



دانشگاه فنی و حرفه ای  
دانشکده فنی و حرفه ای دختران دکتر شریعتی

پایان نامه کارشناسی پیوسته  
رشته مهندسی کامپیوتر گرایش نرم افزار

**تحلیل ، طراحی و پیاده سازی سایت شبکه اجتماعی "گالری هنری تیدا" با استفاده از PHP (MVC)**

نگارش:  
الهام حسینی

استاد راهنما:  
سید حسین احمد پناه

تاریخ دفاع: دی ماه 1399

بِسْمِ اللّٰهِ الرَّحْمٰنِ الرَّحِيْمِ

## **تشکر و قدردانی:**

نمی‌توانم معنایی بالاتر از تقدیر و تشکر بر زبانم جاری سازم و سپاس خود را در وصف استادان خویش آشکار نمایم، که هر چه گوییم و سرایم، کم گفته‌ام.  
خدایا توفیق خدمتی سرشار از شور و نشاط و همراه و همسو با علم و دانش و پژوهش جهت رشد و شکوفایی ایران کهنسال عنایت بفرما.

## چکیده

این سایت طراحی یک سایت شبکه اجتماعی به منظور ایجاد ارتباط بین هنرمندان سراسر دنیا و عرضه اثار هنرمندان سایت "گالری هنری تیدا گلد" می باشد.

این سایت با زبان اسکریپ نویسی php و توسط فریم ورک laravel با معماری MVC توسعه داده شده است.

سایت قابلیت های مختلفی از جمله گذاشتن لایک و کامنت برای آثار دلخواه، قابلیت ایجاد پست و کراپ عکس و همچنین بخش ادمین برای مدیریت کاربران، پست ها، ایجاد دسته بندی برای آثار هنری و مدیریت نظرات از قابلیت های اصلی بخش ادمین سایت به شمار می آید.

این سایت علاوه بر داشتن امکانات اولیه سایت اجتماعی قابلیت توسعه در بخش های مختلف را دارا می باشد، از جمله برگزاری گالری هنری آنلайн، قابلیت ذخیره پست های محبوب قابلیت گذاشتن امتیاز برای آثار یا report کاربران مختلف ، خرید و فروش آثار و بسیاری قابلیت دیگر بسته به نوع نیاز.

**کلیدواژه:** سایت شبکه اجتماعی، فریم ورک لاراول، معماری MVC ، گالری هنری

## فهرست نوشتار

1.....	فصل 1: مقدمه
3.....	1-1- انگیزه
3.....	1-2- مروری بر پیشینه و کارهای مشابه
3.....	1-3- هدف
3.....	1-4- رئوس مطالب سایر فصل‌ها
5.....	فصل 2: تجزیه و تحلیل نیازمندی‌ها
6.....	2-1- مقدمه
6.....	2-2- نمودار ERD
7.....	2-3- نمودار Use Case
8.....	2-3-1- کنشگر (Actor)
8.....	2-3-2- کاربر سایت
9.....	2-3-3- ادمین سایت
10.....	2-4- نمودار Activity
10.....	2-4-1- نمودار فعالیت اضافه کردن پست
11.....	2-4-2- نمودار فعالیت اضافه کردن نظر
12.....	2-4-3- نمودار فعالیت follow-unfollow
13.....	2-4-4- نمودار فعالیت ورود و ثبت نام
14.....	2-4-5- نمودار فعالیت جستجو
15.....	2-5- نمودار Class
17.....	2-6- نمودار Sequence
17.....	2-6-1- نمودار ترتیب اضافه کردن پست یا عکس
18.....	2-6-2- نمودار ترتیب ویرایش پست
19.....	2-6-3- نمودار ترتیب اضافه یا حذف دنبال کردن
19.....	2-6-4- نمودار ترتیب ثبت نام در سایت
20.....	2-6-5- نمودار ترتیب ورود به سایت
21.....	2-6-6- نمودار ترتیب جستجو
22.....	2-6-7- نمودار ترتیب اضافه و حذف کردن کامنت
23.....	2-7- نتیجه‌گیری
24.....	فصل 3: ساختار داده‌ها و بانک اطلاعات
25.....	3-1- مقدمه
25.....	3-2- طراحی
26.....	3-2-2- جزئیات جدول users
27.....	3-2-3- جزئیات جدول posts
28.....	3-2-4- جزئیات جدول photos
28.....	3-2-5- جزئیات جدول categories
29.....	3-2-6- جزئیات جدول category_posts
29.....	3-2-7- جزئیات جدول hashtags
30.....	3-2-8- جزئیات جدول likes
30.....	3-2-9- جزئیات جدول comments
32.....	3-2-10- جزئیات جدول followers

33	3-3- نرمافزار مدیریت پایگاه دادهها
33	3-4- نتیجه‌گیری
34	فصل 4: پیاده‌سازی
35	4-1- مقدمه
35	4-2- کدنویسی
36	4-2-2- دایرکتوری app
37	4-2-2-2- دایرکتوری Http واقع در دایرکتوری app
38	4-2-3- دایرکتوری config
38	4-2-4- دایرکتوری database
39	4-2-5- دایرکتوری public
40	4-2-6- دایرکتوری resources
41	4-2-7- دایرکتوری routes
42	4-2-8- دایرکتوری vendor
42	4-3- اجرای نرمافزار
43	4-3-1- فرایند های مهم سایت
43	4-3-1-1- login-register
46	4-3-1-2- قابلیت responsive بودن صفحات سایت
47	4-3-1-3- استفاده از imageCropper در پروژه
50	4-3-1-4- قابلیت like-dislike-unlike-undislike صفحه بدون refresh
52	4-3-1-5- جستجو
55	4-3-1-6- نظر دهی
59	7-1-3-4- بخش ادمین سایت
59	4-4- نتیجه‌گیری
61	فصل 5: جمع‌بندی و پیشنهادها
62	5-1- نتیجه گیری
62	5-2- پیشنهادهایی برای کارهای آتی
63	منابع
65	پیوست‌ها

