ĐỀ THI THỬ - XÂY DỰNG WEB API

Thời gian làm bài: 120 phút
Tổng điểm: 100 điểm
Được phép: Sử dụng ChatGPT, tài liệu, Internet
Công nghệ: Tùy chọn (Laravel, Node.js/Express, Django, ASP.NET Core, Spring Boot, v.v.)

MÔ TẢ DỰ ÁN

Bạn cần xây dựng một RESTful API cho hệ thống quản lý thư viện sách. API cho phép người dùng đăng ký, đăng nhập, quản lý sách, mượn/trả sách, và xem lịch sử mượn sách.

YÊU CẦU CƠ SỞ DỮ LIỆU

Database Schema:

Bảng 1: `users`

- `id` (Primary Key, Auto Increment): ID duy nhất
- `email` (String, Unique, Not Null): Email đăng nhập
- `password` (String, Not Null): Mật khẩu đã hash
- `full_name` (String, Not Null): Họ và tên đầy đủ
- `phone_number` (String, Nullable): Số điện thoại
- `address` (String, Nullable): Địa chỉ
- `role` (Enum/String, Default: "student"): Vai trò ("student", "teacher", "admin")
- `created_at` (Timestamp): Thời gian tạo
- `updated_at` (Timestamp): Thời gian cập nhật

Bảng 2: `books`

- `id` (Primary Key, Auto Increment): ID duy nhất
- `title` (String, Not Null): Tiêu đề sách
- `author` (String, Not Null): Tác giả
- `isbn` (String, Unique, Not Null): Mã ISBN
- `category` (Enum/String, Not Null): Thể loại ("Novel", "Science", "History", "Technology")
- `description` (Text, Nullable): Mô tả sách
- `total_copies` (Integer, Default: 1): Tổng số bản copy
- `available_copies` (Integer, Default: 1): Số bản copy còn lại
- `published_year` (Integer, Nullable): Năm xuất bản
- `publisher` (String, Nullable): Nhà xuất bản
- `created_at` (Timestamp): Thời gian tạo
- `updated_at` (Timestamp): Thời gian cập nhật

Bảng 3: `borrows`

- `id` (Primary Key, Auto Increment): ID duy nhất

- `user_id` (Foreign Key → users.id, Not Null): ID người dùng
- `book_id` (Foreign Key → books.id, Not Null): ID sách
- `borrow_date` (Date, Not Null): Ngày mượn
- `due_date` (Date, Not Null): Ngày hẹn trả
- `return_date` (Date, Nullable): Ngày thực tế trả sách
- `status` (Enum/String, Default: "borrowed"): Trạng thái ("pending", "borrowed", "returned", "overdue")
- `notes` (Text, Nullable): Ghi chú
- `created_at` (Timestamp): Thời gian tạo
- `updated_at` (Timestamp): Thời gian cập nhật

Ràng buộc và Quan hệ:

- Foreign Key: `borrows.user_id` → `users.id` (ON DELETE RESTRICT)
- Foreign Key: `borrows.book_id` → `books.id` (ON DELETE RESTRICT)
- Unique constraint: Một user không thể mượn cùng một book khi `status != "returned"` (có thể dùng composite unique hoặc check trong logic)
- Khi mượn sách: `books.available_copies` giảm đi 1
- Khi trả sách: `books.available_copies` tăng lên 1
- `books.available_copies` không được < 0

☑ YÊU CẦU CHỨC NĂNG

Phần 1: Thiết lập Project và Database (15 điểm)

1. **Khởi tạo Project** (5 điểm)
    - Tạo project với framework tùy chọn
    - Cấu hình database connection (MySQL, PostgreSQL, hoặc SQLite)
    - Thiết lập environment variables (.env file)
    - Cấu hình CORS (cho phép Flutter app gọi API)
2. **Tạo Database Migration** (7 điểm)
    - Tạo migration cho bảng `users`
    - Tạo migration cho bảng `books`
    - Tạo migration cho bảng `borrows`
    - Tạo foreign key constraints
    - Tạo indexes cho các trường thường query (email, isbn, user_id, book_id)
3. **Seeder Data** (3 điểm)
    - Tạo seeder với ít nhất:
        - 5 users mẫu (1 admin, 2 teachers, 2 students)
        - 10 books mẫu (phân bổ các category)
        - 5 borrows mẫu (cả borrowed và returned)

Phần 2: Authentication & Authorization (20 điểm)

2.1. Đăng ký User (5 điểm)

Endpoint: `POST /api/auth/register`

Request Body:

```
{
  "email": "user@example.com",
  "password": "password123",
  "full_name": "Nguyễn Văn A",
  "phone_number": "0123456789",
  "address": "123 Đường ABC",
  "role": "student"
}
```

