Answer To the Question of assignment-23

1. App Feature: X-Shop Customer Management and Promotional Email Syste

Description:

The X-Shop application is designed to provide shop owners with a user-friendly platform to manage their customer list and send promotional emails to their customers. The app allows for easy CRUD (Create, Read, Update, Delete) operations on customer data and streamlines the process of sending

2. Creating an Income Expense Tracker

You are tasked with developing an Income Expense Tracker web application using Laravel to help users manage their finances effectively. The application will allow users to track their income and expenses, view financial summaries, and perform basic financial calculations. The assignment involves implementing various Laravel concepts, including database migrations, Eloquent ORM, form validation, and data manipulation.

Requirements:

Database Setup:

Create a new database and configure the database connection in the .env file.

Design and implement the necessary database tables for the application. You need at least two tables, one for income records and one for expense records. Include relevant fields such as amount, description, date, and category.

Models and Relationships:

Create two models: Income and Expense, representing the income and expense records, respectively.

Define the relationships between the Income and Expense models and the user model. Each income and expense record should belong to a specific user.

User Authentication:

Implement user registration and login functionalities using Laravel's built-in authentication system.

Restrict access to the income and expense tracking features to authenticated users only.

Income and Expense Creation:

Provide a form for users to add new income and expense records.

Implement server-side validation to ensure the amount, description, and date fields are required and correctly formatted.

Displaying Records:

Create separate views to display the list of income and expense records for the logged-in user.

Implement pagination to display a limited number of records per page.

Financial Calculations:

Calculate and display the total income and total expenses for the user.

Calculate and display the net income (income - expenses) for the user.

Filtering and Sorting:

Allow users to filter income and expense records based on categories or date ranges.

Provide options for sorting records by date or amount in ascending or descending order.

Data Manipulation:

Implement the ability to edit and delete income and expense records.

Styling and User Experience:

Apply CSS styling to ensure a user-friendly and visually appealing interface.

Make the application responsive to different screen sizes.

Step 1: Database Setup

Configure the .env file with your database credentials

Step 2: Design and Implement Database Tables

Create migration files for the Income and Expense tables:

```
php artisan make:migration create_incomes_table

php artisan make:migration create_expenses_table
```

Edit the generated migration files:

```
// database/migrations/2023_08_01_create_incomes_table.php

public function up()

{

    Schema::create('incomes', function (Blueprint $table) {

        $table->id();

        $table->unsignedBigInteger('user_id');

        $table->decimal('amount', 10, 2);

        $table->string('description');

        $table->date('date');

        $table->timestamps();

    });

}
```

Run migrations:

```
php artisan migrate
```

Step 3: Models and Relationships

Create models for Income and Expense:

```
php artisan make:model Income

php artisan make:model Expense
```

Define relationships:

```
// app/Models/Income.php
public function user()
{
    return $this->belongsTo(User::class);
}
```

Step 4: User Authentication

Set up user authentication using Laravel Jetstream:

```
composer require laravel/jetstream
php artisan jetstream:install livewire
composer require laravel/jetstream
php artisan jetstream:install livewire
```

Step 5: Income and Expense Creation

Create forms for adding income and expense records:

```
<!-- resources/views/incomes/create.blade.php -->
<form action="{{ route('incomes.store') }}" method="POST">
    @csrf
    <input type="text" name="amount" placeholder="Amount">
    <input type="text" name="description" placeholder="Description">
    <input type="date" name="date">
    <button type="submit">Add Income</button>
</form>
```

Implement server-side validation:

```
// app/Http/Controllers/IncomeController.php
public function store(Request $request)
{
    $validatedData = $request->validate([
        'amount' => 'required|numeric',
        'description' => 'required|string',
        'date' => 'required|date',
    ]);

    $income = new Income($validatedData);
    $income->user_id = auth()->user()->id;
    $income->save();

    return redirect()->route('incomes.index');
}
```

Step 6: Displaying Records

Create a view to display income records:

```
<!-- resources/views/incomes/index.blade.php -->
@foreach ($incomes as $income)
    {{ $income->amount }} - {{ $income->description }} - {{ $income->date }}
@endforeach

{{ $incomes->links() }}
```

Step 7: Financial Calculations

Calculate total income, expenses, and net income

```
// In your controller method
$totalIncome = auth()->user()->incomes()->sum('amount');
$totalExpenses = auth()->user()->expenses()->sum('amount');
$netIncome = $totalIncome - $totalExpenses;
```

Step 8: Filtering and Sorting

Implementing filtering and sorting allows users to customize how they view their income and expense records.

