```
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";
import { message } from "antd";
import { clone, cloneDeep } from "lodash";
import moment from "moment";
import DeXuatMuaSamService from "services/sale/DeXuatMuaSamService";
import { v4 as uuidv4 } from "uuid";
//#region dexuatmuasam

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

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

export const getall = createAsyncThunk(
  "dexuatmuasam/getall",
  async (data, { rejectWithValue }) => {
    try {
      const response = await DeXuatMuaSamService.getall(data);
      return response?.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getbyid = createAsyncThunk(
  "dexuatmuasam/getbyid",
  async (data, { rejectWithValue }) => {
    try {
      const response = await DeXuatMuaSamService.getbyid(data);
      return response?.data?.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const deletebyid = createAsyncThunk(
  "dexuatmuasam/deletebyid",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
```

```
      const response = await DeXuatMuaSamService.delete(payload.id);
      if (onSuccess) onSuccess(response);
      return response;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

//#endregion

//#region chitietdexuatmuasam
export const createchitet = createAsyncThunk(
  "dexuatmuasam/createchitet",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      delete payload.onError;
      const response = await DeXuatMuaSamService.createChiTiet(data);
      if (onSuccess) onSuccess(response.data.data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getdsdexuatbyid = createAsyncThunk(
  "dexuatmuasam/getdsdexuatbyid",
  async (data, { rejectWithValue }) => {
    try {
      const response = await DeXuatMuaSamService.getDsChiTietByID(data);
      return response?.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

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

export const addrow = createAsyncThunk(
  "dexuatmuasam/addrow",
  async (data, { rejectWithValue, getState }) => {
    try {
      const state = getState();
      let item=state.deXuatMuaSam.chitietdexuatmuasamlist.find(x=>x.tenhanghoa === null)
      if (data && !item) {
        const newData = {
          id: uuidv4(),
          mahh: null,
          tenhanghoa: null,
          soluong: null,
          ghichu: null,
          donvitinh: null,
          dongia: null,
```

```
          thanhtien: null,
        };
        return newData;
      }
      return rejectWithValue();
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
const removeAccents = (str) => {
  return str.normalize("NFD").replace(/[\u0300-\u036f]/g, "");
};

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

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

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

export const getTrangThaiXuLyDon = createAsyncThunk(
  "dexuatmuasam/getTrangThaiXuLyDon",
  async (data, { rejectWithValue }) => {
    try {
      const response = await DeXuatMuaSamService.getTrangThaiXuLyDon(data);
      const res=response.data;
      let noidung={
        maDeXuat:res?.madexuat,
        tenDeXuat:res?.tendexuat,
        ngayDeXuat:moment(res?.ngaydexuat).format("DD-MM/YYYY HH:mm"),
        tenTrangThai:res?.tenTrangThai,
        chitietdexuatmuasam:res.chitietdexuatmuasam
      }
      return noidung;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
```

```
    }
  }
);

//#endregion
const initialState = {
  loading: false,
  dexuatmuasamlist: [],
  dexuatmuasamdetail: {},
  totalcount: 0,
  chitietdexuatmuasamlist: [],
  bangchitietdexuatmuasam: [],
  thongTinCheckXuLyDonHang: null,
};

export const dexuatmuasamSlice = createSlice({
  name: "deXuatMuaSam",
  initialState,
  reducers: {
    showLoading: (state) => {
      state.loading = true;
    },
    reSetDataChiTietDeXuatMuaSam: (state) => {
      state.chitietdexuatmuasamlist = [];
    },
    updateRowChiTietDeXuat: (state, action) => {
      var data = cloneDeep(action.payload);
      const { onSuccess, onError } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      delete payload.onError;
      state.chitietdexuatmuasamlist = payload;
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(create.pending, (state) => {
        state.loading = true;
      })
      .addCase(create.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(create.rejected, (state, action) => {
        state.loading = false;
      })

      .addCase(getall.pending, (state) => {
        state.loading = true;
      })
      .addCase(getall.fulfilled, (state, action) => {
        state.loading = false;
        state.dexuatmuasamlist = action.payload?.data.data;
        state.totalcount = action.payload?.data.totalCount;
        state.bangchitietdexuatmuasam = action.payload.detail;
      })
      .addCase(getall.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getbyid.pending, (state) => {
        state.loading = true;
      })
      .addCase(getbyid.fulfilled, (state, action) => {
        state.loading = false;
        state.dexuatmuasamdetail = action.payload;
      })
      .addCase(getbyid.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(update.pending, (state) => {
        state.loading = true;
      })
      .addCase(update.fulfilled, (state, action) => {
        state.loading = false;
      })
```

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

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

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

.addCase(getDmHangHoa.fulfilled, (state, action) => {
  state.loading = false;

})
.addCase(getDmHangHoa.rejected, (state, action) => {
  state.loading = false;
})
.addCase(getDmHangHoa.pending, (state) => {
  state.loading = true;
})
.addCase(guiYeuCauXuLyDon.fulfilled, (state, action) => {
  state.loading = false;
})
.addCase(guiYeuCauXuLyDon.rejected, (state, action) => {
  state.loading = false;
})
.addCase(guiYeuCauXuLyDon.pending, (state) => {
  state.loading = true;
})
.addCase(getTrangThaiXuLyDon.fulfilled, (state, action) => {
  state.loading = false;
  state.thongTinCheckXuLyDonHang = action.payload;
})
.addCase(getTrangThaiXuLyDon.rejected, (state, action) => {
  state.loading = false;
})
.addCase(getTrangThaiXuLyDon.pending, (state) => {
  state.loading = true;
})

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

```
        state.loading = true;
      })
      .addCase(addrow.fulfilled, (state, action) => {
        if (action.payload) {
          state.chitietdexuatmuasamlist=[action.payload,...state.chitietdexuatmuasamlist];
        }
      })
      .addCase(addrow.rejected, (state, action) => {
        state.loading = false;
      });
  },
});

export const {
  showLoading,
  dexuatmuasamlist,
  dexuatmuasamdetail,
  addRowChiTietDeXuatMuaSam,
  reSetDataChiTietDeXuatMuaSam,
  addRow,
  updateRowChiTietDeXuat,
  bangchitietdexuatmuasam,
  thongTinCheckXuLyDonHang,
} = dexuatmuasamSlice.actions;

export default dexuatmuasamSlice.reducer;
```