ĐẠI HỌC PHENIKAA

**A logo for a university

Description automatically generatedTRƯỜNG CÔNG NGHỆ THÔNG TIN**

**BÁO CÁO BÀI TẬP KẾT THÚC HỌC PHẦN**

**MÔN HỌC**: **Thiết kế ứng dụng web**   
**TÊN ĐỀ TÀI: Website Quản lý sinh viên**

**Lớp: Thiết kế web nâng cao (COUR01.TH1)**

**Nhóm**: **09**

**Giảng viên hướng dẫn: TS. Nguyễn Thị Thùy Liên**

| Họ và tên | Mã sinh viên | Gmail |
| --- | --- | --- |
| Nguyễn Minh Dương | 23010441 | 23010441@st.phenikaa-uni.edu.vn |
| Nguyễn Văn Tú | 23010312 | 23010909@st.phenikaa-uni.edu.vn |

**Hà Nội, 06-2025**

**BẢNG PHÂN CÔNG NHIỆM VỤ**

|  |  |  |  |
| --- | --- | --- | --- |
| **Họ và tên** | **Mã**  **sinh viên** | **Công việc được giao** | **Khối lượng công việc** |
| Nguyễn Minh Dương | 23010441 | - Xây dựng dashboard - Phát triển module quản lý sinh viên (Student) - Phát triển module quản lý giáo viên (Teacher) - Phát triển module quản lý khóa học (Course) - Xây dựng chức năng phân công giáo viên cho khóa học - Viết báo cáo tổng kết | 1/2 |
| Nguyễn Văn Tú | 23010312 | - Thiết kế cơ sở dữ liệu và mối quan hệ giữa các bảng - Triển khai hệ thống xác thực và phân quyền người dùng - Kiểm thử và sửa lỗi - Tối ưu hóa hiệu suất hệ thống  - Xây dựng module quản lý lớp học, quản lý tham gia, thanh toán của hệ thống website | 1/2 |

[**CHƯƠNG 1: GIỚI THIỆU CHUNG** 5](#_Toc201418890)

[**1.1. Tổng quan về Laravel Student Management Web** 5](#_Toc201418891)

[**1.2 Mục tiêu của ứng dụng** 5](#_Toc201418892)

[**1.3 Các tính năng chính** 5](#_Toc201418893)

[**1.4 Công nghệ và kiến trúc** 6](#_Toc201418894)

[**1.5 Kết quả đạt được** 7](#_Toc201418895)

[**1.6 Thách thức và bài học kinh nghiệm** 7](#_Toc201418896)

[**Chương 2: Kiến trúc, công nghệ và phương pháp phát triển** 7](#_Toc201418897)

[**2.1. Kiến trúc MVC** 8](#_Toc201418898)

[**2.1.1. Model** 9](#_Toc201418899)

[**2.1.2 View** 13](#_Toc201418900)

[**2.1.3 Controller** 16](#_Toc201418901)

[**2.1.4 Đăng ký, đăng nhập bằng package Fortify** 23](#_Toc201418902)

[**2.2. Thiết kế cơ sở dữ liệu** 25](#_Toc201418903)

[**2.2.1 Sơ đồ kết nối các bảng** 25](#_Toc201418904)

[**2.2.2 Cấu trúc bảng** 26](#_Toc201418905)

[**2.3. Công nghệ sử dụng** 30](#_Toc201418906)

[**2.3.1 Framework: Laravel (PHP)** 30](#_Toc201418907)

[**2.3.2 Cơ sở dữ liệu: MySQL** 30](#_Toc201418908)

[**2.3.3 Giao diện người dùng: Blade Template, Bootstrap, CSS, HTML, JavaScript** 30](#_Toc201418909)

[**2.4. Phương pháp phát triển** 31](#_Toc201418910)

[**2.4.1 Phát triển theo hướng mô-đun (Modular Development)** 31](#_Toc201418911)

[**2.4.2 Áp dụng Best Practices của Laravel** 31](#_Toc201418912)

[**2.4.3 Xây dựng giao diện thân thiện và dễ sử dụng** 32](#_Toc201418913)

[**Chương 3: Chức năng và Kết quả đạt được** 32](#_Toc201418914)

[**3.1 Chức năng chính** 32](#_Toc201418915)

[**3.2 Kết quả đạt được** 34](#_Toc201418916)

[**Chương 4: Giao diện** 35](#_Toc201418917)

[**4.1. Giao diện Đăng nhập/Đăng ký** 35](#_Toc201418918)

[**4.2 Giao diện Dashboard** 36](#_Toc201418919)

[**4.3 Giao diện StudentManagement** 37](#_Toc201418920)

[**4.4 Giao diện Teacher Management** 38](#_Toc201418921)

[**4.5 Giao diện Class** 40](#_Toc201418922)

[**4.6 Giao diện Enrollment** 41](#_Toc201418923)

[**4.7 Giao diện Payment** 42](#_Toc201418924)

[**4.8 Một vài hình ảnh giao diện DarkMode** 43](#_Toc201418925)

[**Chương 5: Kết luận** 44](#_Toc201418926)

# **CHƯƠNG 1: GIỚI THIỆU CHUNG**

## **1.1. Tổng quan về Laravel Student Management Web**

**Laravel Student Management Web** là một ứng dụng web được phát triển dựa trên framework Laravel, một trong những framework PHP mạnh mẽ và phổ biến nhất hiện nay. Ứng dụng này được thiết kế nhằm hỗ trợ các cơ sở giáo dục, bao gồm trường học, giáo viên và quản trị viên, trong việc quản lý thông tin sinh viên một cách hiệu quả, khoa học và tự động hóa. Với các tính năng đa dạng và giao diện thân thiện, ứng dụng cung cấp một giải pháp toàn diện để quản lý các khía cạnh quan trọng của hệ thống giáo dục như thông tin sinh viên, lớp học, điểm số, lịch học, và các hoạt động liên quan khác.

Ứng dụng được xây dựng dựa trên các nguyên tắc phát triển phần mềm hiện đại, tận dụng các tính năng vượt trội của Laravel như Eloquent ORM, Blade templating, middleware, và hệ thống bảo mật tích hợp. Điều này không chỉ đảm bảo hiệu suất cao mà còn mang lại sự linh hoạt, dễ dàng mở rộng và bảo trì trong tương lai. Trong bối cảnh giáo dục ngày càng hiện đại hóa, **Laravel Student Management Web** đáp ứng nhu cầu số hóa quản lý thông tin, giúp giảm thiểu công việc thủ công, tăng cường độ chính xác và tiết kiệm thời gian cho người dùng.

## **1.2 Mục tiêu của ứng dụng**

Mục tiêu chính của **Laravel Student Management App** là cung cấp một nền tảng trực quan, dễ sử dụng để hỗ trợ các tổ chức giáo dục trong việc quản lý thông tin sinh viên một cách hiệu quả. Cụ thể, ứng dụng hướng đến:

* **Tự động hóa quy trình quản lý**: Giảm thiểu các công việc thủ công như nhập liệu, kiểm tra điểm số, hoặc quản lý lịch học thông qua các chức năng tự động.
* **Tăng cường tính chính xác**: Đảm bảo dữ liệu sinh viên, điểm số và thông tin lớp học được lưu trữ và xử lý chính xác, tránh sai sót do con người.
* **Cải thiện khả năng truy cập**: Cho phép quản trị viên, giáo viên và thậm chí sinh viên (nếu được cấp quyền) truy cập thông tin mọi lúc, mọi nơi thông qua giao diện web.
* **Tùy chỉnh và mở rộng**: Cung cấp một nền tảng linh hoạt, cho phép tích hợp thêm các tính năng mới hoặc tùy chỉnh theo nhu cầu cụ thể của từng cơ sở giáo dục.

