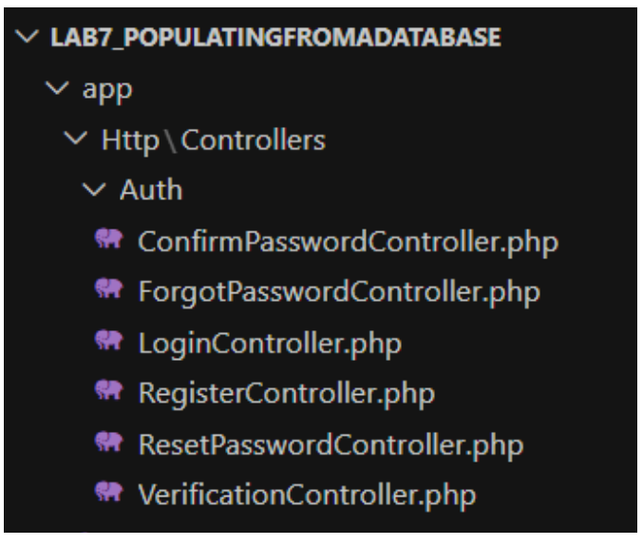
POPULATING FROM A DATABASE

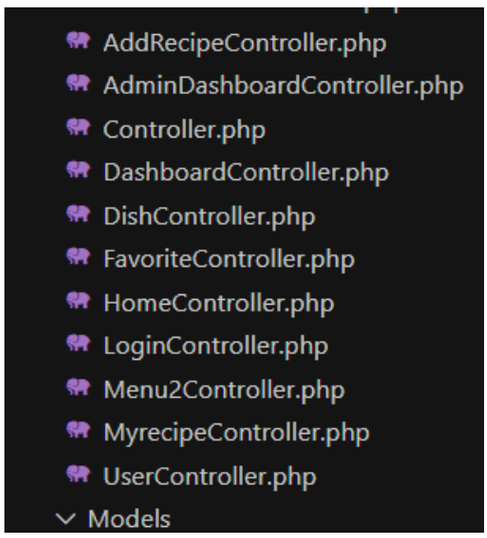
PART 1: CREATE A DATABASE AND CREATE A CONNECTION TO IT.

- **Create controllers that load the dashboard, feed, or equivalent pages.**

```
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.
```

```
C:\Users\user\Herd\lab6_populatingpages>php artisan make:controller Menu2Controller

  INFO  Controller [C:\Users\user\Herd\lab6_populatingpages\app\Http\Controllers\Menu2Controller.php] created successfully.
```





EXPLANATION:

To create a dashboard in Laravel, simply run these commands in the terminal. This controller is typically used to handle the logic for every page that needed a controller (eg,. the dashboard and menu of our application, etc)

- **Register controllers in routes to link method to URLs**

```
Route::get('/login', [LoginController::class, 'showLoginForm'])->name('login');
Route::post('/login', [LoginController::class, 'login']);

Route::get('/sign-up', [LoginController::class, 'showSignUpForm'])->name('sign-up');
Route::post('/sign-up', [LoginController::class, 'signUp']);



// USER VIEW

Route::get('/menu2', [Menu2Controller::class, 'index'])->name('menu2');

Route::get('/faq2', function () {
    return view('faq2');
});
Route::view('/faq2', 'faq2')->name('faq2');

Route::get('/profile', [ProfileController::class, 'show'])->name('profile');
Route::get('/my-recipe', [AllRecipeController::class, 'index']);
Route::get('/favorites', function () {
    return view('favorites');
});
Route::view('/favorites', 'favorites')->name('favorites');
```

