```
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";
import { cloneDeep } from "lodash";
import HRRecordService from "services/HRRecordService";
import Utils from "utils";
import { initHoSo } from "views/app-views/hrm/hr-records/c1";

export const getHSAllLevel = createAsyncThunk(
  "hrRecord/getHSAllLevel",
  async (_, { rejectWithValue }) => {
    try {
      const response = await HRRecordService.getAllLevel();
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const searchHSC1 = createAsyncThunk(
  "hrRecord/searchHSC1",
  async (data, { rejectWithValue }) => {
    try {
      const response = await HRRecordService.searchGrid(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const addNewHSC1 = createAsyncThunk(
  "hrRecord/addNewHSC1",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await HRRecordService.addHSC1(data);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const updateHSC1 = createAsyncThunk(
  "hrRecord/updateHSC1",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await HRRecordService.updateHSC1(data);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const deleteHSC1 = createAsyncThunk(
  "hrRecord/deleteHSC1",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, id } = data;
      const response = await HRRecordService.deleteHSC1(id);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const changeStateHSC1 = createAsyncThunk(
  "hrRecord/changeStateHSC1",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, id } = data;
      const response = await HRRecordService.changeStateHSC1(id);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

//Hoso2

export const getAllDMHSC1 = createAsyncThunk(
  "hrRecord/getAllDMHSC1",
  async (_, { rejectWithValue }) => {
    try {
      const response = await HRRecordService.getAllDmHSC1();
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getAllDMHSC2 = createAsyncThunk(
  "hrRecord/getAllDMHSC2",
  async (id, { rejectWithValue }) => {
    try {
      const response = await HRRecordService.getAllDmHSC2(id);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getAllHSC2 = createAsyncThunk(
  "hrRecord/getAllHSC2",
  async (data, { rejectWithValue }) => {
    try {
      const response = await HRRecordService.getAllHSC2(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const addNewHSC2 = createAsyncThunk(
  "hrRecord/addNewHSC2",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await HRRecordService.addHSC2(data);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const updateHSC2 = createAsyncThunk(
  "hrRecord/updateHSC2",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await HRRecordService.updateHSC2(data);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const deleteHSC2 = createAsyncThunk(
  "hrRecord/deleteHSC2",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, id } = data;
      const response = await HRRecordService.deleteHSC2(id);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const changeStateHSC2 = createAsyncThunk(
  "hrRecord/changeStateHSC2",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, id } = data;
      const response = await HRRecordService.changeStateHSC2(id);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

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

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

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

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

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

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

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

const initialState = {
  loading: false,
  searchHs1: [],
  hs2List: [],
  dmHs1List: [],
  dmHs2List: [],
  menuHoSo: [],
  employeeProfile: [],
  HsCateList: { loading: false, data: [] },
};

export const hrRecordSlice = createSlice({
  name: "hrRecord",
  initialState,
  reducers: {
    showLoading: (state) => {
      state.loading = true;
    },
    setHoSo1: (state, action) => {
      state.searchHs1 = action.payload;
    },
    setHoSo2: (state, action) => {
      state.hs2List = action.payload;
    },
    setEmployeeProfile: (state, action) => {
      state.employeeProfile = action.payload;
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(getHSAllLevel.pending, (state) => {
        state.HsCateList.loading = true;
      })
      .addCase(getHSAllLevel.fulfilled, (state, action) => {
        state.HsCateList.loading = false;
        state.HsCateList.data = action.payload;
      })
      .addCase(getHSAllLevel.rejected, (state, action) => {
        state.HsCateList.loading = false;
      })
      .addCase(searchHSC1.pending, (state) => {
        state.loading = true;
      })
