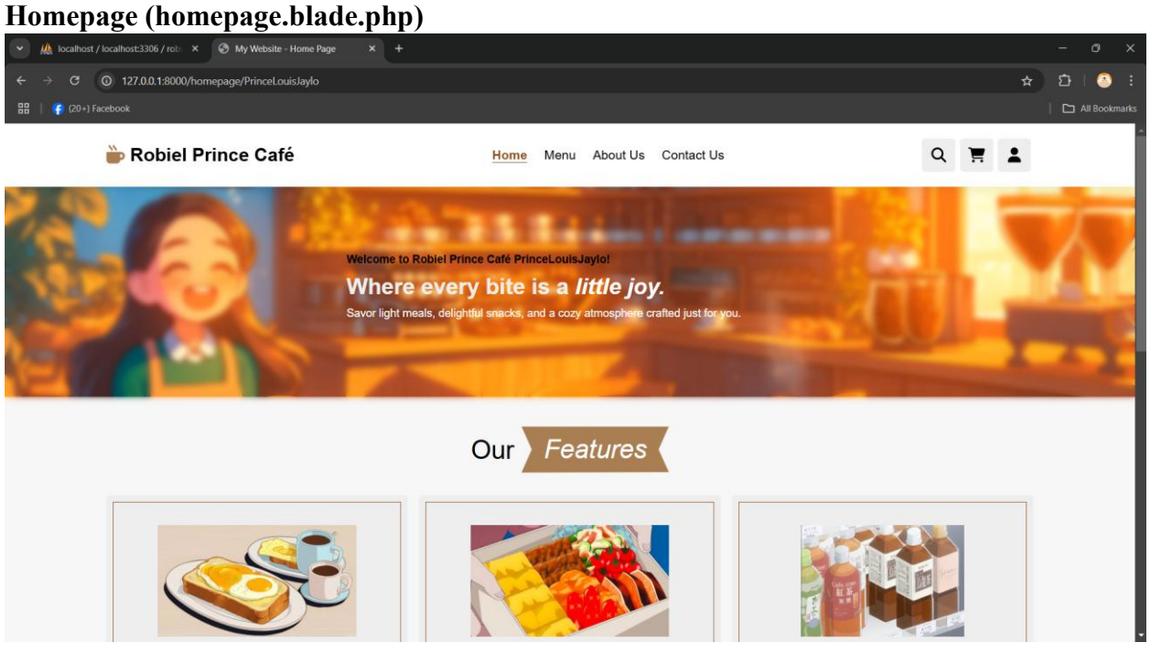
Prince Louis M. Jaylo **BSIT 3-C** **IT Elec 1 – Web Development**

Documentation: Robiel Prince Café Website using “Populating from a Database”

This Lab activity is a Laravel-based web application for a Family café business named RobielPrince Café proposed by my mother and father. So, I thought of making a website for it in the idea that someday it will help out the business. It features a homepage, a menu page, and a login system. The web application dynamically fetches and displays data from a database.

