#### **Backend**

#### **a. Đối tượng sử dụng**

* **Học sinh cấp 3 (Thí sinh)**: Là học sinh từ lớp 10 đến lớp 12, tham gia các bài thi trắc nghiệm để kiểm tra kiến thức.
* **Giáo viên (Admin)**: Quản lý bài thi, câu hỏi, và kết quả của học sinh.
* **Không có đăng ký công khai**: Chỉ giáo viên (admin) được phép tạo tài khoản cho học sinh, đảm bảo hệ thống chỉ dành cho học sinh trong trường.

#### **b. Tính năng điều chỉnh**

1. **Tính năng cho Giáo viên (Admin)**:
   * **Quản lý bài thi**:
     + Tạo bài thi theo môn học cấp 3 (Toán, Lý, Hóa, Văn, Anh, Sử, Địa).
     + Đặt thời gian làm bài (ví dụ: 45 phút, phù hợp với tiết học).
     + Chọn lớp áp dụng (lớp 10A1, 11B2, 12C3, v.v.).
   * **Quản lý câu hỏi**:
     + Chỉ hỗ trợ câu hỏi trắc nghiệm.
     + Câu hỏi được phân loại theo môn học và chương (ví dụ: Toán - Đại số - Phương trình bậc hai).
     + Hỗ trợ nhập các ký tự toán học (chi tiết ở phần sau).
   * **Quản lý học sinh**:
     + Thêm học sinh theo lớp (lớp 10A1, 11B2, v.v.).
     + Nhập thông tin học sinh: Họ tên, Mã học sinh, Lớp.
     + Không cho phép học sinh tự đăng ký (giáo viên tạo tài khoản).
   * **Xem kết quả**:
     + Xem kết quả theo lớp, môn học, hoặc học sinh.
     + Xuất báo cáo điểm số (Excel/CSV) để nộp cho nhà trường.
2. **Tính năng cho Học sinh**:
   * **Đăng nhập**:
     + Đăng nhập bằng tài khoản do giáo viên cung cấp (mã học sinh + mật khẩu mặc định).
   * **Tham gia bài thi**:
     + Xem danh sách bài thi theo môn học và lớp.
     + Làm bài trắc nghiệm với giao diện đơn giản, dễ hiểu.
     + Có đồng hồ đếm ngược và nút nộp bài.
   * **Xem kết quả**:
     + Xem điểm số, số câu đúng/sai, và đáp án đúng.
     + Xem lịch sử thi (các bài thi đã làm).
   * **Hồ sơ cá nhân**:
     + Xem thông tin: Họ tên, Mã học sinh, Lớp.
     + Đổi mật khẩu (tùy chọn).

#### **c. Tính năng bổ sung**

* **Giới hạn quyền truy cập**:
  + Học sinh chỉ thấy bài thi của lớp mình (ví dụ: Học sinh lớp 10A1 không thấy bài thi của 10A2).
* **Môn học cấp 3**:
  + Tập trung vào các môn chính: Toán, Lý, Hóa, Sinh, Văn, Sử, Địa, Anh.
* **Thống kê**:
  + Giáo viên có thể xem thống kê: Điểm trung bình của lớp, số học sinh đạt trên trung bình, v.v.

### **2. Lưu trữ và hiển thị đề bài có ký tự toán học**

#### **a. Vấn đề với ký tự toán học**

* Các ký tự toán học (như phân số ab \frac{a}{b} ba​, căn bậc hai x \sqrt{x} x​, tích phân ∫ \int ∫, lũy thừa x2 x^2 x2) không thể hiển thị trực tiếp dưới dạng văn bản thô trong HTML.
* Để lưu trữ và hiển thị các ký tự toán học, chúng ta cần:
  1. **Lưu trữ**: Lưu dưới dạng mã LaTeX (một ngôn ngữ định dạng toán học phổ biến).
  2. **Hiển thị**: Dùng thư viện như **MathJax** hoặc **KaTeX** để render LaTeX thành ký tự toán học trên giao diện.

