**Lộ trình NodeJS (ExpressJS)**

**Buổi 1: Giới thiệu Tổng quan về NodeJS và Môi trường**

* + NodeJS là gì? Giải thích các khái niệm cốt lõi: Event-Driven (điều khiển bởi sự kiện) và Non-blocking I/O (I/O không chặn).
  + Kiến trúc của NodeJS: Vai trò và sự phối hợp giữa **V8 Engine** (của Google Chrome) và **Libuv** (thư viện C++ xử lý bất đồng bộ).
  + **Event Loop**: Diễn giải chi tiết cách vòng lặp sự kiện xử lý các tác vụ, phân biệt call stack, callback queue, và các phase của nó.
  + Hướng dẫn cài đặt môi trường phát triển: Cài đặt Node.js, NPM/Yarn, VS Code và các extensions hữu ích (ví dụ: Prettier, ESLint).
  + Làm quen và sử dụng NodeJS REPL (Read-Eval-Print Loop) để chạy thử nhanh các đoạn mã JavaScript.

**Buổi 2: Module và Lập trình Bất đồng bộ Cơ bản**

* + Hệ thống Module trong NodeJS: So sánh chi tiết hai cơ chế **CommonJS** (require, module.exports) và **ES Modules** (import, export).
  + Giới thiệu các **Module Cốt lõi** (Core Modules) quan trọng:
    - fs (File System): Các phương thức đọc/ghi file, bao gồm cả phiên bản đồng bộ (blocking) và bất đồng bộ (non-blocking).
    - path: Các tiện ích để làm việc với đường dẫn file và thư mục một cách an toàn trên các hệ điều hành khác nhau.
  + Ôn tập về Lập trình Bất đồng bộ: Nhắc lại về Callbacks, minh họa vấn đề "Callback Hell" và giới thiệu **Promises** như một giải pháp thay thế ưu việt.

**Buổi 3: Làm chủ Async/Await, Streams và Events**

* + **Async/Await**: Giới thiệu cú pháp "syntactic sugar" này để làm việc với Promises, giúp code trông giống như code đồng bộ, dễ đọc và dễ bảo trì hơn.
  + Xử lý lỗi chuyên nghiệp khi dùng Async/Await bằng khối lệnh try...catch.
  + **Streams**: Giải thích khái niệm Stream là gì và tại sao nó cực kỳ hiệu quả để xử lý các tập dữ liệu lớn (như file video, log file) mà không tốn nhiều bộ nhớ.
  + **Event Emitter**: Tìm hiểu về mô hình publish-subscribe được tích hợp sẵn trong NodeJS thông qua class EventEmitter của module events.

**Buổi 4: Bắt đầu với Express.js và Routing**

* + Giới thiệu **Express.js**: Tại sao nó là framework web tối giản và phổ biến nhất cho NodeJS?
  + Hướng dẫn cài đặt Express và cấu trúc một thư mục dự án Express cơ bản.
  + **Routing cơ bản**: Tìm hiểu cách định nghĩa các route cho các phương thức HTTP khác nhau: app.get(), app.post(), app.put(), app.delete().
  + Đối tượng request và response: Khám phá cách lấy dữ liệu từ request (req) và cách gửi phản hồi về cho client (res) với các phương thức như res.send(), res.json(), res.status().
  + Phục vụ các file tĩnh (HTML, CSS, JavaScript phía client, hình ảnh) bằng middleware express.static.

**Buổi 5: Express Nâng cao: Middleware và Cấu trúc Router**

* + **Middleware là gì?**: Giải thích sâu về khái niệm "phần mềm trung gian" trong Express, là các hàm có quyền truy cập vào req, res và hàm next() để chuyển sang middleware tiếp theo.
  + Sử dụng các middleware của bên thứ ba phổ biến: morgan (để ghi log request), cors (để xử lý Cross-Origin Resource Sharing), express.json() (để đọc req.body dạng JSON).
  + Tổ chức code khoa học và dễ quản lý hơn bằng cách sử dụng express.Router để nhóm các route liên quan.
  + Cách lấy các loại dữ liệu từ request: Route parameters (req.params), Query strings (req.query), và Request body (req.body).