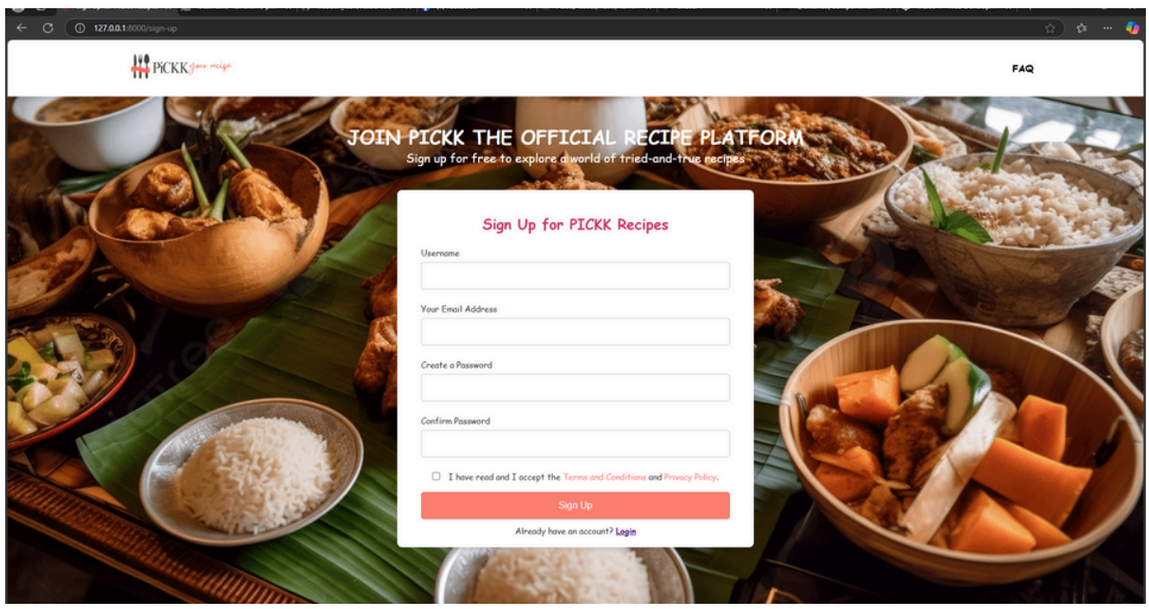
```
Route::get('/', [HomeController::class, 'index'])->name('home');

Route::middleware(['auth'])->group(function () {

    // Regular user dashboard
    Route::get('/dashboard', [DashboardController::class, 'index'])->name('

    // Admin dashboard route
    Route::get('/admin', [AdminDashboardController::class, 'index'])->name(
    Route::get('/admin-users', [AdminUserController::class, 'index'])->name
    Route::delete('/admin-users/{id}', [AdminUserController::class, 'delete
    Route::delete('/admin/delete/{id}', [AdminDashboardController::class, '
    Route::put('/admin/approve/{id}', [AdminDashboardController::class, 'app


});
```