## فهرست شکل‌ها

7	..... شکل 2-1. شکل نمودار ERD جداول پایگاه داده
8	..... شکل 2-2. رابطه actor ها در نمودار usecase
9	..... شکل 2-3. نمودار usecase برای user
10	..... شکل 2-4. نمودار usecase برای admin
11	..... شکل 2-5. نمودار فعالیت اضافه کردن پست
12	..... شکل 2-6. نمودار فعالیت اضافه کردن نظر
12	..... شکل 2-7. نمودار فعالیت follow-unfollow
13	..... شکل 2-8. نمودار فعالیت ورود و ثبت نام
14	..... شکل 2-9. نمودار فعالیت جستجو
15	..... شکل 2-10. نمودار کلاس
17	..... شکل 2-11. نمودار ترتیب اضافه کردن پست یا عکس
18	..... شکل 2-12. نمودار ترتیب ویرایش پست
19	..... شکل 2-13. نمودار ترتیب اضافه یا حذف دنبال کردن
20	..... شکل 2-14. نمودار ثبت نام در سایت
21	..... شکل 2-15. نمودار ترتیب ورود به سایت
22	..... شکل 2-16. نمودار ترتیب جستجو
23	..... شکل 2-17. نمودار ترتیب اضافه کردن کامنت
23	..... شکل 2-18. نمودار ترتیب حذف کردن کامنت
25	..... شکل 3-1. نمای کلی جداول پایگاه داده
26	..... شکل 3-2. جزئیات جدول users
27	..... شکل 3-3. جزئیات جدول posts
28	..... شکل 3-4. جزئیات جدول photos
28	..... شکل 3-5. جزئیات جدول categories
29	..... شکل 3-6. جزئیات جدول category_posts
29	..... شکل 3-7. جزئیات جدول hashtags
30	..... شکل 3-8. جزئیات جدول likes
30	..... شکل 3-9. جزئیات جدول comments
31	..... شکل 3-10. مدل Comment و رابطه polymorphic
31	..... شکل 3-11. مدل Post و رابطه polymorphic
32	..... شکل 3-12. جزئیات جدول followers
32	..... شکل 3-13. مدل User و رابطه با جدول followers
36	..... شکل 4-1. پوشه بندی لاراول
37	..... شکل 4-2. دایرکتوری app
37	..... شکل 4-3. دایرکتوری Http
38	..... شکل 4-4. دایرکتوری Controllers و Middleware
39	..... شکل 4-5. دایرکتوری migrations –migration users
41	..... شکل 4-6. دایرکتوری resources
42	..... شکل 4-7. دایرکتوری routes
43	..... شکل 4-8. ظاهر صفحه register در سایت
44	..... شکل 4-9. Register view codes

46	..... شکل 4-10. صفحه اصلی هنرمند سایت full page
47	..... شکل 4-11. طراحی responsive سایت
47	..... شکل 4-12. کردن عکس قبل از آپلود Crop
48	..... شکل 4-13. کد imageCropper controller مربوط به
50	..... شکل 4-14. قابلیت like در سایت
52	..... شکل 4-15. بخش جستجو برای هنرمند
53	..... شکل 4-16. Layout اصلی کاربر هنرمند
54	..... شکل 4-17. استفاده از include برای جستجو
54	..... شکل 4-18. کد مربوط به جستجو کاربر هنرمند بخش اول
55	..... شکل 4-19. کد مربوط به جستجو کاربر هنرمند بخش دوم
55	..... شکل 4-20. ظاهر بخش نظردهی سایت
56	..... شکل 4-21. کد script مربوط به بخش نظرات بخش اول
56	..... شکل 4-22. کد script مربوط به بخش نظرات بخش دوم
57	..... شکل 4-23. کد نوشته شده در blade برای comment بخش اول
57	..... شکل 4-24. کد نوشته شده در blade برای comment بخش دوم
58	..... شکل 4-25. کد مربوط به include comment بخش اول
58	..... شکل 4-26. کد مربوط به include comment بخش دوم
59	..... شکل 4-27. صفحه اصلی داشبورد ادمین بخش اول
59	..... شکل 4-28. صفحه اصلی داشبورد ادمین بخش دوم
66	..... شکل 5-1. محیط برنامه xampp
67	..... شکل 5-2. فایل env.

# فصل 1:

## مقدمه

امروزه یکی از پرینتندۀ ترین و پرطرفدارترین سایت‌ها، سایت‌های شبکه‌های اجتماعی است به طوری که سایت شبکه‌های اجتماعی، بعد از موتورهای جستجو، تبدیل به پراستفاده‌ترین خدمت اینترنتی شده‌اند. اساس کار شبکه‌های اجتماعی ایجاد ارتباط میان افراد و به اشتراک گذاری محتوا با یکدیگر است به گونه‌ای که هر شخص دارای پروفایل شخصی می‌باشد که در آن از اطلاعات پایه تا علائق و سلایق خود را قرار می‌دهد، دوستان جدیدی پیدا می‌کند و دوستان قدیمی خود را در جریان تغییرات زندگی‌اش قرار می‌دهد.