#### **b. Cách lưu trữ đề bài có ký tự toán học**

* **Lưu dưới dạng LaTeX**:
  + Trong cơ sở dữ liệu, cột content của bảng questions sẽ lưu nội dung câu hỏi dưới dạng mã LaTeX.
  + Ví dụ:
    - Câu hỏi: "Tính 2x+3x−1 \frac{2x + 3}{x - 1} x−12x+3​ khi x=2 x = 2 x=2."
    - Lưu trong database: Tính \( \frac{2x + 3}{x - 1} \) khi \( x = 2 \).
  + Tương tự, các đáp án cũng có thể chứa LaTeX:
    - Đáp án A: x=7 x = 7 x=7
    - Lưu: \( x = 7 \)

#### **c. Cơ sở dữ liệu (điều chỉnh)**

Dựa trên yêu cầu mới, mình sẽ cập nhật cơ sở dữ liệu:

**Bảng users (Học sinh và Giáo viên):**

| **Cột** | **Kiểu dữ liệu** | **Mô tả** |
| --- | --- | --- |
| id | INT (PK, AI) | ID người dùng |
| student\_id | VARCHAR(20) | Mã học sinh (nếu là học sinh) |
| full\_name | VARCHAR(100) | Họ tên |
| email | VARCHAR(100) | Email (dùng cho giáo viên) |
| password | VARCHAR(255) | Mật khẩu (mã hóa) |
| role | ENUM('teacher', 'student') | Vai trò (giáo viên/học sinh) |
| class | VARCHAR(10) | Lớp (10A1, 11B2, v.v.) |
| created\_at | DATETIME | Thời gian tạo |

**Bảng exams (Bài thi):**

| **Cột** | **Kiểu dữ liệu** | **Mô tả** |
| --- | --- | --- |
| id | INT (PK, AI) | ID bài thi |
| title | VARCHAR(255) | Tiêu đề bài thi |
| subject | VARCHAR(50) | Môn học (Toán, Lý, Hóa) |
| class | VARCHAR(10) | Lớp áp dụng (10A1, 11B2) |
| start\_time | DATETIME | Thời gian bắt đầu |
| end\_time | DATETIME | Thời gian kết thúc |
| duration | INT | Thời gian làm bài (phút) |
| created\_by | INT (FK → users) | ID giáo viên tạo bài thi |
| created\_at | DATETIME | Thời gian tạo |

**Bảng questions (Câu hỏi trắc nghiệm):**

| **Cột** | **Kiểu dữ liệu** | **Mô tả** |
| --- | --- | --- |
| id | INT (PK, AI) | ID câu hỏi |
| exam\_id | INT (FK → exams) | ID bài thi |
| content | TEXT | Nội dung câu hỏi (LaTeX) |
| options | JSON | 4 đáp án (A, B, C, D) |
| correct\_answer | CHAR(1) | Đáp án đúng (A/B/C/D) |
| points | INT | Số điểm (mặc định 1) |
| difficulty | ENUM('easy', 'medium', 'hard') | Độ khó |
| topic | VARCHAR(50) | Chủ đề (Phương trình, Hóa hữu cơ) |
| created\_at | DATETIME | Thời gian tạo |

#### **Bảng submissions (Bài làm của thí sinh)**

| **Cột** | **Kiểu dữ liệu** | **Mô tả** |
| --- | --- | --- |
| id | INT (PK, AI) | ID bài làm |
| exam\_id | INT (FK → exams) | ID bài thi |
| user\_id | INT (FK → users) | ID thí sinh |
| start\_time | DATETIME | Thời gian bắt đầu |
| end\_time | DATETIME | Thời gian nộp bài |
| score | FLOAT | Điểm số |
| status | ENUM('in\_progress', 'completed') | Trạng thái bài làm |