Filtering Example:

Let's say you want to filter records based on categories. Add a dropdown to your view where users can select a category:

```
<!-- resources/views/incomes/index.blade.php -->

<form action="{{ route('incomes.index') }}" method="GET">

  <select name="category">

    <option value="salary">Salary</option>

    <option value="bonus">Bonus</option>

    <!-- Other category options -->

  </select>

  <button type="submit">Filter</button>

</form>
```

In your controller, use the selected category to filter the records:

```
// In your controller method

$filteredIncomes = auth()->user()->incomes();


if ($request->has('category')) {

  $filteredIncomes->where('category', $request->category);

}


$incomes = $filteredIncomes->paginate(10);
```

Sorting Example:

Let's sort records by date and amount. Add buttons to your view for sorting:

```
<!-- resources/views/incomes/index.blade.php -->

<a href="{{ route('incomes.index', ['sort' => 'date_asc']) }}">Sort by Date Ascending</a>

<a href="{{ route('incomes.index', ['sort' => 'date_desc']) }}">Sort by Date Descending</a>

<a href="{{ route('incomes.index', ['sort' => 'amount_asc']) }}">Sort by Amount Ascending</a>

<a href="{{ route('incomes.index', ['sort' => 'amount_desc']) }}">Sort by Amount Descending</a>
```

In your controller, handle the sorting:

```
// In your controller method
$sort = $request->get('sort');

if ($sort === 'date_asc') {
    $incomes = $incomes->orderBy('date', 'asc');
} elseif ($sort === 'date_desc') {
    $incomes = $incomes->orderBy('date', 'desc');
} elseif ($sort === 'amount_asc') {
    $incomes = $incomes->orderBy('amount', 'asc');
} elseif ($sort === 'amount_desc') {
    $incomes = $incomes->orderBy('amount', 'desc');
}

$incomes = $incomes->paginate(10);
```

Step 9: Data Manipulation

Implementing editing and deleting capabilities for income and expense records.

Editing Example:

Create an edit form for income records:

```
<!-- resources/views/incomes/edit.blade.php -->
<form action="{{ route('incomes.update', $income) }}" method="POST">
    @csrf
    @method('PUT')
    <input type="text" name="amount" value="{{ $income->amount }}">
    <input type="text" name="description" value="{{ $income->description }}">
    <input type="date" name="date" value="{{ $income->date }}">
    <button type="submit">Update Income</button>
</form>
// In your controller method
```

```
$income->update($validatedData);
return redirect()->route('incomes.index');
```

Deleting Example:

Add a delete button to each record in the view:

```
<!-- resources/views/incomes/index.blade.php -->
@foreach ($incomes as $income)
   <!-- Display income details -->
   <form action="{{ route('incomes.destroy', $income) }}" method="POST">
       @csrf
       @method('DELETE')
       <button type="submit">Delete</button>
   </form>
@endforeach

// In your controller method
$income->delete();
return redirect()->route('incomes.index');
```

Step 10: Styling and User Experience

Apply CSS styling to create an attractive and responsive user interface. You can use CSS frameworks like Bootstrap or Tailwind CSS to style your application. For example, using Bootstrap:

Install Bootstrap via CDN or npm.

Apply classes to your HTML elements to style them.

Make use of responsive design classes to ensure your application works well on various screen sizes.

For example, adding Bootstrap classes to a form element:

```
<form action="{{ route('incomes.store') }}" method="POST" class="form-inline">
   <!-- Form inputs here -->
```

```
    <button type="submit" class="btn btn-primary">Add Income</button>
</form>
```

Income and Expense Creation - Route

In your routes/web.php file, define routes for creating new income and expense records:

```
use App\Http\Controllers\IncomeController;
use App\Http\Controllers\ExpenseController;

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

    // Route for showing the form to add new income
    Route::get('/incomes/create', [IncomeController::class, 'create'])->name('incomes.create');
    // Route for storing new income
    Route::post('/incomes', [IncomeController::class, 'store'])->name('incomes.store');

    // Route for showing the form to add new expense
    Route::get('/expenses/create', [ExpenseController::class, 'create'])->name('expenses.create');
    // Route for storing new expense
    Route::post('/expenses', [ExpenseController::class, 'store'])->name('expenses.store');

    // ...
});
```