هر شبکه اجتماعی بسته به تعداد کاربران و هدف از ایجادش، امکاناتی را می‌تواند به اعضای خود ارائه دهد، که این امکانات عبارتند از:

- امکان عضویت
- جست و جو در میان اعضا
- ارسال پیام برای یکدیگر
- آپلود عکس
- نمایش یکسری اطلاعات اولیه از هر فرد (نمایش سن، مکان سکونت، ایمیل، رنگ چشم و مو، دین، نژاد و ...)
- نمایش اعضا آنلاین در سایت
- و.... امکانات بسیار دیگری که می‌تواند در سایت موجود باشد.

حال با توجه به امکانات گسترده‌ای که می‌توان در طراحی سایت شبکه اجتماعی قرار داد، ابتدا باید اهداف طراحی سایت را مشخص کرده و سپس با توجه به این اهداف امکاناتی را به سایت اضافه کرد.

برای طراحی چنین وب سایت هایی نیاز دارید مطالبی را در این مورد این شبکه های اجتماعی و ایجاد آن بدانید. مطالبی از قبیل اینکه :

- هدف از ایجاد این جوامع
- میزان پیشرفت این جوامع
- تعداد کاربران عضو
- امکانات کاربران
- امنیت پروفایل های شخصی
- امنیت کلی وب سایت
- تامین منابع مالی
- طراحی سایت شبکه های اجتماعی

گسترده‌گی استفاده از این جوامع و تامین امنیت کافی برای کاربران یکی از مهمترین مسائل برای این جوامع خواهد بود.

## 1-1- انگیزه

انگیزه از طراحی سایت شبکه اجتماعی با موضوع گالری هنری فراهم آوردن بستری خاص برای هنرمندان و دوستداران هنر های تجسمی<sup>۱</sup> است ، تا آثار خود را به اشتراک گذارند و با هنرمندان دیگر در ارتباط قرار گیرند. همچنین این سایت با توسعه بیشتر می تواند منبعی برای کسب درآمد و تبلیق و نقد و بررسی آثار هنری باشد.

## 1-2- مروری بر پیشینه و کارهای مشابه

هرچند بستری خاص برای آثار هنری یک عرصه جدید به شمار می رود اما شبکه های اجتماعی موفق بسیاری مانند اینستاگرام<sup>۲</sup>، فیسبوک<sup>۳</sup>، یوتیوب<sup>۴</sup>، توئیتر<sup>۵</sup> و بسیاری دیگر قبل از به عرصه گذاشته اند.

## 1-3- هدف

هدف از طراحی سایت شبکه اجتماعی اغلب جمع کردن انسان ها در کنار هم در دنیای مجازی می باشد. با توجه به استفاده از ابزار های ارتباطی فراوان در جامعه، بسیاری از مردم در دنیا از هم فاصله های فراوانی گرفته اند و از حال و احوال خود و دوستان و خانواده خود با خبر نمی باشند. متخصصان فناوری با توجه به این نیاز جامعه با طراحی سایت شبکه اجتماعی قصد دارند که روابط مردم را از طریق جوامع مجازی با هم پیوند دهند. البته در کنار این هدف، هدف کسب درآمد نیز دلیل بزرگی برای راه اندازی این وب سایت ها می باشد. حال ما اگر قصد داریم اقدام به راه اندازی اینچنین وب سایت هایی نماییم، باید تمام نکات و حالت های ایجاد و طراحی سایت شبکه های مجازی را بدانیم.

## 1-4- رئوس مطالب سایر فصلها

در فصل های بعد به بررسی ساختار قسمت های مختلف سایت و جداول پایگاه داده و معماری

---

Visual arts	<sup>1</sup>
Instagram	<sup>2</sup>
facebook	<sup>3</sup>
youtube	<sup>4</sup>
twitter	<sup>5</sup>

و مدلسازی پروژه با استفاده از زبان مدلسازی UML می پردازیم.

فصل 2:

## تجزیه و تحلیل نیازمندی‌ها

## 2-1- مقدمه

قبل از شروع هر پروژه‌ای، باید آن را از جنبه‌های مختلف مورد تحلیل قرار داد. یکی از اولین گام‌ها برای شروع پروژه، فهمیدن آن است. پس باید نیازمندی‌های پروژه، مورد تحلیل واقع شود. در این مرحله، انتظارات کاربران از سیستمی که پیاده سازی خواهد شد تعریف می‌شوند. نیازمندی‌های کاربر در قالب سناریو نوشته شده و نمودارهای مدلسازی یکپارچه (UML<sup>1</sup>) آن طراحی می‌شود. نمودار مورد کاربرد<sup>2</sup> یکی از نمودارهای زبان مدل سازی یکپارچه می‌باشد.

نمودارهای UML ای که در این بخش مورد بررسی قرار می‌گیرند:

- نمودار کلاس – Class
- نمودار مورد کاربرد – Use case diagram
- نمودار توالی – Sequence
- نمودار فعالیت – Activity

اما ابتدا نمودار ERD<sup>3</sup> که روابط موجودیت‌ها را نمایش میدهد در بخش بعد مشاهده خواهیم کرد.

## 2-2- نمودار ERD

نمودار 2-1 روابط بین موجودیت‌ها و جداول دیتابیس<sup>4</sup> و کلید اصلی<sup>5</sup> و کلید خارجی<sup>6</sup> را به خوبی نشان میدهد.

