

**ON-JOB TRAINING PROJECT**

**Sinh viên ngành Kỹ thuật phần mềm**

**Report 2 – Tuần 3 và 4**

Sinh viên thực hiện: Trần Tấn Phát

Mã số sinh viên: SE183281

Giảng viên hướng dẫn: Ngô Đăng Hà An

**– Hồ Chí Minh, Tháng 10 2025 –**

**1.Giới thiệu tổng quan**

**1.1 Mục tiêu MOOC2**

MOOC2 nhằm thực hành kỹ năng phát triển ứng dụng full-stack với 3 thành phần chính:

Frontend (ReactJS): State management, API integration, UI design

Backend (.NET MVC/API): REST API, Entity Framework, CRUD operations

Database (PostgreSQL): Database design, SQL queries

**1.2 Hệ thống đã xây dựng**

Library Management System - Hệ thống quản lý thư viện với các chức năng:

Quản lý người dùng (Admin/User roles)

Quản lý sách và danh mục

Hệ thống mượn/trả sách

Authentication với JWT

**1.3 Công nghệ sử dụng**

Frontend: React 19 + Axios + CSS3

Backend: ASP.NET Core 8 + Entity Framework Core + JWT

Database: PostgreSQL

Tools: VS Code + Git + GitHub

**1.4 Kiến trúc hệ thống**

**React App (3000) ←→ ASP.NET Core API (5053) ←→ PostgreSQL Database**

**1.5 Kết quả đạt được**

✅ Hoàn thành 100% yêu cầu MOOC2

✅ Ứng dụng full-stack hoạt động ổn định

✅ 4 tables, 20+ API endpoints, 6 UI pages

**2.Database**

**2.1 Thiết kế và tối ưu cơ sở dữ liệu**

**Database Schema Design:**

**Users ──(1:N)──→ Borrows ──(N:1)──→ Books ──(N:1)──→ Categories4 Tables chính:**

+Users Table: Quản lý thông tin người dùng hệ thống

+Categories Table: Phân loại sách theo thể loại

+Books Table: Quản lý kho sách trong thư viện

+Borrows Table: Theo dõi việc mượn/trả sách

|  |  |  |
| --- | --- | --- |
| Tên bảng | Thuộc tính | Mục đích của thuộc tính |
| users | id | ID duy nhất của người dùng |
| username | Tên đăng nhập hệ thống |
| email | Email đăng nhập và liên lạc |
| password | Mật khẩu đã mã hóa BCrypt |
| fullname | Họ tên đầy đủ của người dùng |
| role | Phân quyền: admin hoặc user |
| isactive | Trạng thái kích hoạt tài khoản |
| categories | id | ID duy nhất của danh mục |
| name | Tên danh mục sách |
| description | Mô tả chi tiết về danh mục |
| books | id | ID duy nhất của sách |
| title | Tên sách |
| author | Tác giả của sách |
| isbn | Mã số sách quốc tế |
| publisher | Nhà xuất bản |
| totalcopies | Tổng số bản sách có trong thư viện |
| availablecopies | Số bản sách còn có thể mượn |
| categoryid | Liên kết đến danh mục sách |
| borrows | id | ID duy nhất của phiếu mượn |
|  | userid | Liên kết đến người mượn sách |
| bookid | Liên kết đến sách được mượn |
| borrowdate | Ngày mượn sách |
| duedate | Hạn trả sách |
| returndate | Ngày trả sách thực tế |
| status | Trạng thái mượn: borrowed/returned |
| notes | Ghi chú về việc mượn sách |

**2.2 Thực hành các truy vấn SQL**

**Truy vấn cơ bản (CRUD):**

|  |
| --- |
| -- Thêm người dùng mới  INSERT INTO users (username, email, password, fullname, role)  VALUES ('newuser', 'user@example.com', 'hashedpassword', 'Nguyen Van A', 'user');  -- Lấy danh sách sách còn có thể mượn  SELECT title, author, availablecopies  FROM books WHERE availablecopies > 0;  -- Cập nhật số sách khi mượn  UPDATE books SET availablecopies = availablecopies - 1 WHERE id = 1; |

