Basic Authentication App With Laravel 8 and Bootstrap

1. Create a new Laravel project

```
laravel new basic-auth-app --git
cd basic-auth-app
php artisan serve
```

2. Edit .env file and setup database settings

Make database settings. Database name, user and password. Change the name of the app (APP_NAME property). Don't forget to create database.

```
APP_NAME="Basic Authentication App"
...

DB_DATABASE=basic_auth_app
```

3. Create the MainController and Login Page

Create the MainController:

```
php artisan make:controller MainController
```

Open the MainController.php at app\Http\Controllers folder and add the following login function:

```
    function login()
    {
        return view('auth.login');
    }
```

Open the web.php file at routes folder and add the login root as follows:

```
use App\Http\Controllers\MainController;

Route::get('/auth/login', [MainController::class, 'login'])->name('auth.login');
```

Create the view for the login page:

- Create a auth folder under resources\views folder.

- Under auth folder create a file named login.blade.php as follows:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>

    <!-- Bootstrap from CDN -->
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/bootstrap.min.cs
s">

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.min.js">
</script>
</head>

<body>
    <div class="container">
        <div class="row" style="margin-top: 45px;">
            <div class="col-md-4 col-md-offset-4">
                <h4>Login</h4>
                <hr>
                <form action="" method="post">
                    <div class="form-group">
                        <label for="email">E-mail</label>
                        <input type="text" class="form-control" name="email"
placeholder="Enter e-mail address">
                    </div>
                    <div class="form-group">
                        <label for="password">Password</label>
                        <input type="password" class="form-control"
name="password" placeholder="Enter password">
                    </div>
                    <button type="submit" class="btn btn-block btn-
primary">Sign In</button>
                    <br>
                    <a href="">I don't have an account, create an account.
</a>
                </form>
            </div>
        </div>
    </div>
</body>

</html>
```

We have creted the login page. You can test the login page at: http://localhost:8000/auth/login

Now let's commit our changes to our git repository:

- ```
  git add --all
  git commit -m "Login page created."
  ```

- You can do the same thing using the VS Code editor's git source tracking tool.

4. Create Registration Page

- Edit the MainController.php file and add the function for the registration in the MainController class.

  ```
  function register()
  {
      return view('auth.register');
  }
  ```

- Create the register route in web.php file:

  ```
  Route::get('/auth/register', [MainController::class, 'register'])-
  >name('auth.register');
  ```

- Create register view under resources\views\auth folder: register.blade.php

  ```
  <!DOCTYPE html>
  <html lang="en">

  <head>
      <meta charset="UTF-8">
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <title>Register</title>
      <!-- Bootstrap from CDN -->
      <link rel="stylesheet"
  href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/bootstrap.min.cs
  s">

      <script
  src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.min.js">
  </script>
  </head>

  <body>

  ```
  ```

```
            <div class="col-md-4 offset-md-4">
                <h4>Register</h4>
                <hr>
                <form action="" method="post">
                    <div class="form-group">
                        <label for="name">Name</label>
                        <input type="text" class="form-control" name="name"
placeholder="Enter e-mail address">
                    </div>
                    <div class="form-group">
                        <label for="email">E-mail</label>
                        <input type="text" class="form-control" name="email"
placeholder="Enter e-mail address">
                    </div>
                    <div class="form-group">
                        <label for="password">Password</label>
                        <input type="password" class="form-control"
name="password" placeholder="Enter password">
                    </div>
                    <button type="submit" class="btn btn-block btn-
primary">Sign Up</button>
                    <br>
                    <a href="">I already have an account, Sign In.</a>
                </form>
            </div>
        </div>
    </div>
</body>

</html>
```

- Correct the anchor tags in both login and register pages:

  login.blade.php:

```
<a href="{{ route('auth.register') }}">I don't have an account, create an
account.</a>
```

  register.blade.php:

```
<a href="{{ route('auth.login') }}">I already have an account, Sign In.</a>
```

4. Create the form actions on the registeration page

- Create the save route:

  web.php:

```
Route::post('/auth/save', [MainController::class, 'save'])-
>name('auth.save');
```

- Edit the register.blade.php file to add the action property to the form. Also add the '@csrf' directive to prevent cross-site request forgery:

```
<form action="{{ route('auth.save') }}" method="post">
@csrf
```

- Add save function to MainController.php file:

```
function save(Request $request)
{
    // Validate the form data
    $request->validate([
        'name' => 'required',
        'email' => 'required|email',
        'password' => 'required|min:5|max:12'
    ]);
}
```

