Olivine Limited

Address: House# 310 (5th Floor), Road# 21, New D.O.H.S. Mohakhali, Dhaka-1206, Bangladesh

Name: MD AZIZUL ARIF

Mob: 01726058935; Gmail: azizularifnahim52@gmail.com

Assignment: Technical skill test

Abstract:

Laravel (MIT License) is one of the most prominent backend framework of PHP programming language. Long time development process is main turning point of Laravel framework. Apart from Laravel, some other PHP framework like CakePHP, Symfony, CodeIgniter, Yii are also notable framework. But Laravel is most powerful because of its strong architecture. Also it is noticeable that Laravel follow specific development process of MVC pattern. In this skill test assignment my main focus was to follow along with the task list. All the given task was completed with best possible solution. Validation process and email message process was tricky but fun. In my scope, I will try to explain my project with proper documentation, so that other developer and user can understand easily.

Keywords: ***Laravel, Registration form, form validation, dynamic mail, database.***

Introduction:

This assignment was done with Laravel framework (PHP), Bootstrap 4, HTML, CSS, mailtrap (mail server). Built in feature of laravel artisan was main focal point of faster development process. I will explain each section step by step throughout this documentation.

Project setup:

Project setup can be done with composer [1], which is available to download. It is a PHP package-manager for maintaining dependencies. It is often compare with node package-manager of Javascript. Xampp server is require for MySQL database and application running on server. Bootstrap 4 CDN can be integrate to application by copying link from Bootstrap 4 website [2]. Mail server can be found from mailtrap

website [3], which is very common tools to use mail checking. Node package manager setup was default and webpack.mix was not used for this project, as everything done from scratch.

System:

Hardware setup used is this assignment are down below:

- Intel-core i5-7200U processor laptop
- 8GB RAM
- 480GB SSD
- Windows 10-operating system
- XAMPP 7.4.27
- PHPmyadmin for MySQL database

Coding:

Let us look at some of the coding with explanation. Web.php file will be responsible for handling route from website. Here UserController imported from ‘controllers’ folder, where all the business logic was included. This is based on MVC structure, where controller set connection between database and view. Below is the screenshot for route.

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

Route::get('user', [UserController::class, 'create']);
Route::post('user/create', [UserController::class, 'store']);
```

Now that I have completed routing, let us look at Database design. Database design follow laravel built in setup. Here, user will be counted based on id and it will auto increment. Email and phone number must be unique, so that it does not match with other user.

```php
public function up()
{
    Schema::create('users', function (Blueprint $table) {
        $table->id();
        $table->string('name');
        $table->string('email')->unique();
        $table->string('phone')->unique();
        $table->timestamp('email_verified_at')->nullable();
        $table->string('password');
        $table->rememberToken();
        $table->timestamps();
    });
}
```

Database design needs to migrated, after mentioning which field needs to be ‘fillable’ at User.php file. At this point, artisan migration command can help this schema to push towards database. Now, UserController file needs to fill up with logic like below. Where data store functionality and validation logic was written. Below is screenshot for store function:

```php
public function store(Request $request)
{
    $request->validate(
        [
            'name'=>'required|string|max:20',
            'email'=>'required|string|unique:users,email',
            'phone'=>'required|numeric|digits:11',
            'password'=>'required|alpha_num|min:6',
            'confirm_password'=>'required|same:password'
        ]
    );

    $dataArray = array(
        "name"          =>          $request->name,
        "email"         =>          $request->email,
        "phone"         =>          $request->phone,
        "password"      =>          $request->password
    );

    $user = User::create($dataArray);
    Mail::to($dataArray['email'])->send(new WelcomeMail($user));
    if(!is_null($user)){
        return back()->with("success", "Success! Registration done, Check your mail");

    }
    else{
        return back()->with("failed", "Alert! Failed to register");
    }
```

Now that, some core feature is finished, let us see how to send message to registered user by mailtrap website. For this, built in queue data structure implementation from mail method was used. Below is the screenshot for that:

```php
class WelcomeMail extends Mailable
{
    use Queueable, SerializesModels;

    public $user;
    /**
     * Create a new message instance.
     *
     * @return void
     */
    public function __construct($user)
    {
        //
        $this->user = $user;
    }

