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

export const searchGridAPI = createAsyncThunk(
  "employee/searchGridAPI",
  async (data, { rejectWithValue }) => {
    try {
      const response = await EmployeeService.searchGrid(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getAllEmployee = createAsyncThunk(
  "employee/getAllEmployee",
  async (data, { rejectWithValue }) => {
    try {
      const response = await EmployeeService.searchEmployees(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getAllEmployeeSignProcess = createAsyncThunk(
  "employee/getAllEmployeeSignProcess",
  async (data, { rejectWithValue }) => {
    try {
      const response = await EmployeeService.searchEmployees(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getDetailAPI = createAsyncThunk(
  "employee/getDetailAPI",
  async (id, { rejectWithValue }) => {
    try {
      const response = await EmployeeService.detail(id);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getEmployeeById = createAsyncThunk(
  "employee/getEmployeeById",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, id } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await EmployeeService.detail(id);
      if (onSuccess) onSuccess(response.data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const createEmployeeAPI = createAsyncThunk(
  "employee/createEmployeeAPI",
  async (data, { rejectWithValue }) => {
```

```
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await EmployeeService.create(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const exportExcelEmployee = createAsyncThunk(
  "employee/exportExcelEmployee",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await EmployeeService.exportExcel(payload);
      if (onSuccess) onSuccess(response.data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const importExcelEmployee = createAsyncThunk(
  "employee/importExcelEmployee",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, formData } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await EmployeeService.importExcel(formData);
      if (onSuccess) onSuccess(response.data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const importDsnvEmployee = createAsyncThunk(
  "employee/importDsnvEmployee",
  async (transId, { rejectWithValue }) => {
    try {
      const response = await EmployeeService.importDsnv(transId);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const importHistoryEmployee = createAsyncThunk(
  "employee/importHistoryEmployee",
  async (data, { rejectWithValue }) => {
    try {
      const response = await EmployeeService.importHistory(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const importEmployeeApply = createAsyncThunk(
  "employee/importEmployeeApply",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
```

```
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await EmployeeService.importDsnvApplier(data);
      if (onSuccess) onSuccess(response.data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const updateEmployeeAPI = createAsyncThunk(
  "employee/updateEmployeeAPI",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await EmployeeService.update(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const deleteEmployeeAPI = createAsyncThunk(
  "employee/deleteEmployeeAPI",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const response = await EmployeeService.delete(data?.id);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const changeStateEmployeeAPI = createAsyncThunk(
  "employee/changeStateEmployeeAPI",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await EmployeeService.changeState(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getAllEmployeeSalary = createAsyncThunk(
  "employee/getAllEmployeeSalary",
  async (id, { rejectWithValue }) => {
    try {
      const response = await EmployeeService.getAllSalary(id);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getAllEmployeeWorkProcess = createAsyncThunk(
  "employee/getAllEmployeeWorkProcess",
  async (id, { rejectWithValue }) => {
    try {
      const response = await EmployeeService.getAllWorkProcess(id);
```

```
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getAllEmployeeChangeHistory = createAsyncThunk(
  "employee/getAllEmployeeChangeHistory",
  async (data, { rejectWithValue }) => {
    try {
      const response = await EmployeeService.getAllChangeHistory(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const upSertEmployeeSalary = createAsyncThunk(
  "employee/upSertEmployeeSalary",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await EmployeeService.upSertSalary(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const upSertEmployeeProcess = createAsyncThunk(
  "employee/upSertEmployeeProcess",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await EmployeeService.upSertWorkProcess(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const exportTLLuongExcelEmployee = createAsyncThunk(
  "employee/exportTLLuongExcelEmployee",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await EmployeeService.exportTLLuongExcelEmployee(
        payload
      );
      if (onSuccess) onSuccess(response.data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const importTLLuongEmployee = createAsyncThunk(
  "employee/importTLLuongEmployee",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, formData } = data;
      const payload = cloneDeep(data);
```

```
      delete payload.onSuccess;
      const response = await EmployeeService.importTLLuongEmployee(formData);
      if (onSuccess) onSuccess(response.data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
const initialState = {
  loading: false,
  employeeList: [],
  employeeAllList: [],
  employeeSignProcessList: [],
  employeeDetail: null,
  importEmployeeList: [],
  importHistory: [],
  wageList: [],
  workProcessList: [],
  historyList: [],
};

export const employeeSlice = createSlice({
  name: "employee",
  initialState,
  reducers: {
    showLoading: (state) => {
      state.loading = true;
    },
    setHisInfo: (state, action) => {
      state.hisInfoList = action.payload;
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(searchGridAPI.pending, (state) => {
        state.loading = true;
      })
      .addCase(searchGridAPI.fulfilled, (state, action) => {
        state.loading = false;
        state.employeeList = action.payload;
      })
      .addCase(searchGridAPI.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getAllEmployee.pending, (state) => {
        state.loading = true;
      })
      .addCase(getAllEmployee.fulfilled, (state, action) => {
        state.loading = false;
        state.employeeAllList = action.payload;
      })
      .addCase(getAllEmployee.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getAllEmployeeSignProcess.pending, (state) => {
        state.loading = true;
      })
      .addCase(getAllEmployeeSignProcess.fulfilled, (state, action) => {
        state.employeeSignProcessList = action.payload;
        state.loading = false;
      })
      .addCase(getAllEmployeeSignProcess.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(createEmployeeAPI.pending, (state) => {
        state.loading = true;
      })
      .addCase(createEmployeeAPI.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(createEmployeeAPI.rejected, (state, action) => {
        state.loading = false;
      })
```