Unified Modeling Language <sup>1</sup>

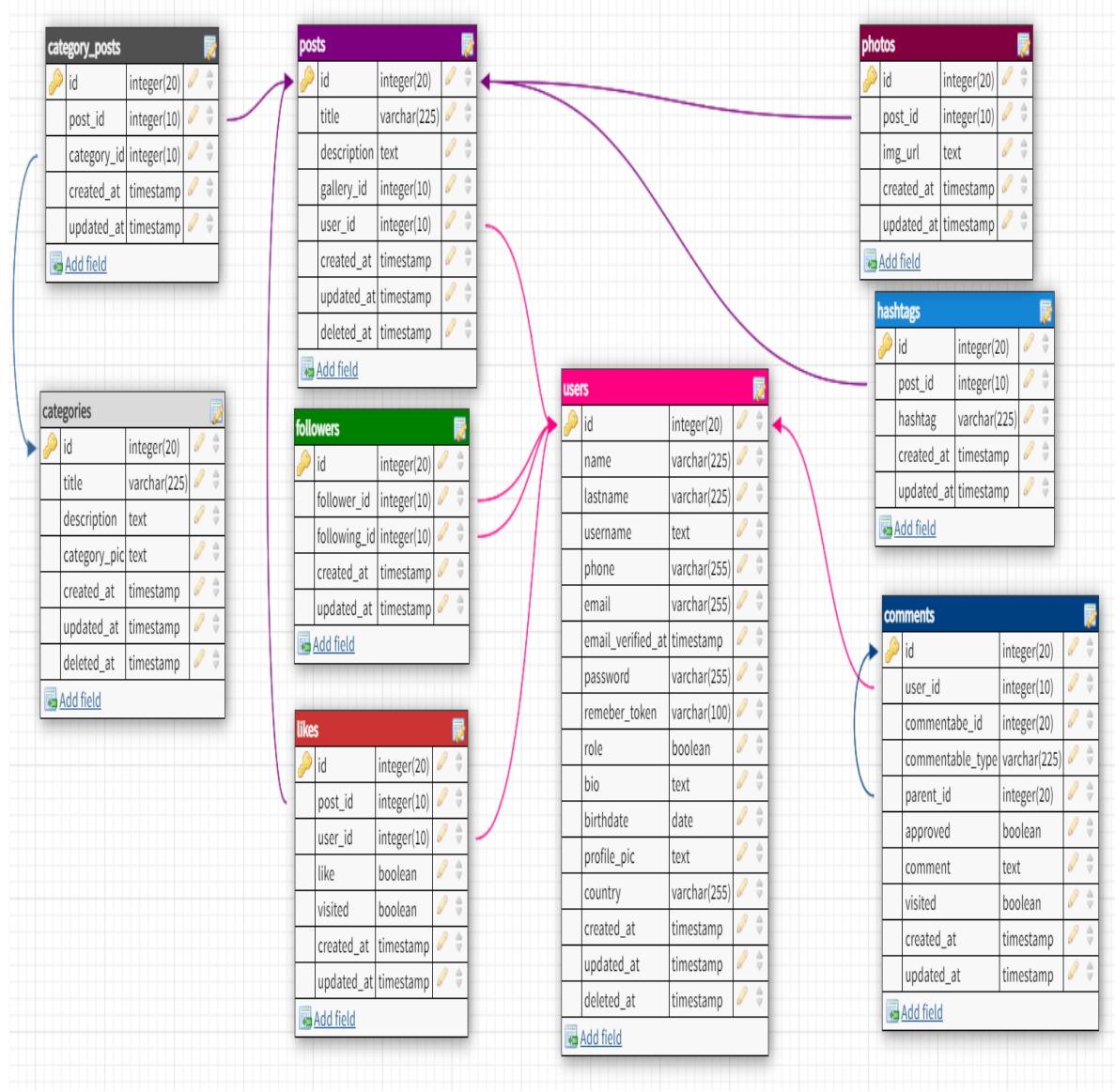
Usecase diagram <sup>2</sup>

Entity Relationship Diagram <sup>3</sup>

database <sup>4</sup>

Primary Key <sup>5</sup>

Foreign Key <sup>6</sup>



شکل 1-2. شکل نمودار ERD جداول پایگاه داده

## Use Case - 2- نمودار

نیازمندی‌های کاربر در قالب سناریو نوشته شده و نمودارهایی مانند نمودار مورد کاربرد برای درک بهتر آنها تولید می‌شوند. نمودار مورد کاربرد یکی از نمودارهای زبان مدل سازی یکپارچه می‌باشد. وظیفه این نمودار مدل کردن خدمات یک سیستم می‌باشد. علاوه بر این، نمودار مورد کاربرد یک مدل بسیار انتزاعی از سیستم ارائه می‌کنید به طوری که توسط اکثر افراد پروژه نرم افزاری قابل فهم است. حتی این نمودار به قدری ساده است که توسط یک کاربر دارای اطلاعات عمومی نیز قابل فهم خواهد بود.

## (Actor) 2-3-2- کنشگر

نقشی که کاربر در ارتباط با سیستم ایفا می‌کند، کنشگر<sup>۱</sup> نامیده می‌شود. کنشگر آغاز کننده‌ی فعالیتی در سیستم است. در اینجا توجه به نقش کاربر است و نه شخص کاربر. کلیه‌ی کاربران از جمله سایر سیستم‌ها کنشگر سیستم محسوب می‌شوند. در این سیستم ما دو کنشگر داریم، کاربر<sup>۲</sup> ساده سیستم که در سایت ثبت نام می‌کند و قابلیت‌های سایت در اختیار او قرار می‌گیرد و کاربر ادمین<sup>۳</sup> که به صورت مدیر در سایت معرفی شده و مدیریت سایت را به عهده دارد. در شکل زیر رابطه بین actor‌ها را مشاهده می‌کنید که به صورت ارثبری<sup>۴</sup> کاربر ساده از ادمین هست.