**Buổi 6: Giới thiệu SQL và Thiết kế Database với MySQL**

* + Giới thiệu về Cơ sở dữ liệu Quan hệ (RDBMS) và tại sao chúng vẫn rất phổ biến.
  + Các bước để thiết kế một CSDL tốt: Xác định thực thể (entities), thuộc tính (attributes), và các mối quan hệ (relationships).
  + Hướng dẫn thiết kế CSDL cụ thể cho dự án Blog, bao gồm các bảng users, posts, categories và các mối quan hệ giữa chúng (1-n, n-n).
  + Giới thiệu các kiểu dữ liệu phổ biến trong MySQL: INT, VARCHAR, TEXT, DATETIME, BOOLEAN.

**Buổi 7: NodeJS và MySQL: Kết nối và CRUD**

* + Hướng dẫn cách kết nối ứng dụng NodeJS đến MySQL Server bằng thư viện mysql2.
  + Giới thiệu về **Connection Pooling**: Tại sao nó là một kỹ thuật bắt buộc phải có trong môi trường production để quản lý kết nối hiệu quả và tăng hiệu năng.
  + Thực hiện các câu lệnh SQL cơ bản (INSERT, SELECT, UPDATE, DELETE) từ ứng dụng NodeJS.
  + Xây dựng cấu trúc code để tương tác với CSDL một cách gọn gàng, ví dụ tạo một module database.js để quản lý kết nối.

**Buổi 8: MySQL Nâng cao và Bảo mật**

* + Tìm hiểu về các phép nối bảng (JOINs): INNER JOIN (lấy phần chung) và LEFT JOIN (lấy toàn bộ bảng trái).
  + Sử dụng các hàm tổng hợp (COUNT, SUM, AVG) kết hợp với mệnh đề GROUP BY để thống kê dữ liệu.
  + **SQL Injection**: Giải thích chi tiết đây là loại tấn công gì, mức độ nguy hiểm và cách hacker thực hiện nó.
  + **Prepared Statements**: Trình bày đây là cơ chế phòng chống SQL Injection hiệu quả nhất bằng cách tách câu lệnh SQL và dữ liệu đầu vào.

**Buổi 9: Xác thực với JWT (Authentication)**

* + Phân biệt rõ hai khái niệm thường bị nhầm lẫn: **Authentication** (Xác thực - Bạn là ai?) và **Authorization** (Phân quyền - Bạn được làm gì?).
  + Tìm hiểu về kỹ thuật băm mật khẩu an toàn bằng thư viện **bcrypt**. Giải thích tại sao không bao giờ được lưu mật khẩu dạng plain text.
  + **JSON Web Token (JWT)**: Phân tích cấu trúc 3 phần của một token (Header, Payload, Signature) và luồng hoạt động của nó trong việc xác thực stateless.

**Buổi 10: Phân quyền và Bảo vệ Route (Authorization)**

* + Hướng dẫn cách viết một middleware auth (hay verifyToken) để kiểm tra tính hợp lệ của JWT được gửi lên trong header của request.
  + Sau khi xác thực token, middleware sẽ lấy thông tin người dùng (ví dụ: user ID) từ payload và gắn vào đối tượng request để các xử lý sau có thể sử dụng.
  + Áp dụng middleware này để bảo vệ các route yêu cầu phải đăng nhập mới được truy cập.
  + Mở rộng với khái niệm Phân quyền dựa trên vai trò (Role-Based Access Control - RBAC), ví dụ admin vs. user.

**Buổi 11: Xử lý Lỗi và Logging Chuyên sâu**

* + Tìm hiểu cơ chế xử lý lỗi mặc định của Express và hạn chế của nó.
  + Xây dựng một **middleware xử lý lỗi tập trung**: một middleware đặc biệt có 4 tham số (error, req, res, next) để bắt tất cả các lỗi trong ứng dụng.
  + Hướng dẫn tạo các lớp Error tùy chỉnh (ví dụ: ApiError) kế thừa từ lớp Error gốc để gửi về các response lỗi có cấu trúc và thông điệp rõ ràng.
  + Ghi log hiệu quả với thư viện **Winston**: cấu hình các "transports" (ghi ra console, file) và các định dạng log khác nhau.

