

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.
```

✓ LAB7_POPULATINGFROMADATABASE

- ✓ app
 - ✓ Http\Controllers
 - ✓ Auth
 - (confirmPasswordController.php)
 - (forgotPasswordController.php)
 - (loginController.php)
 - (registerController.php)
 - (resetPasswordController.php)
 - (verificationController.php)

- (addRecipeController.php)
- (adminDashboardController.php)
- (controller.php)
- (dashboardController.php)
- (dishController.php)
- (favoriteController.php)
- (homeController.php)
- (loginController.php)
- (menu2Controller.php)
- (myrecipeController.php)
- (userController.php)

✓ Models

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 VIEWS
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('user-dashboard');

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

});
```

```

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.

The image consists of two screenshots of a web application interface. Both screenshots feature a background image of various traditional Asian dishes, including bowls of rice, soups, and stir-fries, arranged on banana leaves.

Top Screenshot (Sign-up Page):

- The URL is 127.0.0.1:8000/sign-up.
- The title is "JOIN PICKK THE OFFICIAL RECIPE PLATFORM".
- The sub-instruction is "Sign up for free to explore a world of tried-and-true recipes".
- A central modal window titled "Sign Up for PICKK Recipes" contains fields for "Username", "Your Email Address", "Create a Password", and "Confirm Password".
- Below the fields is a checkbox: "I have read and I accept the [Terms and Conditions](#) and [Privacy Policy](#)".
- A large orange "Sign Up" button is at the bottom of the modal.
- Below the modal, a link says "Already have an account? [Login](#)".

Bottom Screenshot (Login Page):

- The URL is 127.0.0.1:8000/login.
- The title is "Log in".
- The sub-instruction is "Please enter email address and password".
- A central modal window contains fields for "Email Address" (with placeholder "Enter your email (Example: hello@gmail.com)") and "Password" (with placeholder "Enter your password...").
- An orange "Login" button is at the bottom of the modal.
- Below the modal, a link says "Don't have an account? [Sign Up](#)".
- At the very bottom of the page, there are footer links: "©2024 PICKK Recipe", "Terms of Use", and "Privacy Policy".

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

- Populate at least 1 page with content.

ADD RECIPE PAGE

Uploading personal recipes is easy! Add yours to your favorites, share with friends, family, or the PICKKS Recipe community.

Recipe Title

Ube Halaya

Description

Ube Halaya, also known as Ube Jam, is a sweet and creamy Filipino dessert made from purple yam. This vibrant and luscious treat is often enjoyed on its own or as a topping for other Filipino delicacies such as halo-halo, cakes, and pastries. Its rich, buttery texture and unique earthy-sweet flavor make it a beloved dessert across the Philippines.

Photo

Choose File ube-halaya.jpg

Use JPEG or PNG. Must be at least 960 x 960. Max file size: 30MB

Select a category: Desserts

Ingredients

Enter one ingredient per line. Include the quantity (i.e. tablespoons, cloves) and any special preparation (i.e. minced, crushed, chopped). Use optional headers to organize the different parts of the recipe (i.e. Marinade, Sauce, Toppings).

- 1kg (2.2 lbs) Fresh or frozen ube (purple yam), boiled and grated
- 1 can (14 oz) sweetened condensed milk
- 1 can (12 oz) evaporated milk
- 1 cup coconut milk (optional, for creaminess)
- 1/2 cup granulated sugar (adjust to taste)
- 1/2 cup unsalted butter or margarine
- 1 teaspoon vanilla extract

Instructions

Enter step-by-step instructions for the recipe. Start with any prep work (i.e., marinating, chopping) and move through the cooking process. Provide clear, concise directions for each step, and include cooking times, temperature, and any special tips if needed.

- Cook Ube: Boil fresh ube until soft, then mash or grate it. If using frozen ube, thaw it.
- Melt Butter: In a large pan, melt the butter over medium heat.
- Mix Ingredients: Add the ube, condensed milk, evaporated milk, and coconut milk (if using). Stir well.
- Sweeten and Flavor: Add sugar and vanilla extract. Keep stirring to mix everything evenly.
- Thicken: Lower the heat and cook while stirring constantly for 30–40 minutes until the mixture thickens and pulls away from the pan.
- Cool: Transfer to containers, smooth the top, and let it cool. Refrigerate before serving.

SUBMIT RECIPE

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 creamy Filipino dessert made from purple yam. This vibrant and luscious treat is often enjoyed on its own or as a topping for other Filipino delicacies such as halo-halo, cakes, and pastries. Its rich, buttery texture and unique earthy-sweet flavor make it a beloved dessert across the Philippines.	1 kg (2.2 lbs) fresh or frozen ube (purple yam), boiled and grated1 can (14 oz) sweetened condensed milk1 can (12 oz) evaporated milk1 cup coconut milk (optional, for creaminess)1/2 cup granulated sugar (adjust to taste)1/2 cup unsalted butter or margarine1 teaspoon vanilla extract	Cook Ube: Boil fresh ube until soft, then mash or grate it. If using frozen ube, thaw it.Melt Butter: In a large pan, melt the butter over medium heat.Mix Ingredients: Add the ube, condensed milk, evaporated milk, and coconut milk (if using). Stir well.Sweeten and Flavor: Add sugar and vanilla extract. Keep stirring to mix everything evenly.Thicken: Lower the heat and cook while stirring constantly for 30–40 minutes until the mixture thickens and pulls away from the pan.Cool: Transfer to containers, smooth the top, and let it cool. Refrigerate before serving.	0	desserts	approved	2024-12-21 12:47:14	2024-12-21 12:50:46

With selected: [Edit](#) [Copy](#) [Delete](#) [Export](#)

ADD RECIPE CONTROLLER

