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

export const getAllDuAn = createAsyncThunk(
  "quanLyDuAn/getAllQuanLyDuAn",
  async (search, { rejectWithValue }) => {
    try {
      const response = await QuanLyDuAnService.GetAllDuAn(search);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const getAllDuAnUser = createAsyncThunk(
  "quanLyDuAn/getAllQuanLyDuAnUser",
  async (search, { rejectWithValue }) => {
    try {
      const response = await QuanLyDuAnService.GetAllDuAnUser(search);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const getAllDuAnBranch = createAsyncThunk(
  "quanLyDuAn/getAllQuanLyDuAnBranch",
  async (search, { rejectWithValue }) => {
    try {
      const response = await QuanLyDuAnService.GetAllDuAnBranch(search);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getDuAn = createAsyncThunk(
  "quanLyDuAn/getDuAn",
  async (id, { rejectWithValue }) => {
    try {
      const response = await QuanLyDuAnService.GetDuAn(id);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const XoaDuAn = createAsyncThunk(
  "quanLyDuAn/XoaDuAn",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await QuanLyDuAnService.XoaDuAn(payload.id);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const UpSertDuAn = createAsyncThunk(
  "quanLyDuAn/UpSertDuAn",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await QuanLyDuAnService.UpSertDuAn(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const DuyetDuAn = createAsyncThunk(
  "quanLyDuAn/duyetDuAn",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await QuanLyDuAnService.DuyetDuAn(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getAllCongViec = createAsyncThunk(
  "quanLyDuAn/getAllCongViec",
  async (boloc, { rejectWithValue }) => {
    try {
      const response = await QuanLyDuAnService.GetAllCongViec(boloc);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getCongViec = createAsyncThunk(
  "quanLyDuAn/getCongViec",
  async (id, { rejectWithValue }) => {
    try {
      const response = await QuanLyDuAnService.GetCongViec(id);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const XoaCongViec = createAsyncThunk(
  "quanLyDuAn/XoaCongViec",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await QuanLyDuAnService.XoaCongViec(data.id);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const UpSertCongViec = createAsyncThunk(
  "quanLyDuAn/UpSertCongViec",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await QuanLyDuAnService.UpSertCongViec(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const XoaCongViecChiTiet = createAsyncThunk(
  "quanLyDuAn/XoaCongViecChiTiet",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await QuanLyDuAnService.XoaCongViecChiTiet(payload.id);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const getAllCongViecChiTiet = createAsyncThunk(
  "quanLyDuAn/getAllCongViecChiTiet",
  async (boloc, { rejectWithValue }) => {
    try {
      const response = await QuanLyDuAnService.GetAllCongViecChiTiet(boloc);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const getAllCongViecChiTietByCV = createAsyncThunk(
  "quanLyDuAn/getAllCongViecChiTietByCV",
  async (boloc, { rejectWithValue }) => {
    try {
      const response = await QuanLyDuAnService.GetAllCongViecChiTietByCV(boloc);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getCongViecChiTiet = createAsyncThunk(
  "quanLyDuAn/getCongViecChiTiet",
  async (id, { rejectWithValue }) => {
    try {
      const response = await QuanLyDuAnService.GetCongViecChiTiet(id);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const GetAllDuAnThanhVien = createAsyncThunk(
  "quanLyDuAn/GetAllDuAnThanhVien",
  async (id, { rejectWithValue }) => {
    try {
      const response = await QuanLyDuAnService.GetAllDuAnThanhVien(id);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const UpSertCongViecChiTiet = createAsyncThunk(
  "quanLyDuAn/UpSertCongViecChiTiet",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await QuanLyDuAnService.UpSertCongViecChiTiet(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const UpDownCongViec = createAsyncThunk(
  "quanLyDuAn/UpDownCongViec",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await QuanLyDuAnService.UpDownCongViec(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const UpDownCongViecChiTiet = createAsyncThunk(
  "quanLyDuAn/UpDownCongViecChiTiet",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await QuanLyDuAnService.UpDownCongViecChiTiet(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
const initialState = {
  loading: false,
  QuanLyDuAnList: [],
  QuanLyDuAnUserList: [],
  QuanLyDuAnBranchList: [],
  QuanLyDuAnDetail: {},
  CongViecList: [],
  CongViecDetail: {},
  CongViecChiTietList: [],
  CongViecChiTietDetail: {},
  GetAllDuAnThanhVienList: [],
  ChiTietCVByCvList: [],
};

export const QuanLyDuAnSlice = createSlice({
  name: "quanLyDuAn",
  initialState,
  reducers: {
    showLoading: (state) => {
      state.loading = true;
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(getAllCongViecChiTietByCV.pending, (state) => {
        state.loading = true;
      })
      .addCase(getAllCongViecChiTietByCV.fulfilled, (state, action) => {
        state.loading = false;
        state.ChiTietCVByCvList = action.payload;
      })
      .addCase(getAllCongViecChiTietByCV.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getAllDuAn.pending, (state) => {
        state.loading = true;
      })
      .addCase(getAllDuAn.fulfilled, (state, action) => {
        state.loading = false;
        state.QuanLyDuAnList = action.payload;
      })
      .addCase(getAllDuAnUser.fulfilled, (state, action) => {
        state.loading = false;
        state.QuanLyDuAnUserList = action.payload;
      })
      .addCase(GetAllDuAnThanhVien.fulfilled, (state, action) => {
        state.loading = false;
        state.GetAllDuAnThanhVienList = action.payload;
      })
      .addCase(getAllDuAnBranch.fulfilled, (state, action) => {
        state.loading = false;
        state.QuanLyDuAnBranchList = action.payload;
      })
      .addCase(getAllDuAn.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getDuAn.fulfilled, (state, action) => {
        state.loading = false;
        state.QuanLyDuAnDetail = action.payload;
      })
      .addCase(getAllCongViec.pending, (state) => {
        state.loading = true;
      })
      .addCase(getAllCongViec.fulfilled, (state, action) => {
        state.loading = false;
        state.CongViecList = action.payload;
      })
      .addCase(getAllCongViec.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getCongViec.fulfilled, (state, action) => {
        state.loading = false;
        state.CongViecDetail = action.payload;
      })
      .addCase(getAllCongViecChiTiet.pending, (state) => {
        state.loading = true;
      })
      .addCase(getAllCongViecChiTiet.fulfilled, (state, action) => {
        state.loading = false;
        state.CongViecChiTietList = action.payload;
      })
      .addCase(getAllCongViecChiTiet.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getCongViecChiTiet.fulfilled, (state, action) => {
        state.loading = false;
        state.CongViecChiTietDetail = action.payload;
      });
  },
});

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

export default QuanLyDuAnSlice.reducer;
```