**Thiết kế kiến trúc hệ thống (System Architecture Design)**

Hệ thống quản lý điểm danh học sinh sẽ tuân theo mô hình kiến trúc **MVC (Model-View-Controller)** trên nền tảng **Client-Server**, cụ thể hơn là kiến trúc **phân lớp (N-Tier Architecture)**, một cách tiếp cận phổ biến và mạnh mẽ cho các ứng dụng web. Kiến trúc này giúp tách biệt các mối quan tâm, tăng tính module hóa, dễ bảo trì và mở rộng.

1. **Các kiến trúc chính**

Hệ thống được chia thành bốn lớp logic chính, mỗi lớp có trách nghiệm riêng biệt:

1. **Lớp trình bày (Presentation Layer/Client-side):**

* **Mục đích:** Xử lý giao diện người dùng (UI) và tương tác trực tiếp với người dùng cuối. Nó chịu trách nghiệm hiển thị thông tin và gửi yêu cầu từ người dùng đến lớp nghiệp vụ.
* **Công nghệ:**
* **HTML, CSS (TailwindCSS, Bootstrap):** Cấu trúc và tạo điều kiện cho giao diện.
* **JavaScript:** Xử lý các tương tác phía client.
* **Laravel Blade Templates:** Templates engine để render HTML động từ server.
* **Laravel Livewire:** Livewire cho phép viết các thành phần giao diện động bằng PHP, hoạt động như một cầu nối giữa frontend và backend mà không cần viết nhiều JavaScript tuỳ chỉnh. Các Livewire components (AttendanceFilter, AttendanceTable, các form CRUD) sẽ thuộc lớp này
* **Chức năng chính:**
* Hiển thị trang đăng nhập, dashboard, các trang quản lý (học sinh, lớp học).
* Cung cấp các form để nhập dữ liệu (điểm danh, thêm học sinh/lớp).
* Hiển thị bảng dữ liệu.
* Xử lý các sự kiện từ người dùng (click, submit, nhập liệu) và gửi yêu cầu (qua livewire) về server.

1. **Lớp Logic nghiệp vụ (Business Logic Layer/Application Layer):**

* **Mục đích:** Chứa tất cả các quy tắc nghiệp vụ, logic xử lý chính của ứng dụng. Đây là nơi các yêu cầu từ lớp trình bày được xử lý, xác thực và chuyển đổi trước khi tương tác với dữ liệu.
* **Công nghệ:**
* **Laravel Controller:** Nhập yêu cầu từ client (qua Livewire hoặc HTTP request thông thường), điều phối luồng xử lý.
* **Laravel Services/Business Logic Classes:** Nơi chứa các logic phức tạp, quy tắc nghiệp vụ (ví dụ: logic điểm danh, tính toán thống kê, xử lý phân quyền chi tiết).
* **Laravel Policies/Gates:** Quản lý quyền hàn truy cập của người dùng dựa trên vai trò (Admin, Teacher).
* **Chức năng chính:**
* Xác thực người dùng (đăng nhập, đăng ký).
* Kiểm tra tính hợp lệ của dữ liệu đầu vào (validation).
* Áp dụng các quy tắc kinh doanh (Chỉ Admin mới được xoá lớp, chỉ Teacher mới được điểm danh lớp mình).
* Thưc hiện các thao tác CRUD lên dữ liệu thông qua lớp truy cập dữ liệu.
* Tính toán các chỉ số thống kê (tỷ lệ chuyên cần).

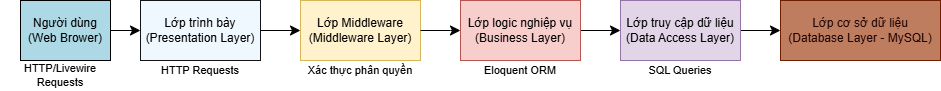
1. **Lớp truy cập dữ liệu (Data Access Layer/Persistence Layer):**

* **Mục đích:** Trừu tượng hoá việc truy cập và thao tác với cơ sở dữ liệu. Lớp này cung cấp một API để lớp nghiệp vụ có thể tương tác với dữ liệu và không cần biết chi tiết về cơ chế lưu trữ.
* **Công nghệ:**
* **Laravel Eloquent ORM (Object-Relational Mapping):** Cung cấp một giao diện dễ sử dụng để tương tác với cơ sở dữ liệu như các đối tượng PHP. Các Model của Eloquent (ví dụ: User, Student, Grade, Attendace) sẽ đại diện cho các bảng trong CSDL và xử lý các thao tác CRUD.
* **Laravel Migrations:** Quản lý cấu trúc cơ sử dữ liệu và các thay đổi lược đồ.
* **Chức năng chính:**
* Kết nối đến cơ sở dữ liệu.
* Thực hiện các truy vấn đọc, ghi, cập nhật, xoá dữ liệu.
* Đảm bảo tính toàn vẹn của dữ liệu ở mức cơ bản (ví dụ: khoá ngoại).

