Here's a structured 5-week development plan for your **Node.js Express backend for the Learning Management System (LMS)**.

**Week 1: Project Setup & Authentication System**

**Objectives**

* Initialize the project with Express.js, JavaScript, and necessary dependencies.
* Set up folder structure (controllers/, routes/, models/, middleware/, config/).
* Configure **ESLint, Prettier, Nodemon, and environment variables (.env)**.
* Connect **MongoDB (or PostgreSQL) using Mongoose/Sequelize**.
* Implement **JWT authentication** for both **admins and users**.
* Develop authentication endpoints:
  + **Admin Signup/Login/Logout**
  + **User Signup/Login/Logout**
  + **Email Verification**
  + **Forgot/Reset Password**
  + **Update Admin/User Details**
  + **Check Auth**
* Test authentication routes using **Postman**.

**Week 2: Learner and Course Management**

**Objectives**

* Create the **Learner model** and implement CRUD operations:
  + **Create Learner**
  + **Get All Learners**
  + **Get Learner by ID**
  + **Update Learner**
  + **Delete Learner**
* Create the **Course model** and implement CRUD operations:
  + **Create Course**
  + **Get All Courses**
  + **Get Course by ID**
  + **Update Course**
  + **Delete Course**
* Add role-based access control (RBAC) to ensure **only admins** can manage learners and courses.
* Validate request payloads using **Express Validator or Zod**.
* Implement **middleware for authentication and authorization**.
* Test the endpoints using **Postman**.

**Week 3: Invoice and Revenue Management**

**Objectives**

* Create the **Invoice model** and implement CRUD operations:
  + **Create Invoice**
  + **Get All Invoices**
  + **Get Invoice by ID**
  + **Update Invoice**
  + **Delete Invoice**
* Implement **revenue tracking** endpoints:
  + **Get Total Revenue**
  + **Get Revenue by Date Range**
* Ensure **only admins** can access revenue-related endpoints.
* Add pagination and filtering for invoices and revenue reports.
* Test all new endpoints using **Postman**.

**Week 4: API Documentation & Refinements**

**Objectives**

* Integrate **Swagger UI** for interactive API documentation:
  + Document all API endpoints in swagger.json or swagger.js.
  + Ensure **clear descriptions, request bodies, and responses** are included.
* Implement **rate limiting and security enhancements**:
  + **Helmet** for security headers.
  + **CORS** configuration.
  + **Rate limiting (express-rate-limit)**.
* Optimize database queries and response times.
* Fix any **bugs** and **refactor the code** for better maintainability.

**Week 5: Jest Testing & Final Deployment**

**Objectives**

* Write **Jest tests** for controllers:
  + **Authentication Controllers**
  + **Learner Controllers**
  + **Course Controllers**
  + **Invoice & Revenue Controllers**
* Use **Supertest** for integration testing.
* Achieve **high test coverage (80-90%)**.
* Set up **CI/CD pipeline** (GitHub Actions or another tool).
* Deploy the API to **Azure Web Services**.
* Perform **final end-to-end testing** before launch.

**Final Deliverables**

✅ Fully functional **LMS backend API**  
✅ Secured authentication system  
✅ Learner, Course, Invoice & Revenue management  
✅ **Swagger UI documentation**  
✅ **Jest unit & integration tests**  
✅ **Deployed API on Azure**