1. Screenshots of Rendered Pages

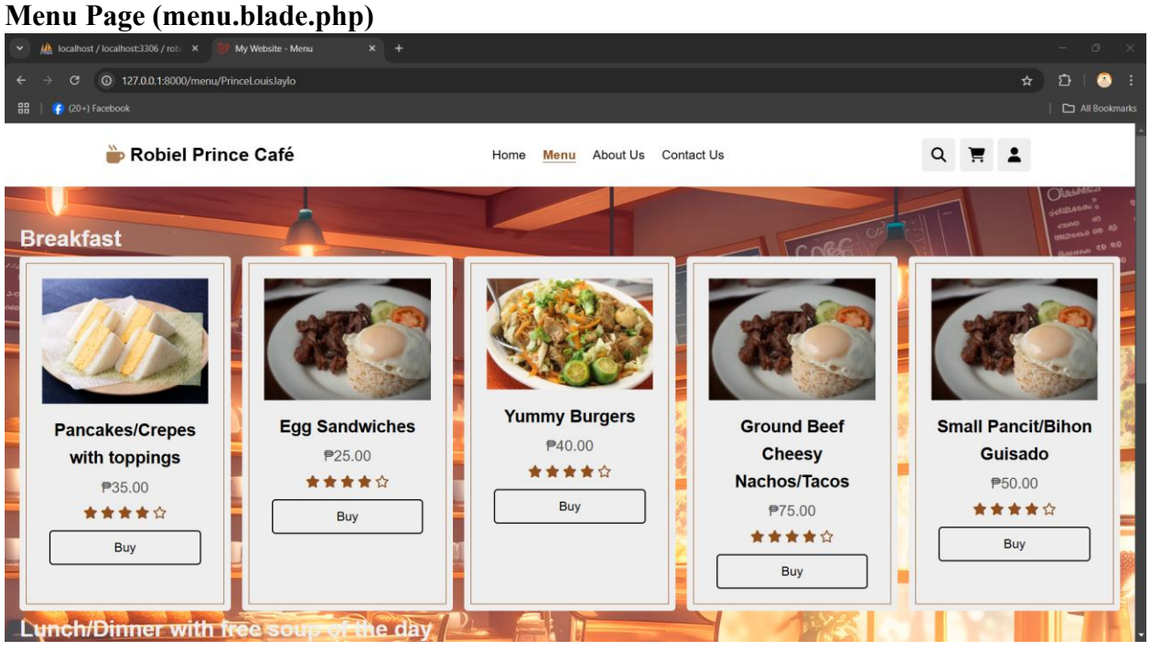
Homepage (homepage.blade.php)



The homepage greets the user with a welcome message, shows the café's features (e.g., breakfast, lunch, drinks), and lists "Best Sellers" with images, prices, and ratings.

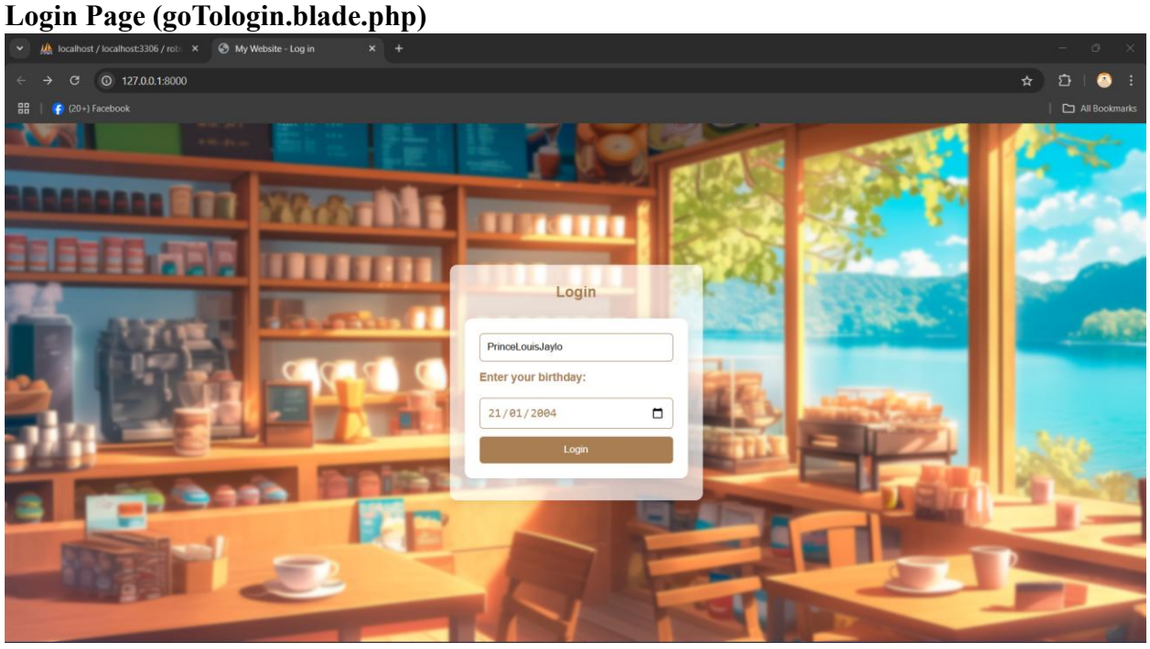
The welcome message with the user's name (Welcome to Robiel Prince Café Guest! or the users name e.g., PrinceLouisJaylo). The features section with images and descriptions (e.g., "Morning Delights"). The Best Sellers section with product names, prices, and ratings.