شکل 2-2. رابطه actor‌ها در نمودار usecase

هر کدام از نقشها، محدوده وظایف مشخصی دارند که برخی از آنها مشترک می‌باشند. در این قسمت به قابلیت‌های منحصر به فرد هر نقش اشاره می‌کنیم

## 2-3-2- کاربر سایت

کاربر باید ابتدا برای بار اول در سایت ثبت نام کند و سپس میتواند به لیست قابلیت‌های زیر که در شکل 2-3 نمودار user usecase نیز قابل مشاهده هست دسترسی پیدا کند:

- ورود به سایت
- جستجو
- خروج از سایت
- دنبال کردن یا قطع دنبال کاربر دیگری

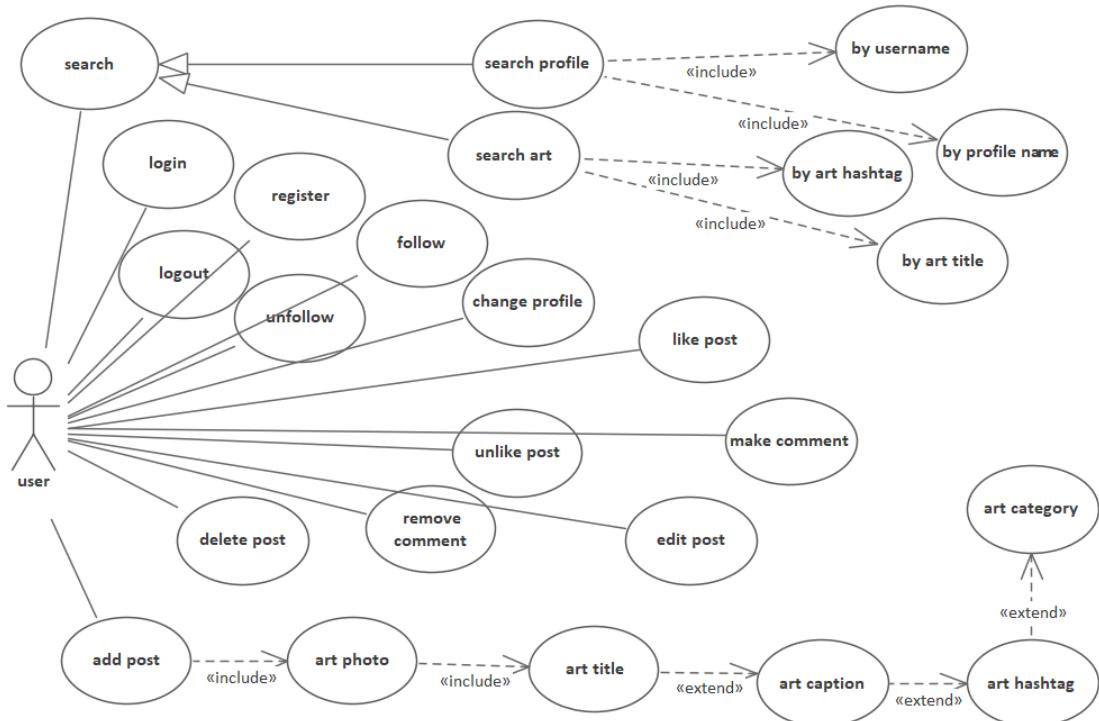
actor<sup>1</sup>

user<sup>2</sup>

admin<sup>3</sup>

generalize<sup>4</sup>

- ویرایش پروفایل<sup>1</sup>
- لایک<sup>2</sup> و دیسلایک<sup>3</sup> کردن یک پست<sup>4</sup>
- ایجاد و یا حذف کامنت<sup>5</sup>
- ایجاد ، ویرایش و یا حذف پست

