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

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

export const getAllCaLamViec = createAsyncThunk(
  "qlCongSlice/getAllCaLamViec",
  async (_, { rejectWithValue }) => {
    try {
      const response = await QuanLyCongService.getAllCaLamViec();
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

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

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

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

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

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

// Ký hiệu chấm công
export const searchSymbol = createAsyncThunk(
  "qlCongSlice/searchSymbol",
  async (search, { rejectWithValue }) => {
    try {
      const response = await QuanLyCongService.searchSymbol(search);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

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

```
// Lý do nghỉ
export const searchReason = createAsyncThunk(
  "qlCongSlice/searchReason",
  async (search, { rejectWithValue }) => {
    try {
      const response = await QuanLyCongService.searchReason(search);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

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

// Bảng chấm công

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

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

export const addNewDsBangCong = createAsyncThunk(
  "qlCongSlice/addNewDsBangCong",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await QuanLyCongService.addDsBangCong(data);
      if (onSuccess) onSuccess(response.data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
```

```
);

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

export const deleteDsBangCong = createAsyncThunk(
  "qlCongSlice/deleteDsBangCong",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, id } = data;
      const response = await QuanLyCongService.deleteBangCong(id);
      if (onSuccess) onSuccess(response.data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

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

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

export const upSertChiTietBangCongNhanVien = createAsyncThunk(
  "qlCongSlice/upSertChiTietBangCongNhanVien",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await QuanLyCongService.upSertBc(payload);
      if (onSuccess) onSuccess(response.data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
```

```
    }
  }
);

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

export const deleteNhanVienById = createAsyncThunk(
  "qlCongSlice/deleteNhanVienById",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, id } = data;
      const response = await QuanLyCongService.delNhanVienById(id);
      if (onSuccess) onSuccess(response.data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const chotBangCong = createAsyncThunk(
  "qlCongSlice/chotBangCong",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const response = await QuanLyCongService.chotBangCong(data);
      if (onSuccess) onSuccess(response.data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const rebuildResultCongNhanVien = createAsyncThunk(
  "qlCongSlice/rebuildResultCongNhanVien",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await QuanLyCongService.rebuildResultCongNhanVien(data);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

const initialState = {
  loading: false,
  calamviecList: [],
  allcalamviecList: [],
  calamviecDetail: {},
  symbolList: [],
  reasonList: [],
  bangCong: {},
  bangCongDetail: {},
  ChiTietBCList: [],
  ngayLamViecList: [],
};
```

```
export const qlCongSliceSlice = createSlice({
  name: "qlCong",
  initialState,
  reducers: {
    showLoading: (state) => {
      state.loading = true;
    },
    setHisInfo: (state, action) => {
      state.hisInfoList = action.payload;
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(gridSearchCaLamViec.pending, (state) => {
        state.loading = true;
      })
      .addCase(gridSearchCaLamViec.fulfilled, (state, action) => {
        state.loading = false;
        state.calamviecList = action.payload;
      })
      .addCase(gridSearchCaLamViec.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getAllCaLamViec.pending, (state) => {
        state.loading = true;
      })
      .addCase(getAllCaLamViec.fulfilled, (state, action) => {
        state.loading = false;
        state.allcalamviecList = action.payload;
      })
      .addCase(getAllCaLamViec.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getCaLamViecById.pending, (state) => {
        state.loading = true;
      })
      .addCase(getCaLamViecById.fulfilled, (state, action) => {
        state.loading = false;
        state.calamviecDetail = action.payload;
      })
      .addCase(getCaLamViecById.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(createCaLamViec.pending, (state) => {
        state.loading = true;
      })
      .addCase(createCaLamViec.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(createCaLamViec.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(updateCaLamViec.pending, (state) => {
        state.loading = true;
      })
      .addCase(updateCaLamViec.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(updateCaLamViec.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(deletedCaLamViec.pending, (state) => {
        state.loading = true;
      })
      .addCase(deletedCaLamViec.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(deletedCaLamViec.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(searchSymbol.pending, (state) => {
        state.loading = true;
      })
```

```
.addCase(searchSymbol.fulfilled, (state, action) => {
  state.loading = false;
  state.symbolList = action.payload;
})
.addCase(searchSymbol.rejected, (state) => {
  state.loading = false;
})
.addCase(upSertSymbol.pending, (state) => {
  state.loading = true;
})
.addCase(upSertSymbol.fulfilled, (state, _action) => {
  state.loading = false;
})
.addCase(upSertSymbol.rejected, (state) => {
  state.loading = false;
})
.addCase(searchReason.pending, (state) => {
  state.loading = true;
})
.addCase(searchReason.fulfilled, (state, action) => {
  state.loading = false;
  state.reasonList = action.payload;
})
.addCase(searchReason.rejected, (state) => {
  state.loading = false;
})
.addCase(upSertReason.pending, (state) => {
  state.loading = true;
})
.addCase(upSertReason.fulfilled, (state, _action) => {
  state.loading = false;
})
.addCase(upSertReason.rejected, (state) => {
  state.loading = false;
})
.addCase(importBangChamCong.pending, (state) => {
  state.loading = true;
})
.addCase(importBangChamCong.fulfilled, (state, action) => {
  state.loading = false;
})
.addCase(importBangChamCong.rejected, (state, action) => {
  state.loading = false;
})

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

.addCase(searchDsBangCong.fulfilled, (state, action) => {
  state.loading = false;
  state.bangCong = action.payload;
})
.addCase(searchDsBangCong.rejected, (state, action) => {
  state.loading = false;
})
.addCase(searchDsBangCong.pending, (state) => {
  state.loading = true;
})

.addCase(getByIdDsBangCong.fulfilled, (state, action) => {
  state.loading = false;
  state.bangCongDetail = action.payload;
})
.addCase(getByIdDsBangCong.rejected, (state, action) => {
  state.loading = false;
})
.addCase(getByIdDsBangCong.pending, (state) => {
```

```
  state.loading = true;
})

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

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

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

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

.addCase(searchChiTietBangCong.fulfilled, (state, action) => {
  state.loading = false;
  state.ChiTietBCList = action.payload.data;
  state.ngayLamViecList = action.payload.ngayLamViecList;
  state.bangCongCate = action.payload;
  state.thietLapCongList = action.payload.thietLapCongList;
})
.addCase(searchChiTietBangCong.rejected, (state, action) => {
  state.loading = false;
})
.addCase(searchChiTietBangCong.pending, (state) => {
  state.loading = true;
})

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

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

```
        state.loading = true;
      })
      .addCase(saveBangChamCongImport.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(saveBangChamCongImport.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(saveBangChamCongImport.pending, (state) => {
        state.loading = true;
      })
      .addCase(chotBangCong.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(chotBangCong.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(chotBangCong.pending, (state) => {
        state.loading = true;
      })
      .addCase(rebuildResultCongNhanVien.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(rebuildResultCongNhanVien.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(rebuildResultCongNhanVien.pending, (state) => {
        state.loading = true;
      });
  },
});

export const { showLoading } = qlCongSliceSlice.actions;

export default qlCongSliceSlice.reducer;
```