Menu Page (menu.blade.php)



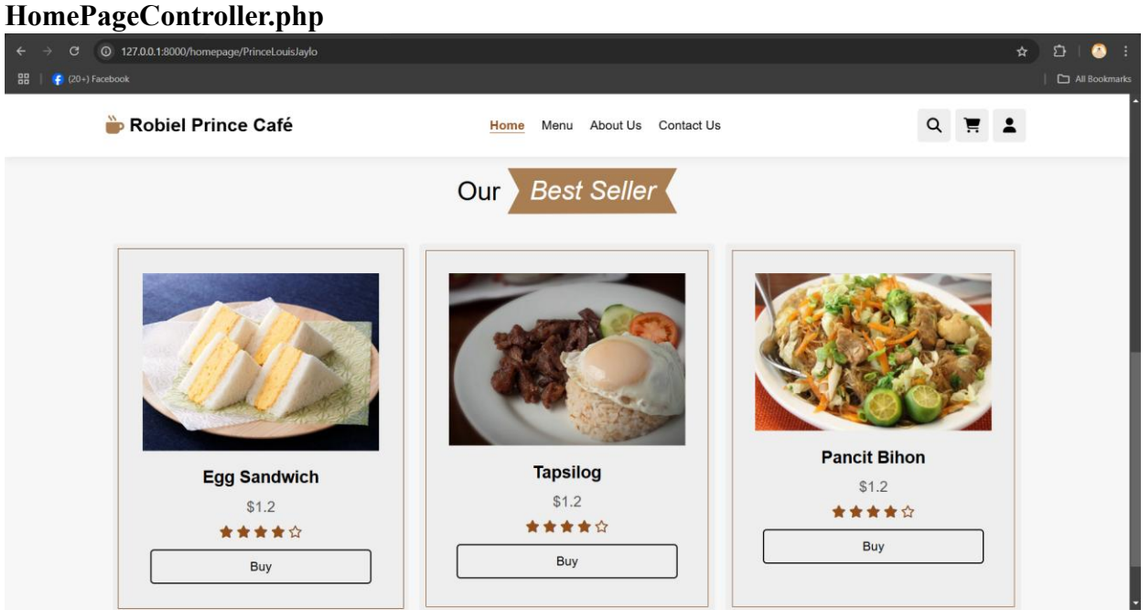
The menu page displays a list of food items dynamically fetched from the database. Each item includes a name, price, image, and rating.