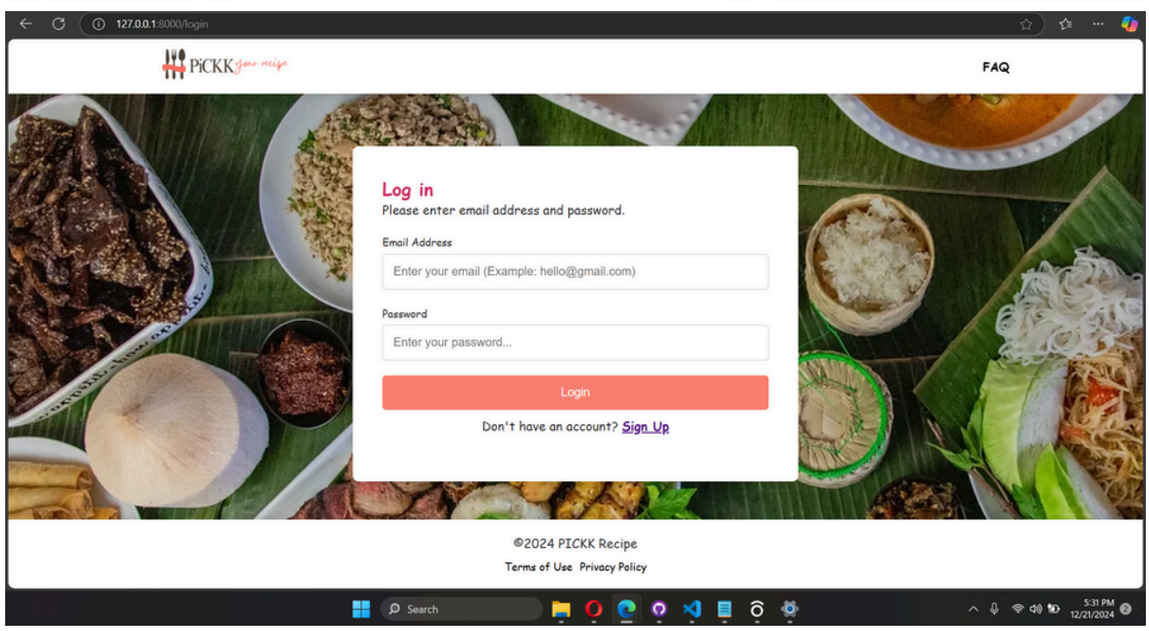
```php
Route::get('/dish', function () {
    return view('dish');  // This will render the adobo.blade.php view
})->name('dish');
Route::get('/created', [MyrecipeController::class, 'getRecipesByUser'])->name('created');

Route::post('/my-recipe', [AddRecipeController::class, 'store'])->name('my-recipe');


Auth::routes();
Route::get('/menu/{recipeId}', [DishController::class, 'showRecipe']);
Route::post('/comments', [DishController::class, 'store'])->name('comments.store');
Route::post('/favorite/add', [DishController::class, 'addFavorite'])->name('favorite.add');
Route::post('/favorite/remove', [DishController::class, 'removeFavorite'])->name('favorite.remove');
Route::get('/favorites', [FavoriteController::class, 'favorites'])->name('favorites');

Route::get('/recipe/{id}', [DishController::class, 'showRecipe'])->name('recipe.show');

Route::get('/dish/{recipeId}', [DishController::class, 'showRecipe'])->name('dish');
```

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. When a user visits /menu2, it uses the Menu2Controller and calls its index method to display the menu and so on.

PART 2: MAKE AN ACCOUNT REGISTRATION AND LOGIN PAGE.

