System Project

Subject:

“Имплементација на софтверски системи со слободен и отворен код”

EDUADMIN

Student:

Shpresa Zendelji, 191038

Mentor:

Јоксимоски Бобан

Contents

Overview

The admin panel for private professors is a robust and comprehensive system that simplifies the management of teachers, students, courses, and payments in a private teaching organization. Leveraging cutting-edge technologies such as Laravel (PHP) with Filament for the backend, Vue.js for the frontend, and MySQL for the database, the system integrates seamlessly to centralize and automate administrative processes.

Key features include role-based access control, ensuring that administrators and teachers only access the resources they are authorized to manage. Teachers can manage their assigned courses and students, while administrators have full control over all aspects of the system. The panel supports automated email notifications for overdue or unpaid payments, which can be sent in bulk, saving time and effort. Additionally, advanced filtering options allow users to sort and search through data effortlessly.

Designed with responsiveness in mind, the admin panel works smoothly across desktops, tablets, and smartphones, making it accessible and easy to use on various devices. This user-friendly system significantly enhances the organization's operational efficiency, providing a centralized solution for daily management tasks.

Model Structure Overview

The model structure of the admin panel for private professors is organized to manage users (both teachers and admins), students, courses, and payments effectively. Below is a detailed breakdown of each model, its relationships, and some key attributes.

User Model

The `User` model handles both admin and teacher users, distinguished by the `role` attribute. Teachers are essentially users with the role of "teacher", whereas admins have more comprehensive privileges across the platform. The model manages user authentication, relationships with courses, and role-based access control.

Table Schema

```php
Schema::create('users', function (Blueprint $table) {
   $table->id();
   $table->string('name');
   $table->string('email')->unique();
```

```
    $table->timestamp('email_verified_at')->nullable();
    $table->string('password');
    $table->enum('role', ['admin', 'teacher'])->default('teacher');
    $table->string('embg')->unique(); // Unique identification for all users
    $table->date('birthday')->nullable();
    $table->string('phone_number')->nullable();
    $table->string('address')->nullable();
    $table->rememberToken();
    $table->timestamps();
});
```

Attributes

- `role`: Differentiates between admins and teachers.
- `name`, `email`, `embg`: Basic identity attributes.
- `birthday`, `phone_number`, `address`: Additional personal details.

Relationships

- `courses`: A `User` (when the role is teacher) can have multiple courses they are responsible for.

Key Functions

- isTeacher(): This function checks if a user is a teacher:

```php
public function isTeacher(): bool {
    return $this->role === 'teacher';
}
```

The role-based functionality ensures that teachers only have access to their own courses and related students. This is enforced through middleware, restricting teacher users from accessing administrative functionalities such as payments and the management of other users.

Student Model

The `Student` model is structured to manage all the data related to students. Each student is uniquely identified by an `EMBG` number (a unique identifier) and is associated with multiple courses and payments.

Table Schema

```php
Schema::create('students', function (Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->string('embg')->unique();
    $table->date('birthday');
    $table->string('email')->unique();
    $table->string('phone_number');
    $table->string('address');
    $table->timestamps();
});
```

Attributes:

- `embg`: Unique identifier for each student.
- `email`, `phone_number`, `address`, `birthday`: Key personal information.

Relationships:

- `courses`: A student can be enrolled in multiple courses, tracked through a pivot table (`course_student`) with an additional `points` field.
- `payments`: A student can have multiple associated payments, tied to different courses

Computed Attributes

- Enrolled Courses Count:

  This function returns the count of ongoing or upcoming courses the student is enrolled in.

  ```php
  public function getEnrolledCoursesCountAttribute(): int {
      return $this->courses()->where('status', 'ongoing')
                          ->orWhere('status', 'waiting_to_start')
                          ->count();
  }
  ```

- Completed Courses Count:

  This function tracks the number of courses a student has completed.

  ```php
  public function getCompletedCoursesCountAttribute(): int {
      return $this->courses()->where('status', 'done')->count();
  }
  ```

Course Model

The `Course` model defines all the courses available in the system. A course is tied to a teacher (user) and has multiple students. Courses are tracked with attributes such as start date, schedule, and status (whether ongoing, completed, or waiting to start).

Table Schema

```php
Schema::create('courses', function (Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->foreignId('user_id')->constrained('users')->onDelete('cascade'); // Teacher ID
    $table->date('start_date');
    $table->enum('status', ['ongoing', 'done', 'waiting_to_start']);
    $table->string('schedule'); // Schedule of the course (e.g., weekdays, times)
    $table->timestamps();
});
```

Attributes:

- `status`: Defines the current state of the course (ongoing, completed, or yet to start).
- `schedule`: Specifies the days and times when the course takes place.