```
app > Http > Controllers > AddRecipeController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Models\Recipe;
6  use Illuminate\Http\Request;
7  use Illuminate\Support\Facades\Validator;
8  use Illuminate\Support\Facades\Auth;
9  use Illuminate\Support\Facades\Log;
10
11 class AddRecipeController extends Controller
12 {
13     public function store(Request $request)
14     {
15         try {
16             Log::info('AddRecipeController: Store method called', [
17                 'user_id' => Auth::id(),
18                 'request_data' => $request->all()
19             ]);
20             Log::info('Request Data:', $request->all());
21
22             // Validate incoming data
23             $validator = Validator::make($request->all(), [
24                 'recipetitle' => 'required|string|max:255',
25                 'description' => 'required|string',
26                 'photo' => 'required|image|mimes:jpeg,png,jpg,gif|max:2048',
27                 'ingredients' => 'nullable|string',
28                 'categories' => 'nullable|string',
29                 'instructions' => 'nullable|string', // Validate the instructions,
30             ]);
31
32             if ($validator->fails()) {
33                 Log::warning('AddRecipeController: Validation failed', [
34                     'errors' => $validator->errors()->toArray()
35                 ]);
36                 return response()->json(['errors' => $validator->errors()], 400);
37             }
38
39         } catch (\Exception $e) {
40             Log::error('AddRecipeController: Validation failed', [
41                 'error' => $e->getMessage(),
42                 'trace' => $e->getTraceAsString()
43             ]);
44             return response()->json(['error' => 'Validation failed'], 400);
45
46         }
47
48         // Handle photo upload
49         $photoPath = null;
50         if ($request->hasFile('photo')) {
51             $photoPath = $request->file('photo')->store('recipe_photos', 'public');
52             Log::info('AddRecipeController: Photo uploaded', [
53                 'photo_path' => $photoPath
54             ]);
55         }
56
57         // Save recipe
58         $recipe = Recipe::create([
59             'name' => $request->recipetitle,
60             'description' => $request->description,
61             'user_id' => Auth::id(),
62             'image' => $photoPath,
63             'ingredients' => $request->ingredients,
64             'categories' => $request->category,
65             'ratings' => 0,
66             'status' => 'pending',
67             'instructions' => $request->instructions, // Save the instructions
68         ]);
69
70         Log::info('AddRecipeController: Recipe saved successfully', [
71             'recipe_id' => $recipe->id,
72             'recipe_data' => $recipe->toArray()
73         ]);
74
75         return response()->json([
76             'message' => 'Recipe added successfully!'
77         ], 201);
78
79     } catch (\Illuminate\Database\QueryException $e) {
80         Log::error('AddRecipeController: Database error occurred', [
81             'error_message' => $e->getMessage(),
82             'trace' => $e->getTraceAsString()
83         ]);
84         return response()->json(['error' => 'Database error occurred'], 500);
85     } catch (\Exception $e) {
86         Log::error('AddRecipeController: Unexpected error occurred', [
87             'error_message' => $e->getMessage(),
88             'trace' => $e->getTraceAsString()
89         ]);
90         return response()->json(['error' => 'An unexpected error occurred'], 500);
91     }
92 }
```

RECIPE MODEL