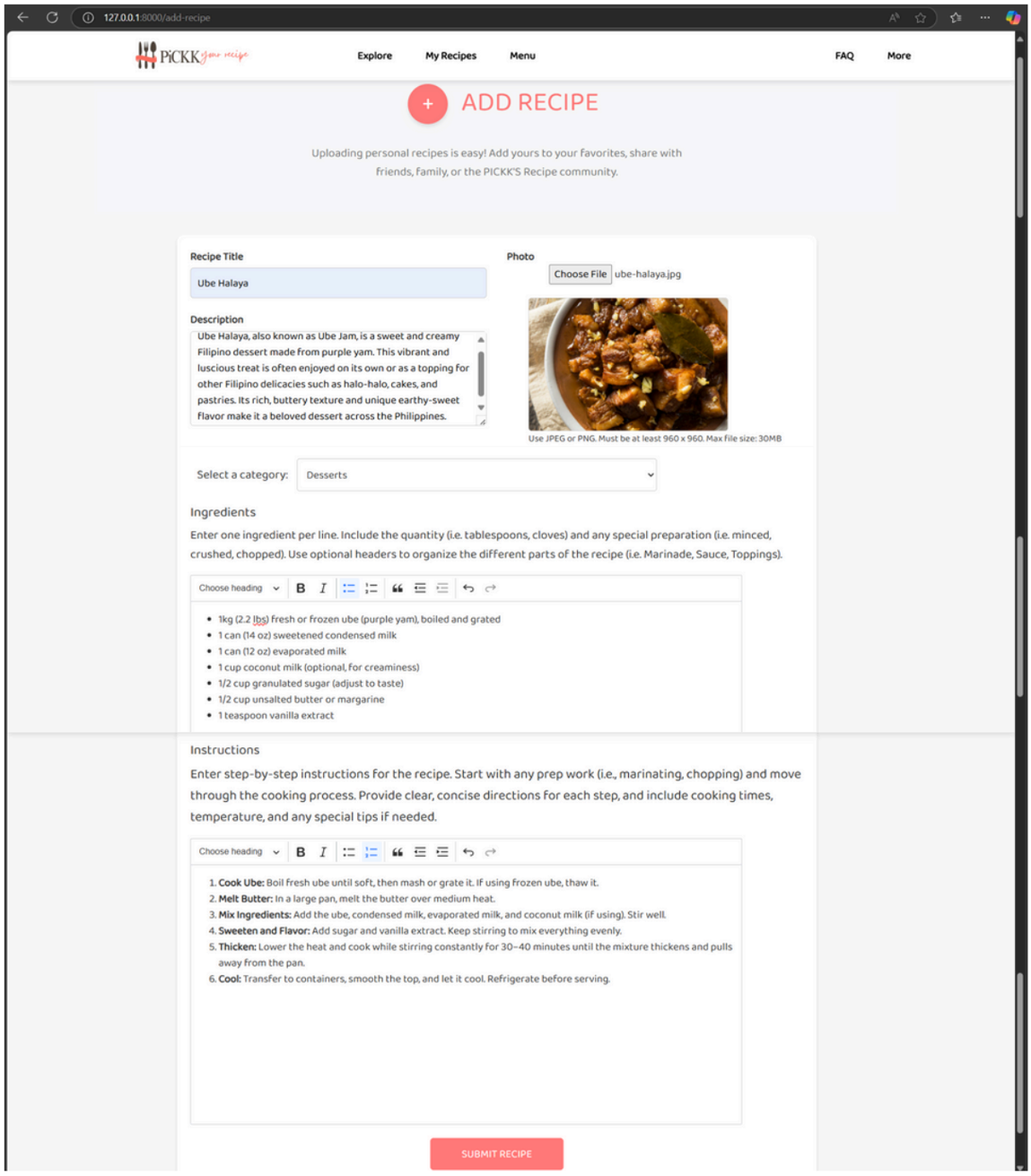




PART 3: POPULATE YOUR PAGES USING DATA FETCHED FROM A DATABASE.

- **Populate at least 1 page with content.**

ADD RECIPE PAGE



DATABASE TABLE

	id	name	user_id	image	description	ingredients	instructions	ratings	categories	status	created_at	updated_at
Delete	1	Ube Halaya	2	recipe_photos/Frqiao0feOZmXu6zXsKamarbgiJ1OC5mSYAe...	Ube Halaya, also known as Ube Jam, is a sweet and ...	1 kg (2.2 lbs) fresh or frozen ube (purple...	Cook Ube: Boil fresh ube ...	0	desserts	approved	2024-12-21 12:47:14	2024-12-21 12:50:46

With selected: Edit Copy Delete Export

ADD RECIPE CONTROLLER

AddRecipeController.php

app > Http > Controllers > AddRecipeController.php > ...

```php
<?php

namespace App\Http\Controllers;

use App\Models\Recipe;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Log;

class AddRecipeController extends Controller
{
    public function store(Request $request)
    {
        try {
            Log::info('AddRecipeController: Store method called', [
                'user_id' => Auth::id(),
                'request_data' => $request->all()
            ]);
            Log::info('Request Data:', $request->all());

            // Validate incoming data
            $validator = Validator::make($request->all(), [
                'recipetitle' => 'required|string|max:255',
                'description' => 'required|string',
                'photo' => 'required|image|mimes:jpeg,png,jpg,gif|max:2048',
                'ingredients' => 'nullable|string',
                'categories' => 'nullable|string',
                'instructions' => 'nullable|string', // Validate the instructions,
            ]);

            if ($validator->fails()) {
                Log::warning('AddRecipeController: Validation failed', [
                    'errors' => $validator->errors()->toArray()
                ]);
                return response()->json(['errors' => $validator->errors()], 400);
            }
```

```php
    // Handle photo upload
    $photoPath = null;
    if ($request->hasFile('photo')) {
        $photoPath = $request->file('photo')->store('recipe_photos', 'public');
        Log::info('AddRecipeController: Photo uploaded', [
            'photo_path' => $photoPath
        ]);
    }

    // Save recipe
    $recipe = Recipe::create([
        'name' => $request->recipetitle,
        'description' => $request->description,
        'user_id' => Auth::id(),
        'image' => $photoPath,
        'ingredients' => $request->ingredients,
        'categories' => $request->category,
        'ratings' => 0,
        'status' => 'pending',
        'instructions' => $request->instructions, // Save the instructions
    ]);

    Log::info('AddRecipeController: Recipe saved successfully', [
        'recipe_id' => $recipe->id,
        'recipe_data' => $recipe->toArray()
    ]);

    return response()->json([
        'message' => 'Recipe added successfully!'
    ], 201);

} catch (\Illuminate\Database\QueryException $e) {
    Log::error('AddRecipeController: Database error occurred', [
```

```php
                return response()->json([
                    'message' => 'Recipe added successfully!'
                ], 201);

            } catch (\Illuminate\Database\QueryException $e) {
                Log::error('AddRecipeController: Database error occurred', [
                    'error_message' => $e->getMessage(),
                    'trace' => $e->getTraceAsString()
                ]);
                return response()->json(['error' => 'Database error occurred'], 500);

            } catch (\Exception $e) {
                Log::error('AddRecipeController: Unexpected error occurred', [
                    'error_message' => $e->getMessage(),
                    'trace' => $e->getTraceAsString()
                ]);
                return response()->json(['error' => 'An unexpected error occurred'], 500);
            }
        }
    }

