INTERNSHIP

Candidate Name: Dona S Lawrence

Date of submission: 14-01-2024

Assignment

- Task :
  Backend CRUD operation. (Node.js, Express.js, MongoDB)
- Modules:
  Student (name, roll no, & mobile No, classId)
  Class (standard, division)
- API's:

  Handle the logical part of these API's carefully

  Create Student & Class

  Update Student's Class with standard and division.

  Delete Student & Class

  Read All Students in a class with standard and division.

  Read All Students in a standard.

Code

```
const express = require('express');

const mongoose = require('mongoose');

const bodyParser = require('body-parser');


const app = express();

const port = 3000;


// Connect to MongoDB

mongoose.connect('mongodb://localhost:27017/school', {

  useNewUrlParser: true,

  useUnifiedTopology: true,

});


// Define schemas

const classSchema = new mongoose.Schema({

  standard: String,

  division: String,

});


const studentSchema = new mongoose.Schema({

  name: String,

  rollNo: String,

  mobileNo: String,

  classId: {

    type: mongoose.Schema.Types.ObjectId,

    ref: 'Class',

  },
});

// Create models
const Class = mongoose.model('Class', classSchema);
const Student = mongoose.model('Student', studentSchema);

app.use(bodyParser.json());

// Create class
app.post('/class', async (req, res) => {
  try {
    const { standard, division } = req.body;
    const newClass = new Class({ standard, division });
    await newClass.save();
    res.json(newClass);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

// Create student
app.post('/student', async (req, res) => {
  try {
    const { name, rollNo, mobileNo, classId } = req.body;
    const newStudent = new Student({ name, rollNo, mobileNo, classId });
    await newStudent.save();
    res.json(newStudent);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

// Update student's class
app.put('/student/:id', async (req, res) => {
  try {
    const { standard, division } = req.body;
    const updatedStudent = await Student.findByIdAndUpdate(
      req.params.id,
      { $set: { classId: { standard, division } } },
      { new: true }
    );
    res.json(updatedStudent);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

// Delete student
app.delete('/student/:id', async (req, res) => {
  try {
    await Student.findByIdAndDelete(req.params.id);
    res.json({ message: 'Student deleted successfully' });
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

// Read all students in a class
```

```
app.get('/students/class/:classId', async (req, res) => {
  try {
    const students = await Student.find({ classId: req.params.classId });
    res.json(students);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

// Read all students in a standard
app.get('/students/standard/:standard', async (req, res) => {
  try {
    const students = await Student.find().populate({
      path: 'classId',
      match: { standard: req.params.standard },
    });
    res.json(students.filter(student => student.classId));
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```