**Truy vấn kết hợp (JOIN):**

|  |
| --- |
| -- Lấy sách kèm tên danh mục  SELECT b.title, b.author, c.name as category\_name  FROM books b  JOIN categories c ON b.categoryid = c.id;  -- Lịch sử mượn sách của người dùng  SELECT u.fullname, b.title, br.borrowdate, br.status  FROM borrows br  JOIN users u ON br.userid = u.id  JOIN books b ON br.bookid = b.id; |

**Truy vấn thống kê:**

|  |
| --- |
| -- Số sách theo danh mục  SELECT c.name, COUNT(b.id) as so\_luong  FROM categories c  LEFT JOIN books b ON c.id = b.categoryid  GROUP BY c.name;  -- Tìm sách quá hạn  SELECT u.fullname, b.title, br.duedate  FROM borrows br  JOIN users u ON br.userid = u.id  JOIN books b ON br.bookid = b.id  WHERE br.status = 'borrowed' AND br.duedate < CURRENT\_TIMESTAMP; |

**Back-end**

**3.1 Xây dựng API với ASP.NET Core**

Cấu hình hệ thống: Backend được xây dựng với ASP.NET Core 8.0, sử dụng Entity Framework Core kết nối PostgreSQL. CORS được cấu hình cho phép frontend localhost:3000 truy cập. Swagger được tích hợp để test API.

5 Controllers đã triển khai:

AuthController - Xử lý đăng nhập

UsersController - Quản lý người dùng

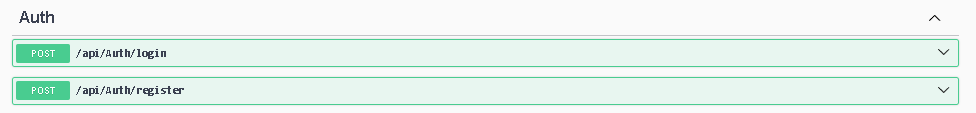
CategoryController - Quản lý danh mục sách

BookController - Quản lý thông tin sách

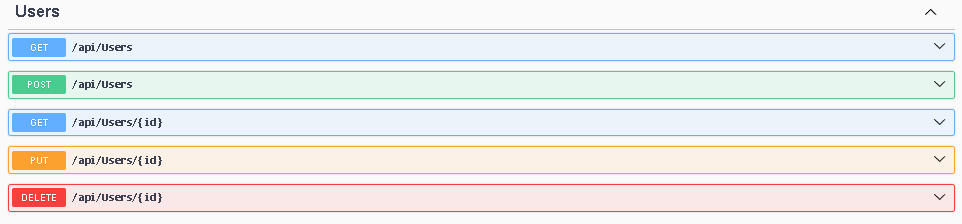
BorrowController - Quản lý mượn trả

API Endpoints đã hoàn thành:

Auth: POST /api/Auth/login



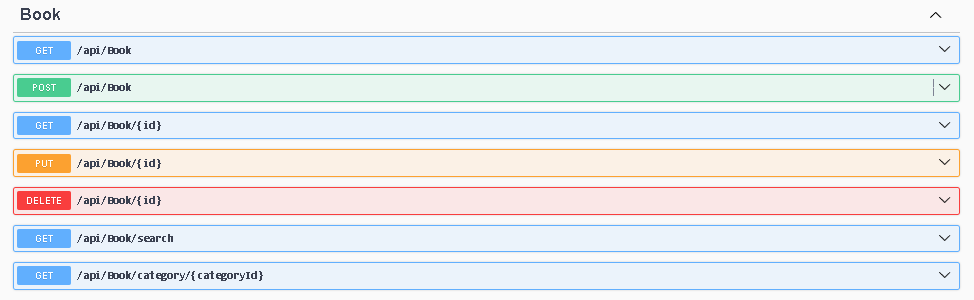
Users: GET, GET/{id}, POST, PUT/{id}, DELETE/{id}