```

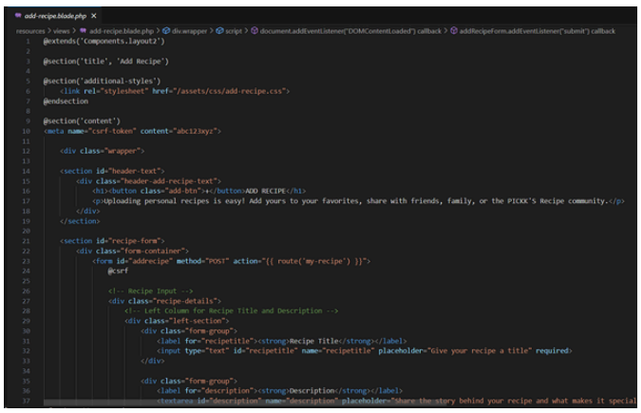
RECIPE MODEL

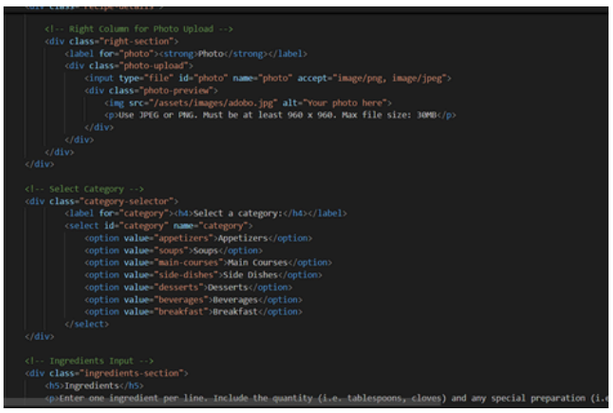
Recipe.php

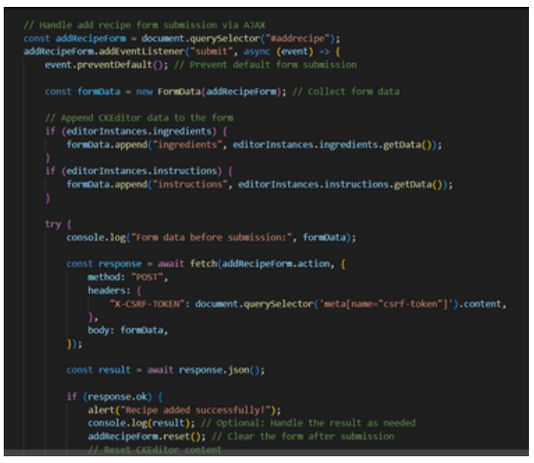
app > Models > Recipe.php > ...

```php
<?php

namespace App\Models;

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

class Recipe extends Model
{
    use HasFactory;

    protected $fillable = [
        'name',
        'description',
        'user_id',
        'image',
        'instructions',
        'ingredients',
        'ratings',
        'categories',
        'status'
    ];
    public function usersFavorited()
    {
        return $this->belongsToMany(User::class, 'favorites', 'recipe_id', 'user_id');
    }
    public function user()
{
    return $this->belongsTo(User::class);
}
}

