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

export const searchGridAPI = createAsyncThunk(
  "department/searchGridAPI",
  async (data, { rejectWithValue }) => {
    try {
      const response = await DepartmentService.searchGrid(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getDepartmentById = createAsyncThunk(
  "department/getDepartmentById",
  async (id, { rejectWithValue }) => {
    try {
      const response = await DepartmentService.departmentById(id);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

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

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

export const deleteDepartmentAPI = createAsyncThunk(
  "department/deleteDepartmentAPI",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const response = await DepartmentService.delete(data?.id);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
```

```
  }
);

export const getAllHospital = createAsyncThunk(
  "department/getAllHospital",
  async (id, { rejectWithValue }) => {
    try {
      const response = await DepartmentService.getAllHospital();
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
const initialState = {
  loading: false,
  hisInfoList: [],
  departmentDetail: null,
  hospitalList: []
};

export const departmentSlice = createSlice({
  name: "department",
  initialState,
  reducers: {
    showLoading: (state) => {
      state.loading = true;
    },
    setHisInfo: (state, action) => {
      state.hisInfoList = action.payload;
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(searchGridAPI.pending, (state) => {
        state.loading = true;
      })
      .addCase(searchGridAPI.fulfilled, (state, action) => {
        state.loading = false;
        state.hisInfoList = action.payload;
      })
      .addCase(searchGridAPI.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getDepartmentById.pending, (state) => {
        state.loading = true;
      })
      .addCase(getDepartmentById.fulfilled, (state, action) => {
        state.loading = false;
        state.departmentDetail = action.payload;
      })
      .addCase(getDepartmentById.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(createDepartmentAPI.pending, (state) => {
        state.loading = true;
      })
      .addCase(createDepartmentAPI.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(createDepartmentAPI.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(updateDepartmentAPI.pending, (state) => {
        state.loading = true;
      })
      .addCase(updateDepartmentAPI.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(updateDepartmentAPI.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getAllHospital.pending, (state) => {
        state.loading = true;
```

```
      })
      .addCase(getAllHospital.fulfilled, (state, action) => {
        state.loading = false;
        state.hospitalList = action.payload;
      })
      .addCase(getAllHospital.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(deleteDepartmentAPI.pending, (state) => {
        state.loading = true;
      })
      .addCase(deleteDepartmentAPI.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(deleteDepartmentAPI.rejected, (state, action) => {
        state.loading = false;
      });
  },
});

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

export default departmentSlice.reducer;
```