```
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";
import { cloneDeep } from "lodash";
import XuatKhoService from "services/sale/XuatKhoService";
import { initXuatKhoCt } from "views/app-views/sale/managements/QuanLyKho/XuatKho";

export const getXuatKhoCTByManx = createAsyncThunk(
  "xuatkho/getXuatKhoCTByManx",
  async (data, { rejectWithValue }) => {
    try {
      const response = await XuatKhoService.getXuatKhoCTByManx(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getTonXuatKho = createAsyncThunk(
  "xuatkho/getTonXuatKho",
  async (data, { rejectWithValue }) => {
    try {
      const response = await XuatKhoService.getTonXuatKho(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const addNewXuatKhoCT = createAsyncThunk(
  "xuatkho/addNewXuatKhoCT",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await XuatKhoService.addXuatKhoCT(data);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const updateXuatKhoCT = createAsyncThunk(
  "xuatkho/updateXuatKhoCT",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await XuatKhoService.updateXuatKhoCT(data);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const deleteXuatKhoCT = createAsyncThunk(
  "xuatkho/deleteXuatKhoCT",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const response = await XuatKhoService.deleteXuatKhoCT(data);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
```

```
);

const initialState = {
  loading: false,
  xuatKhoCTList: [],
  tonXuatKhoList: [],
};

export const xuatkhoSlice = createSlice({
  name: "xuatkho",
  initialState,
  reducers: {
    showLoading: (state) => {
      state.loading = true;
    },
    setHisInfo: (state, action) => {
      state.hisInfoList = action.payload;
    },
    setXuatKhoCTList: (state, action) => {
      state.xuatKhoCTList = action.payload;
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(getXuatKhoCTByManx.pending, (state) => {
        state.loading = true;
      })
      .addCase(getXuatKhoCTByManx.fulfilled, (state, action) => {
        state.loading = false;
        state.xuatKhoCTList = [initXuatKhoCt, ...action.payload];
      })
      .addCase(getXuatKhoCTByManx.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getTonXuatKho.pending, (state) => {
        state.loading = true;
      })
      .addCase(getTonXuatKho.fulfilled, (state, action) => {
        state.loading = false;
        state.tonXuatKhoList = action.payload;
      })
      .addCase(getTonXuatKho.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(addNewXuatKhoCT.pending, (state) => {
        state.loading = true;
      })
      .addCase(addNewXuatKhoCT.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(addNewXuatKhoCT.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(updateXuatKhoCT.pending, (state) => {
        state.loading = true;
      })
      .addCase(updateXuatKhoCT.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(updateXuatKhoCT.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(deleteXuatKhoCT.pending, (state) => {
        state.loading = true;
      })
      .addCase(deleteXuatKhoCT.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(deleteXuatKhoCT.rejected, (state, action) => {
        state.loading = false;
      });
  },
});
```

```
export const { showLoading, setHisInfo, setXuatKhoCTList } =
  xuatkhoSlice.actions;

export default xuatkhoSlice.reducer;
```