```

RECIPE BLADE VIEW







EXPLANATION:

In this part, we implemented dynamic page rendering by fetching data from the database and populating it on at least one page. This involved the following steps:

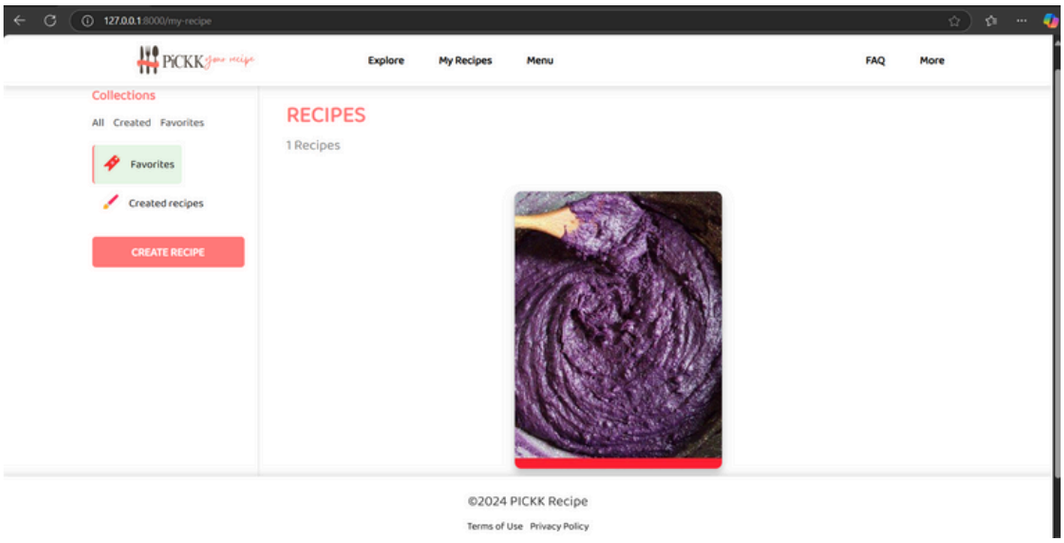
1. *Model: Created/used a model to define the structure and interactions with the database, such as retrieving posts, comments, or pictures.*
2. *Controller: Set up a controller to fetch the required data from the model and prepare it for the view.*
3. *View: Passed the fetched data to the view, where it was dynamically displayed on the page.*

By doing this, we ensured that the page content is loaded dynamically based on the database's current state, making the application interactive and scalable.

RECIPE ROUTES

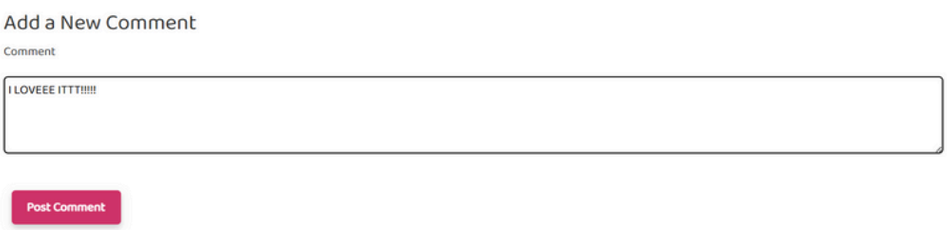
```php
Route::get('/dish', function () {
    return view('dish');  // This will render the adobo.blade.php view
})->name('dish');
Route::get('/created', [MyrecipeController::class, 'getRecipesByUser'])->name('created');

Route::post('/my-recipe', [AddRecipeController::class, 'store'])->name('my-recipe');
```