## **1.3 Các tính năng chính**

Ứng dụng **Laravel Student Management App** được tích hợp nhiều tính năng quan trọng nhằm đáp ứng nhu cầu quản lý trong môi trường giáo dục. Một số tính năng nổi bật bao gồm:

* **Quản lý thông tin sinh viên**: Lưu trữ và quản lý thông tin cá nhân của sinh viên như họ tên, ngày sinh, địa chỉ, email, số điện thoại, và các thông tin liên quan khác. Người dùng có thể thêm, sửa, xóa hoặc tìm kiếm thông tin sinh viên một cách dễ dàng.
* **Đăng ký, đăng nhập:** Cho phép đăng kí, đăng nhâp khi truy câp trang web và khi đăng ký/đăng nhâp thìhệ thống kiểm tra: email đã tồn tại chưa, mật khẩu có đủ độ dài và tính bảo mật không.
* **Quản lý lớp học**: Cho phép tạo, cập nhật và quản lý các lớp học, bao gồm thông tin về giáo viên phụ trách, danh sách sinh viên, và lịch học.
* **Quản lý điểm số**: Hỗ trợ nhập, chỉnh sửa và theo dõi điểm số của sinh viên theo từng môn học, kỳ học, hoặc năm học. Ứng dụng cũng cung cấp tính năng tổng hợp điểm trung bình và xếp loại học lực.
* **Báo cáo và thống kê**: Tạo các báo cáo chi tiết về tình hình học tập, tỷ lệ chuyên cần, hoặc các số liệu thống kê khác, hỗ trợ nhà trường trong việc đánh giá và ra quyết định.
* **Quản lý người dùng và phân quyền**: Cho phép phân quyền cho các vai trò khác nhau như quản trị viên, giáo viên, hoặc sinh viên, đảm bảo tính bảo mật và kiểm soát truy cập phù hợp.
* **Giao diện thân thiện**: Sử dụng Blade templating của Laravel để tạo giao diện người dùng trực quan, dễ sử dụng, và tương thích trên nhiều thiết bị.

## **1.4 Công nghệ và kiến trúc**

**Laravel Student Management App** được phát triển dựa trên framework Laravel, tận dụng các thành phần cốt lõi của framework để đảm bảo hiệu suất và tính bảo mật. Một số công nghệ và công cụ được sử dụng trong quá trình phát triển bao gồm:

* **Laravel Framework**: Sử dụng các tính năng như Eloquent ORM để quản lý cơ sở dữ liệu, Blade để xây dựng giao diện, và middleware để xử lý xác thực và phân quyền.
* **Cơ sở dữ liệu**: Sử dụng MySQL hoặc các hệ quản trị cơ sở dữ liệu tương thích với Laravel để lưu trữ thông tin sinh viên, lớp học, điểm số, và các dữ liệu liên quan.
* **HTML, CSS, JavaScript**: Tạo giao diện người dùng responsive, kết hợp với các thư viện như Bootstrap hoặc Tailwind CSS để tăng tính thẩm mỹ và trải nghiệm người dùng.
* **Hệ thống bảo mật**: Tích hợp các cơ chế xác thực (authentication) và phân quyền (authorization) của Laravel để bảo vệ dữ liệu người dùng và ngăn chặn truy cập trái phép.

Về mặt kiến trúc, ứng dụng tuân theo mô hình MVC (Model-View-Controller) của Laravel, trong đó:

* **Model**: Đại diện cho dữ liệu và logic nghiệp vụ, sử dụng Eloquent ORM để tương tác với cơ sở dữ liệu.
* **View**: Hiển thị giao diện người dùng thông qua các template Blade.
* **Controller**: Xử lý các yêu cầu từ người dùng, điều phối dữ liệu giữa Model và View.

## **1.5 Kết quả đạt được**

Trong quá trình phát triển **Laravel Student Management App** cho môn học, nhóm đã đạt được nhiều kết quả đáng chú ý:

1. **Hoàn thành các chức năng cốt lõi**: Ứng dụng đáp ứng đầy đủ các yêu cầu cơ bản như quản lý sinh viên, lớp học, và điểm số, đồng thời đảm bảo tính chính xác và hiệu quả.
2. **Ứng dụng thực tiễn**: Ứng dụng có thể được triển khai trong các cơ sở giáo dục nhỏ và vừa, giúp tối ưu hóa quy trình quản lý.
3. **Kỹ năng phát triển**: Nhóm đã nâng cao kỹ năng lập trình PHP, sử dụng framework Laravel, và làm việc với cơ sở dữ liệu quan hệ. Ngoài ra, việc áp dụng mô hình MVC và các công cụ hiện đại như Git, Composer đã giúp nhóm cải thiện khả năng làm việc nhóm và quản lý mã nguồn.
4. **Khả năng mở rộng**: Ứng dụng được thiết kế với cấu trúc mô-đun, cho phép dễ dàng tích hợp các tính năng mới như quản lý lịch học, thông báo, hoặc tích hợp API bên thứ ba trong tương lai.

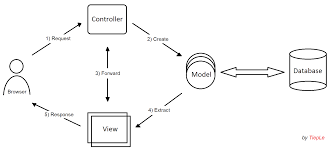
## **1.6 Thách thức và bài học kinh nghiệm**

Trong quá trình phát triển, nhóm đã gặp phải một số thách thức:

* **Thiết kế cơ sở dữ liệu**: Việc thiết kế một cơ sở dữ liệu hiệu quả, đảm bảo tính toàn vẹn dữ liệu và hỗ trợ các truy vấn phức tạp là một thách thức lớn, đặc biệt với các tính năng như quản lý điểm số và báo cáo.
* **Xử lý phân quyền**: Việc triển khai hệ thống phân quyền để đảm bảo chỉ người dùng được ủy quyền mới có thể truy cập vào các chức năng cụ thể đòi hỏi sự cẩn thận trong thiết kế và kiểm thử.
* **Tối ưu hóa hiệu suất**: Với khối lượng dữ liệu lớn, việc tối ưu hóa các truy vấn cơ sở dữ liệu và cải thiện tốc độ tải trang là một vấn đề cần được giải quyết.

# **Chương 2: Kiến trúc, công nghệ và phương pháp phát triển**

## **2.1. Kiến trúc MVC và package**



Hình 1: Kiến trúc MVC

* **Người Dùng (User) và Tương Tác qua Trình Duyệt**  
  Người dùng, chẳng hạn như quản trị viên, giáo viên hoặc sinh viên, tương tác với ứng dụng thông qua **trình duyệt** (Browser) trên giao diện web. Các thao tác bao gồm xem danh sách sinh viên, thêm sinh viên mới, chỉnh sửa thông tin, xóa bản ghi, hoặc thực hiện các chức năng quản lý khác. Những tương tác này được khởi tạo dưới dạng các **yêu cầu HTTP** (HTTP requests) như GET (để xem dữ liệu), POST (để gửi dữ liệu), PUT (để cập nhật), hoặc DELETE (để xóa), được gửi từ trình duyệt đến ứng dụng.
* **View: Giao Diện Người Dùng và Truyền Tương Tác**  
  View đóng vai trò là lớp giao diện, chịu trách nhiệm hiển thị dữ liệu cho người dùng dưới dạng các trang web, biểu mẫu (forms), hoặc bảng dữ liệu, được xây dựng bằng Blade templating của Laravel. View không chỉ trình bày thông tin một cách trực quan (ví dụ: danh sách sinh viên hoặc biểu mẫu nhập điểm), mà còn thu nhận các tương tác từ người dùng, như nhập dữ liệu vào biểu mẫu hoặc nhấp vào nút "Thêm sinh viên". Các tương tác này được View chuyển tiếp đến **Controller** dưới dạng yêu cầu HTTP, kèm theo dữ liệu người dùng nhập (nếu có).
* **Controller: Điều Phối Logic Nghiệp Vụ**  
  Controller hoạt động như trung tâm điều phối, tiếp nhận các yêu cầu từ View thông qua các route được định nghĩa trong tệp web.php. Controller xử lý logic nghiệp vụ, chẳng hạn như xác thực dữ liệu, kiểm tra quyền truy cập, hoặc quyết định hành động cần thực hiện. Để thực hiện các thao tác liên quan đến dữ liệu, Controller tương tác trực tiếp với **Model**, yêu cầu truy xuất, thêm, sửa, hoặc xóa dữ liệu trong cơ sở dữ liệu. Ví dụ, khi người dùng yêu cầu xem danh sách sinh viên, Controller sẽ gọi Model để lấy dữ liệu từ Database và chuyển kết quả trở lại View.
* **Model: Quản Lý Dữ Liệu và Tương Tác với Database**  
  Model đại diện cho các thực thể dữ liệu trong ứng dụng (như Student, Course, Enrollment, v.v.) và chịu trách nhiệm giao tiếp với **Database** (thường là MySQL) thông qua Eloquent ORM của Laravel. Model định nghĩa cấu trúc của các bảng dữ liệu, mối quan hệ giữa chúng (như belongsTo, hasMany), và thực hiện các truy vấn để truy xuất hoặc cập nhật dữ liệu. Ví dụ, Model Student có thể truy vấn danh sách sinh viên hoặc thêm một bản ghi sinh viên mới vào bảng students. Model đảm bảo tính toàn vẹn dữ liệu và cung cấp dữ liệu chính xác cho Controller.
* **Hoàn Thành Luồng: Trả Kết Quả Về Người Dùng**  
  Sau khi **Model** thực hiện truy vấn và trả về dữ liệu từ **Database**, **Controller** nhận dữ liệu này, xử lý thêm nếu cần (như định dạng hoặc tính toán), và truyền kết quả đến **View**. View sau đó chuyển đổi dữ liệu thành giao diện người dùng, chẳng hạn như hiển thị danh sách sinh viên trong một bảng hoặc thông báo thành công khi thêm sinh viên mới. Kết quả cuối cùng được gửi trở lại trình duyệt, cho phép người dùng xem hoặc tiếp tục tương tác với ứng dụng.