      .addCase(searchHSC1.fulfilled, (state, action) => {
        state.loading = false;
        state.searchHs1 = [initHoSo, ...action.payload];
      })
      .addCase(searchHSC1.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(addNewHSC1.pending, (state) => {
        state.loading = true;
      })
      .addCase(addNewHSC1.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(addNewHSC1.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(updateHSC1.pending, (state) => {
        state.loading = true;
      })
      .addCase(updateHSC1.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(updateHSC1.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(deleteHSC1.pending, (state) => {
        state.loading = true;
      })
      .addCase(deleteHSC1.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(deleteHSC1.rejected, (state, action) => {
```

```
      state.loading = false;
    })
    .addCase(changeStateHSC1.pending, (state) => {
      state.loading = true;
    })
    .addCase(changeStateHSC1.fulfilled, (state, action) => {
      state.loading = false;
    })
    .addCase(changeStateHSC1.rejected, (state, action) => {
      state.loading = false;
    })

    .addCase(getAllDMHSC1.pending, (state) => {
      state.loading = true;
    })
    .addCase(getAllDMHSC1.fulfilled, (state, action) => {
      state.loading = false;
      state.dmHs1List = action.payload;
      state.menuHoSo = action.payload.map((x) => ({
        ...x,
        path: Utils.removeVietnameseTones(x.label)
          .toLowerCase()
          .replaceAll(" ", "-"),
      }));
    })
    .addCase(getAllDMHSC1.rejected, (state, action) => {
      state.loading = false;
    })

    .addCase(getAllDMHSC2.pending, (state) => {
      state.loading = true;
    })
    .addCase(getAllDMHSC2.fulfilled, (state, action) => {
      state.loading = false;
      state.dmHs2List = action.payload;
    })
    .addCase(getAllDMHSC2.rejected, (state, action) => {
      state.loading = false;
    })

    .addCase(getAllHSC2.pending, (state) => {
      state.loading = true;
    })
    .addCase(getAllHSC2.fulfilled, (state, action) => {
      state.loading = false;
      state.hs2List = [initHoSo, ...action.payload];
    })
    .addCase(getAllHSC2.rejected, (state, action) => {
      state.loading = false;
    })
    .addCase(addNewHSC2.pending, (state) => {
      state.loading = true;
    })
    .addCase(addNewHSC2.fulfilled, (state, action) => {
      state.loading = false;
    })
    .addCase(addNewHSC2.rejected, (state, action) => {
      state.loading = false;
    })
    .addCase(updateHSC2.pending, (state) => {
      state.loading = true;
    })
    .addCase(updateHSC2.fulfilled, (state, action) => {
      state.loading = false;
    })
    .addCase(updateHSC2.rejected, (state, action) => {
      state.loading = false;
    })
    .addCase(deleteHSC2.pending, (state) => {
      state.loading = true;
    })
    .addCase(deleteHSC2.fulfilled, (state, action) => {
      state.loading = false;
    })
```

```
      .addCase(deleteHSC2.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(changeStateHSC2.pending, (state) => {
        state.loading = true;
      })
      .addCase(changeStateHSC2.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(changeStateHSC2.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getAllHoSoNV.pending, (state) => {
        state.loading = true;
      })
      .addCase(getAllHoSoNV.fulfilled, (state, action) => {
        state.loading = false;
        state.employeeProfile = [initHoSo, ...action.payload];
      })
      .addCase(getAllHoSoNV.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(addNewHoSoNv.pending, (state) => {
        state.loading = true;
      })
      .addCase(addNewHoSoNv.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(addNewHoSoNv.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(updateHoSoNv.pending, (state) => {
        state.loading = true;
      })
      .addCase(updateHoSoNv.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(updateHoSoNv.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(deleteHoSoNv.pending, (state) => {
        state.loading = true;
      })
      .addCase(deleteHoSoNv.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(deleteHoSoNv.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getExportHoSoNhanVien.pending, (state) => {
        state.loading = true;
      })
      .addCase(getExportHoSoNhanVien.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(getExportHoSoNhanVien.rejected, (state, action) => {
        state.loading = false;
      });
  },
});

export const { showLoading, setHoSo1, setHoSo2, setEmployeeProfile } =
  hrRecordSlice.actions;

export default hrRecordSlice.reducer;
```