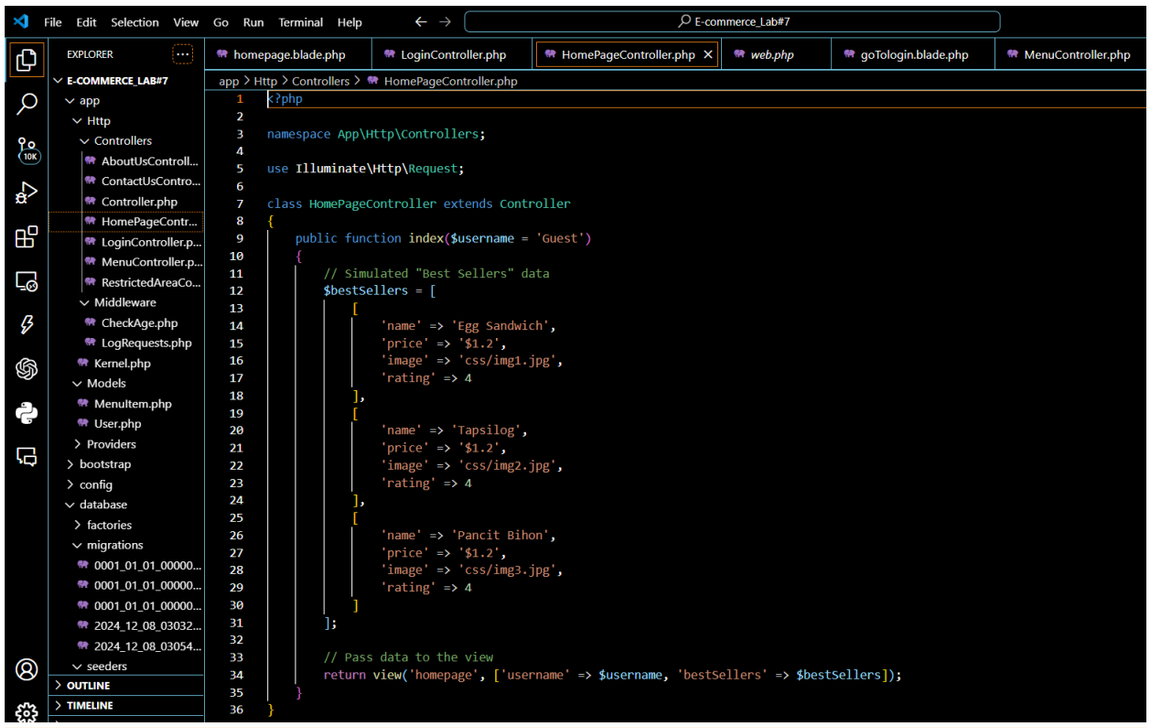
Login Page (goTologin.blade.php)



The login page includes a form where the user can input their name and birthday to log in. Validation errors are displayed if invalid data is submitted. We can see in the **login form** with fields for name and birthday. A submission button "Login". Optionally, a of the **message** when invalid data is entered for example "Please fill out this field" or "Please match the requested format".

2. Explanations for Each Page
HomePageController.php



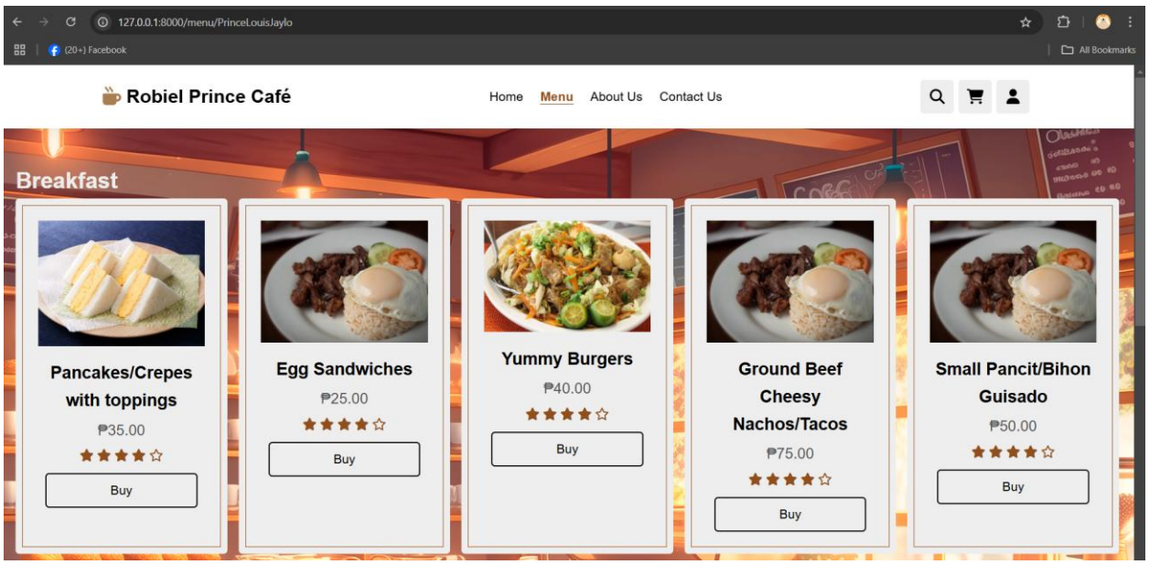



```
Route::get('/homepage/{username}', [HomePageController::class, 'index'])->name('homepage');
```

The HomePageController handles the homepage logic. It simulates "best sellers" data and passes it to the view. The $username parameter defaults to 'Guest' if no username is provided. The $username parameter is retrieved from the route and passed to the view. It allows the page to personalize greetings dynamically.

