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

export const getAllHospital = createAsyncThunk(
  "doctor/getAllHospital",
  async (_, { rejectWithValue }) => {
    try {
      const response = await DoctorManagementService.getAllHospital();
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getAllDoctorManagement = createAsyncThunk(
  "doctor/getAllDoctorManagement",
  async (data, { rejectWithValue }) => {
    try {
      const response = await DoctorManagementService.getAllDoctor(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getDoctorById = createAsyncThunk(
  "doctor/getDoctorById",
  async (id, { rejectWithValue }) => {
    try {
      const response = await DoctorManagementService.getDoctorById(id);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

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

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

```
export const changeStateDoctorManagement = createAsyncThunk(
  "doctor/changeStateDoctorManagement",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await DoctorManagementService.changeStateDoctor(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const deletedDoctorManagement = createAsyncThunk(
  "doctor/deletedDoctorById",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, id } = data;
      const response = await DoctorManagementService.delete(id);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const getPatientCommentById = createAsyncThunk(
  "doctor/getPatientCommentById",
  async (id, { rejectWithValue }) => {
    try {
      const response = await DoctorManagementService.getPatientCommentById(id);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const deletePatientCommentById = createAsyncThunk(
  "doctor/deletePatientCommentById",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, id } = data;
      const response = await DoctorManagementService.deletePatientCommentById(
        id
      );
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
const initialState = {
  loading: false,
  doctorList: [],
  doctorDetail: {},
  hospitalList: [],
  patientCommentList: [],
};

export const doctorManagementSlice = createSlice({
  name: "doctor",
  initialState,
  reducers: {
    showLoading: (state) => {
      state.loading = true;
    },
    setHisInfo: (state, action) => {
      state.hisInfoList = action.payload;
    },
  },
  extraReducers: (builder) => {
```

```
builder
  .addCase(getAllDoctorManagement.pending, (state) => {
    state.loading = true;
  })
  .addCase(getAllDoctorManagement.fulfilled, (state, action) => {
    state.loading = false;
    state.doctorList = action.payload;
  })
  .addCase(getAllDoctorManagement.rejected, (state, action) => {
    state.loading = false;
  })
  .addCase(getDoctorById.pending, (state) => {
    state.loading = true;
  })
  .addCase(getDoctorById.fulfilled, (state, action) => {
    state.loading = false;
    state.doctorDetail = action.payload;
  })
  .addCase(getDoctorById.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(createDoctorManagement.pending, (state) => {
    state.loading = true;
  })
  .addCase(createDoctorManagement.fulfilled, (state, action) => {
    state.loading = false;
  })
  .addCase(createDoctorManagement.rejected, (state, action) => {
    state.loading = false;
  })
  .addCase(updateDoctorManagement.pending, (state) => {
    state.loading = true;
  })
  .addCase(updateDoctorManagement.fulfilled, (state, action) => {
    state.loading = false;
  })
  .addCase(updateDoctorManagement.rejected, (state, action) => {
    state.loading = false;
  })
  .addCase(deletedDoctorManagement.pending, (state) => {
    state.loading = true;
  })
  .addCase(deletedDoctorManagement.fulfilled, (state, action) => {
    state.loading = false;
  })
  .addCase(deletedDoctorManagement.rejected, (state, action) => {
    state.loading = false;
  })
  .addCase(changeStateDoctorManagement.pending, (state) => {
    state.loading = true;
  })
  .addCase(changeStateDoctorManagement.fulfilled, (state, action) => {
    state.loading = false;
  })
  .addCase(changeStateDoctorManagement.rejected, (state, action) => {
    state.loading = false;
  })
  .addCase(getPatientCommentById.pending, (state) => {
    state.loading = true;
  })
  .addCase(getPatientCommentById.fulfilled, (state, action) => {
    state.loading = false;
    state.patientCommentList = action.payload;
  })
```

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

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

export default doctorManagementSlice.reducer;
```