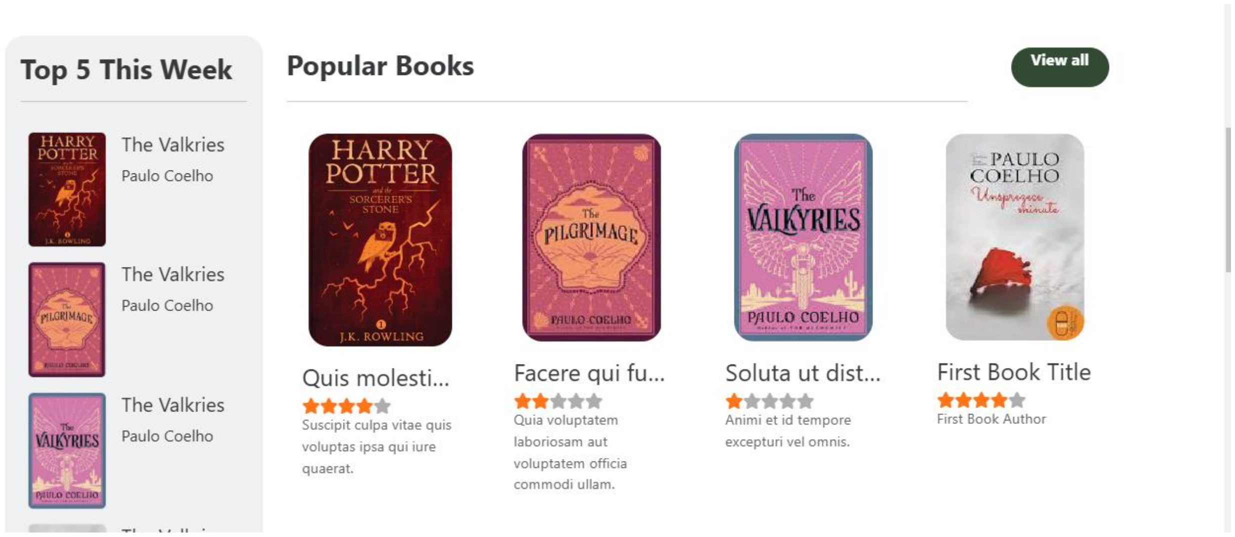
# IT Elec: WEB DEVELOPMENT
Populating from a Database