    /**
     * Build the message.
     *
     * @return $this
     */
    public function build()
    {
        return $this->view('email.send');
    }
}
```

For the frontend part, which will be important for the user interaction. Let us see some coding of the view:

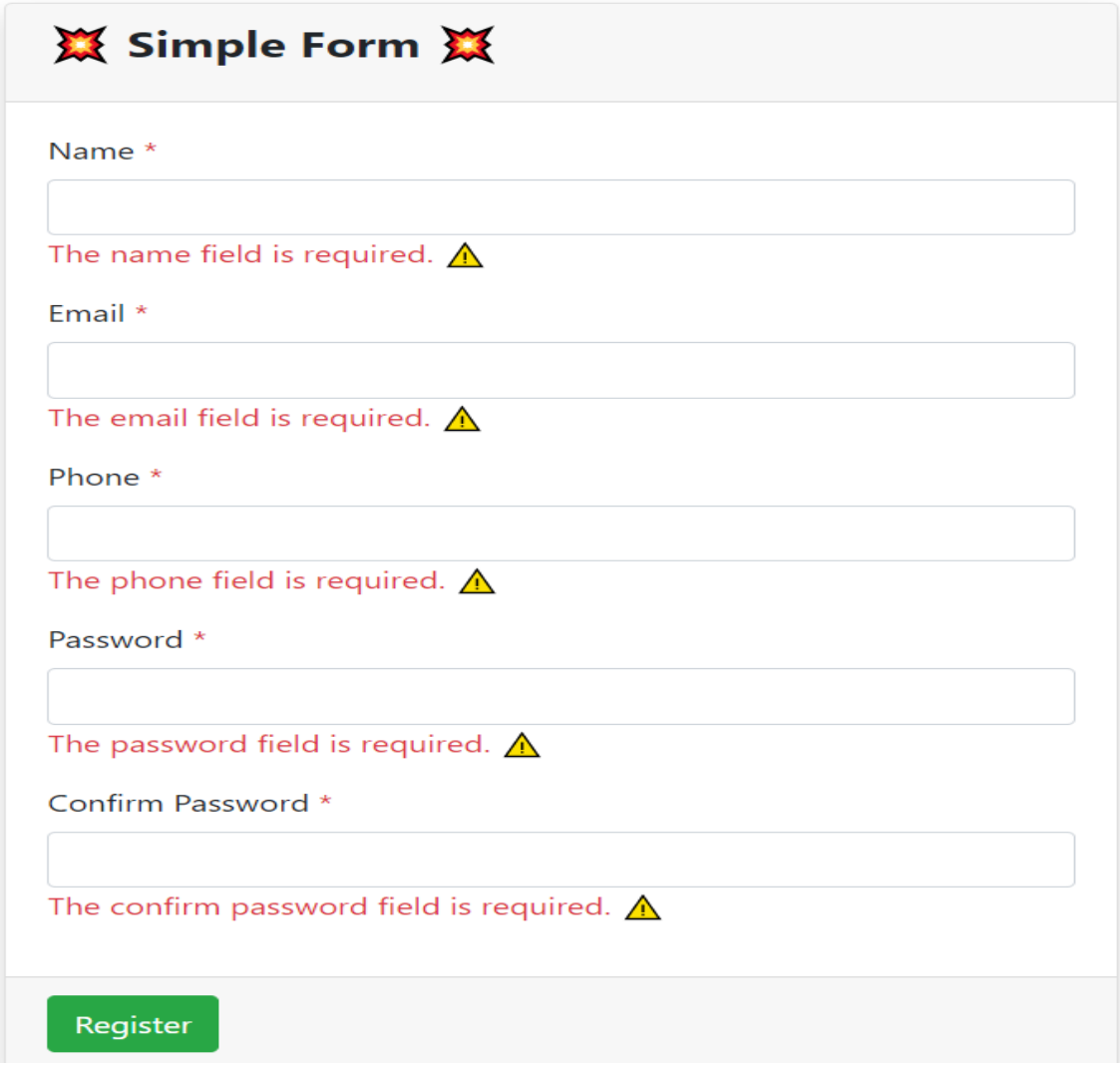
```blade
<!doctype html>
<html lang="en">
  <head>
    <title>Olivine Limited Form</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integ
  </head>
  <body>
    <div class="container mt-5">
        <form action="{{url('user/create')}}" method="POST">
            @csrf
            <div class="row">
                <div class="col-xl-6 m-auto">
                    <div class="card shadow">
                        @if(Session::has('success'))
                            <div class="alert alert-success alert-dismissible">
                                <button type="button" class="close" data-dismiss="alert">✅</button>
                                {{Session::get('success')}}
                            </div>
                        @elseif(Session::has('failed'))
                            <div class="alert alert-danger alert-dismissible">
                                <button type="button" class="close" data-dismiss="alert">❌</button>
                                {{Session::get('failed')}}
                            </div>
                        @endif
```

User documentation:

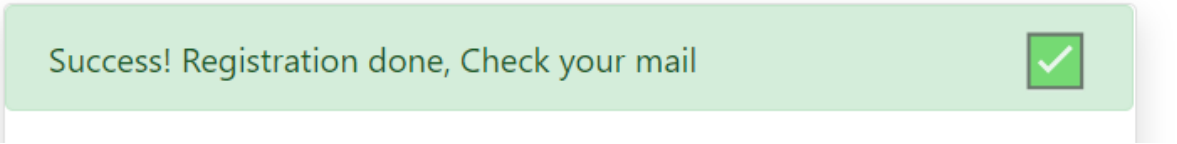
