**Core Features of an LMS**

Here’s what a solid **MVP (Minimum Viable Product)** should include:

**👤 User Management**

* Sign up, login, logout (JWT-based auth).
* Roles: **Admin** (manage everything), **Member** (borrow/return books).

**📖 Book Management**

* Admin can **add, update, delete books**.
* Store book metadata: title, author, ISBN, category, publication year, availability status.
* Optional: upload cover image (start with local storage, later try Cloudinary).

**📦 Borrow & Return**

* Member can borrow a book (if available).
* Return a book → update availability.
* Track due dates, fines (optional advanced feature).

**🔍 Search & Browse**

* Members can **search books** (title, author, category).
* Implement **pagination, filtering, sorting** for book listings.

**🛠️ Extras (Intermediate Level)**

* API versioning (/api/v1/books).
* Input validation & error handling.
* Role-based authorization (Admin vs Member)
* File storage (Cloudinary, AWS S3). ·Logging & monitoring (Winston, Morgan, Prometheus). ·Security (rate limiting, input sanitization, HTTPS, CORS). Graph QL

•

Project skeleton + configs

•

Database + CRUD

•

Auth + validation

•

Pagination/filtering/sorting

•

File uploads

•

Logging (Winston, Morgan)

•

Security (Helmet, rate limiting, sanitization)

•

Deployment (Docker, CI/CD)

**🛠️ Step-by-Step Setup of a NestJS Project**

**2. Setup Core Configurations**

* Add **Config Module** for .env:

npm install @nestjs/config

* Create .env file (DB URL, JWT secret, etc.).
* Use ConfigModule.forRoot({ isGlobal: true }) in app.module.ts.

**3. Database & ORM**

* Choose ORM (Prisma or TypeORM). Example: Prisma:

npm install prisma @prisma/client --save-dev

npx prisma init

* Define schema (Users, Books, BorrowRecords).
* Run migrations.

**4. Modules, Controllers & Services**

* Generate modules for **users, books, auth, borrow-return**:

nest g module books

nest g controller books

nest g service books

* Implement **basic CRUD**.

**5. Validation & Error Handling**

* Install:

npm install class-validator class-transformer

* Enable global validation pipe:

app.useGlobalPipes(new ValidationPipe({ whitelist: true }));

**6. Authentication & Authorization**

* Install JWT + Passport:

npm install @nestjs/jwt @nestjs/passport passport passport-jwt bcrypt

* Add:
  + **Signup/login** endpoints.
  + **JWT strategy**.
  + **Role-based guards** (Admin vs Member).

**7. Pagination, Filtering, Sorting**

* Add query params (page, limit, sortBy, order, filter).
* Implement in repository layer (Prisma/TypeORM queries).

**8. File Upload (Cloudinary or S3)**

* Start with local uploads using @nestjs/platform-express (Multer).
* Later integrate Cloudinary / AWS S3.

**9. Logging & Monitoring (⚡ this is where Winston/Morgan come in)**

👉 Only after your **basic APIs are working**:

* Install **Morgan** (request logging):

npm install morgan

import \* as morgan from 'morgan';

app.use(morgan('dev'));

* Install **Winston** (structured logs):

npm install winston nest-winston

const logger = WinstonModule.createLogger({...});

app.useLogger(logger);

* (Optional) Install Prometheus:

npm install prom-client @willsoto/nestjs-prometheus

**10. Security Enhancements**

* Install:

npm install helmet express-rate-limit xss-clean

* Apply:

app.use(helmet());

app.use(rateLimit({ windowMs: 15 \* 60 \* 1000, max: 100 }));

app.enableCors();

**11. Docker & CI/CD**

* Add Dockerfile & docker-compose.yml (app + PostgreSQL + Redis).
* Later → GitHub Actions / GitLab CI for auto-build & deploy.

**🔑 Key Takeaway**

* **Do NOT install Winston/Morgan at the very beginning**.
  + First → setup NestJS, DB, modules, CRUD, auth.
  + Then → once APIs are working, add logging/monitoring.
* Think in **phases**:
  + Project skeleton + configs
  + Database + CRUD
  + Auth + validation
  + Pagination/filtering/sorting
  + File uploads
  + Logging (Winston, Morgan)
  + Security (Helmet, rate limiting, sanitization)
  + Deployment (Docker, CI/CD)

👉 Would you like me to create a **phase-wise roadmap just for your LMS in NestJS** (Phase 1 = setup & CRUD, Phase 2 = auth + validation, Phase 3 = file storage, Phase 4 = logging/security, Phase 5 = Docker/CI/CD)? This way you’ll know exactly *when* to add Winston, Morgan, and other tools.