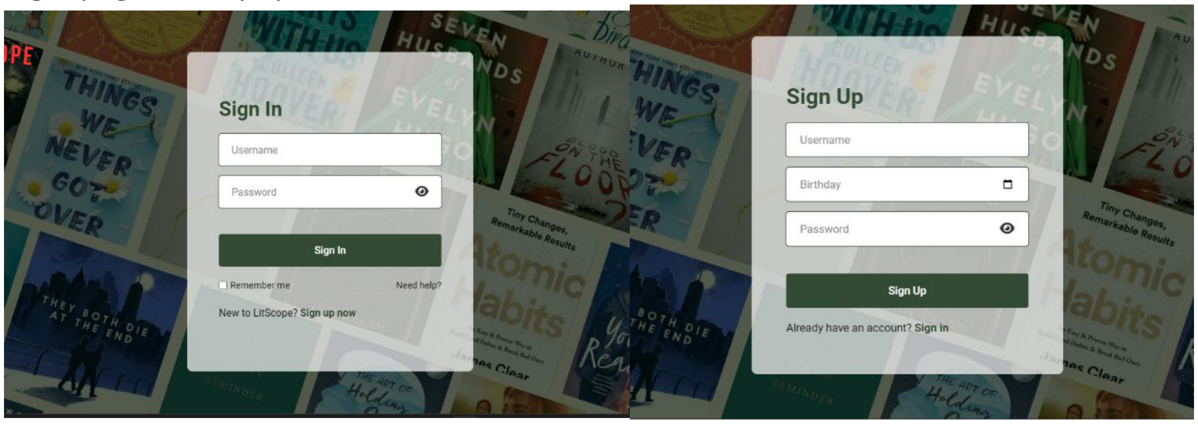
Manuel Andrei Lleva
BSIT-3C

## RENDERED WEB PAGES:

LandingPage.blade.php

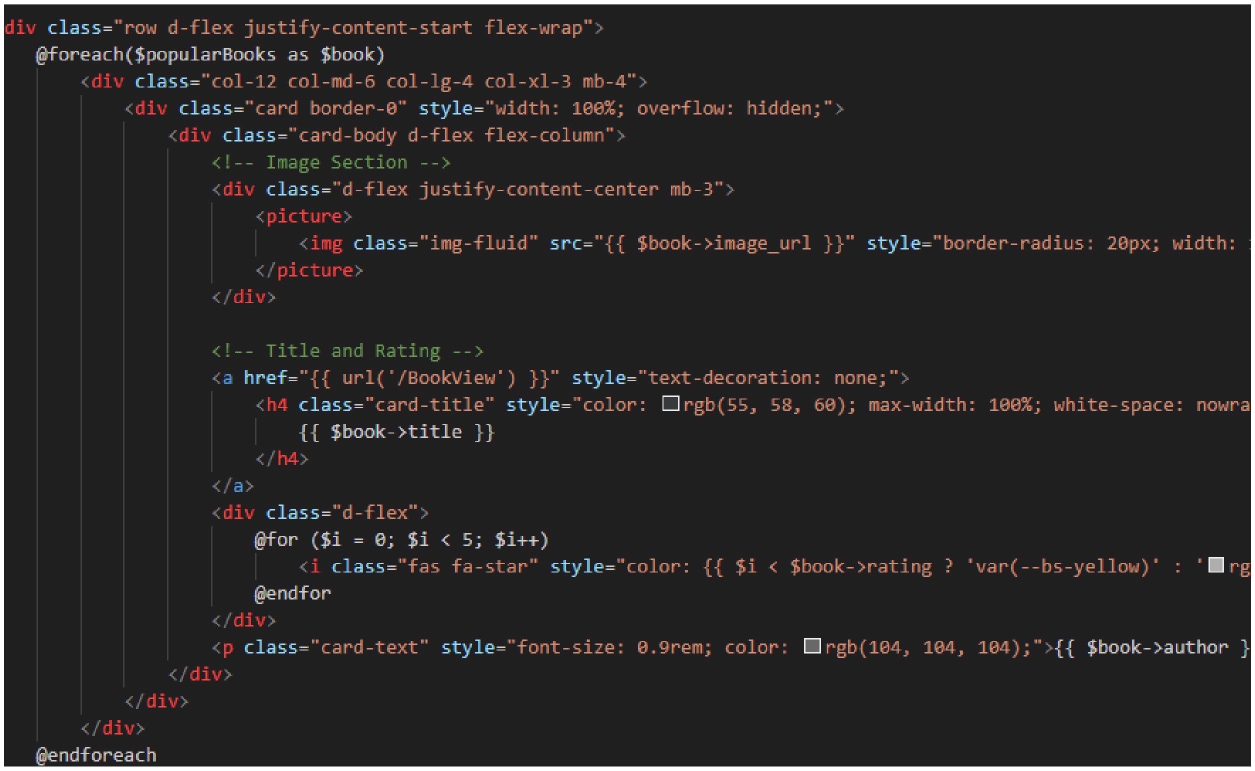


login-page.blade.php



## BLADE FILES:

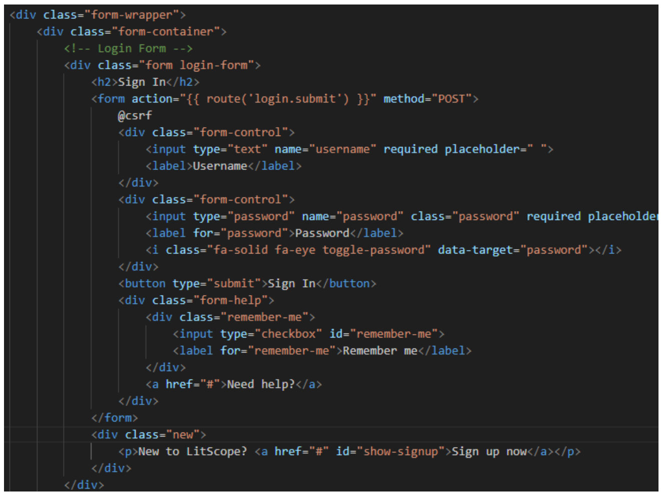
### LandingPage.blade.php



```
div class="row d-flex justify-content-start flex-wrap">
   @foreach($popularBooks as $book)
       <div class="col-12 col-md-6 col-lg-4 col-xl-3 mb-4">
           <div class="card border-0" style="width: 100%; overflow: hidden;">
               <div class="card-body d-flex flex-column">
                   <!-- Image Section -->
                   <div class="d-flex justify-content-center mb-3">
                       <picture>
                           <img class="img-fluid" src="{{ $book->image_url }}" style="border-radius: 20px; width:
                       </picture>
                   </div>

                   <!-- Title and Rating -->
                   <a href="{{ url('/BookView') }}" style="text-decoration: none;">
                       <h4 class="card-title" style="color: rgb(55, 58, 60); max-width: 100%; white-space: nowra
                           {{ $book->title }}
                       </h4>
                   </a>
                   <div class="d-flex">
                       @for ($i = 0; $i < 5; $i++)
                           <i class="fas fa-star" style="color: {{ $i < $book->rating ? 'var(--bs-yellow)' : 'rg
                       @endfor
                   </div>
                   <p class="card-text" style="font-size: 0.9rem; color: rgb(104, 104, 104);">{{ $book->author }
               </div>
           </div>
       </div>
   @endforeach
```

