   * Or run separately:
     + npm --workspace remote-app run dev
     + npm --workspace host-app run dev
   * Open http://localhost:5000 (host). Navigate to Dashboard — host will lazy-load the remote.
6. **Build / static preview (if you prefer build+preview for remote)**
   * Remote build/watch script in the scaffold runs vite build --watch + vite preview simultaneously. This is useful if you prefer rebuild+serve behavior (but note: rebuilds produce full reload rather than true HMR).
   * For full production builds, run npm --workspace <pkg> run build in each package.

**Answer to your layout question**

You do **NOT** need three separate layout folders. The scaffold uses **one layout/ folder inside the host app** that contains three files:

* layout/Layout.jsx — the main container that arranges Header / Sidebar / Main content.
* layout/Header.jsx — top bar (login, user, branding).
* layout/Sidebar.jsx — navigation links (Home, Dashboard, etc.).

Remotes typically do **not** recreate the global layout. They only render into the main area (the content region). That keeps a single consistent layout controlled by the Host.

**How to deploy apps separately (conceptual + steps)**

1. **Build each app** independently:
   * npm --workspace remote-app run build → produces dist/ (including assets/remoteEntry.js).
   * npm --workspace host-app run build → produces host dist/.
2. **Host remote assets on a static host / CDN / object storage**
   * Upload the remote dist contents to a CDN (e.g., S3 + CloudFront, Netlify, Vercel static build, GitHub Pages, etc.). You’ll get a public URL like https://cdn.example.com/remote-app/remoteEntry.js.
3. **Configure Host remotes for production**
   * Update Host’s vite.config.js for production to point the remote name to the CDN URL:
   * remotes: {
   * remote\_app: 'https://cdn.example.com/remote-app/remoteEntry.js'
   * }
   * Alternatively, use a runtime manifest that maps remote names → URLs to support dynamic updates without rebuilding the host.
4. **Deploy Host**
   * Build the Host and deploy it (Netlify, Vercel, static server). Host will load remoteEntry from the CDN at runtime.
5. **Versioning / safe rollouts**
   * Optionally publish remote builds to versioned URLs (e.g., /.../v1.2.0/remoteEntry.js) and control which version the host loads.
   * For safer rollouts, use a manifest server or feature flags that let you swap remote URLs without redeploying Host.

**Host App Folder Structure**

**npm create vite@latest host-app --template react-ts**

**npm install**

**npm install react-router-dom zustand axios @originjs/vite-plugin-federation**

host-app/

├── public/

│ └── index.html

├── src/

│ ├── components/

│ │ ├── Layout/

│ │ │ ├── Header.tsx

│ │ │ ├── SideNav.tsx

│ │ │ └── MainContent.tsx

│ │ └── RemoteMicrofrontend.tsx

│ ├── context/

│ │ └── StoreContext.tsx

│ ├── store/

│ │ └── store.ts

│ ├── routes/

│ │ └── AppRoutes.tsx

│ ├── api/

│ │ └── axiosClient.ts

│ ├── App.tsx

│ ├── index.tsx

├── vite.config.ts

├── package.json

└── tsconfig.json

* Layout/: Holds the layout and panel components (header, side nav, main content).
* context/StoreContext.tsx: Provides React context for global state sharing (integrated with Zustand).
* store/store.ts: Zustand store implementation.
* routes/AppRoutes.tsx: Centralizes route declarations using React Router.
* api/axiosClient.ts: Configures and exports Axios instance.
* RemoteMicrofrontend.tsx: Dynamically loads and renders federated microfrontend.
* **Remote App Folder Structure**

**remote-app/**

* **├── public/**
* **│ └── index.html**
* **├── src/**
* **│ ├── components/**
* **│ │ ├── RemoteFeature.tsx**
* **│ ├── store/**
* **│ │ └── remoteStore.ts**
* **│ ├── api/**
* **│ │ └── remoteAxios.ts**
* **│ ├── App.tsx**
* **│ ├── index.tsx**
* **├── vite.config.ts**
* **├── package.json**
* **└── tsconfig.json**

**We have to follow this**

**micro-frontend-app/**

**│── package.json # root config (workspaces)**

**│── host-app/ # host container**

**│── remote-app/ # remote micro frontend**

**│── shared/ # shared utilities (zustand store, api client, etc.)**

**Final folder structure  
micro-frontend-app/ # Root Folder (Monorepo)**

**│**

**├── package.json # Root config (workspaces, scripts)**

**├── pnpm-workspace.yaml / # (if using pnpm) or yarn workspaces**

**│**

**├── host-app/ # Host Application (container shell)**

**│ ├── package.json**

**│ ├── vite.config.js**

**│ ├── index.html**

**│ └── src/**

**│ ├── main.jsx**

**│ ├── App.jsx**

**│ ├── routes/ # Host app routes**

**│ │ └── index.jsx**

**│ ├── layout/ # Shared Layout for Host**

**│ │ ├── Header.jsx**

**│ │ ├── Sidebar.jsx**

**│ │ └── Layout.jsx # Combines header/sidebar/content**

**│ ├── store/ # Zustand store(s)**

**│ │ └── useAuthStore.js**

**│ ├── context/ # React Context providers**

**│ │ └── ThemeContext.jsx**

**│ ├── api/ # API calls (Axios wrappers)**

**│ │ └── axiosInstance.js**

**│ └── components/ # Reusable UI Components**

**│ └── Loader.jsx**

**│**

**├── remote-app/ # Remote Microfrontend**

**│ ├── package.json**

**│ ├── vite.config.js**

**│ ├── index.html**

**│ └── src/**

**│ ├── main.jsx**

**│ ├── App.jsx**

**│ ├── components/ # Remote shared components**

**│ │ └── Button.jsx**

**│ ├── pages/ # Remote-specific pages**

**│ │ └── Dashboard.jsx**

**│ ├── store/ # (optional) Local Zustand store**

**│ │ └── useCounterStore.js**

**│ └── api/ # (optional) API helpers**

**│ └── dataService.js**

**│**

**└── shared/ # Optional shared libs (utils, constants)**

**├── constants.js**

**└── utils.js**