### **2.1.1. Model**

A diagram of a course

AI-generated content may be incorrect.

Hình 2: Sơ đồ Model của website

Trong **Laravel Student Management App**, các **Model** đóng vai trò cốt lõi, đại diện cho cấu trúc dữ liệu và logic nghiệp vụ của ứng dụng. Mỗi Model tương ứng với một thực thể chính trong hệ thống quản lý giáo dục, được thiết kế để lưu trữ, quản lý và xử lý thông tin một cách khoa học và hiệu quả. Dưới đây là mô tả chi tiết về các Model và các mối quan hệ giữa chúng, được xây dựng dựa trên Eloquent ORM của Laravel để đảm bảo tính linh hoạt và dễ dàng mở rộng.

**2.1.1.1 Các Model Chính**

* **User**  
  Model này quản lý thông tin cá nhân của người dùng, bao gồm:
  + Họ và tên
  + Email
  + Password

Là model quản lý thông tin người dung đăng ký/ đăng nhập, giúp quản lý website

* **Student**  
  Model này quản lý thông tin cá nhân của sinh viên, bao gồm:
  + Họ và tên
  + Địa chỉ
  + Số điện thoại
  + Các thông tin khác như khóa học, lớp học tham gia, chi phi khi tham gia (tùy thuộc vào cấu hình cụ thể).  
    Model Student là trung tâm của hệ thống, cung cấp dữ liệu nền tảng để quản lý các hoạt động liên quan đến sinh viên, chẳng hạn như đăng ký khóa học hoặc theo dõi học phí.
* **Teacher**  
  Model này lưu trữ thông tin về giáo viên, bao gồm:
  + Họ và tên
  + Địa chỉ
  + Số điện thoại
  + Các thông tin bổ sung như trình độ học vấn, hoặc chuyên môn.  
    Teacher đảm bảo việc quản lý thông tin giáo viên được thực hiện một cách nhất quán, hỗ trợ việc phân công giáo viên cho các lớp học hoặc khóa học.
* **Course**  
  Model này quản lý thông tin về các khóa học, bao gồm:
  + Tên khóa học
  + Giáo trình hoặc nội dung học
  + Thời lượng của khóa học (theo giờ, tuần, hoặc tháng).  
    Course đóng vai trò định nghĩa các chương trình học, là nền tảng để tổ chức các lớp học và quản lý đăng ký của sinh viên.
* **Class(batch)**  
  Model này đại diện cho các lớp học cụ thể, với các thông tin như:
  + Tên lớp học
  + Khóa học liên kết (một lớp học thuộc về một khóa học)
  + Ngày bắt đầu và ngày kết thúc (nếu có).  
    Batch giúp tổ chức các nhóm sinh viên học cùng một khóa học trong một khoảng thời gian cụ thể.
* **Enrollment**  
  Model này quản lý các đăng ký học của sinh viên, bao gồm:
  + Mã đăng ký
  + Lớp học (Batch) mà sinh viên tham gia
  + Mã sinh viên (student\_id)
  + Ngày tham gia
  + Phí tham gia

Enrollment giúp quản lý được số lượng sinh viên tham gia các lớp(class), quản lý được môn mà họ chọn

* **Payment**
  + Mã đăng ký
  + Thông tin sinh viên thanh toán
  + Thông tin khóa học, lớp học
  + Ngày thanh toán
  + Chi phí chi tiết
  + Payment giúp quản lý được chi phí khóa học 1 cách hiệu quả

**Ví dụ về đoạn code model Student**A screen shot of a computer

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

Hình 3: Code về Model Student

**2.1.1.2 Mối quan hệ giữa các bảng:**

**- Teacher - Course**: Quan hệ nhiều-nhiều (many-to-many) thông qua bảng trung gian `teacher\_course`

- **Course - Batch**: Quan hệ một-nhiều (one-to-many), một khóa học có thể có nhiều lớp học

- **Student - Enrollment**: Quan hệ một-nhiều (one-to-many), một sinh viên có thể đăng ký nhiều lớp học

- **Batch - Enrollment**: Quan hệ một-nhiều (one-to-many), một lớp học có thể có nhiều sinh viên đăng ký

- **Enrollment - Payment**: Quan hệ một-nhiều (one-to-many), một đăng ký có thể có nhiều lần thanh toán

### **2.1.2 View**

Hệ thống giao diện (Views) trong **Laravel Student Management App** đóng vai trò trung tâm trong việc cung cấp trải nghiệm người dùng trực quan, thân thiện và hiệu quả. Được xây dựng dựa trên công cụ Blade templating của Laravel, các Views đảm bảo tính nhất quán, dễ sử dụng, và hỗ trợ tương tác mượt mà giữa người dùng và hệ thống. Các thành phần giao diện được tổ chức khoa học, cho phép quản trị viên, giáo viên, hoặc sinh viên thực hiện các thao tác quản lý một cách dễ dàng. Dưới đây là mô tả chi tiết về các thành phần giao diện chính của ứng dụng, bao gồm **Layout**, **Dashboard**, và các giao diện **CRUD**.

#### **2.1.2.1 Layout (Bố Cục Giao Diện Chung)**

**Mô tả**  
Layout là khung giao diện chung, đóng vai trò như nền tảng cho toàn bộ ứng dụng, đảm bảo sự nhất quán về thiết kế và trải nghiệm người dùng trên mọi trang. Được triển khai thông qua các template Blade, Layout tích hợp các yếu tố giao diện chính để tạo ra một môi trường điều hướng trực quan và chuyên nghiệp.

**Thành phần chính**

* **Header (Thanh tiêu đề):** Hiển thị logo ứng dụng, tên hệ thống, và thông tin người dùng đăng nhập (như tên, vai trò). Header còn có thể bao gồm các nút đăng xuất hoặc chuyển đổi ngôn ngữ (nếu hỗ trợ đa ngôn ngữ).
* **Sidebar (Thanh điều hướng):** Cung cấp các liên kết nhanh đến các chức năng chính như quản lý sinh viên, giáo viên, khóa học, lớp học, ghi danh, hoặc thanh toán. Sidebar được thiết kế gọn gàng, hỗ trợ thu gọn/mở rộng để tối ưu không gian.
* **Footer (Chân trang):** Chứa thông tin bản quyền, thông tin liên hệ của trường học, hoặc các liên kết đến tài liệu hướng dẫn, chính sách sử dụng.

**Vai trò**  
Layout đảm bảo người dùng dễ dàng định hướng trong hệ thống, chuyển đổi giữa các chức năng mà không gặp trở ngại. Bằng cách duy trì sự đồng nhất về màu sắc, phông chữ, và bố cục, Layout không chỉ nâng cao tính thẩm mỹ mà còn cải thiện hiệu quả sử dụng, giảm thiểu thời gian làm quen với giao diện.

#### **2.1.2.2. Dashboard (Bảng Điều Khiển Tổng Quan)**

**Mô tả**  
Dashboard là giao diện chính xuất hiện ngay sau khi người dùng đăng nhập, cung cấp cái nhìn tổng quan về trạng thái của hệ thống thông qua các số liệu thống kê và biểu đồ trực quan. Được thiết kế để tối ưu hóa khả năng tiếp cận thông tin, Dashboard là công cụ hỗ trợ đắc lực cho quản trị viên và giáo viên trong việc quản lý và ra quyết định.

**Nội dung hiển thị**

* **Thống kê tổng quan:** Tổng số sinh viên, giáo viên, khóa học, và lớp học hiện có trong hệ thống.
* **Biểu đồ phân tích:** Biểu đồ về số lượng học viên theo lớp học, tỷ lệ hoàn thành khóa học, hoặc doanh thu từ học phí (nếu tích hợp chức năng thanh toán).
* **Thông báo và lịch sử:** Các thông báo quan trọng (như lịch học sắp tới) và nhật ký hoạt động gần đây (như thêm sinh viên mới, cập nhật điểm số).
* **Tùy chỉnh theo vai trò:** Dashboard có thể hiển thị nội dung khác nhau tùy thuộc vào vai trò người dùng (quản trị viên thấy toàn bộ dữ liệu, giáo viên chỉ thấy lớp học liên quan).

