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

export const gridSearchBangLuong = createAsyncThunk(
  "quanlyluong/gridSearchBangLuong",
  async (data, { rejectWithValue }) => {
    try {
      const response = await QuanLyLuongService.gridSearchBangLuong(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getBangLuongById = createAsyncThunk(
  "quanlyluong/getBangLuongById",
  async (id, { rejectWithValue }) => {
    try {
      const response = await QuanLyLuongService.getBangLuongById(id);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const createBangLuong = createAsyncThunk(
  "quanlyluong/createBangLuong",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await QuanLyLuongService.create(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const updateBangLuong = createAsyncThunk(
  "quanlyluong/updateBangLuong",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await QuanLyLuongService.update(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const changeStateBangLuong = createAsyncThunk(
  "quanlyluong/changeStateBangLuong",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, id } = data;
      const response = await QuanLyLuongService.changeStateBangLuong(id);
      if (onSuccess) onSuccess(response.data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
```

```
);
export const duyetBangLuong = createAsyncThunk(
  "quanlyluong/duyetBangLuong",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, id } = data;
      const response = await QuanLyLuongService.duyetBangLuong(id);
      if (onSuccess) onSuccess(response.data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

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

export const gridSearchBangLuongNhanVien = createAsyncThunk(
  "quanlyluong/gridSearchBangLuongNhanVien",
  async (data, { rejectWithValue }) => {
    try {
      const response = await QuanLyLuongService.gridSearchBangLuongNhanVien(
        data
      );
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const gridSearchBangCongLuong = createAsyncThunk(
  "quanlyluong/gridSearchBangCongLuong",
  async (data, { rejectWithValue }) => {
    try {
      const response = await QuanLyLuongService.gridSearchBangCongLuong(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const gridSearchBangPhuCapGiamTruLuong = createAsyncThunk(
  "quanlyluong/gridSearchBangPhuCapGiamTruLuong",
  async (data, { rejectWithValue }) => {
    try {
      const response =
        await QuanLyLuongService.gridSearchBangPhuCapGiamTruLuong(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const updateBangLuongNhanVien = createAsyncThunk(
  "quanlyluong/updateBangLuongNhanVien",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
```

```
      delete payload.onSuccess;
      const response = await QuanLyLuongService.updateBangLuongNhanVien(
        payload
      );
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const rebuildResultLuongNhanVien = createAsyncThunk(
  "quanlyluong/rebuildResultLuongNhanVien",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await QuanLyLuongService.rebuildResultLuongNhanVien(
        data
      );
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const bangLuongExport = createAsyncThunk(
  "quanlyluong/bangLuongExport",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, id } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await QuanLyLuongService.export(id);
      if (onSuccess) onSuccess(response.data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
const initialState = {
  loading: false,
  bangluongList: [],
  bangLuongDetail: {},
  bangluongnhanvienList: [],
  bangcongluongList: [],
  bangphucapgiamtruluongList: [],
};

export const quanLyLuongSlice = createSlice({
  name: "quanlyluong",
  initialState,
  reducers: {
    showLoading: (state) => {
      state.loading = true;
    },
    setHisInfo: (state, action) => {
      state.hisInfoList = action.payload;
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(gridSearchBangLuong.pending, (state) => {
        state.loading = true;
      })
      .addCase(gridSearchBangLuong.fulfilled, (state, action) => {
        state.loading = false;
        state.bangluongList = action.payload;
      })
      .addCase(gridSearchBangLuong.rejected, (state, action) => {
        state.loading = false;
```

```
})
.addCase(getBangLuongById.pending, (state) => {
  state.loading = true;
})
.addCase(getBangLuongById.fulfilled, (state, action) => {
  state.loading = false;
  state.bangLuongDetail = action.payload;
})
.addCase(getBangLuongById.rejected, (state, action) => {
  state.loading = false;
})
.addCase(gridSearchBangLuongNhanVien.pending, (state) => {
  state.loading = true;
})
.addCase(gridSearchBangLuongNhanVien.fulfilled, (state, action) => {
  state.loading = false;
  state.bangluongnhanvienList = action.payload.data;
  state.bangLuongCate = action.payload;
})
.addCase(gridSearchBangLuongNhanVien.rejected, (state, action) => {
  state.loading = false;
})
.addCase(gridSearchBangCongLuong.pending, (state) => {
  state.loading = true;
})
.addCase(gridSearchBangCongLuong.fulfilled, (state, action) => {
  state.loading = false;
  state.bangcongluongList = action.payload;
})
.addCase(gridSearchBangCongLuong.rejected, (state, action) => {
  state.loading = false;
})
.addCase(gridSearchBangPhuCapGiamTruLuong.pending, (state) => {
  state.loading = true;
})
.addCase(gridSearchBangPhuCapGiamTruLuong.fulfilled, (state, action) => {
  state.loading = false;
  state.bangphucapgiamtruluongList = action.payload;
})
.addCase(gridSearchBangPhuCapGiamTruLuong.rejected, (state, action) => {
  state.loading = false;
})
.addCase(createBangLuong.pending, (state) => {
  state.loading = true;
})
.addCase(createBangLuong.fulfilled, (state, action) => {
  state.loading = false;
})
.addCase(createBangLuong.rejected, (state, action) => {
  state.loading = false;
})
.addCase(updateBangLuong.pending, (state) => {
  state.loading = true;
})
.addCase(updateBangLuong.fulfilled, (state, action) => {
  state.loading = false;
})
.addCase(updateBangLuong.rejected, (state, action) => {
  state.loading = false;
})
.addCase(deletedBangLuong.pending, (state) => {
  state.loading = true;
})
.addCase(deletedBangLuong.fulfilled, (state, action) => {
  state.loading = false;
})
.addCase(deletedBangLuong.rejected, (state, action) => {
  state.loading = false;
})
.addCase(rebuildResultLuongNhanVien.pending, (state) => {
  state.loading = true;
})
.addCase(rebuildResultLuongNhanVien.fulfilled, (state, action) => {
  state.loading = false;
```

```
      })
      .addCase(rebuildResultLuongNhanVien.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(changeStateBangLuong.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(changeStateBangLuong.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(changeStateBangLuong.pending, (state) => {
        state.loading = true;
      });
  },
});

export const { showLoading } = quanLyLuongSlice.actions;

export default quanLyLuongSlice.reducer;
```