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

export const getAllPosts = createAsyncThunk(
  "posts/getAllPosts",
  async (data, { rejectWithValue }) => {
    try {
      const response = await PostsService.searchGridPosts(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getPostsById = createAsyncThunk(
  "posts/getPostsById",
  async (Id, { rejectWithValue }) => {
    try {
      const response = await PostsService.postsById(Id);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const addNewPosts = createAsyncThunk(
  "posts/addNewPosts",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await PostsService.createPosts(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const updatePostsApi = createAsyncThunk(
  "posts/updatePostsApi",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await PostsService.updatePosts(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const deletePostsApi = createAsyncThunk(
  "posts/deletePostsApi",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, id } = data;
      const response = await PostsService.deletePosts(id);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

const initialState = {
  loading: false,
  postsList: [],
  postsDetail: {},
};

export const postsSlice = createSlice({
  name: "posts",
  initialState,
  reducers: {
    showLoading: (state) => {
      state.loading = true;
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(getAllPosts.pending, (state) => {
        state.loading = true;
      })
      .addCase(getAllPosts.fulfilled, (state, action) => {
        state.loading = false;
        state.postsList = action.payload;
      })
      .addCase(getAllPosts.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getPostsById.pending, (state) => {
        state.loading = true;
      })
      .addCase(getPostsById.fulfilled, (state, action) => {
        state.loading = false;
        state.postsDetail = action.payload;
      })
      .addCase(getPostsById.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(addNewPosts.pending, (state) => {
        state.loading = true;
      })
      .addCase(addNewPosts.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(addNewPosts.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(updatePostsApi.pending, (state) => {
        state.loading = true;
      })
      .addCase(updatePostsApi.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(updatePostsApi.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(deletePostsApi.pending, (state) => {
        state.loading = true;
      })
      .addCase(deletePostsApi.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(deletePostsApi.rejected, (state, action) => {
        state.loading = false;
      });
  },
});

export const { showLoading, setRoleDetail } = postsSlice.actions;

export default postsSlice.reducer;
```