1. **Lớp cơ sở dữ liệu (Database Layer):**

* **Mục đích:** Lưu trữ tất cả dữ liệu của hệ thống một cách bền vững và có tổ chức.
* **Công nghệ:**
* **MySQL:** Hệ quản trị cơ sở dữ liệu quan hệ được sử dụng để lưu trữ của hệ thống.
* **Chức năng chính:**
* Lưu trữ thông tin người dùng, học sinh, lớp học sinh, bàn ghi điểm danh.
* Đảm bảo tính nhất quán và toàn vẹn của dữ liệu.

1. **Sơ đồ kiến trúc hệ thống (Conceptual Diagram)**



* **Giải thích luồng hoạt động:**

1. **Người dùng (Web Browser):**

* **Hành động**: Người dùng (Admin hoặc giáo viên) tương tác với giao diện web thông qua trình duyệt của họ
* Ví dụ:

+ Một giáo viên muốn điểm danh: họ truy cập màn hình điểm danh.

+ Mộ admin muốn them học sinh mới: họ nhấn vào nút “Thêm học sinh” trên màn hình quản lý học sinh.

+ Người dùng muốn đăng nhập: họ nhập email và mật khẩu vào form đăng nhập.

* **Gửi yêu cầu:** Trình duyệt sẽ gửi một yêu cầu HTTP (có thể là yêu cầu thông thường hoặc Livewire requests/callback) đến server. Yêu cầu này chứa thông tin về hành động mà người dùng muốn thực hiện (ví dụ: gửi form, nhấp nút).

1. **Lớp trình bày (Presentation Layer – Frontend):**

* **Mục đích:** Đây là giao diện hiển thị cho người dùng, được xây dựng bằng Laravel Blade Templates, HTML, CSS (TailwindCSS, Bootstrap) và Laravel Livewire.
* **Tiếp nhận yêu cầu:** Lớp này nhận các yêu cầu HTTP từ trình duyệt.
* **Xử lý phía Client:**

+ Livewire: Các Livewire components (như component điểm danh, form CRUD học sinh/lớp) sẽ xử lý các tương tác cục bộ mà không cần tải lại trang. Khi có dữ liệu cần gửi về server (ví dụ: sau khi chọn trạng thái điểm danh, nhập thông tin học sinh), Livewire sẽ đóng gói dữ liệu và tạo ra một yêu cầu HTTP gửi đến server.

+ Validation cơ bản: Một số validation ban đầu có thể được thực hiện ở phía client (ví dụ: kiểm tra định dạng email) trước khi gửi request.

* **Chuyển tiếp yêu cầu:** Yêu cầu (request) từ lớp trình bày sẽ được gửi đến lớp Middleware.

1. **Lớp Middleware (Middleware Layer):**

* **Mục đích:** Đóng vai trò như một bộ lọc cho các yêu cầu HTTP. Mỗi request đi vào server sẽ phải đi qua một hoặc nhiều Middleware đã được cấu hình.
* **Xử lý chính:**

**+** **Xác thực (Authentication):** Middleware như web hoặc auth của Laravel sẽ kiểm tra xem người dùng đã được đăng nhập hay chưa. Nếu người dùng chưa đăng nhập nhưng cố gắng truy cập một trang yêu cầu xác thực (ví dụ: dashboard), Middleware sẽ chặn yêu cầu và chuyển hướng họ đến trang đăng nhập.

* Ví dụ: Khi giáo viên cố gắng truy cập /attendance mà chưa đăng nhập, Middleware sẽ redirect về /login.

**+ Phân quyền (Authorization):** Middleware có thể kiểm tra vai trò của người dùng (Admin hay Teacher) và các quyền cụ thể của họ để đảm bảo rằng họ có đủ quyền thực hiện hành động được yêu cầu.

