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

export const getFeedbackFromCustomer = createAsyncThunk(
  "report/getFeedbackFromCustomer",
  async (data, { rejectWithValue }) => {
    try {
      const response = await ReportService.getFeedBackFromCustomer(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const searchDanhGiaChatLuong = createAsyncThunk(
  "report/searchDanhGiaChatLuong",
  async (data, { rejectWithValue }) => {
    try {
      const response = await ReportService.searchDGCL(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const getFeedbackFromCustomerExport = createAsyncThunk(
  "report/getFeedbackFromCustomerExport",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await ReportService.getFeedBackFromCustomerExport(
        payload
      );
      if (onSuccess) onSuccess(response.data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getAllDoctorReport = createAsyncThunk(
  "report/getAllDoctorReport",
  async (_, { rejectWithValue }) => {
    try {
      const response = await ReportService.getAllDoctor();
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const getReportThisHospital = createAsyncThunk(
  "report/getReportThisHospital",
  async (data, { rejectWithValue }) => {
    try {
      const response = await ReportService.getReportThisHospital(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getReportAllHospital = createAsyncThunk(
  "report/getReportAllHospital",
  async (data, { rejectWithValue }) => {
    try {
      const response = await ReportService.getReportAllHospital(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const getReportKcbAllHospital = createAsyncThunk(
  "report/getReportKcbAllHospital",
  async (data, { rejectWithValue }) => {
    try {
      const response = await ReportService.getReportKcbAllHospital(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const getReportKcbHospital = createAsyncThunk(
  "report/getReportKcbHospital",
  async (data, { rejectWithValue }) => {
    try {
      const response = await ReportService.getReportKcbHospital(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
const initialState = {
  loading: false,
  customerList: [],
  danhGiaList: [],
  tieuChiList: [],
  viTriList: [],
  reportHospital: {},
  reportAllHospital: {},
  reportKcbAllHospital: {},
  reportKcbHospital: {},
  doctorList: [],
};

export const reportSlice = createSlice({
  name: "report",
  initialState,
  reducers: {
    showLoading: (state) => {
      state.loading = true;
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(getFeedbackFromCustomer.pending, (state) => {
        state.loading = true;
      })
      .addCase(getFeedbackFromCustomer.fulfilled, (state, action) => {
        state.loading = false;
        state.customerList = action.payload;
      })
      .addCase(getFeedbackFromCustomer.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getFeedbackFromCustomerExport.pending, (state) => {
        state.loading = true;
      })
      .addCase(getFeedbackFromCustomerExport.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(getFeedbackFromCustomerExport.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getReportAllHospital.pending, (state) => {
        state.loading = true;
      })
      .addCase(getReportAllHospital.fulfilled, (state, action) => {
        state.loading = false;
        state.reportAllHospital = action.payload;
      })
      .addCase(getReportAllHospital.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getReportThisHospital.pending, (state) => {
        state.loading = true;
      })
      .addCase(getReportThisHospital.fulfilled, (state, action) => {
        state.loading = false;
        state.reportHospital = action.payload;
      })
      .addCase(getReportThisHospital.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getReportKcbAllHospital.pending, (state) => {
        state.loading = true;
      })
      .addCase(getReportKcbAllHospital.fulfilled, (state, action) => {
        state.loading = false;
        state.reportKcbAllHospital = action.payload;
      })
      .addCase(getReportKcbAllHospital.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getReportKcbHospital.pending, (state) => {
        state.loading = true;
      })
      .addCase(getReportKcbHospital.fulfilled, (state, action) => {
        state.loading = false;
        state.reportKcbHospital = action.payload;
      })
      .addCase(getReportKcbHospital.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(searchDanhGiaChatLuong.pending, (state) => {
        state.loading = true;
      })
      .addCase(searchDanhGiaChatLuong.fulfilled, (state, action) => {
        state.loading = false;
        state.danhGiaList = action.payload.data;
        state.tieuChiList = action.payload.tieuChiList;
        state.viTriList = action.payload.viTriList;
      })
      .addCase(searchDanhGiaChatLuong.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getAllDoctorReport.pending, (state) => {
        state.loading = true;
      })
      .addCase(getAllDoctorReport.fulfilled, (state, action) => {
        state.loading = false;
        state.doctorList = action.payload;
      })
      .addCase(getAllDoctorReport.rejected, (state, action) => {
        state.loading = false;
      });
  },
});

export const { showLoading } = reportSlice.actions;

export default reportSlice.reducer;
```