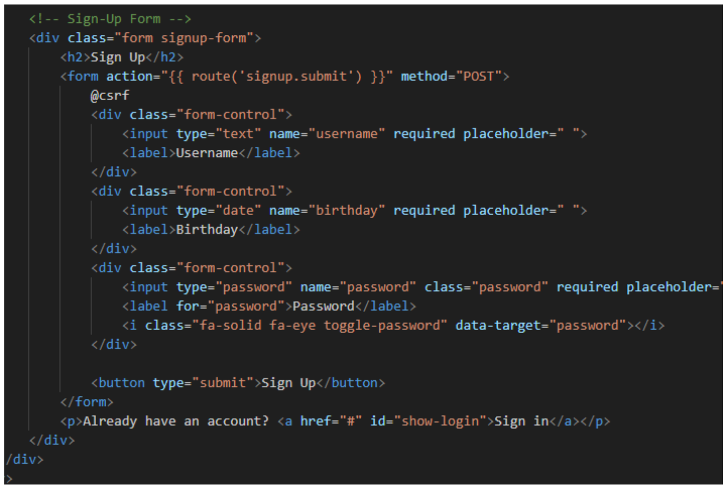
The code generates a list of popular books, displaying each book's image, title, rating, and author using a foreach loop in a Bootstrap card layout.

### login-page.blade.php



```
<div class="form-wrapper">
    <div class="form-container">
        <!-- Login Form -->
        <div class="form login-form">
            <h2>Sign In</h2>
            <form action="{{ route('login.submit') }}" method="POST">
                @csrf
                <div class="form-control">
                    <input type="text" name="username" required placeholder=" ">
                    <label>Username</label>
                </div>
                <div class="form-control">
                    <input type="password" name="password" class="password" required placeholde
                    <label for="password">Password</label>
                    <i class="fa-solid fa-eye toggle-password" data-target="password"></i>
                </div>
                <button type="submit">Sign In</button>
                <div class="form-help">
                    <div class="remember-me">
                        <input type="checkbox" id="remember-me">
                        <label for="remember-me">Remember me</label>
                    </div>
                    <a href="#">Need help?</a>
                </div>
            </form>
            <div class="new">
                <p>New to LitScope? <a href="#" id="show-signup">Sign up now</a></p>
            </div>
        </div>
```

This creates a secure and user-friendly login form. It includes input fields for the username and password, a checkbox for the "Remember me" option, and a submit button. The form's header clearly labels it as a "Sign In" form, and it incorporates a CSRF token for security purposes. The password input field includes a toggle to show or hide the password, enhancing usability. Additionally, there are links for users to get help or to sign up if they don't have an account, making the form comprehensive and easy to navigate.

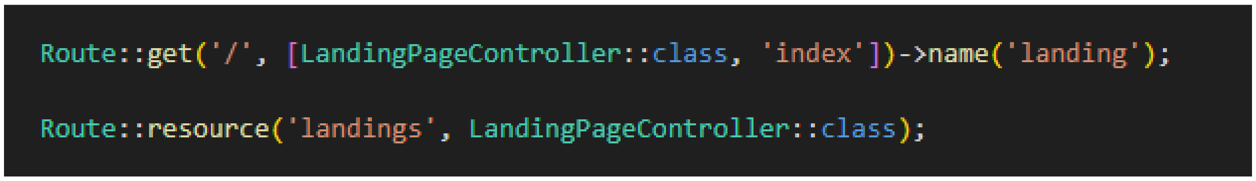


```
<!-- Sign-Up Form -->
<div class="form signup-form">
    <h2>Sign Up</h2>
    <form action="{{ route('signup.submit') }}" method="POST">
        @csrf
        <div class="form-control">
            <input type="text" name="username" required placeholder=" ">
            <label>Username</label>
        </div>
        <div class="form-control">
            <input type="date" name="birthday" required placeholder=" ">
            <label>Birthday</label>
        </div>
        <div class="form-control">
            <input type="password" name="password" class="password" required placeholder="
            <label for="password">Password</label>
            <i class="fa-solid fa-eye toggle-password" data-target="password"></i>
        </div>

        <button type="submit">Sign Up</button>
    </form>
    <p>Already have an account? <a href="#" id="show-login">Sign in</a></p>
</div>
/div>
>
```

This generates a **sign-up form**. It has fields for a username, birthday, and password. The form uses the POST method for submitting data and includes CSRF protection for security. It also features a button to toggle the visibility of the password. There's a link at the bottom for users who already have an account to sign in. In essence, it's a well-structured and secure sign-up interface.

## ROUTES:

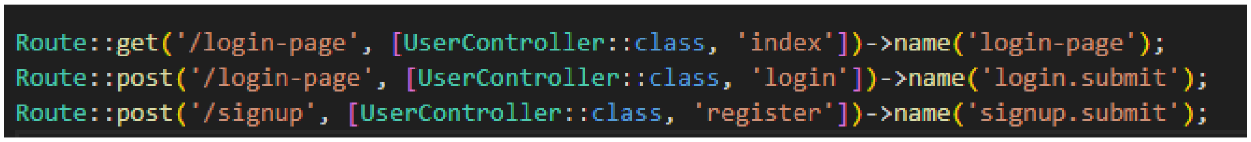
### LandingPage



```
Route::get('/', [LandingPageController::class, 'index'])->name('landing');

Route::resource('landings', LandingPageController::class);
```

This defines two routes in a PHP web application using the Laravel framework. The first route is a GET request to the root URL ('/') that calls the index method of the LandingPageController and names this route landing. The second route is a resource route for 'landings' that maps to the LandingPageController, providing a set of RESTful routes for managing the resource. This structure sets up the foundation for handling requests related to the landing page in the application.

