import Result from "../model/resultModel.js";

import Exam from "../model/examModel.js";

export *const* getAllResult = *async*(*req*,*res*)=>{

    try {

*const* results = await Result.find().populate('examId').populate('subjectResults.subjectId');

        res.status(200).json({

          status: 'success',

          numResult: results.length,

          data: {

            results

          }

        });

      } catch (error) {

        res.status(500).json({ error });

      }

}

export *const* postAllResult = *async*(*req*,*res*)=>{

    try {

*const* result = new Result(req.body);

        await result.save();

        res.status(201).json({

          status: 'success',

          msg: 'Result created',

          data: {

            result

          }

        });

      } catch (error) {

        res.status(400).send(error);

      }

}

export *const* getResultById = *async* (*req*, *res*) => {

    try {

*const* result = await Result.findById(req.params.id).populate('examId').populate('subjectResults.subjectId');

      if (!result) {

        return res.status(404).json({

          msg: 'No result found for this ID'

        });

      }

      res.status(200).json({

        status: 'success',

        data: {

          result

        }

      });

    } catch (error) {

      res.status(500).send(error);

    }

  };

export *const* dropResullt = *async*(*req*,*res*)=>{

    try {

*const* result = await Result.findByIdAndDelete(req.params.id);

        if (!result) {

          return res.status(404).send({ msg: 'No result found for this ID' });

        }

        res.status(204).json({

          status: 'success',

          data: null

        });

      } catch (error) {

        res.status(500).send(error);

      }

}

export *const* upDateResult = *async*(*req*,*res*)=>{

    try {

*const* result = await Result.findByIdAndUpdate(req.params.id, req.body, {

      new: true,

      runValidators: true

    });

    if (!result) {

      return res.status(404).send({ msg: 'No result found for this ID' });

    }

    res.status(200).json({

      status: 'success',

      data: {

        result

      }

    });

  } catch (error) {

    res.status(400).send(error);

  }

}

export *const* getResultsBySubjectAndCategory = *async* (*req*, *res*) => {

    try {

*const* { category, subjectId } = req.params;

*// Find exams with the given category*

*const* exams = await Exam.find({ category });

      if (!exams.length) {

        return res.status(404).json({ msg: 'No exams found for this category' });

      }

*// Extract exam IDs*

*const* examIds = exams.map(*exam* => exam.\_id);

*// Find results that match the given subject and exams in the category*

*const* results = await Result.find({

        examId: { $in: examIds },

        'subjectResults.subjectId': subjectId

      }).populate('examId').populate('subjectResults.subjectId');

      if (!results.length) {

        return res.status(404).json({ msg: 'No results found for this subject in the given category' });

      }

      res.status(200).json({

        status: 'success',

        numResults: results.length,

        data: {

          results

        }

      });

    } catch (error) {

      res.status(500).send(error);

    }

  };

export *const* getAllResultsForStudent = *async* (*req*, *res*) => {

    try {

*const* { studentId } = req.params;

*// Find the student*

*const* student = await User.findById(studentId);

      if (!student) {

        return res.status(404).json({ msg: 'Student not found' });

      }

*// Find all results for the student*

*const* results = await Result.find({ studentId }).populate('examId').populate('subjectResults.subjectId');

      if (!results.length) {

        return res.status(404).json({ msg: 'No results found for this student' });

      }

      res.status(200).json({

        status: 'success',

        numResults: results.length,

        data: {

          results

        }

      });

    } catch (error) {

      res.status(500).send(error);

    }

  };