**Vai trò**  
Dashboard cung cấp một cái nhìn tổng thể, giúp người dùng nhanh chóng nắm bắt tình hình hoạt động của hệ thống. Các biểu đồ và số liệu trực quan hỗ trợ việc phân tích dữ liệu, từ đó đưa ra các quyết định quản lý kịp thời và chính xác. Giao diện này đóng vai trò như một trung tâm thông tin, tăng cường hiệu quả quản lý giáo dục.

#### **2.1.2.3 Giao Diện CRUD (Tạo, Đọc, Cập Nhật, Xóa)**

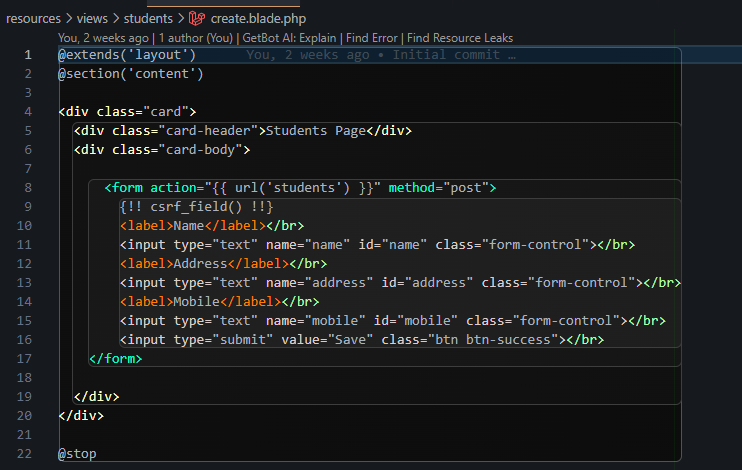
**Mô tả**  
Các giao diện CRUD (Create, Read, Update, Delete) là nền tảng cho việc quản lý các thực thể chính trong ứng dụng, bao gồm sinh viên (Student), giáo viên (Teacher), khóa học (Course), lớp học (Batch), ghi danh (Enrollment), và thanh toán (Payment). Các giao diện này được thiết kế để hỗ trợ các thao tác quản lý dữ liệu một cách hiệu quả, với tính năng tìm kiếm, lọc, và phân trang để tối ưu hóa trải nghiệm người dùng.

**Chi tiết từng giao diện**

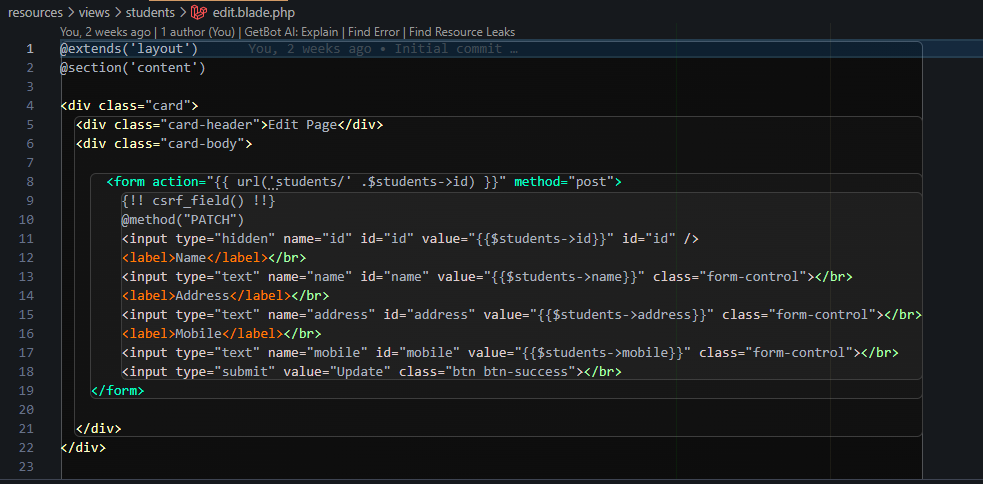
* **Danh sách (List):**  
  Hiển thị dữ liệu dưới dạng bảng, cho phép người dùng xem danh sách các thực thể (ví dụ: danh sách sinh viên với các cột như tên, email, lớp học). Tích hợp các tính năng:
  + **Tìm kiếm:** Tìm kiếm theo tên, mã số, hoặc các trường khác.
  + **Lọc:** Lọc dữ liệu theo tiêu chí như lớp học, khóa học, hoặc trạng thái thanh toán.
  + **Phân trang:** Giới hạn số bản ghi trên mỗi trang để tăng hiệu suất tải.
* **Tạo mới (Create):**  
  Cung cấp biểu mẫu (form) với các trường nhập liệu cần thiết (ví dụ: tên, email, ngày sinh cho sinh viên). Biểu mẫu tích hợp kiểm tra hợp lệ (validation) để đảm bảo dữ liệu chính xác, hiển thị thông báo lỗi hoặc thành công sau khi gửi.
* **Chỉnh sửa (Edit):**  
  Hiển thị biểu mẫu điền sẵn thông tin của bản ghi hiện tại, cho phép người dùng chỉnh sửa và lưu lại. Các kiểm tra hợp lệ tương tự như giao diện tạo mới được áp dụng để đảm bảo tính toàn vẹn dữ liệu.
* **Chi tiết (Detail):**  
  Hiển thị thông tin chi tiết của một bản ghi cụ thể (ví dụ: thông tin sinh viên, lịch sử ghi danh, hoặc các khoản thanh toán). Giao diện này hữu ích khi cần xem toàn bộ thông tin liên quan mà không chỉnh sửa.
* **Xóa (Delete):**  
  Cung cấp chức năng xóa bản ghi, đi kèm với thông báo xác nhận để tránh thao tác nhầm lẫn, đảm bảo an toàn dữ liệu.

**Vai trò**  
Các giao diện CRUD là xương sống của hệ thống, cho phép người dùng thực hiện các thao tác quản lý dữ liệu một cách trực quan và hiệu quả. Với thiết kế thân thiện, tích hợp kiểm tra hợp lệ và thông báo rõ ràng, các giao diện này không chỉ đảm bảo tính chính xác của dữ liệu mà còn nâng cao trải nghiệm người dùng, giúp quản lý giáo dục trở nên dễ dàng và hiệu quả hơn.

**Ví dụ về 1 vài code view của Student trong folder View**



Hình 4: Code create.blade.php của Student trong folder View

Hình 5: Code edit.blade.php của Student trong folder View

### **2.1.3 Controller**

#### **2.1.3.1 Chức Năng Cốt Lõi của Controller**

Controller trong ứng dụng thực hiện các nhiệm vụ quan trọng sau:

* **Tiếp nhận và xử lý yêu cầu (HTTP Requests):** Controller tiếp nhận các yêu cầu từ người dùng thông qua các route được định nghĩa trong tệp web.php, bao gồm các phương thức HTTP như GET, POST, PUT, DELETE.
* **Xác thực dữ liệu đầu vào:** Trước khi xử lý, Controller sử dụng các cơ chế xác thực (validation) của Laravel để đảm bảo dữ liệu người dùng nhập vào hợp lệ, an toàn, và đúng định dạng.
* **Tương tác với Model:** Controller gọi các phương thức của Model để thực hiện các thao tác như truy xuất, thêm mới, cập nhật, hoặc xóa dữ liệu trong cơ sở dữ liệu.
* **Xử lý logic nghiệp vụ:** Controller thực hiện các phép tính hoặc kiểm tra logic phức tạp, ví dụ: tính điểm trung bình của sinh viên, kiểm tra điều kiện ghi danh, hoặc xác định trạng thái thanh toán.
* **Trả kết quả về View hoặc chuyển hướng:** Sau khi xử lý, Controller truyền dữ liệu đến View để hiển thị hoặc thực hiện chuyển hướng (redirect) đến các trang khác kèm thông báo thành công/thất bại.
* **Quản lý phân quyền:** Controller tích hợp middleware để kiểm soát quyền truy cập, ví dụ: chỉ cho phép quản trị viên thực hiện các thao tác như thêm hoặc xóa dữ liệu.