#### **Bảng answers (Câu trả lời của thí sinh)**

| **Cột** | **Kiểu dữ liệu** | **Mô tả** |
| --- | --- | --- |
| id | INT (PK, AI) | ID câu trả lời |
| submission\_id | INT (FK → submissions) | ID bài làm |
| question\_id | INT (FK → questions) | ID câu hỏi |
| answer | CHAR(1) | Đáp án của thí sinh (A/B/C/D) |
| is\_correct | BOOLEAN | Đúng/Sai |

### **3. Mẹo triển khai**

* **Bảo mật**:
  + Dùng JWT để phân quyền: Giáo viên (teacher) và học sinh (student).
  + Học sinh chỉ thấy bài thi của lớp mình (thêm điều kiện WHERE class = ? trong truy vấn).
* **Hiệu suất**:
  + Tạo index cho cột class trong bảng exams và users.
  + Dùng KaTeX thay vì MathJax để render nhanh hơn.
* **Chống gian lận**:
  + Xáo trộn câu hỏi và đáp án trong backend

nếu đã cài navicat premium, nodeJs thì bắt đầu tạo dự án

1.Tạo thư mục dự án:

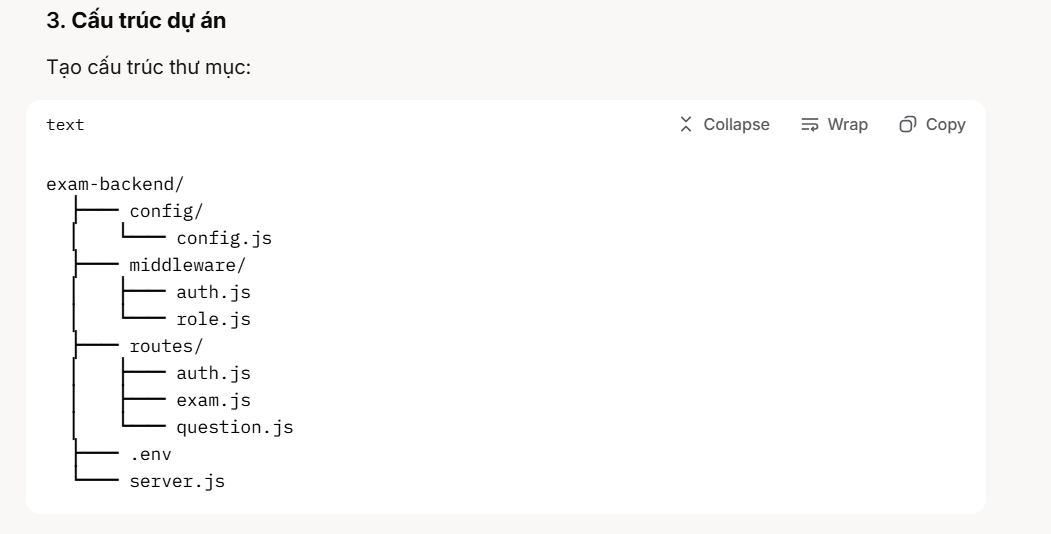
mkdir exam-backend

cd exam-backend

npm init -y

2.Cài đặt các thư viện cần thiết

npm install express sequelize sequelize-cli mysql2 jsonwebtoken bcryptjs dotenv cors





**config/config.json**: Cấu hình kết nối database.

**models/**: Định nghĩa các model (bảng).

**migrations/**: Quản lý thay đổi cấu trúc database.

**seeders/**: Thêm dữ liệu mẫu.

#### **Cấu hình kết nối database**

Mở config/config.json và chỉnh sửa:

{ "development": { "username": "root", "password": "your\_mysql\_password", "database": "exam\_db", "host": "127.0.0.1", "dialect": "mysql" }, "test": { "username": "root", "password": "your\_mysql\_password", "database": "exam\_db\_test", "host": "127.0.0.1", "dialect": "mysql" }, "production": { "username": "root", "password": "your\_mysql\_password", "database": "exam\_db\_production", "host": "127.0.0.1", "dialect": "mysql" } }

**Tạo model cho bảng exams:**

npx sequelize-cli model:generate --name Exam --attributes title:string,subject:string,class:string,startTime:date,endTime:date,duration:integer,createdBy:integer,createdAt:date

Lệnh này sẽ tạo:

* File models/exam.js (model).
* File trong migrations/ (để tạo bảng).

