```
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";
import employmentContractService from "services/EmploymentContractService";
import { cloneDeep } from "lodash";
// Async thunk to fetch contract settings
export const getContractList = createAsyncThunk(
  "contractSlice/getContractList",
  async (data, { rejectWithValue }) => {
    try {
      const response = await employmentContractService.GetContractList(data);
      return response.data;
    } catch (err) {
      console.error("API call failed:", err);
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const updateContractList = createAsyncThunk(
  "contractSlice/updateContractList",
  async (data, { rejectWithValue, dispatch }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await employmentContractService.UpdateContractList(data);
      if (onSuccess) onSuccess(response);

      dispatch(getContractById({ id: response.data }));
      return response.data;
    } catch (err) {
      console.error("API call failed:", err);
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const createContractList = createAsyncThunk(
  "contractSlice/createContractList",
  async (data, { rejectWithValue, dispatch }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await employmentContractService.CreateContractList(data);
      if (onSuccess) onSuccess(response.data);

      dispatch(getContractById({ id: response.data?.id }));
      return response.data;
    } catch (err) {
      console.error("API call failed:", err);
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const deleteContractList = createAsyncThunk(
  "contractSlice/deleteContractList",
  async (data, { rejectWithValue }) => {
    const { onSuccess, id } = data;
    try {
      const response = await employmentContractService.DeleteContract(id);
      if (onSuccess) onSuccess(response);
      return id;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.message || "Error");
    }
  }
);
export const stopContractList = createAsyncThunk(
  "contractSlice/stopContractList",
  async (data, { rejectWithValue, dispatch }) => {
    const { onSuccess } = data;
    try {
```

```
      const response = await employmentContractService.StopContract(data);
      if (onSuccess) onSuccess(response);
      dispatch(getContractById({ id: response.data }));
      return response.data;
    } catch (error) {
      console.error("API call failed:", error);
      return rejectWithValue(error.message || "Error");
    }
  }
);

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

const initialState = {
  loading: false,
  contractList: [],
  initForm: {
    employeeId: null,
    ngayKy: null,
    ngayHieuLuc: null,
    contractTypeId: null,
    ngayHetHan: null,
    comment: null,
  },
  contractDetail: {
    data: null,
    loading: false,
  },
};

const contractSlice = createSlice({
  name: "contract",
  initialState,
  reducers: {
    updateInitForm: (state, action) => {
      state.initForm = { ...state.initForm, ...action.payload };
    },
    resetInitForm: (state) => {
      state.initForm = initialState.initForm;
    },
    resetContractDetail: (state) => {
      state.contractDetail = initialState.contractDetail;
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(getContractList.pending, (state) => {
        state.loading = true;
      })
      .addCase(getContractList.fulfilled, (state, action) => {
        state.loading = false;
        state.contractList = action.payload.result;
      })
      .addCase(getContractList.rejected, (state) => {
        state.loading = false;
      })

      .addCase(updateContractList.pending, (state) => {
        state.loading = true;
      })
      .addCase(updateContractList.fulfilled, (state) => {
```

```
        state.loading = false;
      })
      .addCase(updateContractList.rejected, (state) => {
        state.loading = false;
      })

      .addCase(createContractList.pending, (state) => {
        state.loading = true;
      })
      .addCase(createContractList.fulfilled, (state) => {
        state.loading = false;
      })
      .addCase(createContractList.rejected, (state) => {
        state.loading = false;
      })
      .addCase(stopContractList.pending, (state) => {
        state.loading = true;
      })
      .addCase(stopContractList.fulfilled, (state) => {
        state.loading = false;
      })
      .addCase(stopContractList.rejected, (state) => {
        state.loading = false;
      })
      .addCase(deleteContractList.pending, (state) => {
        state.loading = true;
      })
      .addCase(deleteContractList.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(deleteContractList.rejected, (state, action) => {
        state.loading = false;
      })

      .addCase(getContractById.pending, (state) => {
        state.loading = true;
        state.contractDetail = { ...state.contractDetail, loading: true };
      })
      .addCase(getContractById.fulfilled, (state, action) => {
        state.loading = false;
        state.contractDetail = { data: action.payload, loading: false };
      })
      .addCase(getContractById.rejected, (state, action) => {
        state.loading = false;
        state.contractDetail = { ...state.contractDetail, loading: false };
      });
  },
});
export const { updateInitForm, resetInitForm, resetContractDetail } =
  contractSlice.actions;

export default contractSlice.reducer;
```