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

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

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

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

export const updateDocumentSetting = createAsyncThunk(
  "regulationDocumentSetting/updateDocumentSetting",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await documentService.UpdateSetting(data);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.message || "Error");
    }
  }
);
export const updateRegulationSetting = createAsyncThunk(
  "regulationDocumentSetting/updateRegulationSetting",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await regulationService.UpdateSetting(data);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.message || "Error");
    }
  }
);

export const createDocumentSetting = createAsyncThunk(
  "regulationDocumentSetting/createDocumentSetting",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await documentService.CreateSetting(data);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.message || "Error");
    }
  }
);
export const createRegulationSetting = createAsyncThunk(
  "regulationDocumentSetting/createRegulationSetting",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await regulationService.CreateSetting(data);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.message || "Error");
    }
  }
);

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

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

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

const initialState = {

  nhomQuyCheList: [],
  loading: false,
  documentCatalog: {
    data: [],
    setting: [],
    loading: false,
  },
  regulationCatalog: {
    data: [],
    setting: [],
    loading: false,
  },
};

const regulationDocumentSettingSlice = createSlice({
  name: "regulationDocumentSetting",
  initialState,
  reducers: {},
  extraReducers: (builder) => {
    builder
      .addCase(getDocumentSetting.pending, (state) => {
        state.documentCatalog = {
          ...state.documentCatalog,
          loading: true,
        };
      })
      .addCase(getDocumentSetting.fulfilled, (state, action) => {
        const payload = action.payload || [];
        const filteredPayload = payload.filter(
          (item) => item.inActive === false
        );
        state.documentCatalog = {
          loading: false,
          data: filteredPayload,
          setting: [
            {
              action: "initial",
              isRequired: true,
            },
            ...payload,
          ],
        };
      })
      .addCase(getDocumentSetting.rejected, (state) => {
        state.documentCatalog = {
          ...state.documentCatalog,
          loading: false,
        };
      })
      .addCase(getDsNhomQuyChe.pending, (state) => {
        state.loading = true;
      })
      .addCase(getDsNhomQuyChe.fulfilled, (state, action) => {
        state.loading = false;
        state.nhomQuyCheList = action.payload;
      })
      .addCase(getDsNhomQuyChe.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(updateDocumentSetting.pending, (state) => {
        state.documentCatalog = {
          ...state.documentCatalog,
          loading: true,
        };
      })
      .addCase(updateDocumentSetting.fulfilled, (state, action) => {
        state.documentCatalog = {
          loading: false,
          // data: state.documentCatalog.data.map((item) => {
          //   if (item.id === action.payload.id) {
          //     return action.payload;
          //   }
          //   return item;
          // }),
          // setting: state.documentCatalog.setting.map((item) => {
          //   if (item.id === action.payload.id) {
          //     return action.payload;
          //   }
          //   return item;
          // }),
        };
      })
      .addCase(updateDocumentSetting.rejected, (state, action) => {
        state.documentCatalog = {
          ...state.documentCatalog,
          loading: false,
        };
      })
      .addCase(createDocumentSetting.pending, (state) => {
        state.documentCatalog = {
          ...state.documentCatalog,
          loading: true,
        };
      })
      .addCase(createDocumentSetting.fulfilled, (state, action) => {
        state.documentCatalog = {
          loading: false,
          // data: [action.payload, ...state.documentCatalog.data],
          setting: [
            {
              action: "initial",
              isRequired: true,
            },
            action.payload,
            ...state.documentCatalog.setting.slice(1),
          ],
        };
      })
      .addCase(createDocumentSetting.rejected, (state, action) => {
        state.documentCatalog = {
          ...state.documentCatalog,
          loading: false,
        };
      })
      .addCase(deleteDocumentSetting.pending, (state) => {
        state.documentCatalog = {
          ...state.documentCatalog,
          loading: true,
        };
      })
      .addCase(deleteDocumentSetting.fulfilled, (state, action) => {
        state.documentCatalog = {
          loading: false,
          // data: state.documentCatalog.data.filter((item) => {
          //   return item.id !== action.payload;
          // }),
          // setting: state.documentCatalog.setting.filter((item) => {
          //   return item.id !== action.payload;
          // }),
        };
      })
      .addCase(deleteDocumentSetting.rejected, (state) => {
        state.documentCatalog = {
          ...state.documentCatalog,
          loading: false,
        };
      })
      .addCase(deleteRegulationGroup.pending, (state) => {
        state.documentCatalog = {
          ...state.documentCatalog,
          loading: true,
        };
      })
      .addCase(deleteRegulationGroup.fulfilled, (state, action) => {
        state.documentCatalog = {
          loading: false,
          // data: state.documentCatalog.data.filter((item) => {
          //   return item.id !== action.payload;
          // }),
          // setting: state.documentCatalog.setting.filter((item) => {
          //   return item.id !== action.payload;
          // }),
        };
      })
      .addCase(deleteRegulationGroup.rejected, (state) => {
        state.documentCatalog = {
          ...state.documentCatalog,
          loading: false,
        };
      })
      //REGULATION START