#### **2.1.3.2 Cấu Trúc và Tổ Chức Controller**

Trong **Laravel Student Management App**, mỗi thực thể chính của hệ thống được quản lý bởi một Controller riêng biệt, đảm bảo tính mô-đun và dễ dàng bảo trì. Các Controller chính bao gồm:

* **StudentController**: Quản lý các thao tác liên quan đến sinh viên, như thêm, sửa, xóa, xem chi tiết, hoặc tìm kiếm sinh viên.
* **TeacherController**: Xử lý thông tin giáo viên, bao gồm tạo mới, cập nhật, và quản lý danh sách giáo viên.
* **CourseController**: Quản lý các khóa học, từ việc tạo khóa học mới đến chỉnh sửa hoặc xóa thông tin khóa học.
* **BatchController**: Điều phối các lớp học, hỗ trợ phân công sinh viên vào lớp và quản lý lịch học.
* **EnrollmentController**: Xử lý việc ghi danh sinh viên vào các lớp học hoặc khóa học, đảm bảo tính hợp lệ của dữ liệu ghi danh.
* **PaymentController**: Quản lý các giao dịch thanh toán học phí, bao gồm ghi nhận và theo dõi trạng thái thanh toán.
* **AuthController**: Đảm nhiệm các chức năng xác thực người dùng, như đăng nhập, đăng ký, đổi mật khẩu, hoặc quản lý phiên đăng nhập.

Mỗi Controller được đặt trong thư mục app/Http/Controllers/ và tuân theo cấu trúc chuẩn của Laravel, với các phương thức được thiết kế để xử lý các yêu cầu cụ thể.

#### **2.1.3.3 Các Phương Thức Tiêu Biểu trong Controller**

A screenshot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect.

A screen shot of a computer screen

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

Hình 6: Các phương thức trong StudentController

Để minh họa, dưới đây là các phương thức phổ biến trong **StudentController**, thể hiện các thao tác CRUD (Create, Read, Update, Delete):

* **index()**: Truy xuất danh sách sinh viên từ Model và truyền dữ liệu đến View để hiển thị dưới dạng bảng, hỗ trợ tìm kiếm và phân trang.
* **create()**: Hiển thị biểu mẫu (form) để thêm sinh viên mới, thường là một View như students.create.
* **store(Request $request)**: Xử lý dữ liệu từ biểu mẫu thêm mới, xác thực các trường như tên, email, số điện thoại, sau đó lưu vào cơ sở dữ liệu thông qua Model.
* **show($id)**: Lấy thông tin chi tiết của một sinh viên dựa trên ID và hiển thị trên View students.show.
* **edit($id)**: Truy xuất dữ liệu sinh viên theo ID và hiển thị biểu mẫu chỉnh sửa, cho phép người dùng cập nhật thông tin.
* **update(Request $request, $id)**: Xác thực và lưu các thay đổi từ biểu mẫu chỉnh sửa vào cơ sở dữ liệu.
* **destroy($id)**: Xóa một sinh viên khỏi hệ thống sau khi xác nhận, đảm bảo an toàn dữ liệu.

### **2.1.4 Đăng ký, đăng nhập bằng package Fortify**

#### **2.1.4.1 Các điều kiện ràng buộc và chức năng**

**- Đăng ký:**

* Người dùng nhập thông tin cá nhân như: tên, email, mật khẩu.
* Hệ thống kiểm tra: Email đã tồn tại chưa, mật khẩu có đủ độ dài và tính bảo mật không.
* Nếu hợp lệ, thông tin sẽ được lưu vào cơ sở dữ liệu.
* Hiển thị thông báo đăng ký thành công.

**- Đăng nhâp:**

* Người dùng nhập email và mật khẩu.
* Hệ thống kiểm tra: Thông tin có khớp với dữ liệu đã lưu trong cơ sở dữ liệu không.
* Nếu đúng: cho phép truy cập vào hệ thống.
* Nếu sai: hiển thị thông báo lỗi.

#### **2.1.4.2 Cách hoạt hoạt động**

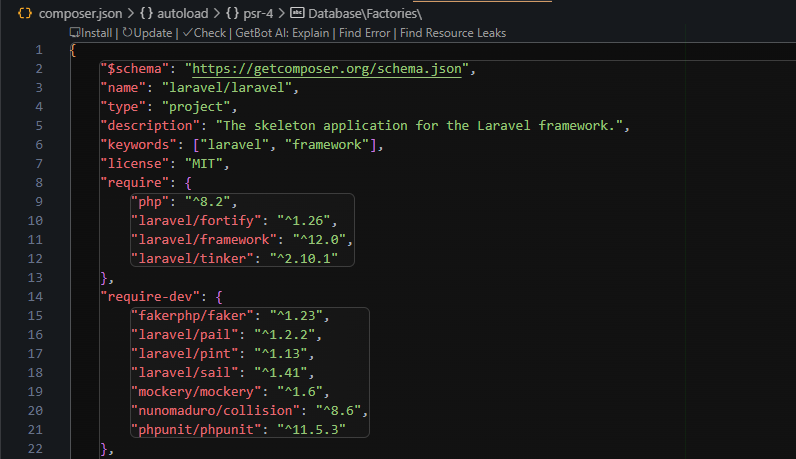
**- Đăng ký (Registration):**

* Khi người dùng gửi form đăng ký, request sẽ được gửi tới route do Fortify cung cấp.
* Fortify gọi file CreatNewUser.php để xác và tạo user mới, sử dụng luật từ file PasswordValidationRules.php
* Nếu thành công, user được lưu vào bảng users

- **Đăng nhập (Login):**

* Người dùng gửi form đăng nhập tới route /login.
* Fortify kiểm tra thông tin đăng nhập (email, password) với bảng users.
* Nếu đúng, session được tạo và người dùng được chuyển hướng tới trang dashboard .

#### **2.1.4.3 Code minh họa**



**Hình 7: Version package Fortify**

A screenshot of a computer program

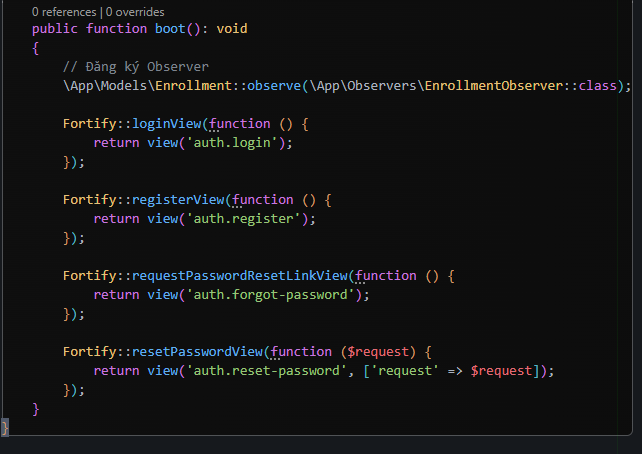
AI-generated content may be incorrect.

**Hình 8: Code của CreatNewUser**

**A computer screen with text on it

AI-generated content may be incorrect.**

**Hình 9: Code của PasswordValidationRules**



**Hình 10: Cấu hình trong AppServiceProvider**

## **2.2. Thiết kế cơ sở dữ liệu**

### **2.2.1 Sơ đồ kết nối các bảng**

A screenshot of a computer

AI-generated content may be incorrect.

### **2.2.2 Cấu trúc bảng**

#### **2.2.2.1 Bảng teachers**

Lưu trữ thông tin chi tiết của các giảng viên

|  |  |  |
| --- | --- | --- |
| Tên trường | Kiểu dữ liệu | Ý nghĩa |
| id | bigint | Khóa chính(Primary Key) Mã giảng viên |
| name | varchar | Tên giảng viên |
| address | varchar | Địa chỉ |
| mobile | varchar | Số điện thoại |
| specialization | varchar | chuyên ngành |
| experience | int | số năm kinh nghiệm |
| qualification | varchar | trình độ chuyên môn |
| join\_date | date | ngày vào làm |
| created\_at | timestamp | ngày tạo bản ghi |
| updated\_at | timestamp | thời điểm cập nhật |

#### **2.2.2.2 Bảng courses**

Quản lý thông tin các khóa học được cung cấp trong hệ thống

|  |  |  |
| --- | --- | --- |
| Tên trường | Kiểu dữ liệu | Ý nghĩa |
| id | bigint | Khóa chính(Primary Key) Mã khóa học |
| name | varchar | Tên khóa học |
| syllabus | varchar | Giáo trình |
| duration | varchar | Thời lượng học |
| created\_at | timestamp | Thời điểm tạo bản ghi |
| updated\_at | timestamp | Thời điểm cập nhật |

