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

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

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

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

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

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

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

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

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

const initialState = {
  documentManageList: {
    data: [],
    loading: false,
  },
  documentManage: {
    data: null,
    loading: false,
  },
  initFormDOC: {
    name: null,
    number: null,
    createdAt: null,
    danhMucId: null,
    description: null,
  },
};

const documentManageSlice = createSlice({
  name: "documentManage",
  initialState,
  reducers: {
    updateFormDOC: (state, action) => {
      state.initFormDOC = {
        ...state.initFormDOC,
        ...action.payload,
      };
    },
    resetFormDOC: (state) => {
      state.initFormDOC = initialState.initFormDOC;
    },
    resetDMDetail: (state) => {
      state.documentManage = initialState.documentManage;
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(getDocumentManage.pending, (state) => {
        state.documentManageList = {
          ...state.documentManageList,
          loading: true,
        };
      })
      .addCase(getDocumentManage.fulfilled, (state, action) => {
        state.documentManageList = {
          loading: false,
          data: action.payload,
        };
      })
      .addCase(getDocumentManage.rejected, (state) => {
        state.documentManageList = {
          ...state.documentManageList,
          loading: false,
        };
      })
      .addCase(getDocumentManageById.pending, (state) => {
        state.documentManage = { ...state.documentManage, loading: true };
      })
      .addCase(getDocumentManageById.fulfilled, (state, action) => {
```

```
        state.documentManage = { data: action.payload, loading: false };
      })
      .addCase(getDocumentManageById.rejected, (state, action) => {
        state.documentManage = { ...state.documentManage, loading: false };
      })
      .addCase(updateDocumentManage.pending, (state) => {
        state.documentManageList = {
          ...state.documentManageList,
          loading: true,
        };
      })
      .addCase(updateDocumentManage.fulfilled, (state) => {
        state.documentManageList = {
          ...state.documentManageList,
          loading: false,
        };
      })
      .addCase(updateDocumentManage.rejected, (state) => {
        state.documentManageList = {
          ...state.documentManageList,
          loading: false,
        };
      })
      .addCase(createDocumentManage.pending, (state) => {
        state.documentManageList = {
          ...state.documentManageList,
          loading: true,
        };
      })
      .addCase(createDocumentManage.fulfilled, (state) => {
        state.documentManageList = {
          ...state.documentManageList,
          loading: false,
        };
      })
      .addCase(createDocumentManage.rejected, (state) => {
        state.documentManageList = {
          ...state.documentManageList,
          loading: false,
        };
      })
      .addCase(deleteDocumentManage.pending, (state) => {
        state.documentManageList = {
          ...state.documentManageList,
          loading: true,
        };
      })
      .addCase(deleteDocumentManage.fulfilled, (state, action) => {
        state.documentManageList = {
          ...state.documentManageList,
          loading: false,
        };
      })
      .addCase(deleteDocumentManage.rejected, (state) => {
        state.documentManageList = {
          ...state.documentManageList,
          loading: false,
        };
      });
  },
});

export const { updateFormDOC, resetFormDOC, resetDMDetail } =
  documentManageSlice.actions;

export default documentManageSlice.reducer;
```