Student Management System

Overview

This document outlines the implementation steps taken to create a Student Management System using Laravel. The system includes three types of users: Admin, Student, and Guest, each with their own dashboard and permissions.

User Roles

- **Admin**: Only one user with full CRUD permissions and access to all user data.
- **Student**: Multiple users who can view user lists (excluding Admin) and create Guest users.
- **Guest**: Multiple users who can view user lists (excluding Admin) but have no CRUD operations.

Implementation Steps

1. **Initiated Laravel Project**
    - Used **Laravel Breeze** for authentication scaffolding.
2. **Installed Node Dependencies**
    - Set up necessary front-end dependencies.
3. **Installed Spatie Roles and Permissions Package**
    - Added package for role management.
4. **Provided Spatie Related Middlewares**
    - Updated **bootstrap/app.php** to include necessary middlewares.
5. **Created Migration File for Database Table**
    - Designed the database schema for users.
6. **Updated Models/User.php**
    - Added **$fillable** properties and used **HasRoles** trait.
7. **Created Seeders**
    - **RoleSeeder**: Inserted default roles: 'Admin', 'Student', 'Guest'.
    - **AdminSeeder**: Created a default Admin user.
8. **Migrated Database**
    - Ran migrations to create the **users** table and seeded it with initial data.

9. **Updated Registration Logic**
    - Modified **create()** and **store()** methods in **app/Http/Controllers/Auth/RegisteredUser Controller.php** to align with database columns and added validation rules.
10. **Created Controllers**
    - **AdminController**: Implemented CRUD operations and user list viewing access.
        - Used **AdminStoreUser Request** for user creation validation.
        - Used **AdminUpdateUser Request** for user update validation.
    - **StudentController**: Implemented user list access (excluding Admin) and guest user creation.
        - Used **StudentStoreGuestUser Request** for guest user creation validation.
    - **GuestController**: Implemented user list access (excluding Admin).
11. **Created View Files**
    - Structured views for different user pages:
        - **Dashboard Views**:
            - **views/dashboard/admin-dashboard.blade.php**
            - **views/dashboard/student-dashboard.blade.php**
            - **views/dashboard/guest-dashboard.blade.php**
        - **Pages Views**:
            - **views/pages/admin/index.blade.php**
            - **views/pages/admin/create.blade.php**
            - **views/pages/admin/edit.blade.php**
            - **views/pages/student/index.blade.php**
            - **views/pages/student/create.blade.php**
            - **views/pages/guest/index.blade.php**
        - **Home Page View**:
            - Updated **welcome.blade.php** to **home-page.blade.php** for a stylish landing page.
12. **Updated Layouts**
    - **app.blade.php**: Added logo and imported Chart.js for graphical representations.
    - **guest.blade.php**: Added logo.

- **navigation.blade.php**: Added logo.
- **register.blade.php**: Updated form according to the data table for user registration.

13. **Initiated Routes**
    - Defined routes for all controller methods with proper role-based middlewares.
14. **Updated Blade Files**
    - **index.blade.php**: Implemented user listing table.
    - **create.blade.php**: Implemented user creation form.
    - **edit.blade.php**: Implemented user data update form.
15. **Created Dashboard Controller**
    - **DashboardController.php**: Implemented graphical interfaces for each user type.
        - Created methods for Admin, Student, and Guest to show relevant data.
    - Added routes for dashboard methods and updated dashboard blade files with graphical view codes using JavaScript templates.

Conclusion

The Student Management System is fully implemented with distinct functionalities for Admin, Student, and Guest users. Each user type has a tailored dashboard with appropriate permissions and graphical representations. The system is built on Laravel, utilizing best practices for role management and user authentication.