#### **2.2.2.3 Bảng teacher\_course**

Thiết lập mối quan hệ N-N giữa giảng viên và khóa học

|  |  |  |
| --- | --- | --- |
| Tên trường | Kiểu dữ liệu | Ý nghĩa |
| id | bigint | Khóa chính(Primary Key) Mã bản ghi |
| teacher\_id | bigint | Khóa ngoại liên kết đến giảng viên (teacher.id) |
| course\_id | bigint | Khóa ngoại liên kết đến khóa học (course.id) |
| role | varchar | Vai trò (ví dụ: giảng viên chính/phụ) |
| is\_primary | tinyint(1) | Có phải giảng viên chính không (1 = có, 0 = không) |
| created\_at | timestamp | Thời điểm tạo |
| updated\_at | timestamp | Thời điểm cập nhật |

#### **2.2.2.4 Bảng batchs (classes)**

Mỗi batch là một lớp học của một khóa cụ thể, có ngày bắt đầu và tên lớp

|  |  |  |
| --- | --- | --- |
| Tên trường | Kiểu dữ liệu | Ý nghĩa |
| id | bigint | Khóa chính(Primary Key) Mã lớp học |
| name | varchar | Tên lớp |
| course\_id | bigint | Khóa ngoại liên kết đến khóa học (batch.id) |
| start\_date | date | Ngày bắt đầu |
| created\_at | timestamp | Thời điểm tạo |
| updated\_at | timestamp | Thời điểm cập nhật |

#### **2.2.2.5 Bảng students**

Lưu thông tin cá nhân của sinh viên

|  |  |  |
| --- | --- | --- |
| Tên trường | Kiểu dữ liệu | Ý nghĩa |
| id | bigint | Khóa chính(Primary Key) Mã sinh viên |
| name | varchar | Họ tên sinh viên |
| gpa | decimal | Điểm GPA trung bình |
| address | varchar | Địa chỉ |
| mobile | varchar | Số điện thoại |
| created\_at | timestamp | Thời điểm tạo |
| updated\_at | timestamp | Thời điểm cập nhật |

#### **2.2.2.6 Bảng enrollments**

Ghi nhận việc sinh viên đăng ký học một lớp học cụ thể (batch). Ngoài ra còn quản lý điểm thành phần, tổng điểm và nhận xét.

|  |  |  |
| --- | --- | --- |
| Tên trường | Kiểu dữ liệu | Ý nghĩa |
| id | bigint | Khóa chính(Primary Key) Mã đăng ký |
| enroll\_no | varchar | Mã số đăng ký (VD: EF123456) |
| batch\_id | bigint | Khóa ngoại liên kết đến lớp học (batch.id) |
| student\_id | bigint | Khóa ngoại liên kết đến sinh viên (student.id) |
| join\_date | date | Ngày đăng ký |
| fees | double | Học phí |
| midterm\_grade | double | Điểm giữa kỳ |
| final\_grade | double | Điểm cuối kỳ |
| assignment\_grade | double | Điểm bài tập |
| total\_grade | double | Tổng điểm |
| grade\_remarks | text | Nhận xét về học lực |
| created\_at | timestamp | Thời điểm tạo |
| updated\_at | timestamp | Thời điểm cập nhật |

#### **2.2.2.7 Bảng payments**

Lưu các khoản học phí mà sinh viên đã nộp

|  |  |  |
| --- | --- | --- |
| Tên trường | Kiểu dữ liệu | Ý nghĩa |
| id | bigint | Khóa chính(Primary Key) Mã đăng thanh toán |
| enrollment\_id | bigint | Khóa ngoại liên kết đến bảng đăng ký (enrollment.id) |
| payment\_date | date | Ngày thanh toán |
| amount | double | Số tiền đã thanh toán |
| created\_at | timestamp | Thời điểm tạo |
| updated\_at | timestamp | Thời điểm cập nhật |

## **2.3. Công nghệ sử dụng**

Để xây dựng và triển khai dự án một cách hiệu quả, tôi đã sử dụng các công nghệ hiện đại và phù hợp với yêu cầu của hệ thống, bao gồm:

### **2.3.1 Framework: Laravel (PHP)**

* + **Laravel** được chọn làm framework chính nhờ tính mạnh mẽ, linh hoạt và cộng đồng hỗ trợ rộng lớn. Framework này cung cấp các tính năng như ORM (Eloquent) để tương tác dễ dàng với cơ sở dữ liệu, middleware để quản lý xác thực, và hệ thống định tuyến (routing) mạnh mẽ, giúp tăng tốc độ phát triển và đảm bảo mã nguồn sạch, dễ bảo trì.
  + Ưu điểm: Laravel hỗ trợ các mẫu thiết kế MVC (Model-View-Controller), giúp tổ chức mã nguồn rõ ràng và tối ưu hóa hiệu suất ứng dụng.

### **2.3.2 Cơ sở dữ liệu: MySQL**

* + **MySQL** được sử dụng làm hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) nhờ hiệu suất cao, khả năng mở rộng và độ tin cậy. MySQL cho phép lưu trữ và truy vấn dữ liệu hiệu quả, phù hợp với các ứng dụng web cần xử lý lượng lớn dữ liệu.
  + Ưu điểm: Tích hợp tốt với Laravel qua Eloquent ORM, hỗ trợ các truy vấn phức tạp và đảm bảo tính toàn vẹn dữ liệu.

### **2.3.3 Giao diện người dùng: Blade Template, Bootstrap, CSS, HTML, JavaScript**

* + **Blade Template**: Công cụ templating của Laravel, cho phép tạo giao diện động, tái sử dụng và dễ bảo trì. Blade giúp tích hợp logic PHP vào giao diện một cách mượt mà, tăng hiệu quả phát triển.
  + **Bootstrap**: Framework CSS được sử dụng để xây dựng giao diện responsive, thân thiện với người dùng trên nhiều thiết bị. Bootstrap cung cấp các thành phần giao diện như button, form, và grid system, giúp tiết kiệm thời gian thiết kế.
  + **CSS/HTML**: Sử dụng để tùy chỉnh giao diện, đảm bảo tính thẩm mỹ và trải nghiệm người dùng mượt mà.
  + **JavaScript**: Được áp dụng để thêm các tính năng tương tác động như xử lý sự kiện, gọi API không đồng bộ (AJAX), và cải thiện trải nghiệm người dùng.

**Lý do lựa chọn công nghệ**

* **Tính phù hợp**: Laravel và MySQL là bộ đôi mạnh mẽ, phù hợp cho các ứng dụng web quy mô vừa và lớn. Blade Template và Bootstrap giúp tối ưu hóa thời gian phát triển giao diện.
* **Hiệu suất và bảo trì**: Các công nghệ này đảm bảo hiệu suất cao, dễ mở rộng và bảo trì lâu dài.
* **Cộng đồng hỗ trợ**: Cả Laravel, MySQL, và Bootstrap đều có cộng đồng lớn, cung cấp tài liệu phong phú và các giải pháp sẵn có trên các nền tảng như X, Stack Overflow, và GitHub.

**Kết luận**

Sự kết hợp giữa Laravel, MySQL, Blade Template, Bootstrap, CSS, HTML, và JavaScript đã tạo nên một hệ thống mạnh mẽ, đáp ứng tốt các yêu cầu về tính năng, giao diện, và hiệu suất. Các công nghệ này không chỉ hỗ trợ phát triển nhanh chóng mà còn đảm bảo khả năng mở rộng và bảo mật cho dự án.

## **2.4. Phương pháp phát triển**

Để đảm bảo dự án được triển khai hiệu quả, dễ bảo trì và đáp ứng yêu cầu kỹ thuật, tôi đã áp dụng các phương pháp phát triển sau:

### **2.4.1 Phát triển theo hướng mô-đun (Modular Development)**

* + Hệ thống được chia thành các mô-đun độc lập, mỗi mô-đun đảm nhiệm một chức năng cụ thể (ví dụ: quản lý người dùng, xử lý dữ liệu, giao diện).
  + Lợi ích: Dễ dàng mở rộng, bảo trì và tái sử dụng mã nguồn. Việc thêm tính năng mới hoặc sửa lỗi chỉ cần tập trung vào mô-đun liên quan mà không ảnh hưởng đến toàn bộ hệ thống.