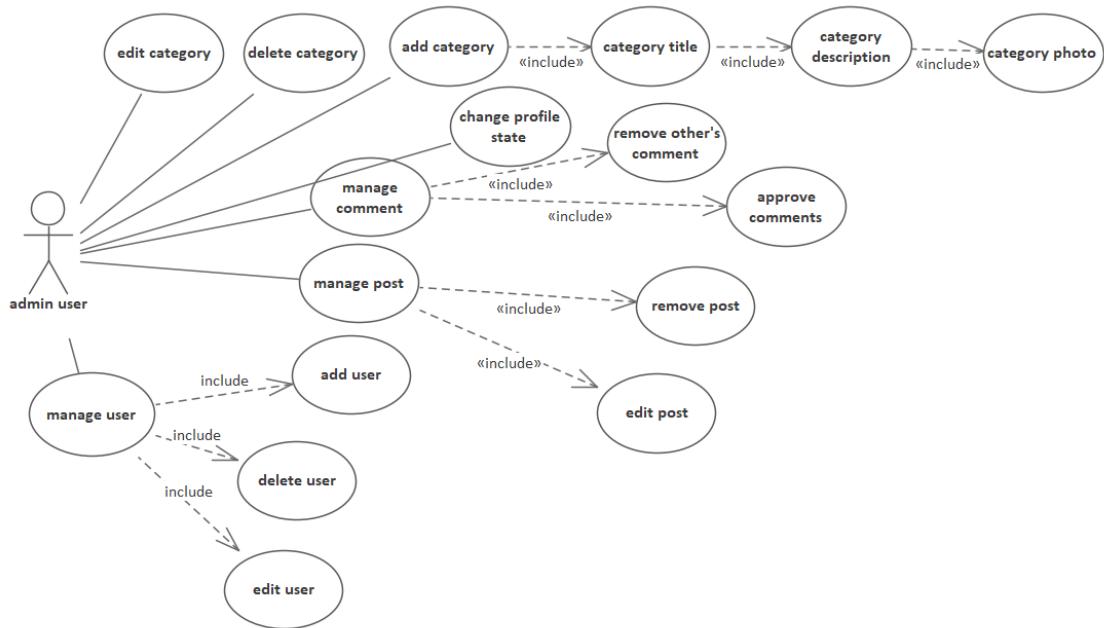


شکل 2-3. نمودار usecase برای user

### 2-3-2- ادمین سایت

دسترسی های ادمین متفاوت از کاربر پس از ثبت نام در سایت به صفحه ادمین هدایت میشود. در این بخش میتواند گزارش هایی از سایت را مشاهده کند و دسترسی هایی که به کاربران و پست ها کامنت ها و مواردی دیگر دارد. شکل 2-4 نمودار usecase را برای ادمین را در زیر میتوانید مشاهده و بررسی کنید.

profile	<sup>1</sup>
like	<sup>2</sup>
dislike	<sup>3</sup>
post	<sup>4</sup>
comment	<sup>5</sup>



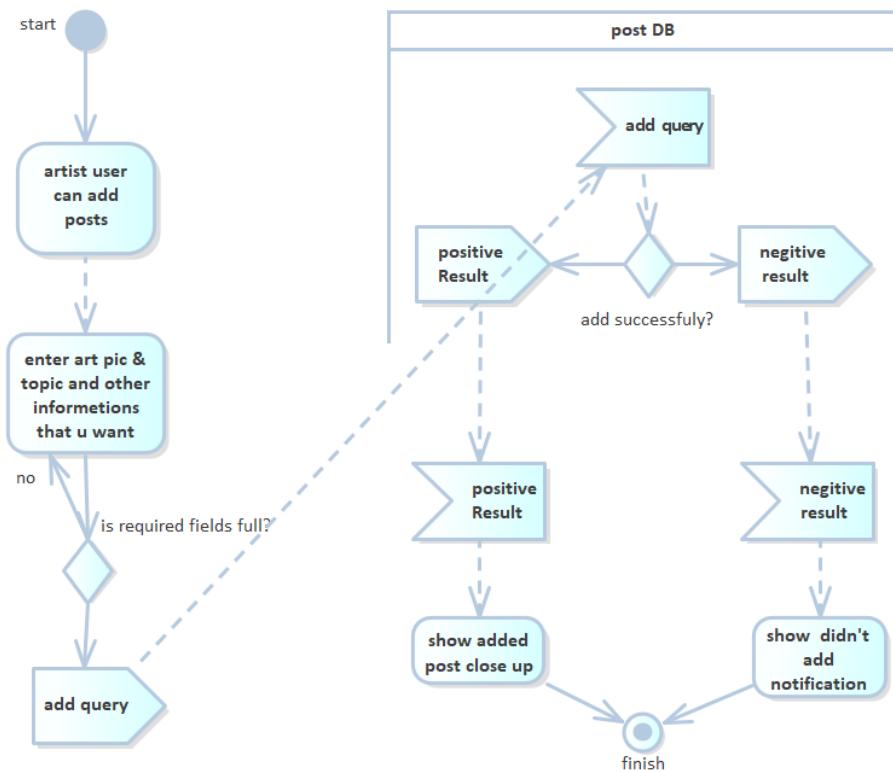
شکل 2-4. نمودار usecase برای admin

## 2-4- نمودار Activity

نمودار فعالیت که رفتار یک شئ را مدل می کند و معادل همان فلوچارت است ولی با یکسری تفاوت جزیی در ساختار و هدف . فعالیت عملی است که به وسیله انسان یا کامپیوتر انجام می شود و در دیدگاه پیاده سازی، متده است. نمودار فعالیت یک فلوچارت است که برای نمایش جریان کنترل از یک فعالیت به فعالیت دیگر به کار می رود. نمودار فعالیت برای نمایش جریان کار و نمایش رفتاری که پردازش‌های موازی دارند مناسب است. در این نمودار صرف نظر از اینکه فاعل رفتار چه کسی است، میتوان رفتار را به خوبی با تقدم و تاخر و بیان شرط های لازم نمایش داد.

### 2-4-2- نمودار فعالیت اضافه کردن پست

شکل 2-5 نمودار فعالیت را برای اضافه کردن مرحله به مرحله پست در سایت نشان میدهد.