**Buổi 12: Upload File với Multer và Cloud Storage**

* + Tìm hiểu về loại request multipart/form-data được sử dụng để upload file.
  + Sử dụng thư viện **Multer** để xử lý các request này, lưu file tạm thời vào bộ nhớ hoặc vào một thư mục trên đĩa.
  + So sánh ưu và nhược điểm giữa việc lưu file trên server (local storage) và trên các dịch vụ lưu trữ đám mây (Cloud Storage).
  + Giới thiệu và hướng dẫn tích hợp với **Cloudinary**, một dịch vụ phổ biến cho việc lưu trữ và xử lý hình ảnh/video.

**Buổi 13: Ứng dụng Real-time với Socket.IO**

* + So sánh các kỹ thuật giao tiếp real-time: HTTP Polling, Long-Polling, Server-Sent Events, và **WebSockets**.
  + Giới thiệu **Socket.IO** như một thư viện mạnh mẽ xây dựng trên WebSockets, cung cấp khả năng tự động fallback và nhiều tính năng tiện lợi.
  + Các sự kiện và phương thức cốt lõi: connection, disconnect, socket.emit(), socket.on(), io.emit().
  + Tìm hiểu về **Broadcasting** (gửi cho tất cả trừ người gửi) và **Rooms** (gửi cho một nhóm client cụ thể).

**Buổi 14: Testing trong NodeJS**

* + Tại sao phải viết test? Tầm quan trọng của việc đảm bảo chất lượng phần mềm.
  + Giới thiệu Kim tự tháp testing (Unit, Integration, E2E).
  + **Unit Testing** với framework **Jest**: Hướng dẫn cách kiểm thử các hàm, module một cách riêng rẽ.
  + **Integration Testing** với thư viện **Supertest**: Hướng dẫn cách kiểm thử toàn bộ luồng của một API endpoint, từ request đến response.
  + Thiết lập một CSDL riêng biệt cho môi trường testing để tránh ảnh hưởng đến dữ liệu development.

**Buổi 15: Tối ưu Hiệu năng và Chuẩn bị Production**

* + Quản lý cấu hình cho các môi trường khác nhau (development, staging, production) một cách an toàn bằng file .env và thư viện **dotenv**.
  + **Tối ưu CSDL**: Tìm hiểu lại về tầm quan trọng của việc đánh chỉ mục (**Indexing**) trong MySQL để tăng tốc độ truy vấn SELECT.
  + Giới thiệu về **PM2**: một trình quản lý tiến trình mạnh mẽ cho NodeJS trong môi trường production, giúp ứng dụng tự khởi động lại khi gặp lỗi, chạy trên nhiều nhân CPU, và giám sát hiệu năng.

**Buổi 16: Triển khai và Tổng kết Phần chính**

* + So sánh các lựa chọn triển khai phổ biến: PaaS (Platform as a Service) như Render, Heroku và IaaS/VPS (Infrastructure as a Service) như AWS EC2, DigitalOcean.
  + **Demo trực tiếp**: Hướng dẫn chi tiết từng bước để triển khai ứng dụng NodeJS + MySQL lên nền tảng **Render.com** (vì Render hỗ trợ cả NodeJS service và MySQL database miễn phí).

**Nâng cao - NoSQL với MongoDB**

**Buổi 17: Giới thiệu NoSQL và MongoDB**

* + Giới thiệu lại về NoSQL như một chủ đề nâng cao. Khi nào thì dự án của bạn nên cân nhắc sử dụng NoSQL?
  + Giới thiệu sâu về **MongoDB** và kiến trúc Document linh hoạt của nó.
  + Cài đặt MongoDB (bản Community trên máy local) và đăng ký **MongoDB Atlas** (dịch vụ cloud).
  + Hướng dẫn sử dụng các lệnh CRUD cơ bản (insertOne, find, updateOne, deleteOne) trong mongosh (MongoDB Shell).

**Buổi 18: Mongoose và Modeling Dữ liệu NoSQL**

* + Giới thiệu **Mongoose** như một lớp ODM (Object Data Modeling) giúp làm việc với MongoDB một cách có cấu trúc hơn.
  + Tìm hiểu về **Schema** (định nghĩa cấu trúc) và **Model** (đối tượng để tương tác với collection).
  + Cách định nghĩa validation, giá trị mặc định, và các middleware (pre/post hooks) trong Mongoose Schema.
  + Modeling Quan hệ trong NoSQL: **Embedding** (lồng document) vs. **Referencing** (tham chiếu bằng ID) và phương thức .populate() để lấy dữ liệu tham chiếu.