* Ví dụ: Nếu một giáo viên cố gắng truy cập trang quản lý người dùng (chỉ dành cho Admin), Middeware sẽ từ chối truy cập và có thể trả về lỗi 403 (Forbidden). Laravel Policies/Gates thường được kiểm tra trong controller, nhưng Middleware cũng có thể được sử dụng để kiểm tra quyền hạn cấp cao hơn cho toàn bộ nhóm route.

**+ Bảo mật:** Middleware Laravel cung cấp các lớp bảo vệ chống lại các cuộc tấn công phổ biến như CSRF.

* **Chuyển tiếp yêu cầu:** Nếu yêu cầu vượt qua tất cả các kiểm tra của Middleware, nó sẽ được chuyển tiếp đến lớp Logic nghiệp vụ.

1. **Lớp logic nghiệp vụ (Business Logic Layer – Laravel Backend):**

* **Mục đích:** Đây là “bộ não” của ứng dụng, nơi chứa các Controllers và Services, xử lý các quy tắc nghiệp vụ cốt lõi.
* **Tiếp nhận yêu cầu:** Một Laravel Controller phù hợp (ví dụ: AttendanceController, StudentController, AuthController) sẽ nhập yêu cầu đã được Middleware cho phép đi qua.
* **Xử lý chính:**

**+ Validation chi tiết:** Controller sẽ thực hiện validation chi tiết hơn về dữ liệu đầu vào (ví dụ: đảm bảo tất cả các trường hợp bắt buộc của học sinh đều được điền, định dạng ngày hợp lệ).

**+ Áp dụng quy tắc nghiệp vụ:** Controller sẽ áp dụng các quy tắc kinh doanh (ví dụ: khi thêm học sinh, kiểm tra xem mã học sinh đã tồn tại chưa).

**+ Điều phối**: Controller điều phối việc thực hiện hành động:

* Ví dụ điểm danh: AttendanceController sẽ nhận dữ liệu điểm danh (Lớp, ngày, trạng thái học sinh), kiểm tra tính hợp lệ, và sau đó yêu cầu lớp truy cập dữ liệu lưu thông tin điểm danh.
* Ví dụ thêm học sinh: StudentController sẽ nhận dữ liệu học sinh, xác thực, và yêu cầu lớp truy cập dữ liệu lưu học sinh mới.
* **Chuyển tiếp yêu cầu:** Controller sẽ gọi lớp truy cập dữ liệu để thực hiện các thao tác với CSDL.

1. **Lớp truy cập dữ liệu (Data Access Layer – Eloquent Models):**

* **Mục đích:** Cung cấp một lóp trừu tượng để tương tác với cơ sở dữ liệu, sử dụng Laravel Eloquent ORM.
* **Tiếp nhận yêu cầu:** Các Eloquent Models (ví dụ: App\Models\Students, App\Models\Attendance, App\Models\Grade) được gọi bởi lớp logic nghiệp vụ.
* **Xử lý chính:**

**+ Chuyển đổi:** Eloquent ORM chuyển đổi các yêu cầu PHP thành các truy vấn SQL tương xứng.

**+ Thao tác CSDL:** Thực hiện các thao tác CRUD (Create, Read, Update, Delete) trên cơ sở dữ liệu.

* Ví dụ: Khi thêm học sinh, Eloquent Model Student sẽ tạo một bản ghi mới trong bảng students.
* Ví dụ; Khi truy vấn danh sách điểm danh, Eloquent sẽ lấy dữ liệu từ bảng attendances.
* **Chuyển tiếp yêu cầu:** Eloquent ORM gửi các truy vấn SQL đến lớp cơ sở dữ liệu.

1. **Lớp cơ sở dữ liệu (Database Layer – MySQL):**

* **Mục đích:** Lưu trữ tất cả dữ liệu của hệ thống một cách bên vững.
* **Tiếp nhận yêu cầu:** MySQL Server nhận các truy vấn SQL từ lớp truy cập dữ liệu.
* **Xử lý chính:** Thực hiện các truy vấn SQL (chèn, cập nhật, xoá đọc) trên các bảng dữ liệu thực tế (ví dụ: users, students, grades, attendances).
* **Trả về kết quả:** MySQL trả về kết quả của truy vấn (dữ liệu hoặc thông báo thành công/thất bại) cho lớp truy cập dữ liệu.

