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

export const getRegulationManage = createAsyncThunk(
  "regulationManage/getRegulationManage",
  async (data, { rejectWithValue }) => {
    try {
      const response = await regulationService.GetManage(data);
      return response.data;
    } catch (err) {
      console.error("API call failed:", err);
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getRegulationManageById = createAsyncThunk(
  "regulationManage/getRegulationManageById",
  async (data, { rejectWithValue }) => {
    const { onSuccess, id } = data;
    try {
      const response = await regulationService.GetManageById(id);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.message || "Error");
    }
  }
);

export const updateRegulationManage = createAsyncThunk(
  "regulationManage/updateRegulationManage",
  async (data, { rejectWithValue, dispatch }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await regulationService.UpdateManage(data);
      if (onSuccess) onSuccess(response);

      dispatch(getRegulationManageById({ id: response.data }));

      return response.data;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.message || "Error");
    }
  }
);

export const createRegulationManage = createAsyncThunk(
  "regulationManage/createRegulationManage",
  async (data, { rejectWithValue, dispatch }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await regulationService.CreateManage(data);
      if (onSuccess) onSuccess(response.data);

      dispatch(getRegulationManageById({ id: response.data?.id }));
      return response.data;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.description || "Error");
    }
  }
);
```

```
export const deleteRegulationManage = createAsyncThunk(
  "regulationManage/deleteRegulationManage",
  async (data, { rejectWithValue }) => {
    const { onSuccess, id } = data;
    try {
      const response = await regulationService.DeleteManage(id);
      if (onSuccess) onSuccess(response);
      return id;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.message || "Error");
    }
  }
);

const initialState = {
  regulationManageList: {
    data: [],
    loading: false,
  },
  regulationManage: {
    data: null,
    loading: false,
  },
  initFormREG: {
    name: null,
    number: null,
    createdAt: null,
    danhMucId: null,
    description: null,
  },
};

const regulationManageSlice = createSlice({
  name: "regulationManage",
  initialState,
  reducers: {
    updateFormREG: (state, action) => {
      state.initFormREG = {
        ...state.initFormREG,
        ...action.payload,
      };
    },
    resetFormREG: (state) => {
      state.initFormREG = initialState.initFormREG;
    },
    resetRMDetail: (state) => {
      state.regulationManage = initialState.regulationManage;
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(getRegulationManage.pending, (state) => {
        state.regulationManageList = {
          ...state.regulationManageList,
          loading: true,
        };
      })
      .addCase(getRegulationManage.fulfilled, (state, action) => {
        state.regulationManageList = {
          loading: false,
          data: action.payload,
        };
      })
      .addCase(getRegulationManage.rejected, (state) => {
        state.regulationManageList = {
          ...state.regulationManageList,
          loading: false,
        };
      })
      .addCase(getRegulationManageById.pending, (state) => {
        state.regulationManage = { ...state.regulationManage, loading: true };
      })
      .addCase(getRegulationManageById.fulfilled, (state, action) => {
        state.regulationManage = { data: action.payload, loading: false };
      })
      .addCase(getRegulationManageById.rejected, (state, action) => {
        state.regulationManage = { ...state.regulationManage, loading: false };
      })
      .addCase(updateRegulationManage.pending, (state) => {
        state.regulationManageList = {
          ...state.regulationManageList,
          loading: true,
        };
      })
      .addCase(updateRegulationManage.fulfilled, (state) => {
        state.regulationManageList = {
          ...state.regulationManageList,
          loading: false,
        };
      })
      .addCase(updateRegulationManage.rejected, (state) => {
        state.regulationManageList = {
          ...state.regulationManageList,
          loading: false,
        };
      })
      .addCase(createRegulationManage.pending, (state) => {
        state.regulationManageList = {
          ...state.regulationManageList,
          loading: true,
        };
      })
      .addCase(createRegulationManage.fulfilled, (state) => {
        state.regulationManageList = {
          ...state.regulationManageList,
          loading: false,
        };
      })
      .addCase(createRegulationManage.rejected, (state) => {
        state.regulationManageList = {
          ...state.regulationManageList,
          loading: false,
        };
      })
      .addCase(deleteRegulationManage.pending, (state) => {
        state.regulationManageList = {
          ...state.regulationManageList,
          loading: true,
        };
      })
      .addCase(deleteRegulationManage.fulfilled, (state, action) => {
        state.regulationManageList = {
          ...state.regulationManageList,
          loading: false,
        };
      })
      .addCase(deleteRegulationManage.rejected, (state) => {
        state.regulationManageList = {
          ...state.regulationManageList,
          loading: false,
        };
      });
  },
});

export const { updateFormREG, resetFormREG, resetRMDetail } =
  regulationManageSlice.actions;

export default regulationManageSlice.reducer;
```