Mở models/exam.js và chỉnh sửa:

'use strict'; const { Model } = require('sequelize'); module.exports = (sequelize, DataTypes) => { class Exam extends Model { static associate(models) { Exam.belongsTo(models.User, { foreignKey: 'createdBy', as: 'creator' }); Exam.hasMany(models.Question, { foreignKey: 'examId', as: 'questions' }); } } Exam.init( { title: DataTypes.STRING, subject: DataTypes.STRING, class: DataTypes.STRING, startTime: DataTypes.DATE, endTime: DataTypes.DATE, duration: DataTypes.INTEGER, createdBy: DataTypes.INTEGER, createdAt: DataTypes.DATE, }, { sequelize, modelName: 'Exam', } ); return Exam; };

#### **Tạo model cho các bảng khác**

Tương tự, tạo model cho các bảng còn lại:

User:

npx sequelize-cli model:generate --name User --attributes studentId:string,fullName:string,email:string,password:string,role:enum:class:string,createdAt:date

'use strict'; const { Model } = require('sequelize'); module.exports = (sequelize, DataTypes) => { class User extends Model { static associate(models) { User.hasMany(models.Exam, { foreignKey: 'createdBy', as: 'exams' }); } } User.init( { studentId: DataTypes.STRING, fullName: DataTypes.STRING, email: DataTypes.STRING, password: DataTypes.STRING, role: DataTypes.ENUM('teacher', 'student'), class: DataTypes.STRING, createdAt: DataTypes.DATE, }, { sequelize, modelName: 'User', } ); return User; };

Cập nhật route exam.js để dùng Sequelize

Tạo file .env

PORT=3000 JWT\_SECRET=your\_jwt\_secret\_key\_here

MYSQL\_HOST=localhost MYSQL\_USER=root

MYSQL\_PASSWORD=your\_mysql\_password (nếu có)

MYSQL\_DATABASE=exam\_db

config/config.js

require('dotenv').config();

module.exports = {

port: process.env.PORT || 3000,

jwtSecret: process.env.JWT\_SECRET || 'default\_secret',

mysql: { host: process.env.MYSQL\_HOST || 'localhost', user: process.env.MYSQL\_USER || 'root', password: process.env.MYSQL\_PASSWORD || '', database: process.env.MYSQL\_DATABASE || 'exam\_db', }, };

**Tạo file server.js**

const express = require('express'); const cors = require('cors'); const authRoutes = require('./routes/auth'); const examRoutes = require('./routes/exam'); const questionRoutes = require('./routes/question'); const { port } = require('./config/config'); const { sequelize } = require('./models'); const app = express(); *// Middleware* app.use(cors()); app.use(express.json()); *// Kết nối Sequelize* sequelize.sync({ force: false }).then(() => { console.log('Database synced'); }); *// Routes* app.use('/api/auth', authRoutes); app.use('/api/exams', examRoutes); app.use('/api/questions', questionRoutes); *// Khởi động server* app.listen(port, () => { console.log(`Server running on port ${port}`); });

Bước 2: Tạo middleware xác thực (middleware/auth.js)

const jwt = require('jsonwebtoken');

const { jwtSecret } = require('../config/config');

const authMiddleware = (req, res, next) => {

const token = req.header('Authorization')?.replace('Bearer ', '');

if (!token) {

return res.status(401).json({ message: 'No token, authorization denied' });