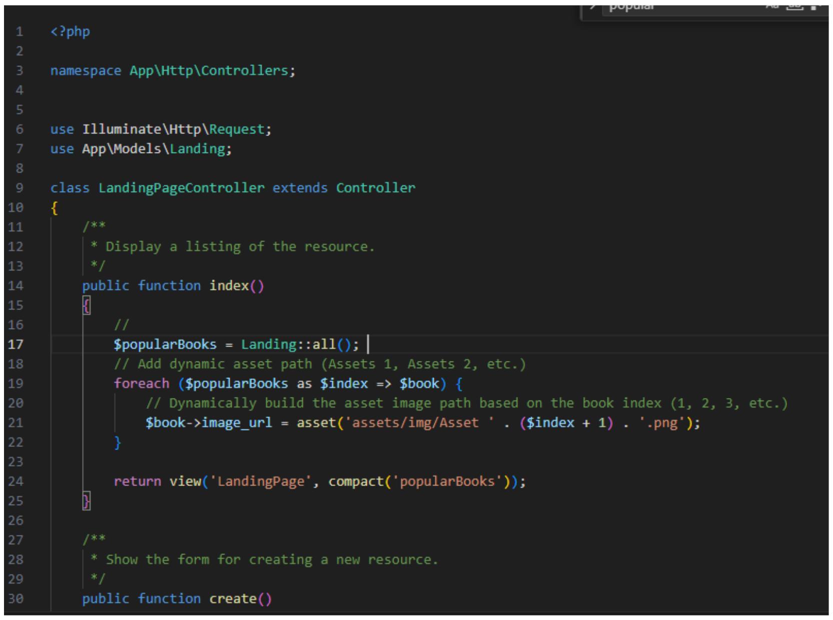
### login-page



```
Route::get('/login-page', [UserController::class, 'index'])->name('login-page');
Route::post('/login-page', [UserController::class, 'login'])->name('login.submit');
Route::post('/signup', [UserController::class, 'register'])->name('signup.submit');
```

This sets up routes in a Laravel web application to handle user login and signup functionalities. It defines a route for displaying the login page, another for submitting the login form, and a third for submitting the signup form, all mapped to methods in the UserController class.

## CONTROLLERS:

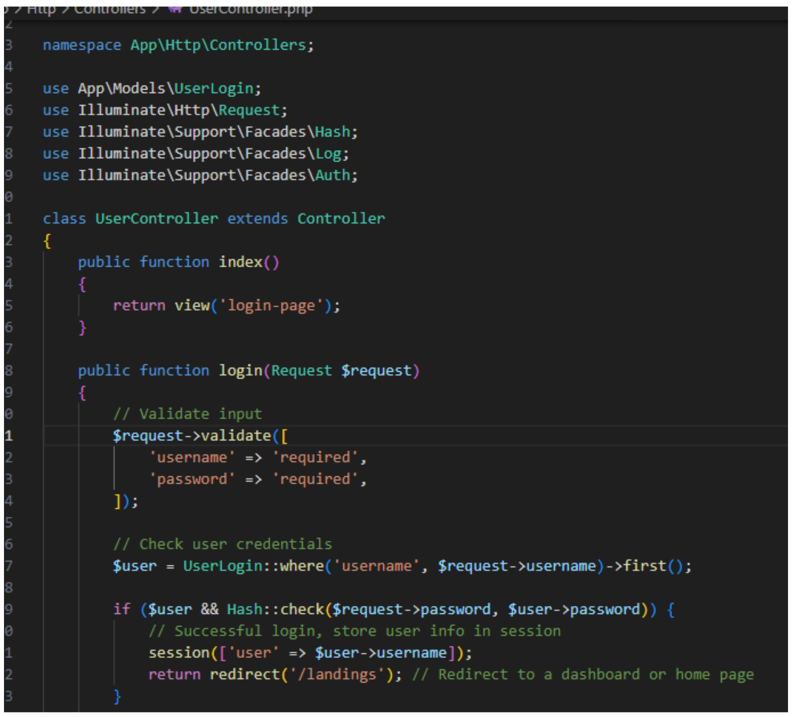


```php
<?php

namespace App\Http\Controllers;


use Illuminate\Http\Request;
use App\Models\Landing;

class LandingPageController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        //
        $popularBooks = Landing::all();
        // Add dynamic asset path (Assets 1, Assets 2, etc.)
        foreach ($popularBooks as $index => $book) {
            // Dynamically build the asset image path based on the book index (1, 2, 3, etc.)
            $book->image_url = asset('assets/img/Asset ' . ($index + 1) . '.png');
        }

        return view('LandingPage', compact('popularBooks'));
    }

    /**
     * Show the form for creating a new resource.
     */
    public function create()
```

LandingPageController.php

This defines user login and signup functionalities. It includes a route to display the login page, another for processing the login form submission, and a third for handling the signup form submission, all mapped to specific methods in the UserController class. This setup enables user authentication and registration within the application.

