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

export const getDSNhapXuat = createAsyncThunk(
  "nhapxuat/getDSNhapXuat",
  async (data, { rejectWithValue }) => {
    try {
      const response = await NhapXuatService.searchGrid(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const getNhapXuatByManx = createAsyncThunk(
  "nhapxuat/getNhapXuatByManx",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await NhapXuatService.getNhapXuatByManx(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
  // async (id, { rejectWithValue }) => {
  //   try {
  //     const response = await NhapXuatService.getNhapXuatByManx(id);
  //     return response.data;
  //   } catch (err) {
  //     return rejectWithValue(err.message || "Error");
  //   }
  // }
);
export const createNhapXuat = createAsyncThunk(
  "nhapxuat/createNhapXuat",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await NhapXuatService.create(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const updateNhapXuat = createAsyncThunk(
  "nhapxuat/updateNhapXuat",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await NhapXuatService.update(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const changeStateNhapXuat = createAsyncThunk(
  "nhapxuat/changeStateNhapXuat",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await NhapXuatService.changeStateNhapXuat(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const deleteNhapXuat = createAsyncThunk(
  "nhapxuat/deleteNhapXuat",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, id } = data;
      const response = await NhapXuatService.deleteNhapXuat(id);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

const initialState = {
  loading: false,
  nhapXuatList: [],
  nhapXuatDetail: {},
};

export const nhapxuatSlice = createSlice({
  name: "nhapxuat",
  initialState,
  reducers: {
    showLoading: (state) => {
      state.loading = true;
    },
    setHisInfo: (state, action) => {
      state.hisInfoList = action.payload;
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(getDSNhapXuat.pending, (state) => {
        state.loading = true;
      })
      .addCase(getDSNhapXuat.fulfilled, (state, action) => {
        state.loading = false;
        state.nhapXuatList = action.payload;
      })
      .addCase(getDSNhapXuat.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getNhapXuatByManx.pending, (state) => {
        state.loading = true;
      })
      .addCase(getNhapXuatByManx.fulfilled, (state, action) => {
        state.loading = false;
        state.nhapXuatDetail = action.payload;
      })
      .addCase(getNhapXuatByManx.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(createNhapXuat.pending, (state) => {
        state.loading = true;
      })
      .addCase(createNhapXuat.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(createNhapXuat.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(updateNhapXuat.pending, (state) => {
        state.loading = true;
      })
      .addCase(updateNhapXuat.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(updateNhapXuat.rejected, (state, action) => {
        state.loading = false;
      });
  },
});

export const { showLoading, setHisInfo } = nhapxuatSlice.actions;

export default nhapxuatSlice.reducer;
```