1. **Phản hồi (Response Flow – ngược chiều):**

* Kết quả từ MySQL được chuyển ngược lại cho Eloquent ORM.
* Eloquent trả về các đối tượng PHP (Model) cho lớp logic nghiệp vụ (Controller).
* Controller xử lý kết quả này (ví dụ: tạo thông báo thành công, lấy dữ liệu để hiển thị) và chuẩn bị phản hồi.
* Phản hồi này được gửi qua lớp Middleware (nơi các Middleware “sau” request có thể xử lý, ví dụ: ghi log response).
* Cuối cùng, lớp trình bày nhận phản hồi từ server, và giao diện người dùng dược cập nhật để hiển thị kết quả (ví dụ: thông báo “Điểm danh thành công”, hiển thị danh sách học sinh đã cập nhật, hiển thị biểu đồ mới).

1. **Lợi ích của kiến trúc đối với dự án**
2. **Tính Module hoá cao (Modularity):**

* Mỗi lớp có một trách nghiệm cụ thể và độc lập. Ví dụ, lớp trình bày chỉ lo hiển thị, lớp logic nghiệp vụ chỉ lo xử lý nghiệp vụ, và lớp truy cập dữ liệu chi lo tương tác với CSDL.
* Điều này giúp dễ dàng hiểu, phát triển và thử nghiệm từng phần của hệ thống một cách riêng biệt.

1. **Dễ bảo trì (Maintainability):**

* Khi có sự thay đổi hoặc sửa lỗi ở một lớp, ảnh hưởng đến các lớp khác được giảm thiểu đáng kể. Ví dụ, nếu bạn thay đổi thiết kế giao diện, logic nghiệp vụ không bị ảnh hưởng. Nếu bạn thay đổi cơ sở dữ liệu (ví dụ: MySQL sang PostgreSQL), chỉ cần thay đổi lớp truy cập dữ liệu mà không ảnh hưởng đến logic nghiệp vụ hay giao diện.
* Code theo chuẩn Laravel và có cấu trúc rõ rang giúp việc bảo trì trở nên dẽ dàng hơn.

1. **Khả năng mở rộng (Scalability):**

* Bạn có thể dễ dàng mở rộng các chức năng mới bằng cách thêm các Controller, Model, Livewire Components hoặc Middleware mới mà không làm ảnh hưởng đến cấu trúc hiện có.
* Kiến trúc này cũng hỗ trợ việc mở rộng về mặt tài nguyên. Ví dụ, nếu lượng người dùng tăng, bạn có thể triển khai lớp trình bày hoặc lớp logic nghiệp vụ trên các server riêng biệt để cân bằng tải.

1. **Tăng cường bảo mật (Enhanced Security):**

* Lớp Middleware đóng vai trò như một “hang rào” đầu tiên, xử lý các tác vụ xác thực và phân quyền trước khi yêu cầu đến được logic nghiệp vụ nhạy cảm.
* Việc hash mật khẩu bằng bcrypt và quản lý phiên đăng nhập an toàn là các biện pháp bảo mật cốt lõi được thực hiện ở đây hoặc bởi Laravel Auth thông qua Middleware.
* Các biện pháp chống tấn công web phổ biến (CSRF, XSS, SQL Injection) được xử lý hiệu quả ở các lớp phù hợp (Middleware cho CSRF, Eloquent ORM cho SQL Ijection).

1. **Tối ưu hoá phát triển (Optimized Development):**

* Việc tận dụng các tính năng sẵn sàng của Laravel (như Eloquent ORM, Routing, Authentication, Validation, Middleware) giúp tăng tốc độ phát triển dự án.
* Sử dụng Livewire cho phép phát triển các giao diện động bằng PHP, giảm thiểu sự cần thiết của việc viết nhiều mã JavaScript tuỳ chỉnh.

1. **Quản lý phức tạp tốt hơn:**

* Khi dự án phát triển và có nhiều tính năng hơn (ví dụ: thêm module quản lý báo cáo, tích hợp thanh toán), kiến trúc phân lớp giúp duy trì sự rõ rang và có tổ chức, tránh tình trạng mã nguồn trở nên hỗn loạn.

1. **Tính tương thích và hiệu suất:**

* Việc xác định rõ các yêu cầu về trình duyệt và hiệu suất (tải trang nhanh, xử lý tác vụ dưới 3 giây hoặc 2 giây cho CRUD) trong yêu cầu phi chức năng được hỗ trợ bởi thiết kế kiến trúc này, giúp đảm bảo ứng dụng hoạt động mượt và ổn định.