}

try {

const decoded = jwt.verify(token, jwtSecret);

req.user = decoded;

next();

} catch (err) {

res.status(401).json({ message: 'Token is not valid' });

}

};

module.exports = authMiddleware;

Bước 3: Tạo middleware phân quyền (middleware/role.js)

const roleMiddleware = (role) => (req, res, next) => {

if (req.user.role !== role) {

return res.status(403).json({ message: 'Access denied' });

}

next();

};

module.exports = roleMiddleware;

Bước 4: Tạo route đăng nhập (routes/auth.js)

const express = require('express');

const router = express.Router();

const bcrypt = require('bcryptjs');

const jwt = require('jsonwebtoken');

const mysql = require('mysql2/promise');

const { mysql, jwtSecret } = require('../config/config');

// Kết nối MySQL

const pool = mysql.createPool(mysql);

// Đăng nhập

router.post('/login', async (req, res) => {

const { studentId, email, password } = req.body;

try {

let user;

if (studentId) {

[user] = await pool.query('SELECT \* FROM users WHERE student\_id = ?', [studentId]);

} else if (email) {

[user] = await pool.query('SELECT \* FROM users WHERE email = ?', [email]);

}

if (!user[0]) {

return res.status(400).json({ message: 'Invalid credentials' });

}

const isMatch = await bcrypt.compare(password, user[0].password);

if (!isMatch) {

return res.status(400).json({ message: 'Invalid credentials' });

}

const payload = {

id: user[0].id,

role: user[0].role,

class: user[0].class,

};

const token = jwt.sign(payload, jwtSecret, { expiresIn: '1h' });

res.json({ token });

} catch (err) {

res.status(500).json({ message: 'Server error' });

}

});

// Tạo tài khoản học sinh (chỉ giáo viên)

router.post('/register-student', authMiddleware, roleMiddleware('teacher'), async (req, res) => {

const { studentId, fullName, class: studentClass, password } = req.body;

try {

const [existing] = await pool.query('SELECT \* FROM users WHERE student\_id = ?', [studentId]);

if (existing[0]) {

return res.status(400).json({ message: 'Student ID already exists' });

}

const hashedPassword = await bcrypt.hash(password, 10);

await pool.query(

'INSERT INTO users (student\_id, full\_name, class, password, role, created\_at) VALUES (?, ?, ?, ?, ?, ?)',

[studentId, fullName, studentClass, hashedPassword, 'student', new Date()]

);

res.json({ message: 'Student registered successfully' });

} catch (err) {

res.status(500).json({ message: 'Server error' });

}

});

module.exports = router;

Bước 5: Tạo route quản lý bài thi (routes/exam.js)