As the application is not hosted on live server, we can use it only in local server. For that, user can go to this link- 127.0.0.1. It will take user to Registration form. For user interaction user needs to follow some rules like below:

- All field from the form needs to be filled up
- Name can be maximum of 20 characters
- Email must contain @ sign
- Phone number must be 11 digit Bangladeshi number pattern
- Password must contain at least 6 characters
- Confirm password must be same as Password

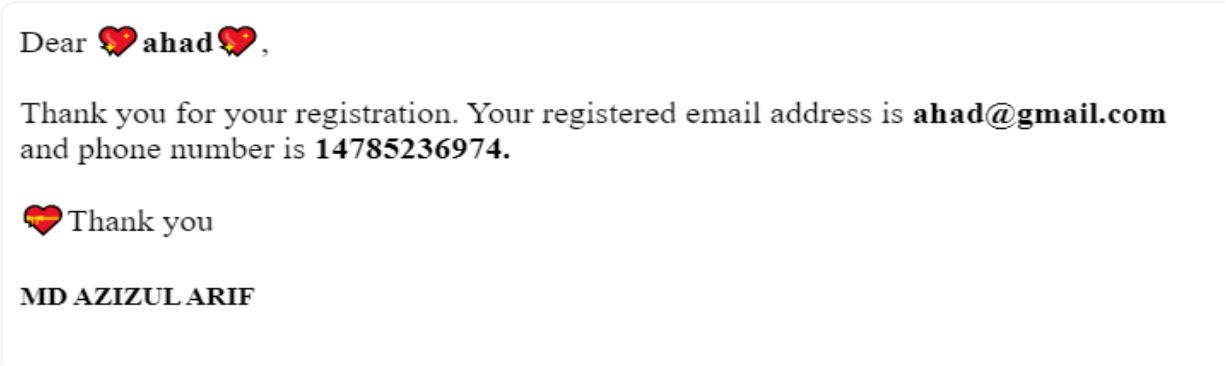
If user does not maintain this pattern, they will get error like this below:



If user follow the pattern that was set by programmer, they can easily register and will get message like below:



Then user will get email message containing their email and password, for further uses. So, let us see that the message from mail:



Hope, user will find this documentation section interesting and give feedback based on their point of view.

Conclusion:

Finally I want to say that, it was interesting and fun to work on this assignment. If programmer can enjoy their coding then, coding feels alive. I believe, I completed all the task that was assigned to me with my heart. So, I hope to get some positive feedback from Olivine Limited.

References:

[1]. Laravel documentation: https://laravel.com/docs/8.x

[2]. Bootstrap: https://getbootstrap.com/docs/4.0/getting-started/introduction/

[3]. Mailtrap: https://mailtrap.io/

[4]. Emoji: http://www.get-emoji.com/

[5]. Blog: https://raygun.com/blog/top-php-frameworks/

[6]. Github: https://github.com/Azizul109/form-olivine