### **2.4.2 Áp dụng Best Practices của Laravel**

* + Tuân thủ mô hình MVC (Model-View-Controller) của Laravel để tổ chức mã nguồn rõ ràng, tăng khả năng bảo trì.
  + Sử dụng **Eloquent ORM** để tương tác an toàn và hiệu quả với cơ sở dữ liệu MySQL, giảm thiểu nguy cơ lỗi SQL Injection.
  + Triển khai **middleware** để quản lý xác thực và phân quyền, đảm bảo bảo mật hệ thống.
  + Tối ưu hiệu năng bằng các kỹ thuật như **caching** (sử dụng Laravel Cache) và **lazy loading** để giảm tải cơ sở dữ liệu.
  + Áp dụng **CSRF protection** và mã hóa dữ liệu để tăng cường bảo mật.

### **2.4.3 Xây dựng giao diện thân thiện và dễ sử dụng**

* + Sử dụng **Blade Template** kết hợp với **Bootstrap** để tạo giao diện responsive, tương thích trên nhiều thiết bị (máy tính, điện thoại, máy tính bảng).
  + Tùy chỉnh **CSS** và **HTML** để đảm bảo giao diện trực quan, thẩm mỹ, và nhất quán.
  + Tích hợp **JavaScript** để thêm các tính năng tương tác động như xử lý form thời gian thực, thông báo, và gọi API không đồng bộ (AJAX), nâng cao trải nghiệm người dùng.

**Kết quả đạt được**

* **Tính mô-đun**: Hệ thống dễ dàng mở rộng, ví dụ: thêm mô-đun mới như phân tích dữ liệu hoặc tích hợp API bên thứ ba.
* **Bảo mật và hiệu năng**: Ứng dụng hoạt động ổn định, an toàn, và đáp ứng nhanh nhờ các best practices của Laravel.
* **Trải nghiệm người dùng**: Giao diện trực quan, thân thiện, phù hợp với đối tượng người dùng đa dạng.

# **Chương 3: Chức năng và Kết quả đạt được**

## **3.1 Chức năng chính**

Hệ thống quản lý sinh viên được phát triển với các chức năng cốt lõi nhằm đáp ứng nhu cầu quản lý thông tin học tập hiệu quả, bao gồm:

**Đăng ký, đăng nhâp sử dụng package Fortify của Laravel:**

**- Đăng ký:**

* Người dùng nhập thông tin cá nhân như: tên, email, mật khẩu.
* Hệ thống kiểm tra: Email đã tồn tại chưa, mật khẩu có đủ độ dài và tính bảo mật không.
* Nếu hợp lệ, thông tin sẽ được lưu vào cơ sở dữ liệu.
* Hiển thị thông báo đăng ký thành công.

**- Đăng nhâp:**

* Người dùng nhập email và mật khẩu.
* Hệ thống kiểm tra: Thông tin có khớp với dữ liệu đã lưu trong cơ sở dữ liệu không.
* Nếu đúng: cho phép truy cập vào hệ thống.
* Nếu sai: hiển thị thông báo lỗi.

**Quản lý sinh viên**:

* Cho phép **thêm**, **sửa**, **xóa** và **xem danh sách** thông tin sinh viên (họ tên, mã số, ngày sinh, email, v.v.).
* Giao diện hiển thị danh sách sinh viên chi tiết, hỗ trợ quản lý dễ dàng và trực quan.

**Quản lý lớp học**:

* Hỗ trợ **thêm**, **sửa**, **xóa** thông tin lớp học (mã lớp, tên lớp, giảng viên phụ trách, lịch học).
* Chức năng **phân công sinh viên** vào lớp, đảm bảo quản lý linh hoạt và chính xác.

**Quản lý môn học**:

* Cho phép **thêm**, **sửa**, **xóa** thông tin môn học (mã môn, tên môn, số tín chỉ, mô tả).
* Đảm bảo thông tin môn học được lưu trữ nhất quán và dễ truy cập.

**Quản lý điểm**:

* Hỗ trợ **nhập điểm**, **sửa điểm** và **xem bảng điểm** của sinh viên theo môn học, lớp học hoặc kỳ học.
* Cung cấp giao diện tổng quan để theo dõi kết quả học tập của từng sinh viên.

**Quản lý enrollment (đăng ký học phần)**:

* Cho phép sinh viên **đăng ký** hoặc **hủy đăng ký** các môn học trong kỳ.
* Quản trị viên có thể xem danh sách sinh viên đã đăng ký theo môn học hoặc lớp, hỗ trợ kiểm soát số lượng và trạng thái đăng ký.

**Quản lý người dùng**:

* Hỗ trợ **thêm**, **sửa**, **xóa** và quản lý tài khoản người dùng (quản trị viên, giáo viên, sinh viên).
* Giao diện quản lý cho phép phân quyền chi tiết, ví dụ: giáo viên chỉ chỉnh sửa điểm, quản trị viên có toàn quyền.

**Quản lý payment (thanh toán)**:

* Cho phép nhập, theo dõi và cập nhật trạng thái thanh toán học phí của sinh viên (đã thanh toán, chưa thanh toán, nợ).
* Hỗ trợ xuất báo cáo thanh toán và gửi thông báo nhắc nhở cho sinh viên.

**Tìm kiếm và lọc dữ liệu**:

* Tích hợp chức năng **tìm kiếm nhanh** theo tên, mã số sinh viên, lớp học, môn học, hoặc trạng thái thanh toán.
* Hỗ trợ **lọc dữ liệu** theo các tiêu chí như lớp, môn học, điểm số, hoặc trạng thái đăng ký, giúp tiết kiệm thời gian tra cứu.

**Xác thực và phân quyền**:

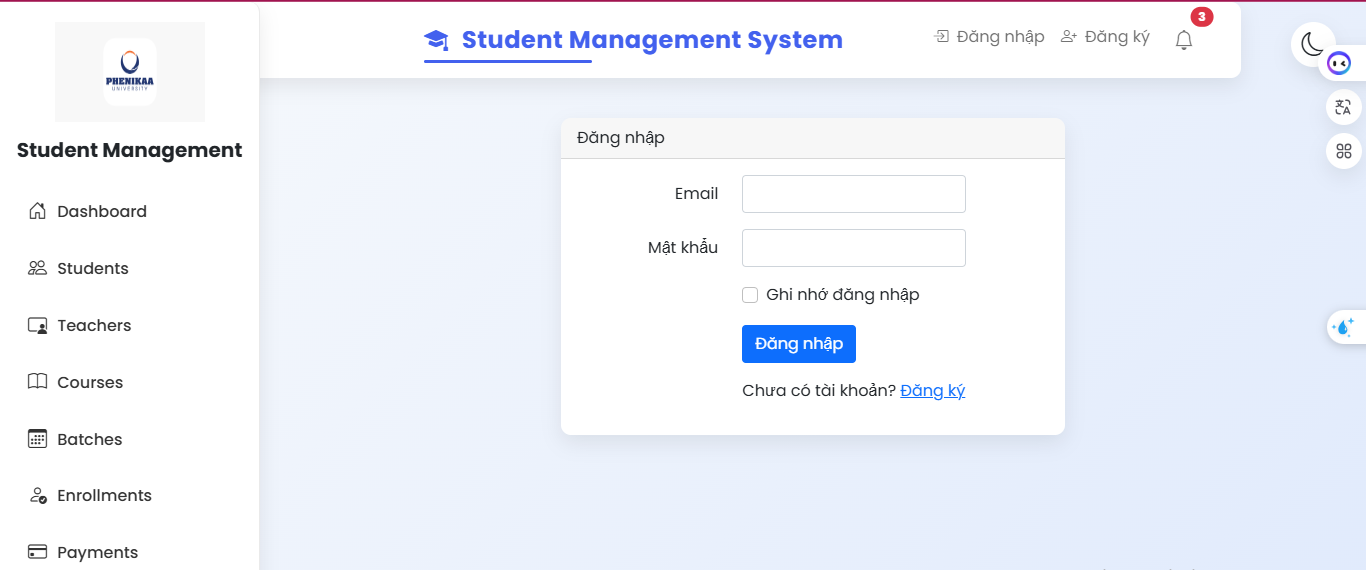
* Hệ thống đăng nhập an toàn với **xác thực người dùng** cho các vai trò (quản trị viên, giáo viên, sinh viên).
* Áp dụng **phân quyền** chi tiết thông qua middleware của Laravel, đảm bảo dữ liệu nhạy cảm được bảo vệ bằng CSRF và các cơ chế bảo mật khác.

