CONTROLLERS

PART 1: CREATE AND REGISTER CONTROLLERS

- **Create HomeController (loads home page) and DashboardController (loads dashboard, feed, or equivalent pages).**

```
C:\Users\user\Herd\lab5_controllers>php artisan make:controller HomeController

  INFO  Controller [C:\Users\user\Herd\lab5_controllers\app\Http\Controllers\HomeController.php] created successfully.

C:\Users\user\Herd\lab5_controllers>php artisan make:controller DashboardController

  INFO  Controller [C:\Users\user\Herd\lab5_controllers\app\Http\Controllers\DashboardController.php] created successfully.
```

EXPLANATION:

To create the HomeController and DashboardController in Laravel, simply run these commands in the terminal. This controller is typically used to handle the logic for the home page and dashboard of our application.

- **Register controllers in routes to link methods to URLs.**

routes > web.php > ...

```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\HomeController;
use App\Http\Controllers\DashboardController;


Route::get('/', [HomeController::class, 'index'])->name('home');
Route::get('/dashboard', [DashboardController::class, 'index'])->name('dashboard');

Route::get('/menu', function () {
    return view('menu');
});

Route::get('/sign-up', function () {
    return view('sign-up');
});

Route::get('/login', function () {
    return view('login');
});
```

EXPLANATION:

This code links URLs to specific controller methods. When a user visits the home page (/), it uses the HomeController and calls its index method to display the home page. Similarly, when a user visits /dashboard, it uses the DashboardController and calls its index method to display the dashboard.

- **Controller Simple Method for Loading Page View**

> Http > Controllers > HomeController.php > Hom

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class HomeController extends Controller
{
    public function index()
    {
        return view('home');
    }
}
```

> Http > Controllers > DashboardController.php > Dashl

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class DashboardController extends Controller
{
    public function index()
    {
        return view('dashboard');
    }
}
```

PART 2: ASSIGN CONTROLLERS TO ROUTES

- **Use middleware (e.g., authentication) to protect specific routes and controllers.**

routes > web.php > ...

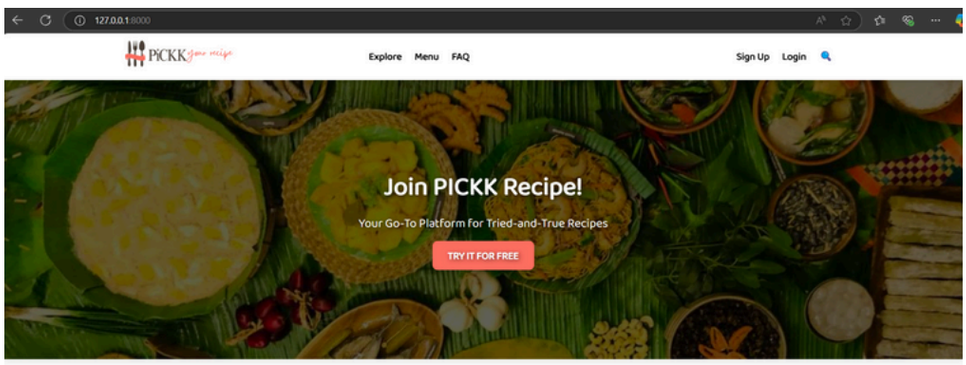
```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\HomeController;
use App\Http\Controllers\DashboardController;


Route::get('/', [HomeController::class, 'index'])->name('home');

Route::middleware(['auth'])->group(function () {
    Route::get('/dashboard/{userId}', [DashboardController::class, 'index'])->name('dashboard');
});

Route::view('/dashboard', 'dashboard')->name('dashboard');

```

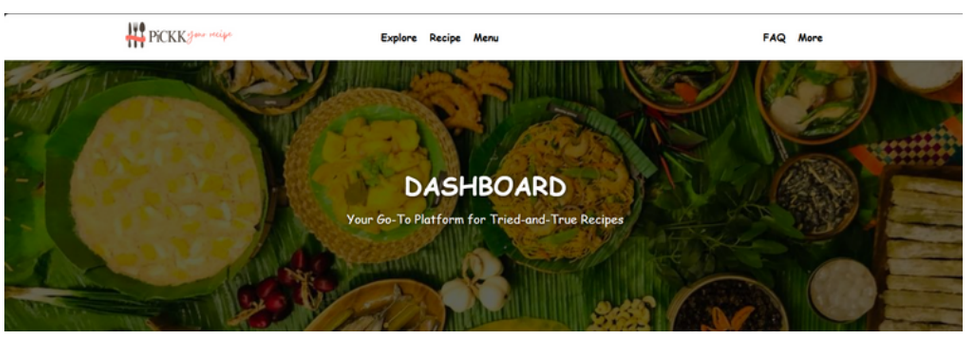
EXPLANATION:

This code uses middleware to protect certain routes, ensuring that only authenticated users can access them. The route for `/dashboard/{userId}` is wrapped inside a middleware group, meaning users must be logged in to view the dashboard. This helps restrict access to specific pages, like the dashboard, to authenticated users only.

- **Test routes (e.g. /, /dashboard) to ensure proper page loading.**

OUTPUT:





PART 3: CONTROLLERS WITH PARAMETERS

- **Modify DashboardController to accept dynamic parameters (e.g., user ID).**

app > Http > Controllers > DashboardController.php > ...

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class DashboardController extends Controller
{
    public function index($userId)
    {
        // Example: Fetch user-specific data based on the user ID
        $user = User::find($userId); // Make sure to import the User model

        return view('dashboard', ['user' => $user]);
    }
}

```

EXPLANATION:

This code modifies the `DashboardController` to accept a dynamic parameter, like a user ID, in the `index` method. When the route for the dashboard is visited, the controller uses the user ID from the URL to fetch the corresponding user's data. It then passes this data to the `dashboard` view, so the page can display information specific to that user.

- **Test parameterized routes to load user-specific data.**

```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\HomeController;
use App\Http\Controllers\DashboardController;


Route::get('/', [HomeController::class, 'index'])->name('home');

Route::middleware(['auth'])->group(function () {
    Route::get('/dashboard/{userId}', [DashboardController::class, 'index'])->name('dashboard');
});

Route::view('/dashboard', 'dashboard')->name('dashboard');
```

EXPLANATION:

This code defines a route that loads user-specific data by using a dynamic parameter, like a user ID, in the URL. The route is protected by the `auth` middleware, meaning only logged-in users can access it.

PART 4: EXPLANATION

- **Controller Logic**

  Controllers in Laravel manage the logic for handling incoming requests. They retrieve data, process it, and return views to display the data. For example, a DashboardController can fetch user-specific data and display it on the dashboard page.

- **Parameter Handling:**

  Parameters in routes allow dynamic data (like a user ID) to be passed into controllers. When defining a route, parameters can be added to the URL (e.g., /dashboard/{userId}). The controller then uses this parameter to fetch the relevant data, such as user information.

- **Route Assignments:**

  Routes define the URLs of your application and link them to controller methods. In Laravel, you use Route::get() to assign URLs to specific controller methods. Middleware can be applied to routes to control access, like ensuring users are authenticated before accessing certain pages.