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**

**Nhóm**: **07**

**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 | 23010909 | 23010909@st.phenikaa-uni.edu.vn |

**Hà Nội, 05-2025**

**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, thiết kế chức năng quản lý sách và Module đọc ghi File | 1/3 |
| Nguyễn Văn Tú | 23010559 | Xây dựng, thiết kế chức năng mượn trả sách, tài khoản, bảo mật | 1/3 |

**MỤC LỤC**

**CHƯƠNG 1: GIỚI THIỆU CHUNG**

**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.

**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:

1. **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.
2. **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.
3. **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.
4. **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.

**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.
* **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.

**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.

**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.

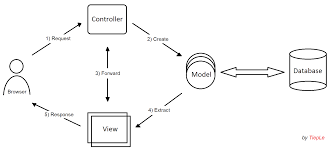
**Thách thức và bài học kinh nghiệm**

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**



* **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.

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.

**Các Model Chính**

1. **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

1. **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 email, ngày sinh (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.
2. **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 email, 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.
3. **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.
4. **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.
5. **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(batch), quản lý được môn mà họ chọn

**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**.

**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. 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.

**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.

**Ví dụ cụ thể**

* **Quản lý sinh viên:**
  + **Danh sách:** Hiển thị bảng sinh viên với các cột như tên, email, lớp học, hỗ trợ tìm kiếm theo tên hoặc lọc theo lớp.
  + **Tạo mới:** Biểu mẫu nhập thông tin sinh viên (tên, email, số điện thoại, ngày sinh, lớp học).
  + **Chỉnh sửa:** Cập nhật thông tin sinh viên, ví dụ: thay đổi số điện thoại hoặc lớp học.
  + **Xóa:** Xóa sinh viên khỏi hệ thống sau khi xác nhận.
* **Quản lý khóa học:**
  + **Danh sách:** Hiển thị danh sách các khóa học với thông tin như tên, thời lượng, giáo viên phụ trách.
  + **Tạo mới:** Nhập thông tin khóa học mới (tên, giáo trình, thời lượng).
  + **Chỉnh sửa/Xóa:** Cập nhật hoặc xóa khóa học khi không còn cần thiết.

**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.