UserController.php

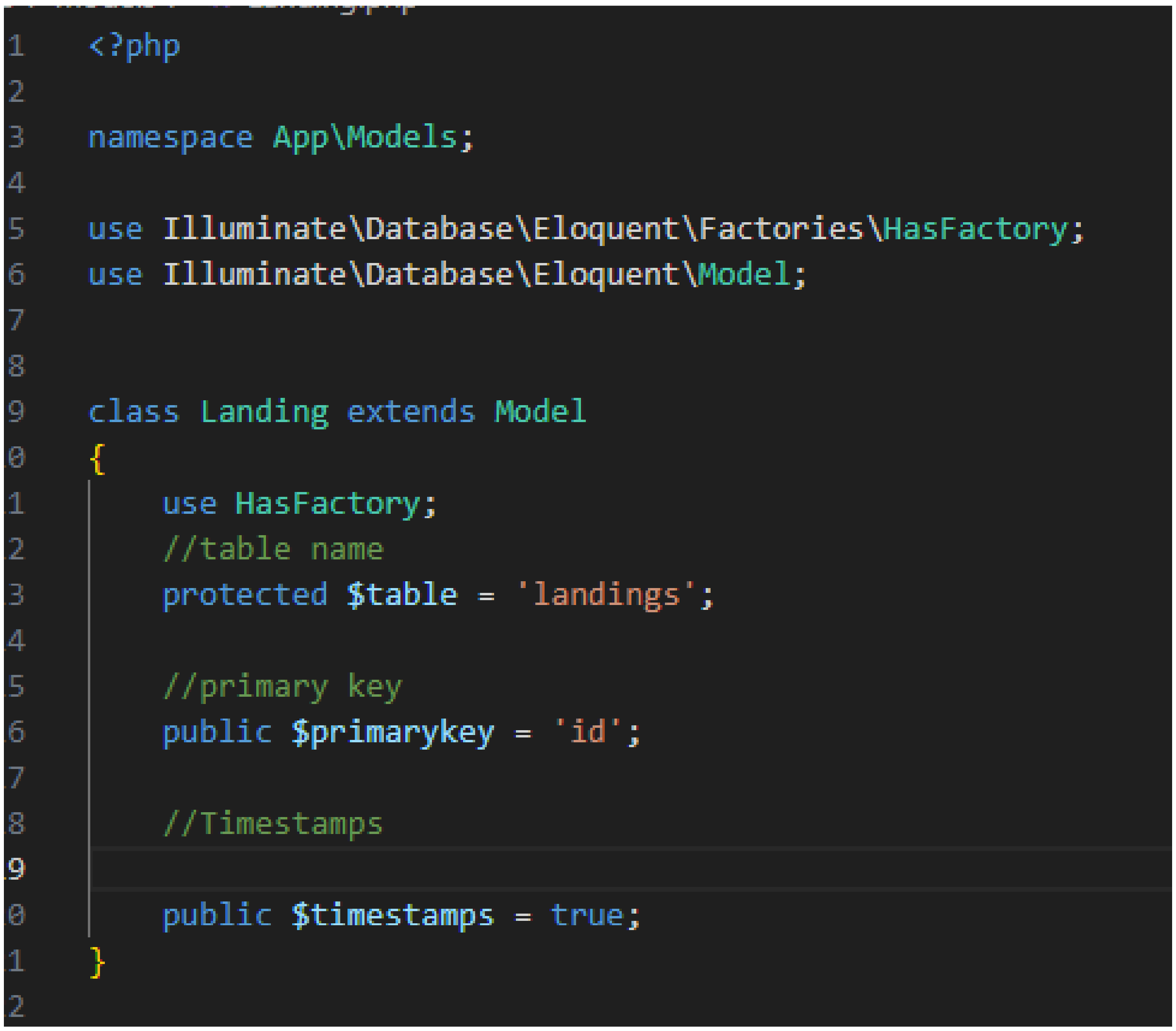


```php
namespace App\Http\Controllers;

use App\Models\UserLogin;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Log;
use Illuminate\Support\Facades\Auth;

class UserController extends Controller
{
    public function index()
    {
        return view('login-page');
    }

    public function login(Request $request)
    {
        // Validate input
        $request->validate([
            'username' => 'required',
            'password' => 'required',
        ]);

        // Check user credentials
        $user = UserLogin::where('username', $request->username)->first();

        if ($user && Hash::check($request->password, $user->password)) {
            // Successful login, store user info in session
            session(['user' => $user->username]);
            return redirect('/landings'); // Redirect to a dashboard or home page
        }
```

This defines user authentication. It includes routes for displaying the login page, processing the login form submission, and handling the signup form submission, all mapped to methods in the UserController class. This setup facilitates user login and registration within the application.