Yêu cầu:

- Validate input (email format, password min 6 ký tự, role hợp lệ)
- Hash password trước khi lưu (bcrypt, argon2, hoặc tương đương)
- Kiểm tra email đã tồn tại chưa
- Trả về status code 201 và user data (không có password)
- Trả về status code 400 nếu validation fail
- Trả về status code 409 nếu email đã tồn tại

Response (Success):

```
{
  "success": true,
  "message": "User registered successfully",
  "data": {
    "id": 1,
    "email": "user@example.com",
    "full_name": "Nguyễn Văn A",
    "phone_number": "0123456789",
    "address": "123 Đường ABC",
    "role": "student",
    "created_at": "2024-01-01T00:00:00Z"
  }
}
```

2.2. Đăng nhập (5 điểm)

Endpoint: `POST /api/auth/login`

Request Body:

```
{
  "email": "user@example.com",
  "password": "password123"
}
```

Yêu cầu:

- Validate email và password
- Kiểm tra email tồn tại
- Verify password
- Tạo JWT token (hoặc session token)
- Trả về token và user info

Response (Success):

```
{
  "success": true,
  "message": "Login successful",
  "data": {
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "user": {
      "id": 1,
      "email": "user@example.com",
      "full_name": "Nguyễn Văn A",
      "role": "student"
    }
  }
}
```

2.3. Middleware Authentication (5 điểm)

- Tạo middleware để verify JWT token
- Áp dụng middleware cho các protected routes
- Trả về 401 nếu token không hợp lệ hoặc hết hạn
- Lưu thông tin user vào request (req.user hoặc tương đương)

2.4. Authorization (5 điểm)

- Tạo middleware kiểm tra role
- Endpoint quản lý users/books chỉ dành cho `admin`
- Endpoint mượn/trả sách dành cho tất cả authenticated users
- Trả về 403 nếu không có quyền

Phần 3: User Management API (15 điểm)

3.1. Lấy danh sách Users (3 điểm)

Endpoint: `GET /api/users`

Yêu cầu:

- Chỉ admin mới được truy cập
- Pagination: `?page=1&limit=10`
- Sorting: `?sort=created_at&order=desc`
- Search: `?search=keyword` (tìm trong email, full_name)
- Trả về danh sách users (không có password)

3.2. Lấy User theo ID (2 điểm)

Endpoint: `GET /api/users/:id`

Yêu cầu:

- Admin hoặc chính user đó mới được xem
- Trả về 404 nếu không tìm thấy

3.3. Cập nhật User (4 điểm)

Endpoint: `PUT /api/users/:id`

Request Body:

```
{
  "full_name": "Nguyễn Văn B",
  "phone_number": "0987654321",
  "address": "456 Đường XYZ"
}
```

Yêu cầu:

- Admin hoặc chính user đó mới được cập nhật
- Validate input
- Không cho phép cập nhật email và role (trừ admin)
- Trả về user đã cập nhật

3.4. Xóa User (3 điểm)

Endpoint: `DELETE /api/users/:id`

Yêu cầu:

- Chỉ admin mới được xóa
- Kiểm tra user có đang mượn sách không (có borrows với status != "returned")
- Nếu đang mượn sách, trả về 400 với message "Cannot delete user with active borrows"
- Nếu không, xóa user và trả về 200

3.5. Lấy thông tin User hiện tại (3 điểm)

Endpoint: `GET /api/auth/me`

Yêu cầu:

- Lấy thông tin user từ token
- Trả về user info (không có password)

Phần 4: Book Management API (20 điểm)

4.1. Lấy danh sách Books (4 điểm)

Endpoint: `GET /api/books`

Yêu cầu:

- Public endpoint (không cần authentication)
- Pagination: `?page=1&limit=10`
- Sorting: `?sort=title&order=asc`
- Search: `?search=keyword` (tìm trong title, author, isbn)
- Filter by category: `?category=Novel`
- Trả về danh sách books

4.2. Lấy Book theo ID (2 điểm)

Endpoint: `GET /api/books/:id`

Yêu cầu:

- Public endpoint
- Trả về 404 nếu không tìm thấy

4.3. Thêm Book (4 điểm)

Endpoint: `POST /api/books`

Request Body:

```
{
  "title": "Flutter Development Guide",
  "author": "John Doe",
  "isbn": "978-1234567890",
  "category": "Technology",
  "description": "A comprehensive guide to Flutter development",
  "total_copies": 5,
  "published_year": 2024,
  "publisher": "Tech Publisher"
}
```