Displaying Records - Route

Define a route to display the list of income and expense records:

```
Route::middleware(['auth'])->group(function () {
    // ...
    // Route for displaying income records
    Route::get('/incomes', [IncomeController::class, 'index'])->name('incomes.index');
    // Route for displaying expense records
    Route::get('/expenses', [ExpenseController::class, 'index'])->name('expenses.index');

    // ...
});
```

Data Manipulation - Editing and Deleting - Routes

Define routes for editing and deleting income and expense records:

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

    // Route for showing the edit form for an income
    Route::get('/incomes/{income}/edit', [IncomeController::class, 'edit'])->name('incomes.edit');
    // Route for updating an income
    Route::put('/incomes/{income}', [IncomeController::class, 'update'])->name('incomes.update');
    // Route for deleting an income
    Route::delete('/incomes/{income}', [IncomeController::class, 'destroy'])->name('incomes.destroy');

    // Route for showing the edit form for an expense
    Route::get('/expenses/{expense}/edit', [ExpenseController::class, 'edit'])->name('expenses.edit');
    // Route for updating an expense
    Route::put('/expenses/{expense}', [ExpenseController::class, 'update'])->name('expenses.update');
    // Route for deleting an expense
    Route::delete('/expenses/{expense}', [ExpenseController::class, 'destroy'])->name('expenses.destroy');

    // ...
});
```

Filtering and Sorting - Route

If you're implementing filtering and sorting using URL parameters, you can update the existing routes to handle these parameters. For example, for filtering by category and sorting:

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


    // Route for displaying income records with filtering and sorting
    Route::get('/incomes', [IncomeController::class, 'index'])->name('incomes.index');


    // ...
});
```

Income and Expense Creation - Controller Methods

1. Create a new income record (store method)
   ```
   // app/Http/Controllers/IncomeController.php
   public function store(Request $request)
   {
       $validatedData = $request->validate([
           'amount' => 'required|numeric',
           'description' => 'required|string',
           'date' => 'required|date',
       ]);

       $income = new Income($validatedData);
       $income->user_id = auth()->user()->id;
       $income->save();

       return redirect()->route('incomes.index');
   }
   ```

   Displaying Records - Controller Methods

2. Display a list of income records (index method)
   ```
   // app/Http/Controllers/IncomeController.php
   public function index()
   {
   ```

```
    $incomes = auth()->user()->incomes()->paginate(10);
    return view('incomes.index', compact('incomes'));
}
```

Styling and User Experience - Apply Bootstrap

First, make sure you have included the Bootstrap CSS and JavaScript files in your layout:

```
<!-- resources/views/layouts/app.blade.php -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Income Expense Tracker</title>
    <!-- Include Bootstrap CSS -->
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
</head>
<body>
    <div class="container">
        @yield('content')
    </div>
    <!-- Include Bootstrap JS -->
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

1. Create/Edit Forms:

```
<!-- resources/views/incomes/create.blade.php -->
@extends('layouts.app')

@section('content')
    <div class="container">
        <h2 class="mt-5">Add New Income</h2>
        <form action="{{ route('incomes.store') }}" method="POST">
            @csrf
            <div class="mb-3">
                <label for="amount" class="form-label">Amount:</label>
                <input type="text" name="amount" class="form-control" required>
            </div>
            <!-- Other form fields -->
            <button type="submit" class="btn btn-primary">Add Income</button>
        </form>
    </div>
@endsection
```

2. Display Records:

```
<!-- resources/views/incomes/index.blade.php -->
@extends('layouts.app')

@section('content')
    <div class="container">
        <h2 class="mt-5">Income Records</h2>
        <table class="table">
            <thead>
                <tr>
                    <th>Amount</th>
                    <th>Description</th>
                    <th>Date</th>
                    <th>Action</th>
                </tr>
            </thead>
            <tbody>
                <!-- Loop through records -->
            </tbody>
        </table>
        {{ $incomes->links() }}
    </div>
@endsection
```

3. Edit Forms:

```
<!-- resources/views/incomes/edit.blade.php -->
@extends('layouts.app')

@section('content')
    <div class="container">
        <h2 class="mt-5">Edit Income Record</h2>
        <form action="{{ route('incomes.update', $income) }}" method="POST">
            @csrf
            @method('PUT')
            <div class="mb-3">
                <label for="amount" class="form-label">Amount:</label>
                <input type="text" name="amount" class="form-control" value="{{ $income-
>amount }}" required>
            </div>
            <!-- Other form fields -->
            <button type="submit" class="btn btn-primary">Update Income</button>
        </form>
    </div>
@endsection
```