MODELS:

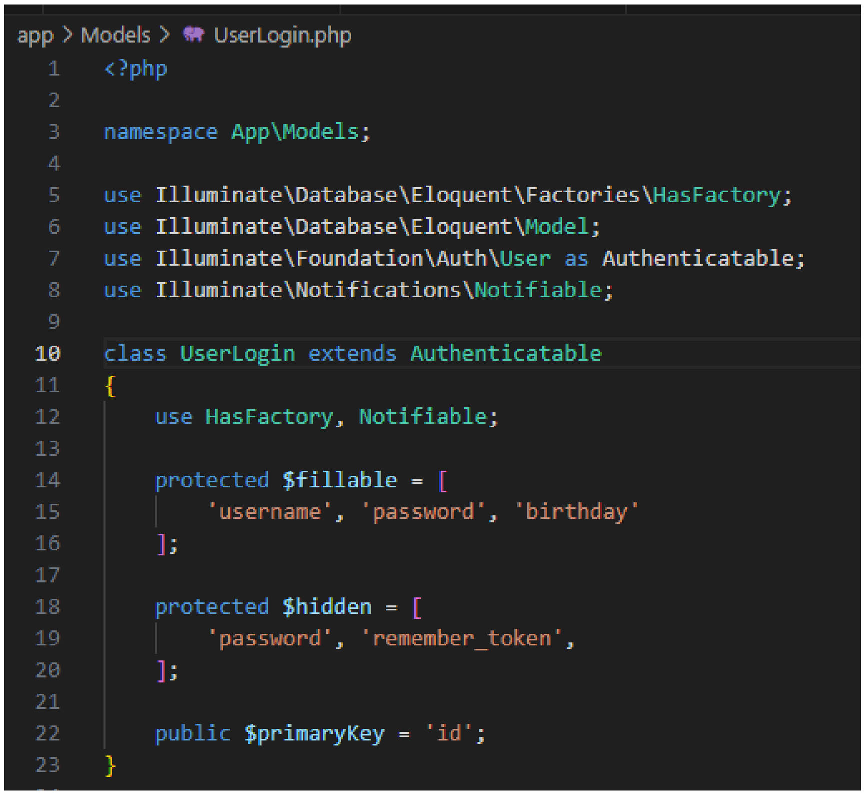
Landing.php



```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;


class Landing extends Model
{
    use HasFactory;
    //table name
    protected $table = 'landings';

    //primary key
    public $primarykey = 'id';

    //Timestamps

    public $timestamps = true;
}

```

This code defines a Laravel Eloquent model named Landing. It sets up the model to interact with the landings table in the database, specifies that the primary key is the id column, and enables automatic management of created_at and updated_at timestamps. The Landing model uses the HasFactory trait, which provides methods for creating instances of the model. This setup allows the Landing model to effectively interact with the corresponding database table and manage records.

UserLogin.php



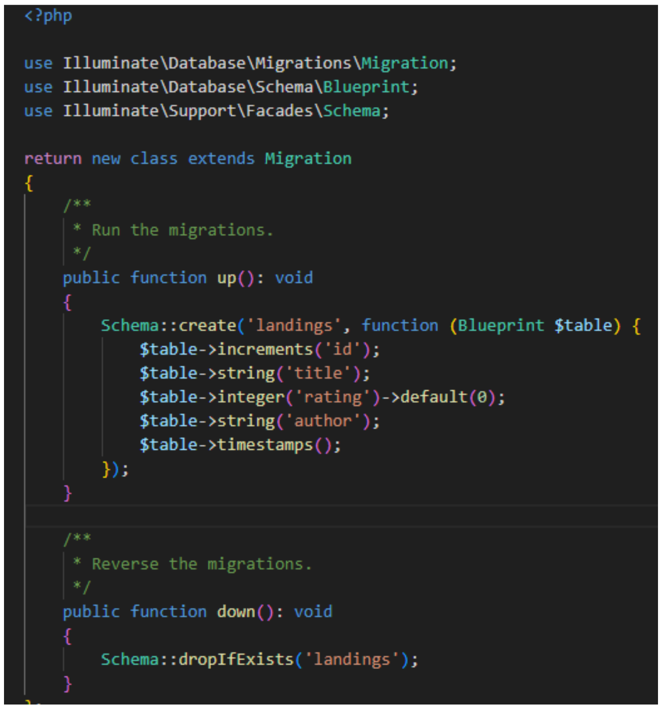
app > Models > UserLogin.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

class UserLogin extends Authenticatable
{
    use HasFactory, Notifiable;

    protected $fillable = [
        'username', 'password', 'birthday'
    ];

    protected $hidden = [
        'password', 'remember_token',
    ];

