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

export const SearchSangKienYTuong = createAsyncThunk(
  "sangkien/SearchSangKienYTuong",
  async (data, { rejectWithValue }) => {
    try {
      const response = await SangKienYTuongService.search(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const GetSangKienCate = createAsyncThunk(
  "sangkien/GetSangKienCate",
  async (_, { rejectWithValue }) => {
    try {
      const response = await SangKienYTuongService.getCategory();
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

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

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

const initialState = {
  loading: false,
  ideaList: [],
  branchList: [],
  linhVucList: [],
  ngaySinhNhat: 0,
  tongSoDangKy: 0,
  tongSoGui: 0,
};

export const sangKienYTuongSlice = createSlice({
```

```
  name: "sangkien",
  initialState,
  reducers: {
    showLoading: (state) => {
      state.loading = true;
    },
    setRoleDetail: (state) => {
      state.roleDetail = state.payload;
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(SearchSangKienYTuong.pending, (state) => {
        state.loading = false;
      })
      .addCase(SearchSangKienYTuong.fulfilled, (state, action) => {
        state.loading = false;
        state.ideaList = action.payload;
      })
      .addCase(SearchSangKienYTuong.rejected, (state, _action) => {
        state.loading = false;
      })
      .addCase(ExportSangKienYTuong.pending, (state) => {
        state.loading = false;
      })
      .addCase(ExportSangKienYTuong.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(ExportSangKienYTuong.rejected, (state, _action) => {
        state.loading = false;
      })
      .addCase(GetSangKienCate.pending, (state) => {
        state.loading = false;
      })
      .addCase(GetSangKienCate.fulfilled, (state, action) => {
        state.loading = false;
        state.branchList = action.payload.branchList;
        state.linhVucList = action.payload.linhVucList.map((x) => ({
          value: x.id,
          label: x.name,
        }));
        state.ngaySinhNhat = action.payload.ngaySinhNhat;
        state.tongSoDangKy = action.payload.tongSoDangKy;
        state.tongSoGui = action.payload.tongSoGui;
      })
      .addCase(GetSangKienCate.rejected, (state, _action) => {
        state.loading = false;
      })
      .addCase(CreateSangKienYTuong.pending, (state) => {
        state.loading = true;
      })
      .addCase(CreateSangKienYTuong.fulfilled, (state, _action) => {
        state.loading = false;
      })
      .addCase(CreateSangKienYTuong.rejected, (state, _action) => {
        state.loading = false;
      });
  },
});

export const { showLoading, setRoleDetail } = sangKienYTuongSlice.actions;

export default sangKienYTuongSlice.reducer;
```