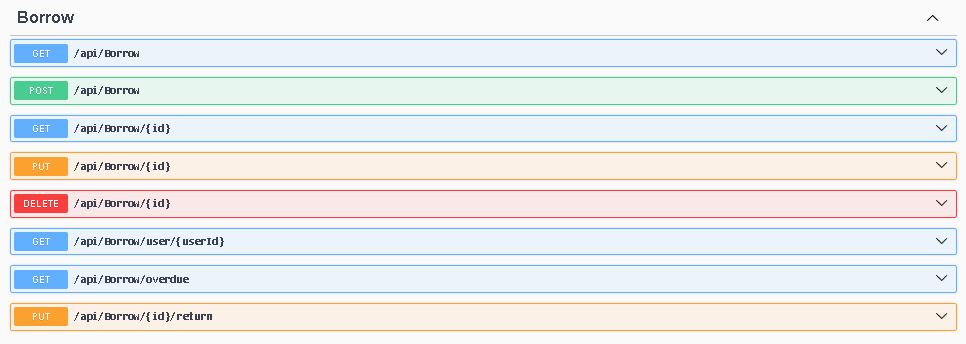
Category: GET, GET/{id}, POST, PUT/{id}, DELETE/{id}



Book: GET, GET/{id}, POST, PUT/{id}, DELETE/{id}



Borrow: GET, GET/{id}, POST, PUT/{id}, DELETE/{id}, PUT/{id}/return



**3.2 Entity Framework Core**

Database Context: OjtDbContext quản lý 4 DbSet: Users, Categories, Books, Borrows với các navigation properties được cấu hình trong OnModelCreating.

LINQ Queries thực tế:

|  |
| --- |
| // Lấy categories kèm books  var categories = await \_context.Categories  .Include(c => c.Books)  .OrderBy(c => c.Name)  .ToListAsync();  // Lấy books kèm category  var books = await \_context.Books  .Include(b => b.Category)  .Include(b => b.Borrows)  .ToListAsync(); |

Migrations: Database được tạo từ migration AddLibraryTables với proper foreign keys và constraints.

**3.3 Authentication & Security**

BCrypt Password Hashing:

|  |
| --- |
| // Hash password khi tạo user  string hashedPassword = BCrypt.Net.BCrypt.HashPassword(user.Password);  // Verify password khi login  bool isValid = BCrypt.Net.BCrypt.Verify(request.Password, user.Password); |

Input Validation: Mỗi controller có validation cho null checks, required fields và business logic validation.

**3.4 CRUD Operations thực tế**

Users CRUD:

GET /api/Users - Lấy tất cả users

GET /api/Users/{id} - Lấy user theo ID

POST /api/Users - Tạo user mới với BCrypt hashing

PUT /api/Users/{id} - Update user information

DELETE /api/Users/{id} - Xóa user

Error Handling:

400 cho bad request và validation errors

401 cho unauthorized login

404 cho resource not found

200/201 cho successful operations

**Front-end**

Quản lý trạng thái ứng dụng (State Management với React Hooks)

**React Hooks được sử dụng:**

useState: Quản lý dữ liệu form đăng nhập, trạng thái loading, và thông báo lỗi trong từng component

useEffect: Tự động khôi phục phiên đăng nhập từ localStorage khi mở lại trang và tải dữ liệu dashboard

useContext: Chia sẻ thông tin người dùng đã đăng nhập cho toàn bộ ứng dụng

useNavigate: Chuyển hướng trang sau khi đăng nhập thành công hoặc khi chưa có quyền truy cập

**Context API:** Lưu trữ thông tin đăng nhập toàn cục và cung cấp các hàm login/logout cho tất cả components.

Giao tiếp với API bằng Axios

**Services Layer:** Tách riêng logic gọi API thành các file service riêng biệt để dễ quản lý và tái sử dụng.

**Các service đã tạo:**

authService: Xử lý đăng nhập và đăng ký người dùng

userService: Quản lý thông tin người dùng (thêm, sửa, xóa, xem)

categoryService: Quản lý danh mục sách

bookService: Quản lý thông tin sách và liên kết với danh mục

borrowService: Xử lý mượn sách và trả sách

**Xử lý lỗi:** Hiển thị thông báo lỗi bằng tiếng Việt khi API trả về lỗi hoặc mất kết nối.

Thiết kế giao diện người dùng với CSS3

**CSS tùy chỉnh:** Sử dụng CSS3 thay vì framework để có thiết kế độc đáo và tải trang nhanh hơn.

**Tính năng giao diện:**

Form validation: Kiểm tra và hiển thị lỗi khi người dùng nhập sai thông tin

Professional styling: Thiết kế giao diện chuyên nghiệp với màu sắc và bố cục nhất quán

Interactive elements: Các nút và form có hiệu ứng khi hover và focus