Relationships:

- `user`: A course belongs to a teacher (user), defining the teacher who is responsible for conducting the course.

  - `students`: Many students can be enrolled in a course, with an additional pivot table (`course_student`) to track the students' progress and points.

  - `payments`: A course can have multiple associated payments (e.g., students paying for enrollment in that specific course).

Computed Attribute

- Student Count:

  ```php
  public function getStudentsCountAttribute(): int {
      return $this->students()->count();
  }
  ```

Payment Model

The `Payment` model manages the financial aspect of student enrollments. It tracks payments made by students for different courses, ensuring accurate records of paid, unpaid, and overdue amounts.

Table Schema

```php
Schema::create('payments', function (Blueprint $table) {
    $table->id();
    $table->foreignId('course_id')->constrained('courses')->onDelete('cascade');
    $table->foreignId('student_id')->constrained('students')->onDelete('cascade');
    $table->enum('currency_of_payments', ['USD', 'EUR', 'MKD']);
    $table->decimal('total_amount', 8, 2);
    $table->decimal('discount', 8, 2)->nullable();
    $table->decimal('paid_amount', 8, 2);
    $table->date('due_date');
    $table->enum('status', ['paid', 'unpaid']);
    $table->timestamps();
});
```

Attributes:

- `total_amount`: The full amount to be paid for a course.
- `discount`: An optional discount applied to the total amount.
- `paid_amount`: The amount the student has paid.
- `due_date`: The last date by which the payment is expected.
- `status`: Tracks whether the payment is `paid` or `unpaid`.

Relationships:

- `course`: A payment is associated with a specific course.
- `student`: A payment is linked to a specific student.

In summary, this model structure provides a robust system for managing key components of a private teaching organization. By centralizing teacher, student, course, and payment management, the system allows efficient tracking and management of essential data, ensuring smooth administrative operations.

Role-Based Access Control

The admin panel implements role-based access control, with two primary roles:

Admin: Full access to all resources and functionality, including teacher, student, course, and payment management.

Teacher: Restricted access, with teachers only able to view and manage the courses they are teaching and the students enrolled in those courses.

Middleware for Role-Based Access

A teacher middleware has been implemented to ensure that teacher users only have access to routes that pertain to their specific courses and students. This middleware checks the user's role and allows access only to the relevant sections of the system. Teachers are not able to modify other courses or access student data unrelated to their courses.

```php
public function handle($request, Closure $next) {
    if (auth()->user()->isTeacher()) {
        // Restrict access to non-course related routes
    }
    return $next($request);
}
```

Technology Stack

Frontend

The frontend of the admin panel is built using Vue.js, providing a dynamic and interactive interface. Vue.js ensures fast rendering and a smooth user experience, particularly when dealing with large amounts of data in tables and forms.

Backend

The backend is powered by Laravel, a PHP framework known for its simplicity, elegance, and performance. Filament, a Laravel-based admin panel framework, is used for handling resource management like teachers, students, courses, and payments. Filament offers powerful tools for creating CRUD (Create, Read, Update, Delete) operations, bulk actions, filters, and custom logic.

Database

The database used is MySQL, which stores all essential data, including user credentials, teacher and student details, course information, and payment records. Relationships between models are managed using Laravel's Eloquent ORM, ensuring data integrity and ease of querying.

Mailing System

A custom mailing system is integrated to send bulk emails for various events, such as overdue payments. The mailing functionality is built using Symfony's Mailer component, allowing admins to notify students of payment statuses automatically.

Key Technical Implementations

Teacher Middleware

A custom middleware ensures that teacher users have limited access based on their role. This middleware allows them to view and manage only the students and courses assigned to them, providing security and limiting data access to unauthorized users.

