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

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

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

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

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

export const deletedAppointment = createAsyncThunk(
  "appointment/deletedAppointmentById",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, id } = data;
      const response = await AppointmentService.delete(id);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
```

```
    }
  }
);
export const changeStateAppointment = createAsyncThunk(
  "appointment/changeStateAppointment",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await AppointmentService.changeStateAppointment(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const getAllHospital = createAsyncThunk(
  "appointment/getAllHospital",
  async (_, { rejectWithValue }) => {
    try {
      const response = await DoctorManagementService.getAllHospital();
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const getAllDoctor = createAsyncThunk(
  "appointment/getAllDoctor",
  async (_, { rejectWithValue }) => {
    try {
      const response = await AppointmentService.getAllDoctor();
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
const initialState = {
  loading: false,
  appointmentList: [],
  appointmentDetail: {},
  hospitalList: [],
  doctorList: [],
};

export const AppointmentSlice = createSlice({
  name: "appointment",
  initialState,
  reducers: {
    showLoading: (state) => {
      state.loading = true;
    },
    setHisInfo: (state, action) => {
      state.hisInfoList = action.payload;
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(getAllAppointment.pending, (state) => {
        state.loading = true;
      })
      .addCase(getAllAppointment.fulfilled, (state, action) => {
        state.loading = false;
        state.appointmentList = action.payload;
      })
      .addCase(getAllAppointment.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getAppointmentById.pending, (state) => {
        state.loading = true;
      })
```

```
      .addCase(getAppointmentById.fulfilled, (state, action) => {
        state.loading = false;
        state.appointmentDetail = action.payload.data;
      })
      .addCase(getAppointmentById.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(createAppointment.pending, (state) => {
        state.loading = true;
      })
      .addCase(createAppointment.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(createAppointment.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(updateAppointment.pending, (state) => {
        state.loading = true;
      })
      .addCase(updateAppointment.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(updateAppointment.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(deletedAppointment.pending, (state) => {
        state.loading = true;
      })
      .addCase(deletedAppointment.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(deletedAppointment.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(changeStateAppointment.pending, (state) => {
        state.loading = true;
      })
      .addCase(changeStateAppointment.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(changeStateAppointment.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(getAllDoctor.pending, (state) => {
        state.loading = true;
      })
      .addCase(getAllDoctor.fulfilled, (state, action) => {
        state.loading = false;
        state.doctorList = action.payload;
      })
      .addCase(getAllDoctor.rejected, (state, action) => {
        state.loading = false;
      });
  },
});

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

export default AppointmentSlice.reducer;
```