```
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";
import { cloneDeep } from "lodash";
import UserService from "services/UserService";

export const getUserBranch = createAsyncThunk(
  "userPermission/getUserBranch",
  async (text, { rejectWithValue }) => {
    try {
      const response = await UserService.getUserBranchPermission(text);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getUserScope = createAsyncThunk(
  "userPermission/getUserScope",
  async (data, { rejectWithValue }) => {
    try {
      const response = await UserService.getUserScopePermissionById(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getUserRole = createAsyncThunk(
  "userPermission/getUserRole",
  async (data, { rejectWithValue }) => {
    try {
      const response = await UserService.getUserRolePermission(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const updateUserBranch = createAsyncThunk(
  "userPermission/updateUserBranch",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, arrData } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await UserService.updateUserBranchPermission(arrData);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const updateUserScope = createAsyncThunk(
  "userPermission/updateUserScope",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, arrData } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await UserService.updateUserScopePermission(arrData);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
```

```
export const updateUserRole = createAsyncThunk(
  "userPermission/updateUserRole",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, arrData } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await UserService.updateUserRolePermission(arrData);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

const initialState = {
  loading: false,
  userBranch: null,
  userScope: null,
  userEmployee: null,
};

export const userPermissionSlice = createSlice({
  name: "userPermission",
  initialState,
  reducers: {
    showLoading: (state) => {
      state.loading = true;
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(getUserBranch.pending, (state) => {
        state.loading = true;
      })
      .addCase(getUserBranch.fulfilled, (state, action) => {
        state.loading = false;
        state.userBranch = action.payload;
      })
      .addCase(getUserBranch.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(updateUserBranch.pending, (state) => {
        state.loading = true;
      })
      .addCase(updateUserBranch.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(updateUserBranch.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getUserScope.pending, (state) => {
        state.loading = true;
      })
      .addCase(getUserScope.fulfilled, (state, action) => {
        state.loading = false;
        state.userScope = action.payload;
      })
      .addCase(getUserScope.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(updateUserScope.pending, (state) => {
        state.loading = true;
      })
      .addCase(updateUserScope.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(updateUserScope.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getUserRole.pending, (state) => {
        state.loading = true;
      })
```

```
      .addCase(getUserRole.fulfilled, (state, action) => {
        state.loading = false;
        state.userEmployee = action.payload;
      })
      .addCase(getUserRole.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(updateUserRole.pending, (state) => {
        state.loading = true;
      })
      .addCase(updateUserRole.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(updateUserRole.rejected, (state, action) => {
        state.loading = false;
      });
  },
});

export const { showLoading } = userPermissionSlice.actions;

export default userPermissionSlice.reducer;
```