```
app > Models > Recipe.php > ...
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Recipe extends Model
9  {
10     use HasFactory;
11
12     protected $fillable = [
13         'name',
14         'description',
15         'user_id',
16         'image',
17         'instructions',
18         'ingredients',
19         'ratings',
20         'categories',
21         'status'
22     ];
23     public function usersFavorited()
24     {
25         return $this->belongsToMany(User::class, 'favorites', 'recipe_id', 'user_id');
26     }
27     public function user()
28     {
29         return $this->belongsTo(User::class);
30     }
31 }
```

RECIPE BLADE VIEW

```

resources/views/add-recipe.blade.php
1 <!-- Header -->
2 <!-- Components/Layout2 -->
3 <!-- Section('title', 'Add Recipe') -->
4 <!-- Section('additional-styles') -->
5   <link rel="stylesheet" href="/assets/css/add-recipe.css">
6 </head>
7 <body>
8 <div class="app">
9   <div>
10    <div name="csrf-token" content="abc123xyz">
11    </div>
12    <div class="wrapper">
13      <div id="header-text">
14        <div class="text-add-recipe-text">
15          <a href="#">ADD RECIPE</a>
16          <p>Uploading personal recipes is easy! Add yours to your favorites, share with friends, family, or the PIXXX Recipe community.<!--/p--&gt;
17        &lt;/div&gt;
18      &lt;/div&gt;
19    &lt;/div&gt;
20  &lt;/div&gt;
21 &lt;div id="recipe-form"&gt;
22   &lt;form id="add-recipe" method="POST" action="{{ route('my-recipe') }}&gt;
23     &lt;input type="hidden" name="csrf-token" value="abc123xyz"&gt;
24     &lt;div class="form-container"&gt;
25       &lt;div class="left-section"&gt;
26         &lt;div class="group"&gt;
27           &lt;label for="recipetitle"&gt;string Recipe Title&lt;/label&gt;
28           &lt;input type="text" id="recipetitle" placeholder="Give your recipe a title" required&gt;
29         &lt;/div&gt;
30         &lt;div class="group"&gt;
31           &lt;label for="description"&gt;string Description&lt;/label&gt;
32           &lt;input type="text" id="description" name="description" placeholder="Share the story behind your recipe and what makes it special" required&gt;
33         &lt;/div&gt;
34       &lt;/div&gt;
35       &lt;div class="right-section"&gt;
36         &lt;div&gt;
37           &lt;label for="photo"&gt;strong Photo&lt;/label&gt;
38           &lt;input type="file" id="photo" name="photo" accept="image/png, image/jpeg"&gt;
39           &lt;div class="photo-preview"&gt;
40             &lt;img alt="Your photo here" src="/assets/images/adobo.jpg"/&gt;
41             &lt;p&gt;Use JPEG or PNG. Must be at least 960 x 960. Max file size: 30MB&lt;/p&gt;
42           &lt;/div&gt;
43         &lt;/div&gt;
44       &lt;/div&gt;
45     &lt;/div&gt;
46     &lt;div class="category-selector"&gt;
47       &lt;label for="category"&gt;Select a category:&lt;/label&gt;
48       &lt;select id="category" name="category"&gt;
49         &lt;option value="appetizers"&gt;Appetizers&lt;/option&gt;
50         &lt;option value="soups"&gt;Soups&lt;/option&gt;
51         &lt;option value="main-courses"&gt;Main Courses&lt;/option&gt;
52         &lt;option value="side-dishes"&gt;Side Dishes&lt;/option&gt;
53         &lt;option value="desserts"&gt;Desserts&lt;/option&gt;
54         &lt;option value="beverages"&gt;Beverages&lt;/option&gt;
55         &lt;option value="breakfast"&gt;Breakfast&lt;/option&gt;
56       &lt;/select&gt;
57     &lt;/div&gt;
58     &lt;div class="ingredients-section"&gt;
59       &lt;h3&gt;Ingredients&lt;/h3&gt;
60       &lt;p&gt;Enter one ingredient per line, include the quantity (i.e., tablespoons, cloves) and any special preparation (i.e., minced, crushed, etc.)&lt;/p&gt;
61       &lt;div class="ingredient-input"&gt;
62         &lt;div class="ingredient-entry"&gt;
63           &lt;input type="text" id="ingredient" type="text" placeholder="e.g. 1/2 cup soy sauce" class="wsyoyg"&gt;&lt;textarea&gt;
64         &lt;/div&gt;
65       &lt;/div&gt;
66     &lt;/div&gt;
67     &lt;div class="instructions-section"&gt;
68       &lt;h3&gt;Instructions&lt;/h3&gt;
69       &lt;p&gt;Enter step-by-step instructions for the recipe. Start with any prep work (i.e., marinating, chopping) and move through the cooking steps.&lt;/p&gt;
70       &lt;div class="instruction-inputs"&gt;
71         &lt;div class="instruction-entry"&gt;
72           &lt;input type="text" id="instruction" type="text" placeholder="Step 1 description...." class="wsyoyg"&gt;&lt;textarea&gt;
73         &lt;/div&gt;
74       &lt;/div&gt;
75     &lt;/div&gt;
76     &lt;div class="submit-button"&gt;
77       &lt;button type="submit" class="btn"&gt;SUBMIT RECIPE&lt;/button&gt;
78     &lt;/div&gt;
79   &lt;/form&gt;
80 &lt;/div&gt;
81 &lt;/body&gt;
82 &lt;/html&gt;
</pre>

```

