MVC Architecture Explanation

What is MVC?

MVC stands for **Model-View-Controller**. It's a design pattern that separates an application into three main components:

1. **Model (M)** - Manages data and business logic
2. **View (V)** - Displays information to the user
3. **Controller (C)** - Handles user input and coordinates between Model and View

Think of it like a restaurant:

- **Model** = Kitchen (where food is prepared, data is processed)
- **View** = Menu & Plate (what customers see)
- **Controller** = Waiter (takes orders from customers, delivers to kitchen, brings food back)

How Laravel Implements MVC

Laravel is built around the MVC pattern, but adds extra layers for flexibility:

```
User Request → Routes → Controller → Model → Database
                            ↓           ↓
                           View    ←   Data
```

Flow Example: User Login

1. **User clicks "Login"** → Goes to URL `/login`
2. **Router** (`routes/web.php`) → Maps `/login` to `AuthController@show_login`
3. **Controller** (`AuthController.php`) → `show_login()` method runs
4. **View** (`login.blade.php`) → HTML form is displayed
5. **User submits form** → Goes to POST `/login`
6. **Controller** → `login_user()` validates and calls...
7. **Model** (`User.php`) → Checks database for email/password
8. **Controller** → Redirects to home page or shows error
9. **View** (`welcome.blade.php` or back to `login.blade.php`)

Models: The "M" in MVC

Purpose: Represent database tables and contain business logic.

Location: `app/Models/`

What they do:

- Define table structure
- Validate data
- Perform calculations
- Contain business rules (e.g., "password must be hashed")

In This Project:

- `User.php`: Represents the `user` table (customers)
  - Methods: `register()`, `login()`, `logout()`, `updateProfile()`, `fullName()`
- `Admin.php`: Represents the `admins` table (staff)
  - Properties: `$fillable`, `$hidden`, `$guard`

Example:

```
// Creating a new user (Model handles data)
$user = new User();
$user->register('John', 'Doe', 'john@example.com', 'password123');
// The model hashes the password and saves to database
```

Views: The "V" in MVC

Purpose: Display HTML to the user.

Location: `resources/views/`

What they do:

- Render HTML with dynamic data
- Use Blade templating (Laravel's template engine)
- Display forms, tables, buttons, etc.

In This Project:

- `auth/login.blade.php`: Login form
- `auth/register.blade.php`: Signup form
- `admin/login.blade.php`: Admin login form
- `admin/dashboard.blade.php`: Admin dashboard

- `layouts/app.blade.php`: Master layout (header, footer)

Example:

```
<!-- View displays data passed from controller -->
<h1>Welcome, {{ $user->name }}!</h1>
```

Controllers: The "C" in MVC

Purpose: Handle HTTP requests and coordinate between Models and Views.

Location: `app/Http/Controllers/`

What they do:

- Receive user input (form data, URL parameters)
- Validate input
- Call Model methods to fetch/save data
- Return Views or redirect to other pages

In This Project:

- `AuthController.php`: Handles user authentication
  - `show_login()` → Returns login view
  - `login_user()` → Processes login, calls User model
  - `show_register()` → Returns registration view
  - `register_user()` → Creates new user
  - `logout_user()` → Logs user out
- `AdminAuthController.php`: Same but for admins

Example:

```
// Controller method
public function login_user(Request $request) {
    // 1. Validate input
    $request->validate([...]);

    // 2. Call Model
    if (Auth::attempt($credentials)) {
        // 3. Return View (redirect)
        return redirect()->route('home');
    }

    // 3. Return View (with error)
    return back()->withErrors([...]);
}
```

Routing: Tying It All Together

Purpose: Map URLs to Controller methods.

Location: `routes/web.php`

How it works:

```
Route::get('/login', [AuthController::class, 'show_login'])->name('login');
Route::post('/login', [AuthController::class, 'login_user']);
```

This means:

- When user visits `/login` → Call `show_login()` → Display login form (View)
- When user submits login form → Call `login_user()` → Check credentials (Model) → Redirect (View)

Where Business Logic Belongs

Code Type	Goes In	Example
Database queries	Model	`User::where('email', $email)->first()`
Validation rules	Controller (or Form Request)	`$request->validate([...])`
Password hashing	Model	`Hash::make($password)`
Displaying data	View	`{{ $user->name }}`
Reusable logic	Service (`app/Services/`)	`AuthManager.php`

How This Project Uses MVC

User Registration Flow

1. **User visits `/register`**
   - **Router**: `routes/web.php` → `AuthController@show_register`
   - **Controller**: Returns a **View**
   - **View**: `auth/register.blade.php` displays the signup form
2. **User fills form and clicks "Register"**
   - **Router**: POST `/register` → `AuthController@register_user`
   - **Controller**: Validates input (email, password, etc.)
   - **Model**: `User.php`'s `register()` method hashes password and saves to database
   - **Controller**: Redirects to home page
   - **View**: `welcome.blade.php` is displayed

Admin vs User Separation

This project has **two parallel MVC stacks**:

Component	Users	Admins
Model	`User.php`	`Admin.php`
Controller	`AuthController.php`	`AdminAuthController.php`
Views	`auth/login.blade.php`	`admin/login.blade.php`
Routes	`/login`, `/register`	`/admin/login`, `/admin/register`
Guard	`web` (default)	`admin` (custom)

They are completely separate to prevent:

- Regular users accessing admin pages
- Admins accidentally using the customer login system

Files That Represent M, V, and C

Models (M)

- `app/Models/User.php`
- `app/Models/Admin.php`

Views (V)

- `resources/views/auth/login.blade.php`
- `resources/views/auth/register.blade.php`
- `resources/views/admin/login.blade.php`
- `resources/views/admin/dashboard.blade.php`
- `resources/views/welcome.blade.php`

Controllers (C)

- `app/Http/Controllers/AuthController.php`
- `app/Http/Controllers/AdminAuthController.php`

Routing (Glue)

- `routes/web.php`

Services (Business Logic Helper)

- `app/Services/AuthManager.php` - Shared authentication logic used by both User and Admin controllers

Benefits of MVC

1. **Separation of Concerns**: HTML code (View) doesn't mix with business logic (Model)
2. **Reusability**: Models can be used by multiple controllers
3. **Testability**: Each component can be tested independently
4. **Maintainability**: Easy to find and fix bugs
5. **Team Collaboration**: Frontend developers work on Views, backend developers work on Models/Controllers

Demonstrating MVC to Your Instructor

Simple Explanation: "Imagine I want to log in. I type my email and password on the login page (View). When I click submit, the AuthController receives my input, asks the User model to check if my credentials are correct in the database, and then either shows me the home page or an error message (another View). The Model handles data, the View handles display, and the Controller coordinates everything."

Show in Code:

1. Open `routes/web.php` → Point to `/login` route
2. Open `AuthController.php` → Show `login_user()` method
3. Open `User.php` → Show how it represents the user table
4. Open `auth/login.blade.php` → Show the HTML form
5. Explain: "Route → Controller → Model → View. That's MVC."