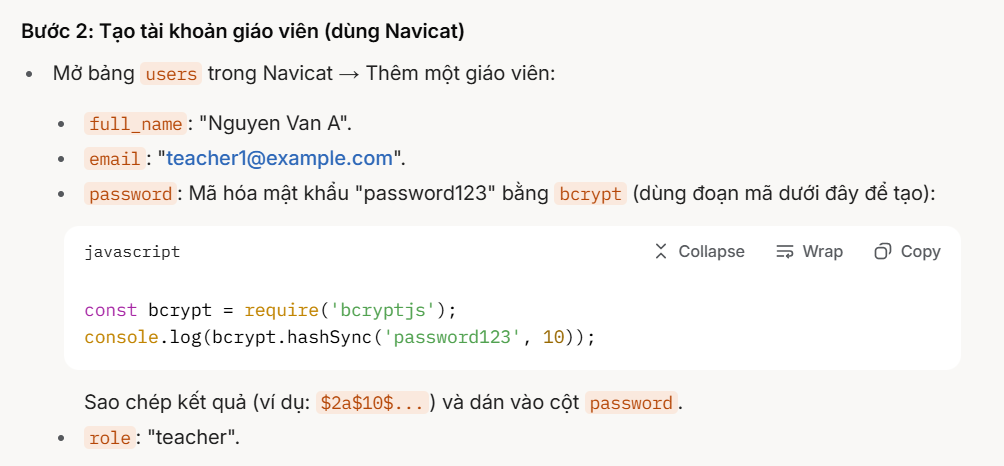
const express = require('express'); const router = express.Router(); const authMiddleware = require('../middleware/auth'); const roleMiddleware = require('../middleware/role'); const { Exam } = require('../models'); *// Tạo bài thi (giáo viên)* router.post('/', authMiddleware, roleMiddleware('teacher'), async (req, res) => { const { title, subject, class: examClass, startTime, endTime, duration } = req.body; try { const exam = await Exam.create({ title, subject, class: examClass, startTime, endTime, duration, createdBy: req.user.id, createdAt: new Date(), }); res.json({ message: 'Exam created successfully', exam }); } catch (err) { res.status(500).json({ message: 'Server error' }); } }); *// Lấy danh sách bài thi* router.get('/', authMiddleware, async (req, res) => { try { let exams; if (req.user.role === 'student') { exams = await Exam.findAll({ where: { class: req.user.class, startTime: { [Op.lte]: new Date() }, endTime: { [Op.gte]: new Date() }, }, }); } else { exams = await Exam.findAll({ where: { createdBy: req.user.id }, }); } res.json(exams); } catch (err) { res.status(500).json({ message: 'Server error' }); } }); module.exports = router;

Bước 6: Tạo route quản lý câu hỏi (routes/question.js)

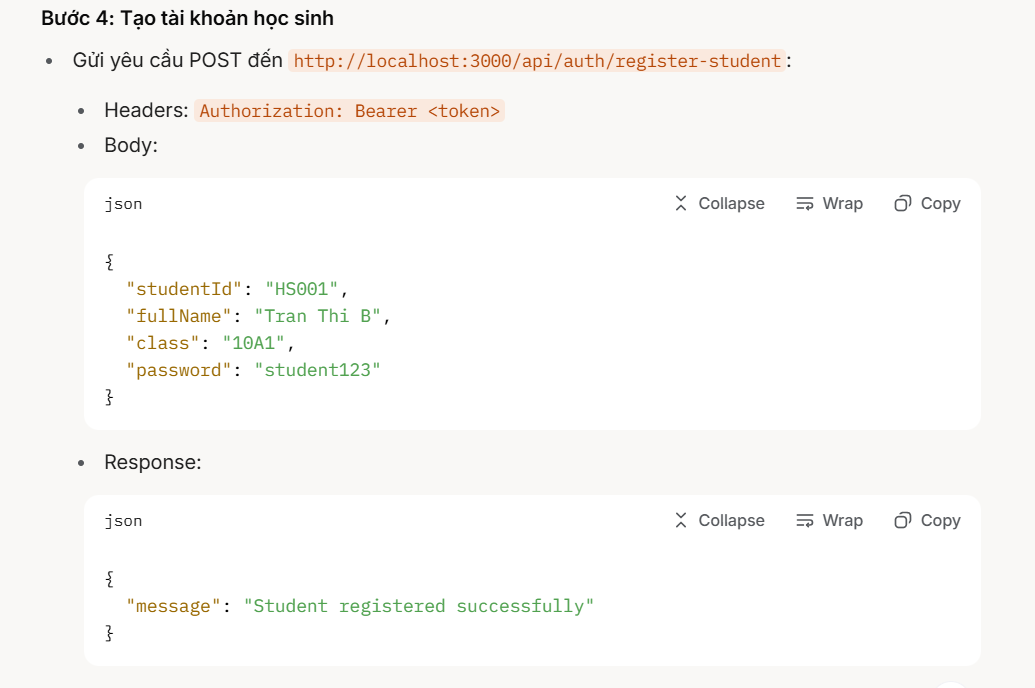
const express = require('express'); const router = express.Router(); const authMiddleware = require('../middleware/auth'); const roleMiddleware = require('../middleware/role'); const mysql = require('mysql2/promise'); const { mysql } = require('../config/config'); const pool = mysql.createPool(mysql); *// Thêm câu hỏi (giáo viên)* router.post('/', authMiddleware, roleMiddleware('teacher'), async (req, res) => { const { content, options, correctAnswer, points, difficulty, topic, examId } = req.body; try { await pool.query( 'INSERT INTO questions (exam\_id, content, options, correct\_answer, points, difficulty, topic, created\_at) VALUES (?, ?, ?, ?, ?, ?, ?, ?)', [examId, content, JSON.stringify(options), correctAnswer, points, difficulty, topic, new Date()] ); res.json({ message: 'Question added successfully' }); } catch (err) { res.status(500).json({ message: 'Server error' }); } }); module.exports = router;

