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

export const getAllNotification = createAsyncThunk(
  "notification/getAllNotification",
  async (data, { rejectWithValue }) => {
    try {
      const response = await NotificationService.getAllNotification(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

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

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

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

export const deletedNotification = createAsyncThunk(
  "notification/deletedHNotificationById",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, id } = data;
      const response = await NotificationService.delete(id);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

const initialState = {
  loading: false,
  notificationList: [],
  notificationDetail: {},
};

export const NotificationSlice = createSlice({
  name: "notification",
  initialState,
  reducers: {
    showLoading: (state) => {
      state.loading = true;
    },
    setHisInfo: (state, action) => {
      state.hisInfoList = action.payload;
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(getAllNotification.pending, (state) => {
        state.loading = true;
      })
      .addCase(getAllNotification.fulfilled, (state, action) => {
        state.loading = false;
        state.notificationList = action.payload;
      })
      .addCase(getAllNotification.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getNotificationById.pending, (state) => {
        state.loading = true;
      })
      .addCase(getNotificationById.fulfilled, (state, action) => {
        state.loading = false;
        state.notificationDetail = action.payload;
      })
      .addCase(getNotificationById.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(updateNotification.pending, (state) => {
        state.loading = true;
      })
      .addCase(updateNotification.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(updateNotification.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(createNotification.pending, (state) => {
        state.loading = true;
      })
      .addCase(createNotification.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(createNotification.rejected, (state, action) => {
        state.loading = false;
      });
  },
});

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

export default NotificationSlice.reducer;
```