```

<!-- Right Column for Photo Upload -->
<div class="right-section">
  <label for="photo">strong Photo</label>
  <input type="file" id="photo" name="photo" accept="image/png, image/jpeg">
  <div class="photo-preview">
    
    <p>Use JPEG or PNG. Must be at least 960 x 960. Max file size: 30MB</p>
  </div>
</div>

<!-- Select Category -->
<div class="category-selector">
  <label for="category">Select a category:</label>
  <select id="category" name="category">
    <option value="appetizers">Appetizers</option>
    <option value="soups">Soups</option>
    <option value="main-courses">Main Courses</option>
    <option value="side-dishes">Side Dishes</option>
    <option value="desserts">Desserts</option>
    <option value="beverages">Beverages</option>
    <option value="breakfast">Breakfast</option>
  </select>
</div>

<!-- Ingredients Input -->
<div class="ingredients-section">
  <h3>Ingredients</h3>
  <p>Enter one ingredient per line, include the quantity (i.e., tablespoons, cloves) and any special preparation (i.e., minced, crushed, etc.)</p>
  <div class="ingredient-input">
    <div class="ingredient-entry">
      <input type="text" id="ingredient" type="text" placeholder="e.g. 1/2 cup soy sauce" class="wsyoyg"><textarea>
    </div>
  </div>
</div>

<!-- Instructions Input -->
<div class="instructions-section">
  <h3>Instructions</h3>
  <p>Enter step-by-step instructions for the recipe. Start with any prep work (i.e., marinating, chopping) and move through the cooking steps.</p>
  <div class="instruction-inputs">
    <div class="instruction-entry">
      <input type="text" id="instruction" type="text" placeholder="Step 1 description...." class="wsyoyg"><textarea>
    </div>
  </div>
</div>