MenuController.php

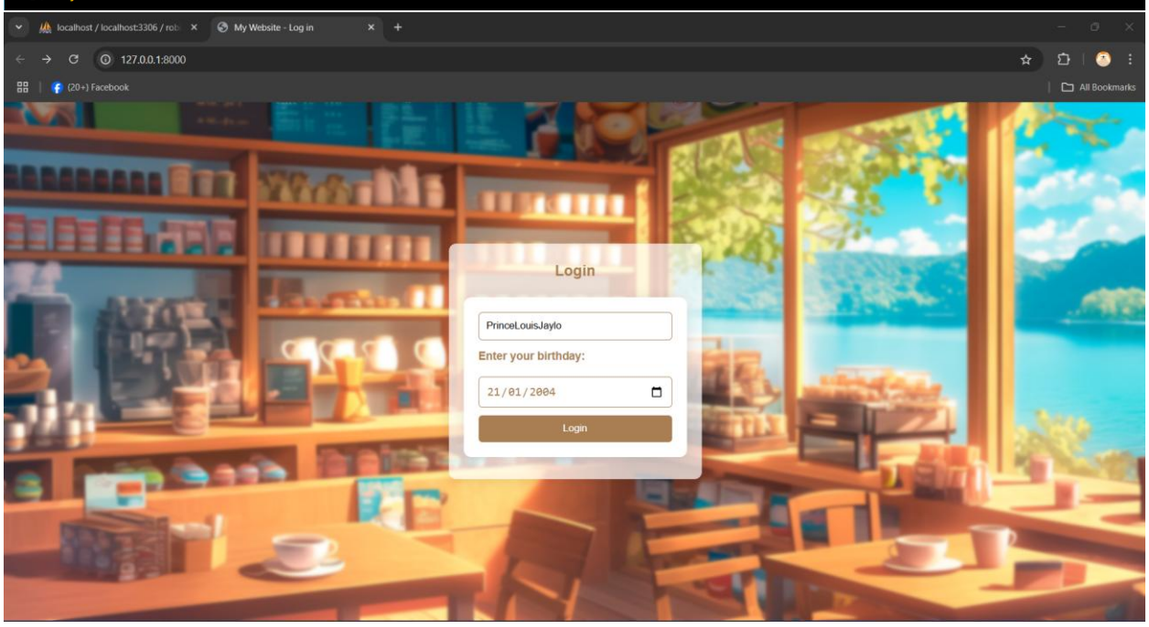
```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class MenuController extends Controller
{
    public function index($username)
    {
        // Simulated menu data
        $menuItems = [
            [
                'name' => 'Pancakes/Crepes with toppings',
                'price' => '₱35.00',
                'image' => 'css/img1.jpg',
                'rating' => 4
            ],
            [
                'name' => 'Egg Sandwiches',
                'price' => '₱25.00',
                'image' => 'css/img2.jpg',
                'rating' => 4
            ],
            [
                'name' => 'Yummy Burgers',
                'price' => '₱40.00',
                'image' => 'css/img3.jpg',
                'rating' => 4
            ],
            [
                'name' => 'Ground Beef Cheesy Nachos/Tacos',
                'price' => '₱75.00',
                'image' => 'css/img2.jpg',
                'rating' => 4
            ],
            [
                'rating' => 4
            ],
            [
                'name' => 'Small Pancit/Bihon Guisado',
                'price' => '₱50.00',
                'image' => 'css/img2.jpg',
                'rating' => 4
            ],
        ];

        // Pass data to the view
        return view('menu', ['username' => $username, 'menuItems' => $menuItems]);
    }
}
```



The MenuController fetches the menu items from the menu_items database table using the MenuItem model and passes the data to the view. The $username parameter is retrieved from the route and used to personalize the menu page.

