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

export const SearchDonXinNghi = createAsyncThunk(
  "qlNghiSlice/SearchDonXinNghi",
  async (data, { rejectWithValue }) => {
    try {
      const response = await QuanLyNghiService.searchDonXinNghi(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const GetDonXinNghiDetail = createAsyncThunk(
  "qlNghiSlice/GetDonXinNghiDetail",
  async (id, { rejectWithValue }) => {
    try {
      const response = await QuanLyNghiService.donXinNghiDetail(id);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

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

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

export const SearchNghiPhepNam = createAsyncThunk(
  "qlNghiSlice/SearchNghiPhepNam",
  async (data, { rejectWithValue }) => {
    try {
      const response = await QuanLyNghiService.searchNghiPhepNam(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const AddNewNghiPhepNam = createAsyncThunk(
```

```
  "qlNghiSlice/AddNewNghiPhepNam",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await QuanLyNghiService.addNewNghiPhepNam(payload);
      if (onSuccess) onSuccess(response.data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

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

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

const initialState = {
  loading: false,
  list: [],
  detail: {},
  phepNamList: [],
};

export const qlNghiSlice = createSlice({
  name: "qlNghi",
  initialState,
  reducers: {
    showLoading: (state) => {
      state.loading = true;
    },
    setHisInfo: (state, action) => {
      state.hisInfoList = action.payload;
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(SearchDonXinNghi.pending, (state) => {
        state.loading = true;
      })
      .addCase(SearchDonXinNghi.fulfilled, (state, action) => {
        state.loading = false;
        state.list = action.payload;
      })
```

```
      .addCase(SearchDonXinNghi.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(GetDonXinNghiDetail.pending, (state) => {
        state.loading = true;
      })
      .addCase(GetDonXinNghiDetail.fulfilled, (state, action) => {
        state.loading = false;
        state.detail = action.payload;
      })
      .addCase(GetDonXinNghiDetail.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(UpSertDonXinNghi.pending, (state) => {
        state.loading = true;
      })
      .addCase(UpSertDonXinNghi.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(UpSertDonXinNghi.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(DeleteDonXinNghi.pending, (state) => {
        state.loading = true;
      })
      .addCase(DeleteDonXinNghi.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(DeleteDonXinNghi.rejected, (state, action) => {
        state.loading = false;
      })

      .addCase(SearchNghiPhepNam.pending, (state) => {
        state.loading = true;
      })
      .addCase(SearchNghiPhepNam.fulfilled, (state, action) => {
        state.loading = false;
        state.phepNamList = action.payload;
      })
      .addCase(SearchNghiPhepNam.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(AddNewNghiPhepNam.pending, (state) => {
        state.loading = true;
      })
      .addCase(AddNewNghiPhepNam.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(AddNewNghiPhepNam.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(UpdateNghiPhepNam.pending, (state) => {
        state.loading = true;
      })
      .addCase(UpdateNghiPhepNam.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(UpdateNghiPhepNam.rejected, (state, action) => {
        state.loading = false;
      });
  },
});

export const { showLoading } = qlNghiSlice.actions;

export default qlNghiSlice.reducer;
```