<!-- Submit Button -->
<div class="submit-button">
  <button type="submit" class="btn">SUBMIT RECIPE</button>
</div>

```

```

// Initialize CKEditor for instructions
CKEDITOR.create(document.querySelector("#instruction"), {
  toolbar: [
    "heading",
    "|",
    "bold",
    "italic",
    "|",
    "bulletedlist",
    "numberedlist",
    "|",
    "blockquote",
    "outdent",
    "indent",
    "|",
    "undo",
    "redo"
  ],
  then(editor) => {
    editorInstances.instructions = editor; // Save the editor instance
    editor.editing.view.change(writer) => {
      writer.setStyle("height", "30px", editor.editing.view.document.getRoot());
      editor.ui.view.element.style.width = "900";
    });
  },
  catch(error) => {
    console.error("Error initializing instructions editor:", error);
  }
});

// Handle add recipe form submission via AJAX
const addRecipeForm = document.querySelector("#add-recipe");
addRecipeForm.addEventListener("submit", async (event) => {
  event.preventDefault();
  const formData = new FormData(addRecipeForm); // Collect form data
  // Append CKEditor data to the form
  if (editorInstances.ingredients) {
    formData.append("ingredients", editorInstances.ingredients.getData());
  }
  if (editorInstances.instructions) {
    formData.append("instructions", editorInstances.instructions.getData());
  }

  try {
    console.log("Form data before submission:", formData);

    const response = await fetch(addRecipeForm.action, {
      method: "POST",
      headers: {
        "X-CSRF-TOKEN": document.querySelector('meta[name="csrf-token"]').content,
      },
      body: formData,
    });

    const result = await response.json();

    if (result.ok) {
      alert("Recipe added successfully!");
      console.log(result); // Optional: Handle the result as needed
      addRecipeForm.reset(); // Clear the form after submission
    } else {
      alert("Error adding recipe.");
      console.error(result.errors); // Log validation errors
    }
  } catch (error) {
    console.error("Error submitting form:", error);
    alert("An unexpected error occurred.");
  }
});

```

```

// Initialize CKEditor for ingredients
CKEDITOR.create(document.querySelector("#ingredient"), {
  toolbar: [
    "heading",
    "|",
    "bold",
    "italic",
    "|",
    "bulletedlist",
    "numberedlist",
    "|",
    "blockquote",
    "outdent",
    "indent",
    "|",
    "undo",
    "redo"
  ],
  then(editor) => {
    editorInstances.ingredients = editor; // Save the editor instance
    editor.editing.view.change(writer) => {
      writer.setStyle("height", "30px", editor.editing.view.document.getRoot());
      editor.ui.view.element.style.width = "900";
    });
  },
  catch(error) => {
    console.error("Error initializing ingredients editor:", error);
  }
});

// Handle add recipe form submission via AJAX
const addRecipeForm = document.querySelector("#add-recipe");
addRecipeForm.addEventListener("submit", async (event) => {
  event.preventDefault();
  const formData = new FormData(addRecipeForm); // Collect form data
  // Append CKEditor data to the form
  if (editorInstances.ingredients) {
    formData.append("ingredients", editorInstances.ingredients.getData());
  }
  if (editorInstances.instructions) {
    formData.append("instructions", editorInstances.instructions.getData());
  }

  try {
    console.log("Form data before submission:", formData);

    const response = await fetch(addRecipeForm.action, {
      method: "POST",
      headers: {
        "X-CSRF-TOKEN": document.querySelector('meta[name="csrf-token"]').content,
      },
      body: formData,
    });

    const result = await response.json();

    if (result.ok) {
      alert("Recipe added successfully!");
      console.log(result); // Optional: Handle the result as needed
      addRecipeForm.reset(); // Clear the form after submission
    } else {
      alert("Error adding recipe.");
      console.error(result.errors); // Log validation errors
    }
  } catch (error) {
    console.error("Error submitting form:", error);
    alert("An unexpected error occurred.");
  }
});

```