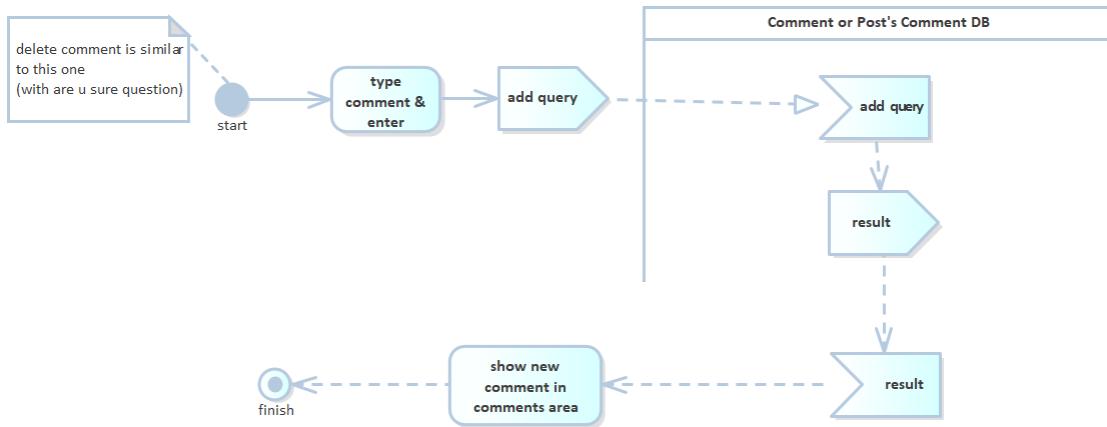


شکل 5-2. نمودار فعالیت اضافه کردن پست

هر نمودار فعالیت یک نقطه شروع و یک نقطه پایان برای فعالیت ذکر شده (در اینجا اضافه کردن پست) دارد. علامت لوزی شکل نشان دهنده شرط برای رفتن به مرحله بعد را نشان میدهد. در شکل مشاهده میکنید که اگر کاربر مقادیر اجباری برای ایجاد پست را وارد کرده باشد در مرحله بعد درخواست ایجاد پست به پایگاه داده فرستاده میشود. و در آخر اگر پست با موفقیت ایجاد شد پیغام مناسب ایجاد پست و اگر ایجاد نشد پیغام خطای مناسب نشان داده میشود.

#### 4-2-2- نمودار فعالیت اضافه کردن نظر

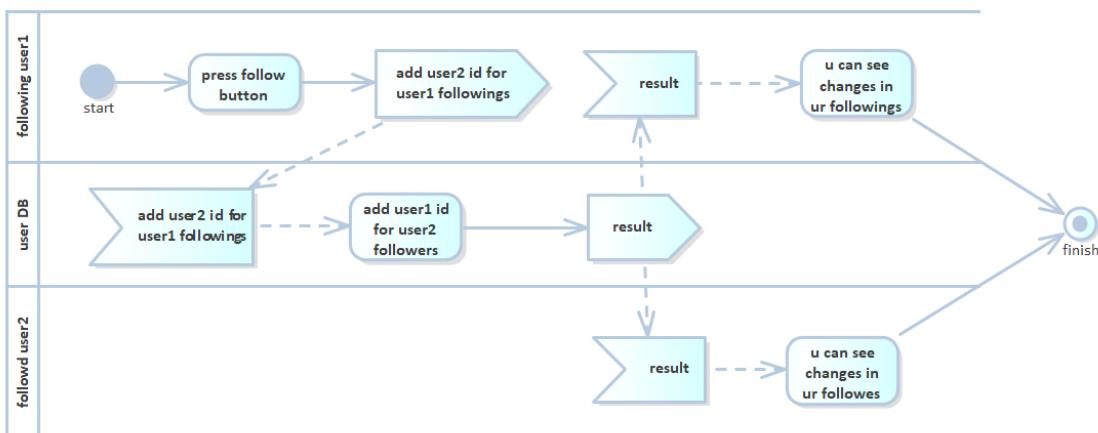
مراحل اضافه کردن نظر در سایت به ترتیب نوشتمن متن نظر توسط کاربر ، زدن دکمه ثبت ، ارسال دستور به پایگاه داده و اضافه کردن آن به جدول نظرات و در آخر نمایش نظر در سایت می باشد. البته دارنده سایت میتواند بخش های مختلفی مانند تایید نظر توسط ادمین و یا نمایش پیغام مناسب ایجاد نظر را هم به این فرایند اضافه کند و آن را توسعه ببخشد.



شکل 6-2. نمودار فعالیت اضافه کردن نظر

### 2-4-3- نمودار فعالیت follow-unfollow

در یک سایت شبکه اجتماعی معمولاً قابلیتی به عنوان follow یا همان دنبال کردن فعالیت های کاربر دیگری در سایت وجود دارد. برنامه نویس میتواند از این طریق قابلیت هایی به سایت اضافه کند مانند صفحه ای برای نمایش فعالیت و پست های کسانی که ما دنبال کننده آن ها هستیم. شکل 7-2 روند follow کردن را نشان میدهد.