```
.addCase(getDetailAPI.pending, (state) => {
  state.loading = true;
})
.addCase(getDetailAPI.fulfilled, (state, action) => {
  state.loading = false;
  state.employeeDetail = action.payload;
})
.addCase(getDetailAPI.rejected, (state, action) => {
  state.loading = false;
})
.addCase(updateEmployeeAPI.fulfilled, (state, action) => {
  state.loading = false;
})
.addCase(updateEmployeeAPI.rejected, (state, action) => {
  state.loading = false;
})
.addCase(updateEmployeeAPI.pending, (state) => {
  state.loading = true;
})
.addCase(deleteEmployeeAPI.fulfilled, (state, action) => {
  state.loading = false;
})
.addCase(deleteEmployeeAPI.rejected, (state, action) => {
  state.loading = false;
})
.addCase(deleteEmployeeAPI.pending, (state) => {
  state.loading = true;
})
.addCase(changeStateEmployeeAPI.pending, (state) => {
  state.loading = true;
})
.addCase(changeStateEmployeeAPI.fulfilled, (state, action) => {
  state.loading = false;
})
.addCase(changeStateEmployeeAPI.rejected, (state, action) => {
  state.loading = false;
})
.addCase(exportExcelEmployee.pending, (state) => {
  state.loading = true;
})
.addCase(exportExcelEmployee.fulfilled, (state, action) => {
  state.loading = false;
})
.addCase(exportExcelEmployee.rejected, (state, action) => {
  state.loading = false;
})
.addCase(importExcelEmployee.pending, (state) => {
  state.loading = true;
})
.addCase(importExcelEmployee.fulfilled, (state, action) => {
  state.loading = false;
})
.addCase(importExcelEmployee.rejected, (state, action) => {
  state.loading = false;
})
.addCase(importHistoryEmployee.pending, (state) => {
  state.loading = true;
})
.addCase(importHistoryEmployee.fulfilled, (state, action) => {
  state.loading = false;
  state.importHistory = action.payload;
})
.addCase(importHistoryEmployee.rejected, (state, action) => {
  state.loading = false;
})
.addCase(importEmployeeApply.pending, (state) => {
  state.loading = true;
})
.addCase(importEmployeeApply.fulfilled, (state, action) => {
  state.loading = false;
})
.addCase(importEmployeeApply.rejected, (state, action) => {
  state.loading = false;
})
```

```
      .addCase(importDsnvEmployee.pending, (state) => {
        state.loading = true;
      })
      .addCase(importDsnvEmployee.fulfilled, (state, action) => {
        state.loading = false;
        state.importEmployeeList = action.payload;
      })
      .addCase(importDsnvEmployee.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getAllEmployeeSalary.pending, (state) => {
        state.loading = true;
      })
      .addCase(getAllEmployeeSalary.fulfilled, (state, action) => {
        state.loading = false;
        state.wageList = action.payload;
      })
      .addCase(getAllEmployeeSalary.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getAllEmployeeWorkProcess.pending, (state) => {
        state.loading = true;
      })
      .addCase(getAllEmployeeWorkProcess.fulfilled, (state, action) => {
        state.loading = false;
        state.workProcessList = action.payload;
      })
      .addCase(getAllEmployeeWorkProcess.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getAllEmployeeChangeHistory.pending, (state) => {
        state.loading = true;
      })
      .addCase(getAllEmployeeChangeHistory.fulfilled, (state, action) => {
        state.loading = false;
        state.historyList = action.payload;
      })
      .addCase(getAllEmployeeChangeHistory.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(upSertEmployeeSalary.pending, (state) => {
        state.loading = true;
      })
      .addCase(upSertEmployeeSalary.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(upSertEmployeeSalary.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(upSertEmployeeProcess.pending, (state) => {
        state.loading = true;
      })
      .addCase(upSertEmployeeProcess.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(upSertEmployeeProcess.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(exportTLLuongExcelEmployee.pending, (state) => {
        state.loading = true;
      })
      .addCase(exportTLLuongExcelEmployee.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(exportTLLuongExcelEmployee.rejected, (state, action) => {
        state.loading = false;
      });
  },
});

export const { showLoading, setHisInfo } = employeeSlice.actions;

export default employeeSlice.reducer;
```