Yêu cầu:

- Chỉ admin mới được thêm
- Validate input (required fields, ISBN unique, category hợp lệ)
- Set `available_copies = total_copies` khi tạo
- Trả về book đã tạo

4.4. Cập nhật Book (4 điểm)

Endpoint: `PUT /api/books/:id`

Request Body:

```
{
  "title": "Updated Title",
  "total_copies": 10
}
```

Yêu cầu:

- Chỉ admin mới được cập nhật
- Validate input
- Khi cập nhật `total_copies`:
  - Tính `available_copies = available_copies + (new_total - old_total)`
  - Đảm bảo `available_copies >= 0`
- Trả về book đã cập nhật

4.5. Xóa Book (3 điểm)

Endpoint: `DELETE /api/books/:id`

Yêu cầu:

- Chỉ admin mới được xóa
- Kiểm tra book có đang được mượn không (có borrows với status != "returned")
- Nếu có, trả về 400 với message "Cannot delete book with active borrows"
- Nếu không, xóa book và trả về 200

4.6. Tìm kiếm Books nâng cao (3 điểm)

Endpoint: `GET /api/books/search`

Query Parameters:

- `q`: Keyword tìm kiếm
- `category`: Lọc theo category
- `author`: Lọc theo author
- `available_only`: true/false - chỉ lấy sách còn available

Yêu cầu:

- Kết hợp nhiều điều kiện tìm kiếm
- Trả về kết quả đã filter

Phần 5: Borrow Management API (20 điểm)

5.1. Mượn Sách (6 điểm)

Endpoint: `POST /api/borrows`

Request Body:

```
{
  "book_id": 1,
  "days_to_borrow": 7
}
```

Yêu cầu:

- Cần authentication
- Validate input (book_id tồn tại, days_to_borrow từ 1-30)
- Kiểm tra `available_copies > 0`
- Kiểm tra user chưa mượn quá 5 sách cùng lúc (status != "returned")
- Kiểm tra user chưa mượn cùng book này (status != "returned")
- Tạo record trong `borrows`:
  - `borrow_date` = hiện tại
  - `due_date` = `borrow_date` + `days_to_borrow` ngày
  - `status` = "borrowed"
- Giảm `available_copies` của book đi 1
- Trả về borrow record đã tạo

Response (Success):

```
{
  "success": true,
  "message": "Book borrowed successfully",
  "data": {
    "id": 1,
    "user_id": 1,
    "book_id": 1,
    "borrow_date": "2024-01-01",
    "due_date": "2024-01-08",
    "status": "borrowed",
    "book": {
      "id": 1,
      "title": "Flutter Guide",
      "author": "John Doe"
```

```
    }
  }
}
```

5.2. Trả Sách (5 điểm)

Endpoint: `PUT /api/borrows/:id/return`

Yêu cầu:

- Cần authentication
- Chỉ user mượn sách đó hoặc admin mới được trả
- Kiểm tra borrow tồn tại và `status != "returned"`
- Cập nhật:
  - `return_date` = hiện tại
  - `status` = "returned"
- Tăng `available_copies` của book lên 1
- Trả về borrow record đã cập nhật

5.3. Lấy lịch sử mượn của User (3 điểm)

Endpoint: `GET /api/users/:id/borrows`

Yêu cầu:

- Cần authentication
- Admin hoặc chính user đó mới được xem
- Pagination
- Filter by status: `?status=borrowed`
- Sắp xếp theo `borrow_date` giảm dần
- Include thông tin book trong response

5.4. Lấy lịch sử mượn của Book (3 điểm)

Endpoint: `GET /api/books/:id/borrows`

Yêu cầu:

- Public endpoint
- Pagination
- Filter by status
- Sắp xếp theo `borrow_date` giảm dần
- Include thông tin user trong response

5.5. Lấy danh sách sách đang mượn (3 điểm)

Endpoint: `GET /api/borrows/active`

Yêu cầu:

- Cần authentication
- Chỉ admin mới được xem tất cả
- User thường chỉ xem của mình
- Filter: `?status=borrowed,overdue`
- Include thông tin user và book
- Highlight sách quá hạn (due_date < hiện tại)

Phần 6: Error Handling & Validation (10 điểm)

1. **Global Error Handler** (4 điểm)
    - Tạo middleware xử lý lỗi toàn cục
    - Xử lý các loại lỗi:
        - Validation errors (400)
        - Authentication errors (401)
        - Authorization errors (403)
        - Not found (404)
        - Database errors (500)
        - Duplicate entry (409)
    - Trả về response format nhất quán:

```
{
  "success": false,
  "message": "Error message",
  "errors": {} // Chi tiết lỗi validation (nếu có)
}
```