- In order to maintain the last input value in the form fields we can use the `old` helper function in the form field values. We can also use the `error` helper function within a `span` tag so that in case an invalid value is entered in the form field we can provide feedback to the user about the error. Edit the register.blade.php file as follows:

```
<form action="{{ route('auth.save') }}" method="post">
    @csrf
    <div class="form-group">
        <label for="name">Name</label>
        <input type="text" class="form-control" name="name"
placeholder="Enter e-mail address"
            value="{{ old('name') }}">
        <span class="text-danger">@error('name') {{ $message }}
@enderror</span>
    </div>
    <div class="form-group">
        <label for="email">E-mail</label>
        <input type="text" class="form-control" name="email"
placeholder="Enter e-mail address"
            value="{{ old('email') }}">
        <span class="text-danger">@error('email') {{ $message }}
@enderror</span>
    </div>
    <div class="form-group">
        <label for="password">Password</label>
```

```
        <input type="password" class="form-control" name="password"
placeholder="Enter password">
        <span class="text-danger">@error('password') {{ $message }}
@enderror</span>
    </div>
    <button type="submit" class="btn btn-block btn-primary">Sign Up</button>
    <br>
    <a href="{{ route('auth.login') }}">I already have an account, Sign In.
</a>
</form>
```

5. Create the Admin model and migrations

- Create the Admin model with migrations

```
php artisan make:model Admin -m
```

- Edit the migrations file to add some columns to the admins table:

```
public function up()
{
    Schema::create('admins', function (Blueprint $table) {
        $table->id();
        $table->text('name');
        $table->text('email');
        $table->text('password');
        $table->timestamps();
    });
}
```

- Create the admins table by running the migrations

```
php artisan migrate
```

- Edit the validation rule in the MainController.php file to check if the entered e-mail in the form is a unique e-mail in the admins table:

```
$request->validate([
    'name' => 'required',
    'email' => 'required|email|unique:admins',
    'password' => 'required|min:5|max:12'
]);
```

6. Edit the the MainController.php to save the user data in the admins table

- Add the Admin model to the MainController.php file:

```
use app\Models\Admin;
```

- Edit the save function in the MainController.php file to save the form values to the database:

```
// Insert the data to the database
$admin = new Admin();
$admin->name = $request->name;
$admin->email = $request->email;
$admin->password = $request->password;
$save = $admin->save();
```

- To check if the data is saved successfully edit the MainController.php as follows:

```
if ($save) {
    return back()->with('success', 'User created successfully');
} else {
    return back()->with('fail', 'Something wrong, try again.');
}
```

- To display these return messages on the registration page edit the register.blade.php file as follows:

```
<form action="{{ route('auth.save') }}" method="post">
    @if (Session::get('success'))
    <div class="alert alert-success">
        {{ Session::get('success') }}
    </div>
    @endif
    @if (Session::get('fail'))
    <div class="alert alert-danger">
        {{ Session::get('fail') }}
    </div>
    @endif
```

- The password data is saved to the table unencrypted. To save the encrypted password instead we can use the `Hash` class. Edit the MainController.php file to make this change:

```
use Illuminate\Support\Facades\Hash;
```

```
...

$admin->password = Hash::make($request->password);
```

- Registration process is created completely.

7. Create the login page functions

- First let's create the password check route for the login page. Edit the web.php file for this:

```
Route::post('/auth/check', [MainController::class, 'check'])-
>name('auth.check');
```

- Edit the login.blade.php file to add the login form action:

```
<form action="{{ route('auth.check') }}" method="post">
    @csrf
```

- Add the check function in the MainController.php file:

```
function check(Request $request)
{
    // Validate the form data
    $request->validate([
        'email' => 'required|email',
        'password' => 'required|min:5|max:12'
    ]);

    // Check user credentials by querying the database
    $userinfo = Admin::where('email', '=', $request->email)->first();

    // If the user is not found return an error message
    if (!$userinfo) {
        return back()->with('fail', 'Not a valid e-mail address.');
    } else {
        // Check if the password is correct
        if (Hash::check($request->password, $userinfo->password)) {
            // If the password is correct save the logged in
            // user id in the session
            // and redirect user to the dashboard
            $request->session()->put('LoggedUser', $userinfo->id);
            return redirect('admin/dashboard');
        } else {
            // If the password is not correct return an error message
            return back()->with('fail', 'Incorrect password.');
        }
```

```
    }
}
```

