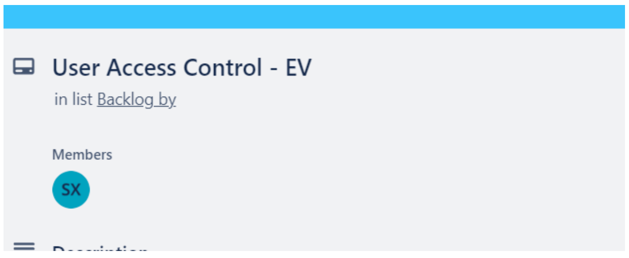
SX

Lead: azmym@deakin.edu.au. Miriam Azmy
Contact: s221071548@deakin.edu.au. AKSHIT SINGH

Task:


User Access Control - EV
in list Backlog by
Members
SX

Infrastructure Security - EV
in list Backlog by
Members
SX

EV react website objectives:

1. Login components
2. Google map API (no API key provided)
3. Firebase.

1. Role base user access control and using state management library work with Firebase E.g. Zustand (see page2 Zustand example). This will partially solve: https://owasp.org/Top10/A01_2021-Broken_Access_Control/

   Note:
   A. using Firebase, setup company collection and user collection, assign each user a role to achieve the role base control.
   B. Firebase rules, within each request, check the role before enable user write the data, here both for EVCFLO and EVOLEON can wrap the component in AuthProvider.
   C. (Optional) Firebase cloud function handles data validation and cleaning to reduce risk of client-side vulnerabilities.
2. Content Security Policy, note these configurations while hosting.
   A. Prevent XSS Attacks: Specify script sources to only allow scripts from trusted domains. Example: script-src 'self' https://trusteddomain.com;.

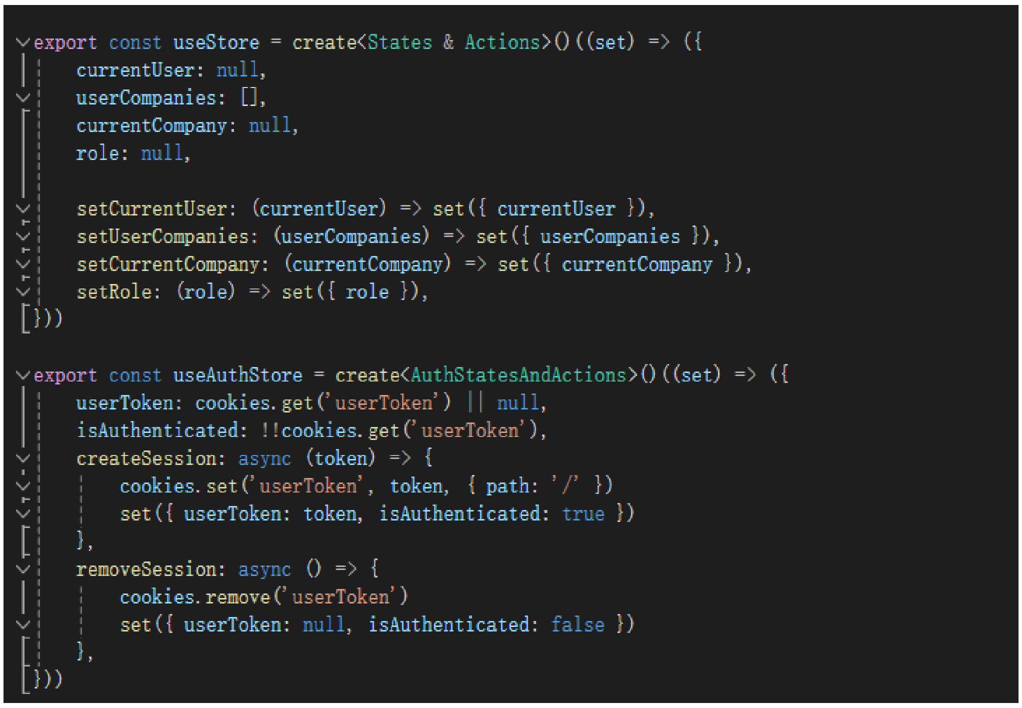
   B. Block Inline Scripts and Styles: Use script-src 'none'; and style-src 'none'; directives to block inline scripts and styles, reducing the risk of XSS.

   C. Restrict Image Sources: Control where images can be loaded from to prevent data exfiltration via image requests. Example: img-src 'self' https://trustedimagesource.com;.

   D. Secure Form Actions: Specify which URLs can be used in form actions to prevent form data from being sent to malicious sites. Example: form-action 'self';.