POST TO CREATED RECIPES



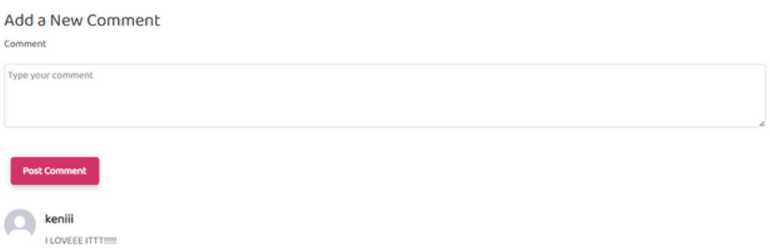
COMMENT

COMMENT VIEW



DATABASE TABLE

COMMENT POST



COMMENT BLADE VIEWS

```
<section id="comment-section">
    @foreach ($comments as $comment)
        <div class="comment">
            <div class="profile-pic">
                <img src="/assets/images/default-pfp.jpg" alt="User profile picture">
            </div>
            <div class="comment-content">
                <p class="username">{{ $comment->username }}</p>
                <p class="text">{{ $comment->comment }}</p>
            </div>
        </div>
    @endforeach
</section>
```

COMMENT CONTROLLER

```php
public function showRecipe($recipeId)
{
    // Query the database using Eloquent or Query Builder
    $recipe = DB::table('recipes')
                ->where('id', $recipeId)
                ->first();

    // Check if recipe exists
    if (!$recipe) {
        abort(404, 'Recipe not found');
    }

    // Fetch comments for the recipe
    $comments = DB::table('comments')
        ->join('users', 'comments.user_id', '=', 'users.id')
        ->select('users.username as username', 'comments.comment', 'comments.created_at')
        ->where('comments.recipe_id', $recipeId)
        ->orderBy('comments.created_at', 'desc')
        ->get();

    // Check if the recipe is favorited by the current user
    $isFavorited = Auth::check() && Favorite::where('user_id', Auth::id())->where('recipe_id

    // Pass the data to the Blade view
    return view('dish', compact('recipe', 'comments', 'isFavorited'));
}
```

```php
public function store(Request $request)
{
    try {
        // Validate the incoming request
        $validated = $request->validate([
            'recipe_id' => 'required|integer|exists:recipes,id',
            'comment' => 'required|string|max:65535',
        ]);

        // Create a new comment
        Comment::create([
            'recipe_id' => $validated['recipe_id'],
            'user_id' => Auth::id(),
            'comment' => $validated['comment'],
        ]);

        // Flash success message to the session
        session()->flash('success', 'Comment successfully added!');

        // Return the view (or redirect to the current page if needed)
        return back();
    } catch (\Exception $e) {
        // Flash error message to the session
        session()->flash('error', 'Failed to add comment. Please try again.');

        // Return the view (or redirect to the current page if needed)
        return back();
    }
}
```

COMMENT MODEL

Comment.php

app > Models > Comment.php > ...

```php
<?php

namespace App\Models;

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

class Comment extends Model
{
    use HasFactory;

    // Add the fillable properties
    protected $fillable = ['recipe_id', 'user_id', 'comment'];
}

```

PART 4: EXPLANATION

- **Controller Logic**
    - *The Controller retrieves and processes the data needed for the page, like user stats, recent activities, or updates. It gathers this information, processes it if necessary, and sends it to the view for display.*
    - *For example in the Menu Controller handles the logic for displaying the menu page, such as the list of available options, categories, or content. It manages the data related to the menu and ensures it is presented properly to the user.*
- **Parameter Handling:**

  In controllers, parameters can be used to customize the data. For example, a user might request a specific dashboard or menu item, and the controller will use parameters to fetch and display the correct data.

- **Route Assignments:**

  Route assignments in controllers map specific URLs to actions in the controller. When a user visits a URL, the route ensures the correct controller action is called to display the data needed

- **Others things you learned while making the laboratory 7.**

  In this laboratory, I learned how to design and implement a full-stack web application. I started by creating a database and establishing a connection to it, then set up routes and controllers to handle navigation and functionality for pages like the dashboard or feed. I also built user authentication features, including registration and login. By fetching data from the database and populating web pages dynamically, I gained hands-on experience with using models, controllers, and views. This helped me understand the MVC architecture and how it all comes together to create interactive, data-driven applications.