ASSIGNMENT 1

1.Tổng Quan Bài Tập

Bài tập này hướng dẫn sinh viên xây dựng **backend API** cho một ứng dụng Quiz (**trắc nghiệm**) bằng **Node.js, Express.js, MongoDB, và Mongoose**. Ứng dụng này sẽ giúp người dùng tạo và quản lý các bài trắc nghiệm (Quiz). Mỗi bài Quiz sẽ chứa nhiều câu hỏi, và hệ thống sẽ hỗ trợ việc thêm, chỉnh sửa, xóa và lấy dữ liệu Quiz. API này có thể dùng cho các hệ thống học tập trực tuyến hoặc ứng dụng thi thử.

-**Ứng dụng sẽ hỗ trợ:**

- **Tạo, cập nhật, lấy dữ liệu và xóa bài quiz & câu hỏi** (CRUD với REST API).
- **Sử dụng Mongoose Models** để kết nối với MongoDB.
- **Tuân theo mô hình MVC** để tổ chức mã nguồn.
- **Sử dụng Express Router** để tạo module quản lý routes.
- **Dùng Mongoose Populate** để lấy dữ liệu liên kết giữa **quizzes và questions**.

Kết quả mong đợi: Một API hoàn chỉnh có thể quản lý các **Quiz và Question** bằng cách sử dụng **GET, POST, PUT, DELETE** với MongoDB.

2.Tóm Tắt Các Bước Thực Hiện

Bước	Nội dung	Mô tả
Bước 1	Cài đặt dự án Node.js và MongoDB.	Khởi tạo dự án bằng `npm init`, cài đặt các package cần thiết.
Bước 2	Tạo Mongoose Models (Quiz.js, Question.js).	Định nghĩa schema cho Quiz và Question, liên kết dữ liệu.
Bước 3	Cấu hình Express Router (quizRoutes.js, questionRoutes.js).	Xây dựng routes cho API theo mô hình MVC.
Bước 4	Viết API RESTful với GET, POST, PUT, DELETE.	Cài đặt các API cần thiết để quản lý Quiz và Question.
Bước 5	Kiểm tra API bằng Postman.	Gửi request thử nghiệm API, kiểm tra response JSON.

3. Yêu Cầu Cần Đạt Được Của Bài Tập

a) Cấu Trúc Dự Án Chuẩn MVC

- Tổ chức mã nguồn theo mô hình **MVC (Model-View-Controller)** để dễ bảo trì.
- Tách biệt **models, routes, controllers** thay vì viết tất cả trong một file.

b) Xây Dựng Mongoose Schema Hợp Lý

- Thiết kế **Mongoose Schema** cho **Quiz** và **Question**.
- **Liên kết dữ liệu** giữa Quiz và Question bằng **Mongoose Populate**.

✓ **Quiz Schema** phải có:

- `title`: Tiêu đề bài quiz.
- `description`: Mô tả về bài quiz.
- `questions`: Danh sách ID của các câu hỏi.

✓ **Question Schema** phải có:

- `text`: Nội dung câu hỏi.
- `options`: Danh sách các lựa chọn.

- `correctAnswerIndex`: Đáp án đúng.
- `keywords`: Danh sách từ khóa để lọc câu hỏi.

c) Xây Dựng API Theo RESTful

- Cung cấp **các API thực hiện CRUD** (Create, Read, Update, Delete) cho **Quiz và Question**.
- Sử dụng **Express Router** để quản lý các routes.

✓ Danh sách API cần có:

HTTP Method	Endpoint	Chức năng
GET	`/quizzes`	Lấy danh sách tất cả quiz (có câu hỏi).
POST	`/quizzes`	Thêm một bài quiz mới.
DELETE	`/quizzes/:id`	Xóa một quiz.
POST	`/quizzes/:quizId/question`	Thêm một câu hỏi vào quiz.
POST	`/quizzes/:quizId/questions`	Thêm nhiều câu hỏi vào quiz.
GET	`/quizzes/:quizId/populate`	Lọc câu hỏi có chứa từ "capital".

d) Kết Nối Thành Công Với MongoDB

- API phải có thể **kết nối MongoDB** bằng **Mongoose**.
- Dùng **MongoDB Atlas hoặc MongoDB cục bộ** (`mongodb://localhost:27017/quizdb`).

e) Kiểm Tra API Bằng Postman

- Tất cả API phải hoạt động đúng khi test bằng **Postman**.
- Đảm bảo **response JSON đúng format** và xử lý lỗi nếu có.

f) Xử Lý Lỗi Và Middleware

- **Bảo vệ API khỏi lỗi server** (dùng `try/catch`).
- Tạo **middleware cho xử lý lỗi** (`errorHandler`).

4. Yêu Cầu Nộp Bài

Sinh viên cần **hoàn thành bài tập và nộp đầy đủ các tài liệu sau đây**:

a. Source Code Dự Án

- **Thư mục dự án đầy đủ**, bao gồm các file sau:
  - `server.js` (hoặc `index.js`) – File khởi động ứng dụng.
  - `models/` – Thư mục chứa các **Mongoose Schema** (`Quiz.js`, `Question.js`).
  - `routes/` – Thư mục chứa **Express Router** (`quizRoutes.js`, `questionRoutes.js`).
  - `.env` (không bắt buộc nộp, chỉ cần ghi trong document nếu có biến môi trường).
- **Yêu cầu về code:**
  - **Mỗi file phải có phần comment đầu file** ghi rõ:

```
/**
 * @file quizRoutes.js
 * @author [Họ và Tên Sinh Viên] - [Mã Sinh Viên]
 * @date [Ngày hoàn thành]
 * @description API routes cho quản lý quiz.
 */
```

  - **Code phải có comment giải thích các function quan trọng.**

b. File Document Báo Cáo (.PDF hoặc .DOCX)

Sinh viên cần nộp một **file tài liệu báo cáo** chứa **hình ảnh minh họa đầy đủ**, bao gồm:

- **Cấu trúc thư mục dự án**
  - **Chụp ảnh toàn bộ thư mục gốc** (`server.js`, `routes/`, `models/`...).
- **Code chính trong dự án**
  - **Chụp màn hình code của các file chính** (server.js, models, routes).

- **Chụp phần comment tên tác giả** trong mỗi file.

- **Kết quả thực thi API trên Postman**
  - **Chụp màn hình Postman** khi thực hiện **GET, POST, DELETE** API.
  - **Hiển thị response JSON đúng format**.
- **Hướng dẫn chạy dự án**
  - Hướng dẫn cách **cài đặt & chạy API** (`npm install`, `node server.js`).

5. Hình Thức Nộp Bài

- **Gửi bài tập dưới dạng file nén (.zip/.rar)**, gồm:
  - **Thư mục source code đầy đủ**.
  - **File tài liệu báo cáo (.PDF hoặc .DOCX)**.
- **Đặt tên file nén theo format:**
  - **`[MãSV]_[Tên]_Assignment1.zip`**
    Ví dụ: `SE12345_NguyenVanA_Assignment1.zip`
- **Tùy chọn:** Video ngắn quay lại thao tác test API.

- **Hạn chót nộp bài:** theo lịch Edunext

6. Lưu Ý Quan Trọng

X Bài nộp không đầy đủ hoặc thiếu file báo cáo sẽ bị trừ điểm.

X Mọi hành vi sao chép code sẽ bị xử lý theo quy định của nhà trường.

X Sinh viên cần kiểm tra kỹ lưỡng trước khi nộp bài.