   E. Prevent Mixed Content: Ensure all content is served over HTTPS by setting block-all-mixed-content;.

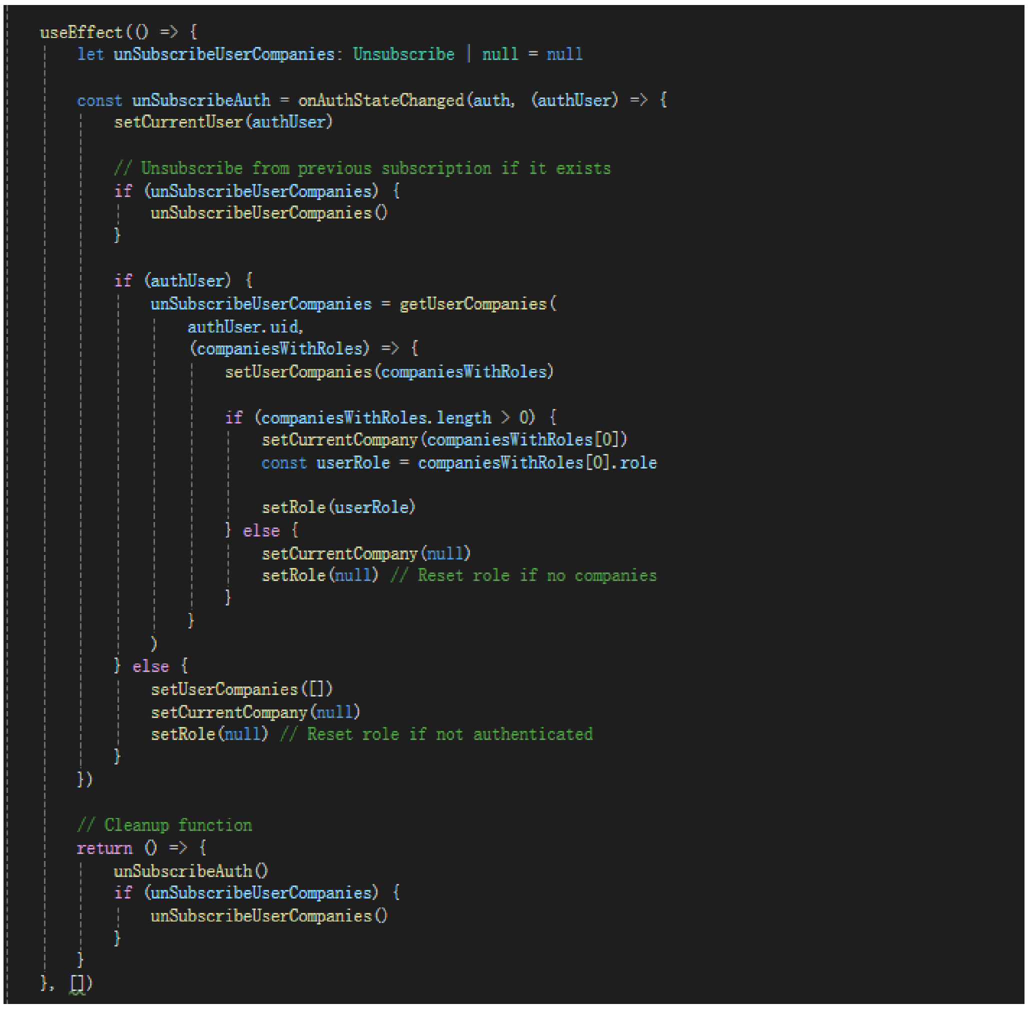
Zustand Example:



```
export const useStore = create<States & Actions>()((set) => ({
    currentUser: null,
    userCompanies: [],
    currentCompany: null,
    role: null,

    setCurrentUser: (currentUser) => set({ currentUser }),
    setUserCompanies: (userCompanies) => set({ userCompanies }),
    setCurrentCompany: (currentCompany) => set({ currentCompany }),
    setRole: (role) => set({ role }),
}))

export const useAuthStore = create<AuthStatesAndActions>()((set) => ({
    userToken: cookies.get('userToken') || null,
    isAuthenticated: !!cookies.get('userToken'),
    createSession: async (token) => {
        cookies.set('userToken', token, { path: '/' })
        set({ userToken: token, isAuthenticated: true })
    },
    removeSession: async () => {
        cookies.remove('userToken')
        set({ userToken: null, isAuthenticated: false })
    },
}))
```



```
  useEffect(() => {
    let unSubscribeUserCompanies: Unsubscribe | null = null

    const unSubscribeAuth = onAuthStateChanged(auth, (authUser) => {
      setCurrentUser(authUser)

      // Unsubscribe from previous subscription if it exists
      if (unSubscribeUserCompanies) {
        unSubscribeUserCompanies()
      }

      if (authUser) {
        unSubscribeUserCompanies = getUserCompanies(
          authUser.uid,
          (companiesWithRoles) => {
            setUserCompanies(companiesWithRoles)

            if (companiesWithRoles.length > 0) {
              setCurrentCompany(companiesWithRoles[0])
              const userRole = companiesWithRoles[0].role

              setRole(userRole)
            } else {
              setCurrentCompany(null)
              setRole(null) // Reset role if no companies
            }
          }
        )
      } else {
        setUserCompanies([])
        setCurrentCompany(null)
        setRole(null) // Reset role if not authenticated
      }
    })

    // Cleanup function
    return () => {
      unSubscribeAuth()
      if (unSubscribeUserCompanies) {
        unSubscribeUserCompanies()
      }
    }
  }, [])
```