LoginController.php

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class LoginController extends Controller
{
    public function goTologin($userId = null)
    {
        $username = 'Guest';

        // If a user ID is provided, fetch or customize data for that user
        if ($userId) {
            // Placeholder logic; in a real app, fetch user info from a database.
            $username = "User{$userId}";
        }

        return view('goTologin', ['username' => $username])->with('hideNavAndFooter', true);
    }

    public function submit(Request $request)
    {
        $request->validate([
            'username' => 'required|alpha'
        ], [
            'username.alpha' => 'The username should only contain alphabetic characters.',
            'username.required' => 'The username field is required.'
        ]);

        $username = $request->input('username', 'Guest');
        return redirect("/homepage/$username");
    }
}
```



The LoginController displays the login form. Upon submission, it validates the username and redirects to the homepage with the provided username. The login form takes user input (username) and validates it. The username is passed as a route parameter to personalize the homepage.

.env file

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=robielprincecafe
DB_USERNAME=root
DB_PASSWORD=
```

This is where I connected the data base of the website using XAMPP

Models/MenuItem.php

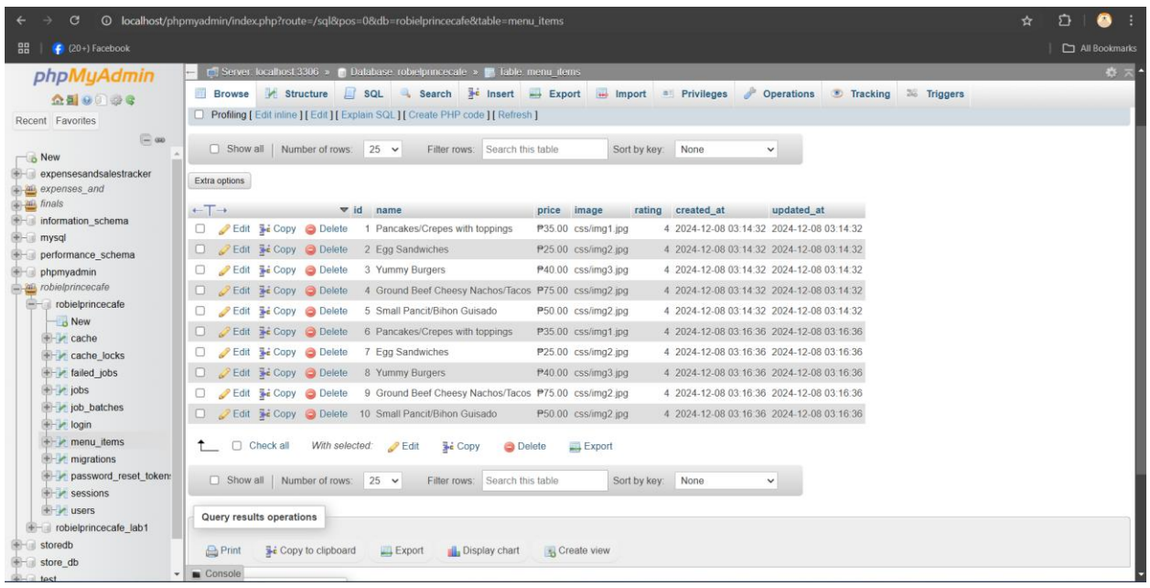
```php
<?php

namespace App\Models;

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

class MenuItem extends Model
{
    use HasFactory;
}
```

This MenuItem file is a Laravel model. A model represents a table in the database and allows you to interact with it. For example: It will be used to fetch or save menu items (like dishes) from/to the database. When the MenuController needs menu data, it can use this model to get it easily. The HasFactory part helps create fake menu items for testing or adding sample data during development.

Migration Files

Menu:

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up()
    {
        Schema::create('menu_items', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('price');
            $table->string('image');
            $table->integer('rating');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('menu_items');
    }
};
```



This migration creates a menu_items table in the database with columns for the item's id, name, price, image, rating, and timestamps (created_at and updated_at). The up() method builds the table, while the down() method removes it if needed, ensuring the database structure aligns with the application requirements.