**Lộ trình NodeJS (NestJS)**

**Buổi 1 & 2: Nền tảng NodeJS**

* *(Giữ nguyên)* Vẫn cần 2 buổi đầu để nắm vững các khái niệm bất đồng bộ (Async/Await), Event Loop, Streams, fs, path. Đây là kiến thức nền không phụ thuộc vào framework.

**Buổi 3: Giới thiệu NestJS và các Khái niệm Cốt lõi**

* NestJS là gì? So sánh với Express. Giới thiệu **NestJS CLI**. Cấu trúc một dự án NestJS. Giới thiệu 3 thành phần cốt lõi: **Modules, Controllers, Services**.

**Buổi 4: Dependency Injection và Providers**

Giải thích sâu về **Dependency Injection (DI)** - trái tim của NestJS. **Providers** là gì? Vòng đời của Provider (Scope). Cách NestJS quản lý các dependencies.

**Buổi 5: Validation với DTO và Pipes**

Giới thiệu **DTO (Data Transfer Object)**. Tại sao nên dùng DTO. Sử dụng các thư viện class-validator và class-transformer. Giới thiệu về **Pipes** trong NestJS, đặc biệt là ValidationPipe.

**Buổi 6: Thiết kế CSDL và Giới thiệu ORM**

* *(Giữ nguyên phần thiết kế CSDL)*. Giới thiệu về **ORM (Object-Relational Mapping)**. Giới thiệu **TypeORM**, ORM được hỗ trợ chính thức bởi NestJS.

**Buổi 7: Kết nối CSDL và Entities với TypeORM**

* Tích hợp NestJS với TypeORM qua module @nestjs/typeorm. Khái niệm **Entity**. Cách tạo các file entity tương ứng với các bảng CSDL.

**Buổi 8: CRUD với Repository Pattern**

* Tìm hiểu sâu về **Repository Pattern** do TypeORM cung cấp. Cách thực hiện các thao tác CRUD (find, findOne, create, save, update, delete).

**Buổi 9: Configuration Management**

* Sử dụng module @nestjs/config để quản lý biến môi trường một cách chuyên nghiệp, thay thế cho dotenv đơn thuần.

**Buổi 10: Authentication với Passport và Guards**

Giới thiệu module @nestjs/passport. Tích hợp Passport-jwt strategy. Giới thiệu khái niệm **Guards** trong NestJS để bảo vệ route.

**Buổi 11: Authorization (Phân quyền)**

* Xây dựng hệ thống phân quyền (RBAC) bằng cách tạo các **Custom Decorators** (ví dụ @Roles) và các **Guards** tùy chỉnh (RolesGuard).

**Buổi 12: Xử lý lỗi với Exception Filters**

* Giới thiệu hệ thống xử lý lỗi của NestJS với **Exception Filters**. Cách tạo một filter tùy chỉnh để bắt tất cả các lỗi và trả về response nhất quán.

**Buổi 13: Upload File với Interceptors**

Cách NestJS xử lý upload file bằng **Interceptors**. Sử dụng FileInterceptor (một wrapper của Multer) để xử lý multipart/form-data.

**Buổi 14: Real-time với Gateways**

Giới thiệu cách NestJS xây dựng ứng dụng real-time với **Gateways**, một lớp trừu tượng phía trên Socket.IO.

**Buổi 15: Testing trong NestJS**

Hệ thống testing tích hợp sẵn của NestJS. Cách viết Unit Test cho Services (với mocking providers) và E2E Test cho Controllers.

**Buổi 16: Triển khai ứng dụng NestJS**

* Các bước cần chuẩn bị để triển khai một ứng dụng NestJS (build project TypeScript ra JavaScript). Sử dụng PM2.

**Buổi 17: Giới thiệu NoSQL và MongoDB**

* *(Giữ nguyên)*

**Buổi 18: Tích hợp MongoDB với Mongoose trong NestJS**

Sử dụng module @nestjs/mongoose. Cách định nghĩa Mongoose Schema và đăng ký Model trong NestJS module.