- Edit the login.blade.php file to include action, to display validation errors, and to keep old values

```
<form action="{{ route('auth.check') }}" method="post">
    @if(Session::get('fail'))
    <div class="alert alert-danger">
        {{ Session::get('fail') }}
    </div>
    @endif
    @csrf
    <div class="form-group">
        <label for="email">E-mail</label>
        <input type="text" class="form-control" name="email"
placeholder="Enter e-mail address"
            value="{{ old('email') }}">
        <span class="text-danger">@error('email') {{ $message }}
@enderror</span>
    </div>
    <div class="form-group">
        <label for="password">Password</label>
        <input type="password" class="form-control" name="password"
placeholder="Enter password">
        <span class="text-danger">@error('password') {{ $message }}
@enderror</span>
    </div>
    <button type="submit" class="btn btn-block btn-primary">Sign In</button>
    <br>
    <a href="{{ route('auth.register') }}">I don't have an account, create
an account.</a>
</form>
```

8. Create the dashboard page

- Edit the web.php file to create the dashboard route

```
Route::get('/admin/dashboard', [MainController::class, 'dashboard'])-
>name('auth.dashboard');
```

- Now create the `dashboard` method in the MainController.php file

```
function dashboard()
{
    // Get the user data from the database to pass to the
    // dashboard page
    $data = ['LoggedUserInfo' => Admin::where('id', '=',
```

```
session('LoggedUser'))->first()];
    return view('admin.dashboard', $data);
}
```

- Create the dashboard view file dashboad.blade.php in resources\views\admin folder

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Dashboard</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/bootstrap.min.cs
s">

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.min.js">
</script>

</head>

<body>
    <div class="container">
        <div class="row">
            <div class="col-md-10 col-md-offset-3">
                <h4>Dashboard</h4>
                <hr>
                <table class="table table-hover">
                    <thead>
                        <th>Name</th>
                        <th>E-mail</th>
                        <th></th>
                    </thead>
                    <tbody>
                        <tr>
                            <td>{{ $LoggedUserInfo['name'] }}</td>
                            <td>{{ $LoggedUserInfo['email'] }}</td>
                            <td><a href="{{ route('auth.logout')
}}">Logout</a></td>

                        </tr>
                    </tbody>
                </table>
            </div>
        </div>
    </div>
</body>
```

```
</html>
```

- Create the logout function in the MainController.php file:

```
function logout()
{
    // If the LoggedUser id is in the session info
    // remove it from session and redirect to the login page.
    if (session()->has('LoggedUser')) {
        session()->pull('LoggedUser');
        return redirect('auth/login');
    }
}
```

- Create the logout route in the web.php file:

```
Route::get('/auth/logout', [MainController::class, 'logout'])-
>name('auth.logout');
```

9. Create middleware to protect the dashboard page

The dashboard page can be seen without loging in if the dashboard url is entered in the address bar directly. We can create a middleware route to prevent direct access to the dashboard.

- Create an authorization check middleware

```
php artisan make:middleware AuthCheck
```

A middlleware file AuthCheck.php is created in app\Http\Middleware folder

- We have to register this middleware in the Kernel.php file under app\Http folder. Edit the Kernel.php file and add the following line inside `$routeMiddleware` array:

```
'AuthCheck' => \App\Http\Middleware\AuthCheck::class,
```

- We have to create a middleware route in the web.php file and put the routes that we want to keep safe inside this middleware route. Change the web.php file as follows:

```
Route::group(['middleware' => ['AuthCheck']], function () {
    Route::get('/auth/login', [MainController::class, 'login'])-
>name('auth.login');
```

```
    Route::get('/auth/register', [MainController::class, 'register'])-
>name('auth.register');

    Route::get('/admin/dashboard', [MainController::class, 'dashboard'])-
>name('auth.dashboard');
});
```

- Then we have to edit the middleware file AuthCheck.php file as follows:

```
public function handle(Request $request, Closure $next)
{
    // If the use is not logged in and tries to access routes other than
'register' or 'login'
    // then redirect user to login page with an error message.
    if (!session()->has('LoggedUser') && ($request->path() != 'auth/login'
&& $request->path() != 'auth/register')) {
        return redirect('auth/login')->with('fail', 'You must be logged
in.');
    }
    // If the user is already logged in and tries to access the login or
register pages
    // then redirect the user back.
    if (session()->has('LoggedUser') && ($request->path() == 'auth/login' ||
$request->path() == 'auth/register')) {
        return back();
    }
    return $next($request);
}
```

- In order to prevent the user go back to the dashboard page after logout edit the AuthCehck.php to pass some parameters to the page header.

```
return $next($request)->header('Cache-Control', 'no-cache, no-store, max-
age=0, must-revalidate')
    ->header('Pragma', 'no-cache')
    ->header('Expires', 'Sat 01 Jan 1900 00:00:00 GMT');
```