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

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

export const getHospitalById = createAsyncThunk(
  "hospitalManager/getHospitalById",
  async (id, { rejectWithValue }) => {
    try {
      const response = await HospitalManagerService.getHosiptalById(id);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

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

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

export const deletedHospitalManagement = createAsyncThunk(
  "hospitalManager/deletedHospitalById",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, id } = data;
      const response = await HospitalManagerService.delete(id);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
```

```
  }
);
export const changeStateHosiptalManagement = createAsyncThunk(
  "hospitalManager/changeStateHosiptalManagement",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await HospitalManagerService.changeStateHosiptal(
        payload
      );
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
const initialState = {
  loading: false,
  hospitalList: [],
  hospitalDetail: {},
};

export const hospitalManagementSlice = createSlice({
  name: "hospitalManager",
  initialState,
  reducers: {
    showLoading: (state) => {
      state.loading = true;
    },
    setHisInfo: (state, action) => {
      state.hisInfoList = action.payload;
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(getAllHospitalManagement.pending, (state) => {
        state.loading = true;
      })
      .addCase(getAllHospitalManagement.fulfilled, (state, action) => {
        state.loading = false;
        state.hospitalList = action.payload;
      })
      .addCase(getAllHospitalManagement.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getHospitalById.pending, (state) => {
        state.loading = true;
      })
      .addCase(getHospitalById.fulfilled, (state, action) => {
        state.loading = false;
        state.hospitalDetail = action.payload;
      })
      .addCase(getHospitalById.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(createHospitalManagement.pending, (state) => {
        state.loading = true;
      })
      .addCase(createHospitalManagement.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(createHospitalManagement.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(updateHospitalManagement.pending, (state) => {
        state.loading = true;
      })
      .addCase(updateHospitalManagement.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(updateHospitalManagement.rejected, (state, action) => {
```

```
        state.loading = false;
      })
      .addCase(deletedHospitalManagement.pending, (state) => {
        state.loading = true;
      })
      .addCase(deletedHospitalManagement.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(deletedHospitalManagement.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(changeStateHosiptalManagement.pending, (state) => {
        state.loading = true;
      })
      .addCase(changeStateHosiptalManagement.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(changeStateHosiptalManagement.rejected, (state, action) => {
        state.loading = false;
      });
  },
});

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

export default hospitalManagementSlice.reducer;
```