    public $primaryKey = 'id';
}
```

This defines a UserLogin model for managing user authentication within a Laravel application. It imports necessary traits and classes from the Laravel framework and specifies which fields can be mass-assigned (username, password, birthday) and which fields should be hidden (password, remember_token). The model also sets the primary key to id and uses the HasFactory trait for factory support, and the Notifiable trait for sending notifications. This setup enables secure handling of user authentication data.

## MIGRATIONS:

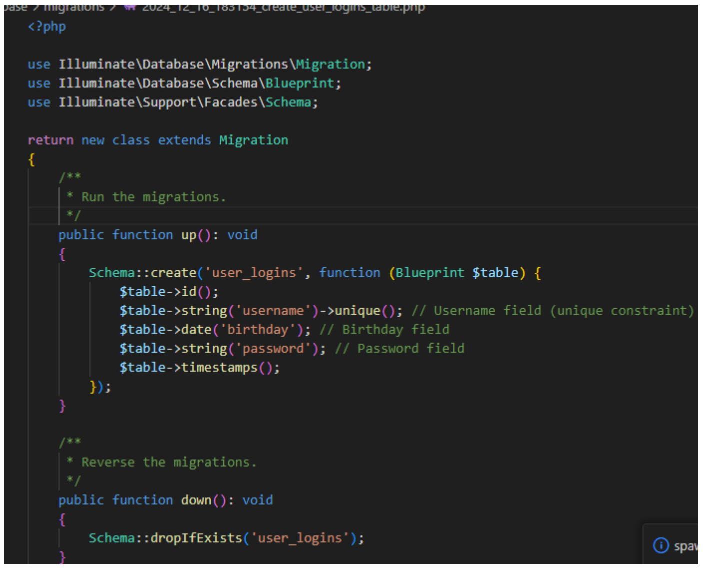
### landings



```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('landings', function (Blueprint $table) {
            $table->increments('id');
            $table->string('title');
            $table->integer('rating')->default(0);
            $table->string('author');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('landings');
    }
```

This defines migration script that creates a landings table with columns for id, title, rating, author, and timestamps. The up method creates the table, while the down method drops it, ensuring the database schema can be rolled back if needed.

### user_logins

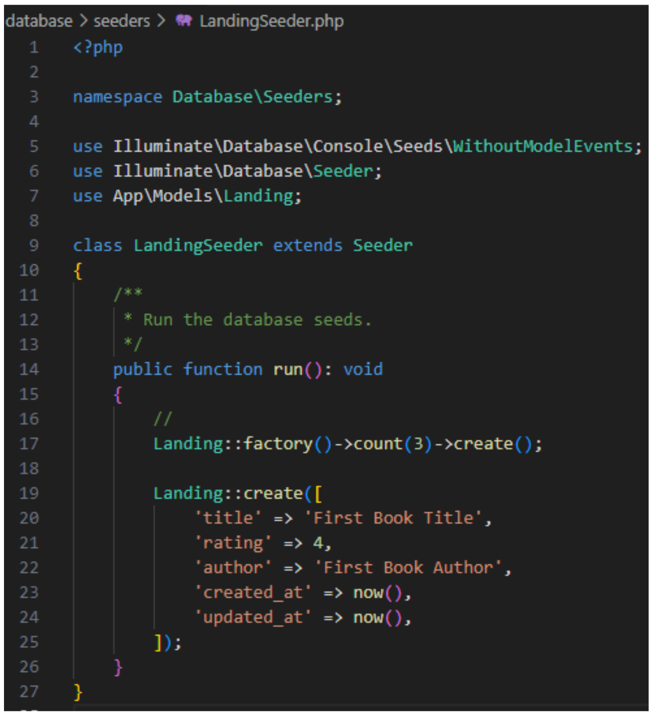


```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('user_logins', function (Blueprint $table) {
            $table->id();
            $table->string('username')->unique(); // Username field (unique constraint)
            $table->date('birthday'); // Birthday field
            $table->string('password'); // Password field
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('user_logins');
    }
```

This defines a migration script for creating a user_logins table in a Laravel application. The up method sets up the table structure with an auto-incrementing primary key (id), unique string for username, date for birthday, string for password, and timestamps (created_at and updated_at). The down method allows for reversing the migration by dropping the user_logins table if it exists. This setup ensures the database schema can be modified and rolled back during the development process.\

## SEEDING:

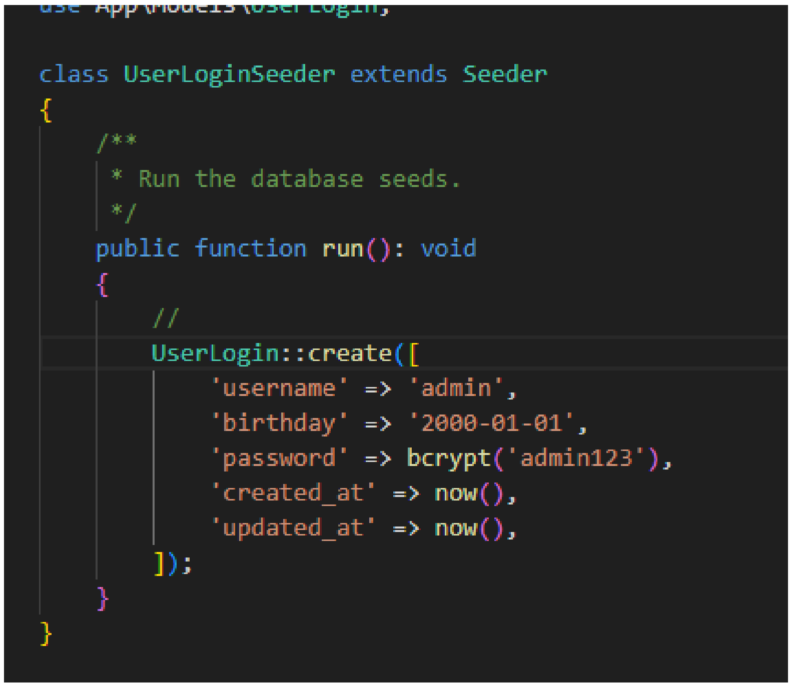
LandingSeeder.php