2. **Input Validation** (3 điểm)
    - Validate tất cả input từ request
    - Sử dụng validation library (Joi, Validator, Laravel Validation, v.v.)
    - Trả về lỗi chi tiết cho từng field

3. **Database Transaction** (3 điểm)
    - Sử dụng transaction cho các operations phức tạp:
        - Mượn sách (tạo borrow + cập nhật available_copies)
        - Trả sách (cập nhật borrow + cập nhật available_copies)
    - Rollback nếu có lỗi

📝 YÊU CẦU KỸ THUẬT

1. **RESTful API Design** (Bắt buộc)
    - Tuân thủ REST conventions
    - Sử dụng HTTP methods đúng (GET, POST, PUT, DELETE)
    - Status codes phù hợp (200, 201, 400, 401, 403, 404, 500)

- URL naming convention nhất quán

2. **Response Format** (Bắt buộc)
   - Tất cả response phải có format nhất quán:

```
{
  "success": true/false,
  "message": "Message",
  "data": {} // hoặc []
}
```

3. **Security** (Bắt buộc)
   - Hash password (bcrypt, argon2, hoặc tương đương)
   - JWT token với expiration time
   - Validate và sanitize input
   - SQL injection prevention (dùng prepared statements/ORM)
4. **Code Organization** (Khuyến khích)
   - Tách biệt Controller, Service/Repository, Model
   - Sử dụng middleware cho authentication/authorization
   - Tái sử dụng code, DRY principle

API DOCUMENTATION

Tạo file `API_DOCUMENTATION.md` hoặc sử dụng Swagger/Postman Collection với:

1. **Danh sách tất cả endpoints**
2. **Request/Response examples** cho mỗi endpoint
3. **Authentication method** (JWT Bearer token)
4. **Error codes và messages**
5. **Sample requests** (có thể dùng Postman Collection)

CHECKLIST NỘP BÀI

- [ ] Project hoàn chỉnh, có thể chạy được
- [ ] Database đã migrate và có seeder data
- [ ] Tất cả endpoints hoạt động đúng
- [ ] Authentication và Authorization hoạt động
- [ ] Validation đầy đủ cho tất cả input
- [ ] Error handling toàn cục
- [ ] Response format nhất quán
- [ ] API Documentation đầy đủ
- [ ] CORS đã cấu hình
- [ ] Code được tổ chức rõ ràng, có comment

- [ ] File README.md hướng dẫn setup và chạy project
- [ ] File .env.example (không commit .env thật)

CÁCH NỘP BÀI

1. Nén toàn bộ project thành file `.zip`
2. Export database schema (SQL dump hoặc migration files)
3. Gửi kèm file `README.md` hướng dẫn:
    - Cách cài đặt dependencies
    - Cách setup database
    - Cách chạy project
    - Cách test API (Postman collection hoặc curl commands)
4. Gửi link GitHub (nếu có) để code review
5. Gửi kèm Postman Collection hoặc API Documentation

TESTING (Khuyến khích, không bắt buộc)

Nếu có thời gian, viết unit tests hoặc integration tests cho:

- Authentication endpoints
- CRUD operations
- Business logic (mượn sách, trả sách)

LƯU Ý

- **Chọn công nghệ phù hợp** - Laravel, Express.js, Django, ASP.NET Core, Spring Boot đều được
- **Chú ý Security** - Không hardcode credentials, hash password, validate input
- **Database Design** - Đảm bảo foreign keys, constraints, indexes hợp lý
- **Error Messages** - Thông báo lỗi rõ ràng, user-friendly
- **Code Quality** - Code sạch, có comment, dễ đọc, dễ maintain
- **Sử dụng ChatGPT hợp lý** - Hiểu code bạn viết, không chỉ copy-paste
- **Test kỹ các trường hợp edge cases** - Hết sách, user mượn quá nhiều, invalid input, v.v.

THANG ĐIỂM CHI TIẾT

Phần	Điểm	Ghi chú
Thiết lập Project và Database	15	Migration, Seeder
Authentication & Authorization	20	Register, Login, JWT, Middleware
User Management API	15	CRUD Users
Book Management API	20	CRUD Books, Search
Borrow Management API	20	Mượn, Trả, Lịch sử
Error Handling & Validation	10	Global handler, Validation, Transaction

Phần	Điểm	Ghi chú
Tổng	**100**	

Bonus điểm (tối đa +10):

- Unit/Integration Tests: +5
- API Documentation chi tiết (Swagger): +3
- Code quality cao (clean code, design patterns): +2

Chúc các bạn làm bài tốt! 🚀