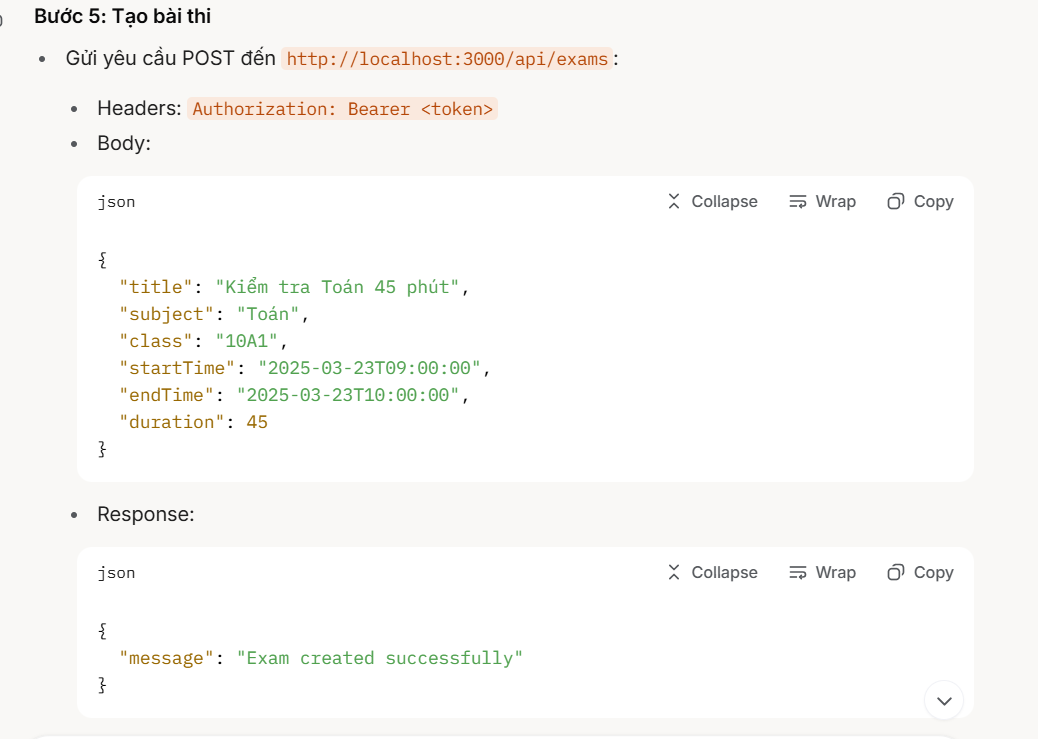
Kiểm tra backend

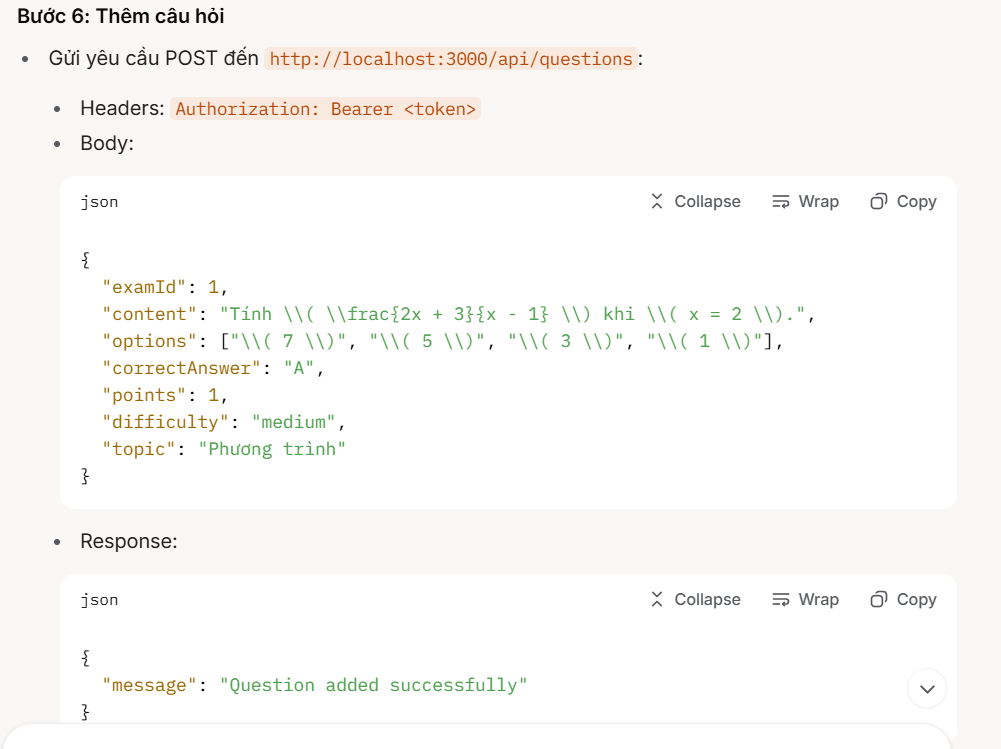
Chạy server: node server.Js hay npm run dev gì đó:

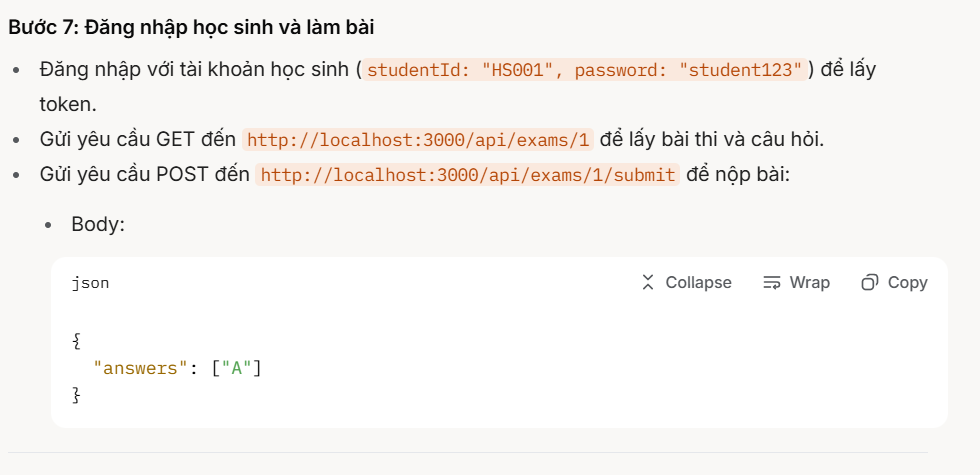












**Frontend:**

**Sử dụng KaTeX** (nhẹ và nhanh hơn MathJax): trong Vue.Js