      // .addCase(getRegulationSetting.pending, (state) => {
      //   state.regulationCatalog = {
      //     ...state.regulationCatalog,
      //     loading: true,
      //   };
      // })
      // .addCase(getRegulationSetting.fulfilled, (state, action) => {
      //   const payload = action.payload || [];
      //   const filteredPayload = payload.filter(
      //     (item) => item.inActive === false
      //   );
      //   state.regulationCatalog = {
      //     loading: false,
      //     data: filteredPayload,
      //     setting: [
      //       {
      //         action: "initial",
      //         isRequired: true,
      //       },
      //       ...payload,
      //     ],
      //   };
      // })
      // .addCase(getRegulationSetting.rejected, (state) => {
      //   state.regulationCatalog = {
      //     ...state.regulationCatalog,
      //     loading: false,
      //   };
      // })
      // .addCase(updateRegulationSetting.pending, (state) => {
      //   state.regulationCatalog = {
      //     ...state.regulationCatalog,
      //     loading: true,
      //   };
      // })
      // .addCase(updateRegulationSetting.fulfilled, (state, action) => {
      //   state.regulationCatalog = {
      //     loading: false,
      //     data: state.regulationCatalog.data.map((item) => {
      //       if (item.id === action.payload.id) {
      //         return action.payload;
      //       }
      //       return item;
      //     }),
      //     setting: state.regulationCatalog.setting.map((item) => {
      //       if (item.id === action.payload.id) {
      //         return action.payload;
      //       }
      //       return item;
      //     }),
      //   };
      // })
      // .addCase(updateRegulationSetting.rejected, (state, action) => {
      //   state.regulationCatalog = {
      //     ...state.regulationCatalog,
      //     loading: false,
      //   };
      // })
      // .addCase(createRegulationSetting.pending, (state) => {
      //   state.regulationCatalog = {
      //     ...state.regulationCatalog,
      //     loading: true,
      //   };
      // })
      // .addCase(createRegulationSetting.fulfilled, (state, action) => {
      //   state.regulationCatalog = {
      //     loading: false,
      //     data: [action.payload, ...state.regulationCatalog.data],
      //     setting: [
      //       {
      //         action: "initial",
      //         isRequired: true,
      //       },
      //       action.payload,
      //       ...state.regulationCatalog.setting.slice(1),
      //     ],
      //   };
      // })
      // .addCase(createRegulationSetting.rejected, (state) => {
      //   state.regulationCatalog = {
      //     ...state.regulationCatalog,
      //     loading: false,
      //   };
      // })
      // .addCase(deleteRegulationSetting.pending, (state) => {
      //   state.regulationCatalog = {
      //     ...state.regulationCatalog,
      //     loading: true,
      //   };
      // })
      // .addCase(deleteRegulationSetting.fulfilled, (state, action) => {
      //   state.regulationCatalog = {
      //     loading: false,
      //     data: state.regulationCatalog.data.filter((item) => {
      //       return item.id !== action.payload;
      //     }),
      //     setting: state.regulationCatalog.setting.filter((item) => {
      //       return item.id !== action.payload;
      //     }),
      //   };
      // })
      // .addCase(deleteRegulationSetting.rejected, (state) => {
      //   state.regulationCatalog = {
      //     ...state.regulationCatalog,
      //     loading: false,
      //   };
      // });
  },
});

export default regulationDocumentSettingSlice.reducer;
```