```

// Handle logout form submission
const logoutForm = document.querySelector("#logout-form");
const logoutButton = document.querySelector("a[href='{{ route('logout') }}']");
logoutButton.addEventListener("click", event) => {
  event.preventDefault(); // Prevent default logout navigation
  logoutForm.submit(); // Trigger the logout form submission manually
};

</script>

```

RECIPE ROUTES

```

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');

```

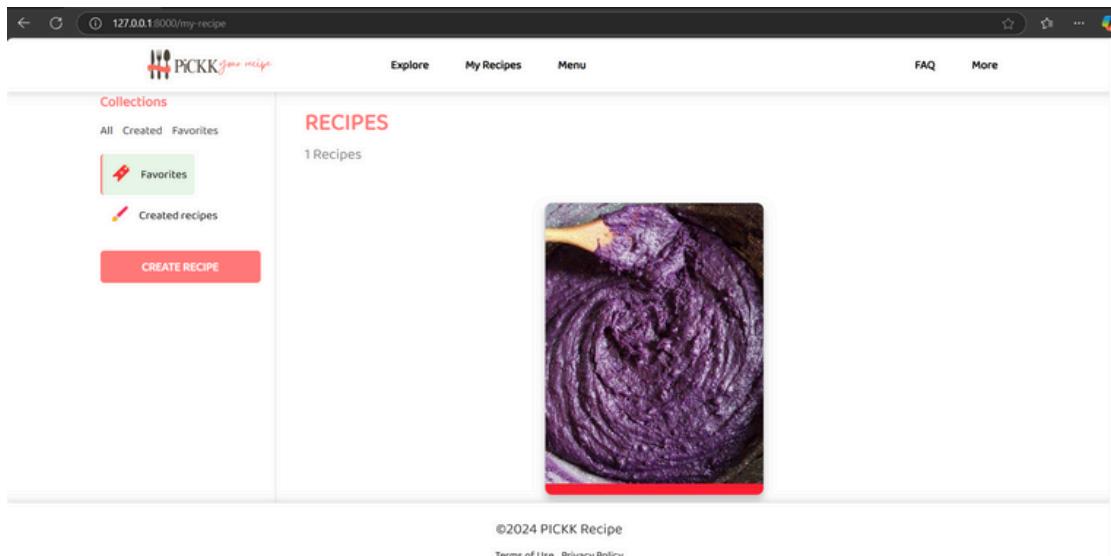
EXPLANATION:

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:

- Model:** Created/used a model to define the structure and interactions with the database, such as retrieving posts, comments, or pictures.
- Controller:** Set up a controller to fetch the required data from the model and prepare it for the view.
- 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.

POST TO CREATED RECIPES



COMMENT

COMMENT VIEW

Add a New Comment

Comment

I LOVEEE ITTT!!!!

Post Comment

DATABASE TABLE

A screenshot of a database management tool showing a table named 'pickk_recipe.comments'. The table has columns: id, recipe_id, user_id, comment, created_at, and updated_at. There are two rows of data. A sidebar on the left shows the database structure with 'pickk_recipe' as the schema containing 'New', 'cache', 'cache_locks', and 'comments' tables, along with 'failed_inhs' and 'tmp' tables.

COMMENT POST

Add a New Comment

Comment

Type your comment

Post Comment

kenii
I LOVEEE ITTT!!!!

COMMENT BLADE VIEWS

```
<section id="comment-section">
    @foreach ($comments as $comment)
        <div class="comment">
            <div class="profile-pic">
                
            </div>
            <div class="comment-content">
                <p class="username">{{ $comment->username }}</p>
                <p class="text">{{ $comment->comment }}</p>
            </div>
        </div>
    @endforeach
</section>
```

COMMENT CONTROLLER

```
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', $recipeId)->exists();

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

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 > ...
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Comment extends Model
9  {
10     use HasFactory;
11
12     // Add the fillable properties
13     protected $fillable = ['recipe_id', 'user_id', 'comment'];
14 }
```

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.