Filament Integration

Filament plays a pivotal role in the design and functionality of this admin panel, providing a highly flexible and extensible interface built on top of Laravel. Filament's tight integration with Laravel ensures that the system is both highly customizable and user-friendly, offering an intuitive dashboard for managing key aspects of the private teaching organization. The project extensively utilizes various Filament features, streamlining the development of CRUD operations, custom filters, bulk actions, and more. Below are some of the Filament functionalities integrated into the system:

Custom Filters

Filament’s filtering capabilities are used extensively throughout the admin panel to make data management more efficient. In the Payments section, custom filters are implemented to allow the admin or teacher to view specific subsets of data based on payment status. For instance, a filter for “Unpaid Overdue” payments allows users to quickly view overdue payments that have not been settled. Similarly, another filter can display "Unpaid but Not Due" payments, helping users differentiate between immediate issues and upcoming payments.

```
Tables\Filters\Filter::make('unpaid_overdue')

  ->label('Unpaid Overdue')

  ->query(function (Builder $query) {

      return $query->where('status', 'unpaid')

                  ->whereDate('due_date', '<', now());

  }),
```

These custom filters enhance data navigation, ensuring that users can easily pinpoint specific records and take action when needed, such as sending reminders for overdue payments.

Bulk Actions

Filament’s bulk action functionality is another critical feature in the system, especially in the Payments module. The admin can perform bulk actions on selected records, such as sending reminder emails to students with unpaid or overdue payments. This bulk email feature greatly reduces the time spent on individual tasks, automating communication and improving operational efficiency.

```
Tables\Actions\BulkAction::make('send_reminder_emails')

  ->action(function (Collection $records) {

      foreach ($records as $record) {

          // Send email logic here

      }

  }),
```

With this integration, admins can quickly notify multiple students about their payment status, ensuring that no overdue payments are missed. This functionality also extends to other sections of the system where bulk actions, such as deleting records or updating statuses, are required.

Custom Forms and Tables

The admin panel uses Filament's customizable forms and tables to handle CRUD operations across all major entities—teachers, students, courses, and payments. The simplicity and flexibility of Filament's form builder allow the system to include a wide range of fields, such as text, dates, select dropdowns, and file uploads.

For example, in the teacher creation or editing forms, fields such as name, email, birthday, EMBG, phone number, and address are provided. The form leverages validation and error handling to ensure that all required fields are filled out accurately before submission. This approach is consistent across the other entities, ensuring uniformity in data management.

```
Forms\Components\TextInput::make('name')
    ->required()
    ->label('Teacher Name')
    ->placeholder('Enter full name'),


Forms\Components\DatePicker::make('birthday')
    ->label('Birthday')
    ->required(),
```

In terms of tables, Filament's tables component is used to display records across the admin panel. For instance, the list of students or courses shows all relevant details in a tabular format, with options to filter, sort, and search records. The tables are also linked with the forms, allowing seamless navigation between viewing, editing, and deleting records.

Custom Dashboard and Widgets

One of the standout features of Filament is the ability to customize the admin dashboard, which is a focal point for admins and teachers alike. The default widgets, such as the "Welcome Widget," are removed to make room for custom widgets that display actionable insights, such as:

- Total number of courses
- Number of teachers currently active
- Payments overview, including unpaid or overdue payments
- Upcoming due dates for payments

These widgets provide users with quick, high-level information about the system, helping them manage their tasks more effectively. The dashboard customization ensures that only the most relevant data is displayed upfront.

Security and Authentication

The system incorporates comprehensive security measures to safeguard user data and ensure that each user only has access to the areas and features they are authorized to manage. At the heart of this system is role-based authentication and authorization. Admins are granted full access to all sections of the admin panel, allowing them to manage teachers, students, payments, and courses comprehensively. Teachers, on the other hand, have restricted access, limited to managing only their assigned courses and students. They cannot view or manage payment records or interact with students outside of their designated courses.

To further ensure data protection, the system securely encrypts sensitive information such as passwords. Laravel's default hashing mechanism, bcrypt, is used to store passwords in a hashed format, which is defined through the protected $casts array. This ensures that even in the event of a data breach, user credentials remain secure. The system also uses the protected $hidden array to keep sensitive attributes like password and remember_token hidden from model output, preventing exposure in API responses or other serialized data.

Additionally, the system implements token-based authentication, including remember_token, to enhance session security. Email verification is another layer of protection, ensuring that only verified users can access their accounts. With these measures in place, the system provides a secure environment where both admins and teachers can confidently manage their respective responsibilities.

Conclusion

The Admin Panel for Private Professors is a robust and efficient solution for managing teachers, students, courses, and payments. By centralizing all essential operations in a single platform, it enhances the overall management efficiency of a private teaching organization. The system's role-based access control, combined with powerful Filament features like filters, bulk actions, and custom forms, ensures that users can easily manage large datasets and take swift actions when necessary.

With features like email notifications, overdue payment tracking, and comprehensive filtering, the admin panel provides both flexibility and control to administrators and teachers alike.