```
database > seeders > LandingSeeder.php
<?php

namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use App\Models\Landing;

class LandingSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        //
        Landing::factory()->count(3)->create();

        Landing::create([
            'title' => 'First Book Title',
            'rating' => 4,
            'author' => 'First Book Author',
            'created_at' => now(),
            'updated_at' => now(),
        ]);
    }
}
```

This Laravel seeder script used to populate the landings table with initial data. It defines a LandingSeeder class that extends Laravel's Seeder class. Within the run method, it creates a new record in the landings table with specific attributes: a title ("First Book Title"), a rating (4), and an author ("First Book Author"). The created_at and updated_at timestamps are set to the current time. This seeder helps in setting up the database with sample data for testing and development purposes.

UserLoginSeeder.php

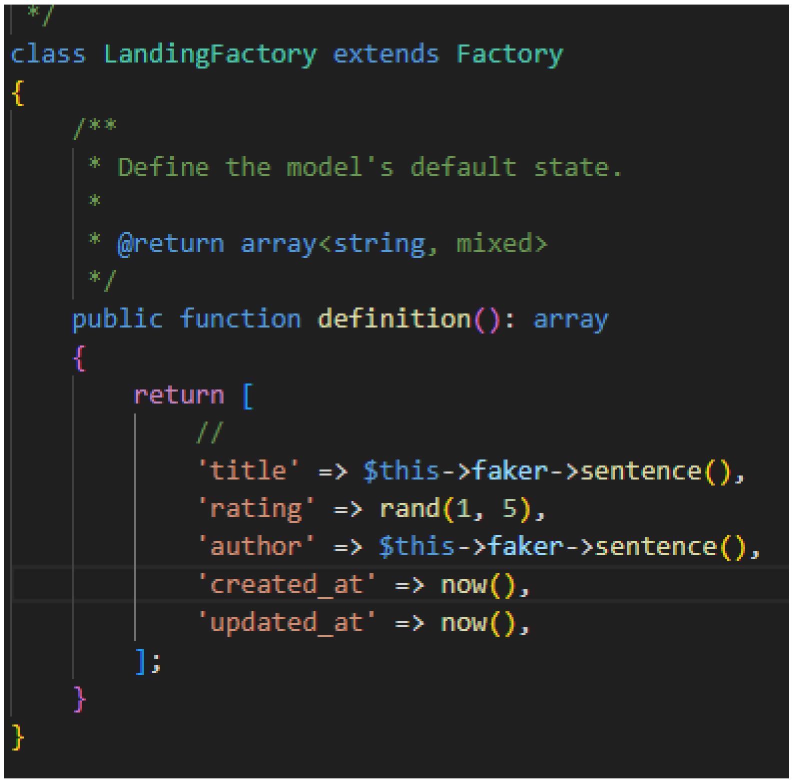


```
use App\Models\UserLogin;

class UserLoginSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        //
        UserLogin::create([
            'username' => 'admin',
            'birthday' => '2000-01-01',
            'password' => bcrypt('admin123'),
            'created_at' => now(),
            'updated_at' => now(),
        ]);
    }
}
```

This defines a Laravel seeder class named UserLoginSeeder used to populate the UserLogin table with initial data. The run method creates a new user record with the username 'admin', birthday '2000-01-01', and a hashed password 'admin123'. The created_at and updated_at fields are set to the current timestamp. This seeder helps set up a default admin user for testing or initial setup purposes.

## FACTORY:

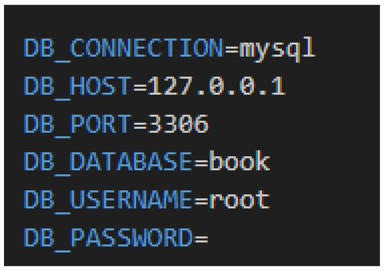
LandingFactory.php



```php
 */
class LandingFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array<string, mixed>
     */
    public function definition(): array
    {
        return [
            //
            'title' => $this->faker->sentence(),
            'rating' => rand(1, 5),
            'author' => $this->faker->sentence(),
            'created_at' => now(),
            'updated_at' => now(),
        ];
    }
}
```

This defines a factory class named LandingFactory for generating instances of the Landing model with default values in a Laravel application. The definition method in the class uses the Faker library to generate a random sentence for the title and author attributes, a random integer between 1 and 5 for the rating attribute, and sets the created_at and updated_at timestamps to the current date and time. This factory is useful for creating default data for the Landing model, aiding in testing and seeding databases during development.

## DATABASE:



```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=book
DB_USERNAME=root
DB_PASSWORD=
```

These settings configure the connection to the MySQL database, specifying the type of database (DB_CONNECTION), the host address (DB_HOST), the port number (DB_PORT), the database name (DB_DATABASE), the username (DB_USERNAME), and an empty password (DB_PASSWORD). The document mentions modifying these settings to update the necessary configuration for the application.