## **3.2 Kết quả đạt được**

* **Hiệu quả quản lý**:
  + Hệ thống tự động hóa quy trình quản lý sinh viên, lớp học, môn học và điểm số, giảm thiểu sai sót so với phương pháp thủ công.
  + Chức năng tìm kiếm và lọc giúp quản trị viên truy xuất thông tin nhanh chóng, nâng cao hiệu suất làm việc.
* **Tính bảo mật**:
  + Dữ liệu được bảo vệ an toàn nhờ xác thực người dùng, mã hóa và phân quyền chặt chẽ, đáp ứng yêu cầu bảo mật cơ bản của hệ thống giáo dục.
* **Trải nghiệm người dùng**:
  + Giao diện thân thiện, xây dựng bằng **Blade Template** và **Bootstrap**, đảm bảo tính responsive và dễ sử dụng trên nhiều thiết bị.
  + Các tính năng tương tác như nhập liệu, tìm kiếm và xem báo cáo được tối ưu hóa, mang lại trải nghiệm mượt mà.
* **Khả năng mở rộng**:
  + Nhờ thiết kế mô-đun và sử dụng Laravel, hệ thống dễ dàng tích hợp thêm các tính năng mới như báo cáo thống kê, xuất file Excel, hoặc tích hợp với các API bên ngoài.
* **Hiệu suất**:
  + Sử dụng **MySQL** và **Eloquent ORM** giúp truy vấn dữ liệu nhanh chóng, ngay cả với khối lượng dữ liệu lớn.
  + Tối ưu hóa hiệu năng bằng caching và lazy loading, đảm bảo hệ thống hoạt động ổn định.

# **Chương 4: Giao diện**

## **4.1. Giao diện Đăng nhập/Đăng ký**



A screenshot of a computer

AI-generated content may be incorrect.

## **4.2 Giao diện Dashboard**

A screenshot of a computer

AI-generated content may be incorrect.

## **4.3 Giao diện StudentManagement**

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## **4.4 Giao diện Teacher Management**

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## **4.5 Giao diện Class**

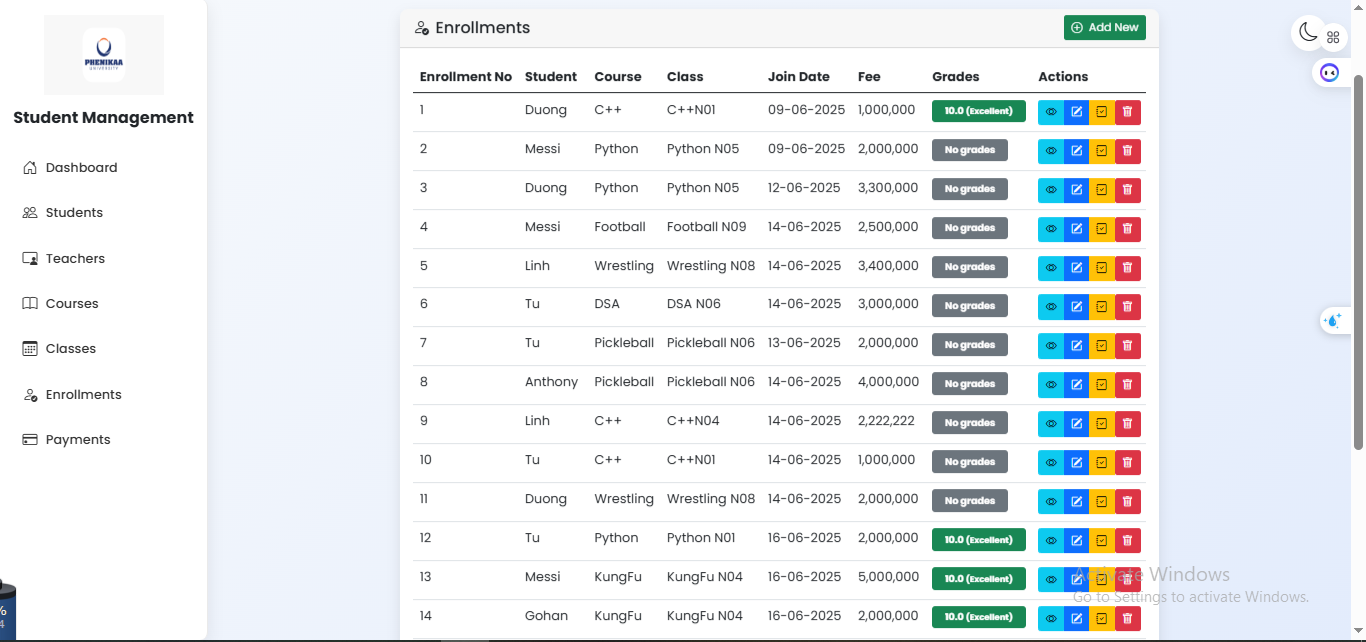
A screen shot of a computer

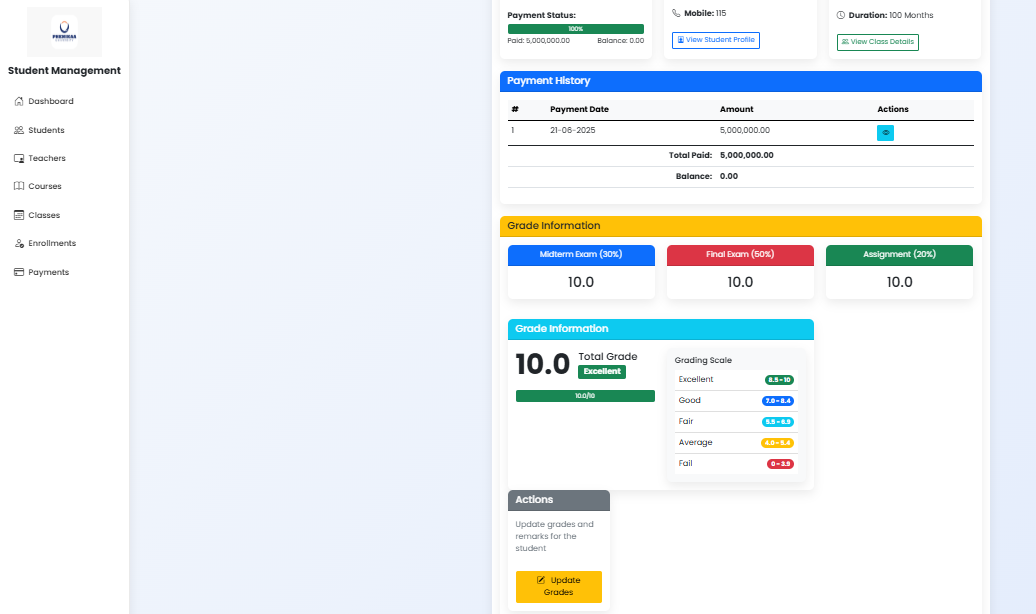
AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## **4.6 Giao diện Enrollment**





## **4.7 Giao diện Payment**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## **4.8 Một vài hình ảnh giao diện DarkMode**



A screen shot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

# **Chương 5: Kết luận**

Laravel Student Management Website không chỉ là một ứng dụng quản lý sinh viên đơn thuần, mà là một giải pháp toàn diện và tối ưu, được thiết kế đặc biệt cho các tổ chức giáo dục mong muốn hiện đại hóa và tự động hóa quy trình quản lý sinh viên một cách chuyên nghiệp và hiệu quả.

Bằng việc khai thác tối đa tiềm năng của Laravel framework – một trong những PHP framework mạnh mẽ, phổ biến và có cộng đồng hỗ trợ lớn nhất hiện nay – cùng với việc áp dụng kiến trúc MVC (Model-View-Controller) rõ ràng và khoa học, dự án đảm bảo tính dễ bảo trì, dễ phát triển và khả năng làm việc nhóm hiệu quả.

Ứng dụng này không chỉ đáp ứng xuất sắc các nghiệp vụ quản lý sinh viên cốt lõi như quản lý thông tin cá nhân, quá trình học tập, điểm số, mà còn đặt nền tảng vững chắc cho việc tích hợp và phát triển các tính năng phức tạp hơn trong tương lai. Từ việc quản lý lịch học, tài liệu, các khoản phí, cho đến việc xây dựng cổng thông tin tương tác cho sinh viên và giảng viên, tất cả đều có thể được mở rộng một cách linh hoạt.

Với Laravel Student Management Website, các đơn vị giáo dục sẽ sở hữu một công cụ mạnh mẽ, giúp tiết kiệm thời gian, giảm thiểu sai sót, nâng cao hiệu quả quản lý và mang đến trải nghiệm tốt hơn cho cả người quản lý, giảng viên và sinh viên. Đây chính là bước đệm quan trọng để tiến tới một môi trường giáo dục số hóa, năng động và tiên tiến.