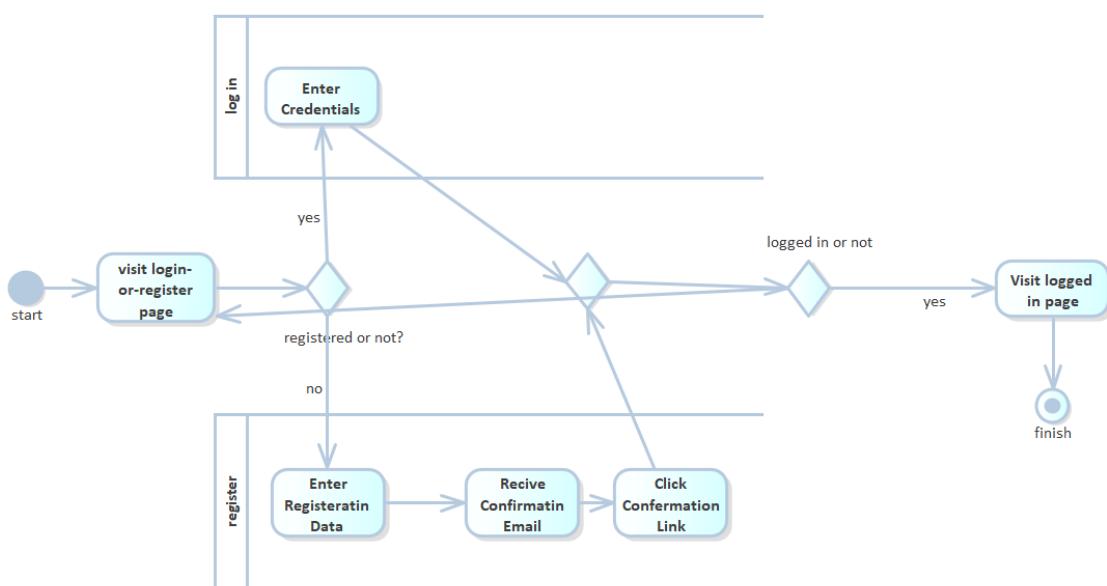


شکل 7-2. نمودار فعالیت follow-unfollow

با زدن دکمه follow در صفحه کاربر دلخواه او را به لیست<sup>1</sup> های خود اضافه میکنیم و با زدن دوباره روی همان دکمه unfollow<sup>2</sup> انجام میشود و آن کاربر از لیست following های ما حذف میشود. و همینطور اگر کاربر دیگری ما را دنبال کند به لیست following های ما اضافه میشود و با unfollow<sup>3</sup> کردن ما از لیست follower ها پک میشود.

#### 4-2-2. نمودار فعالیت ورود<sup>4</sup> و ثبت نام<sup>5</sup>

کاربر ای که برای اولین بار وارد این سایت میشود باید ابتدا در سایت ثبت نام کند. برای دفعات بعدی فقط با زدن ایمیل و پسورد میتواند به سایت ورود کند و دیگر لازم به ثبت نام نمی باشد. شکل 2-8 این مراحل را به خوبی نشان میدهد.



شکل 2-8. نمودار فعالیت ورود و ثبت نام

کاربر با ثبت نام به جدول کاربران افزوده میشود اما با ورود تنها صحت اطلاعات وارد شده او

<sup>1</sup> در حال دنبال کردن

<sup>2</sup> دنبال نکردن

<sup>3</sup> دنبال کننده

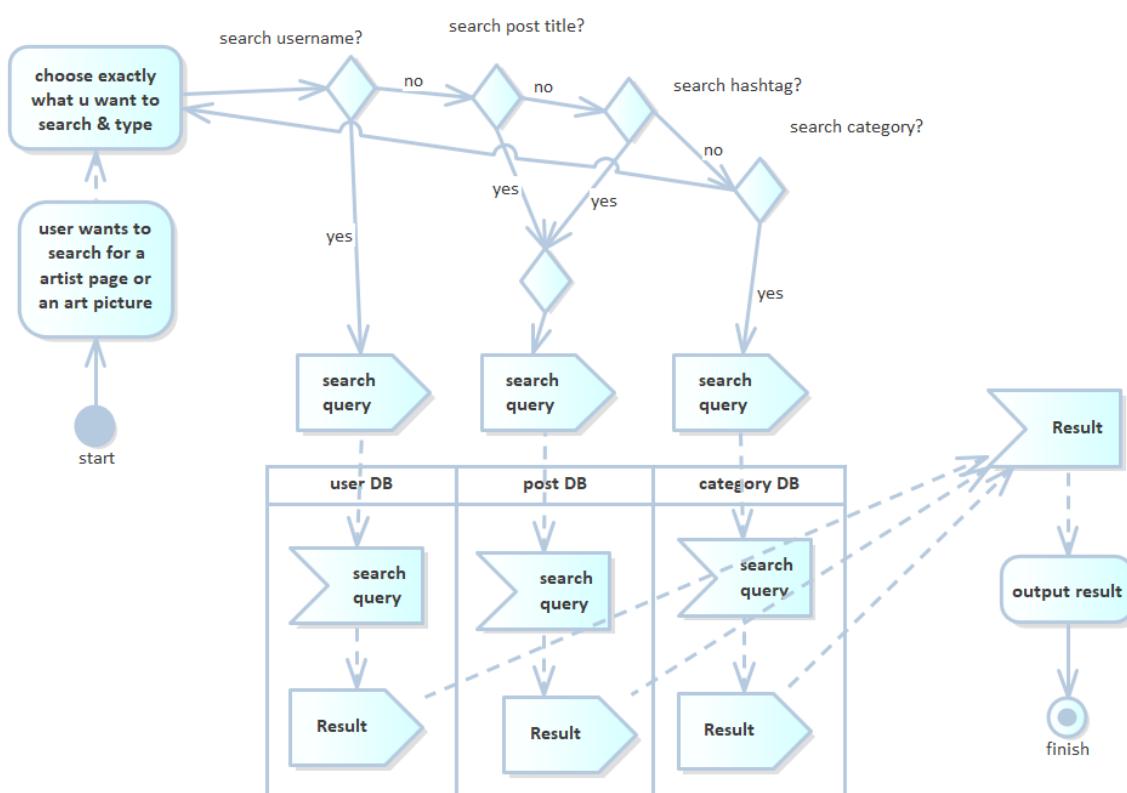
<sup>4</sup> login

<sup>5</sup> register

با جدول کاربران بررسی میشود.

## 2-4-5- نمودار فعالیت جستجو

در سایت کاربر میتواند اطلاعات مختلفی را جستجو کند. نام کاربری<sup>۱</sup> افراد، هشتگ<sup>۲</sup> های هر پست ، عنوان<sup>۳</sup> هر پست ، و دسته<sup>۴</sup> ای هر پست که در این سایت همان سبک هر اثر هنری میباشد. با انتخاب نوع جستجو، دستور جستجو به جدول مربوطه مراجعه کرده و نتایج مشابه عبارت جستجو شده را بر می گرداند. این مراحل را میتوانید در نمودار فعالیت شکل 2-9 مشاهده کنید.

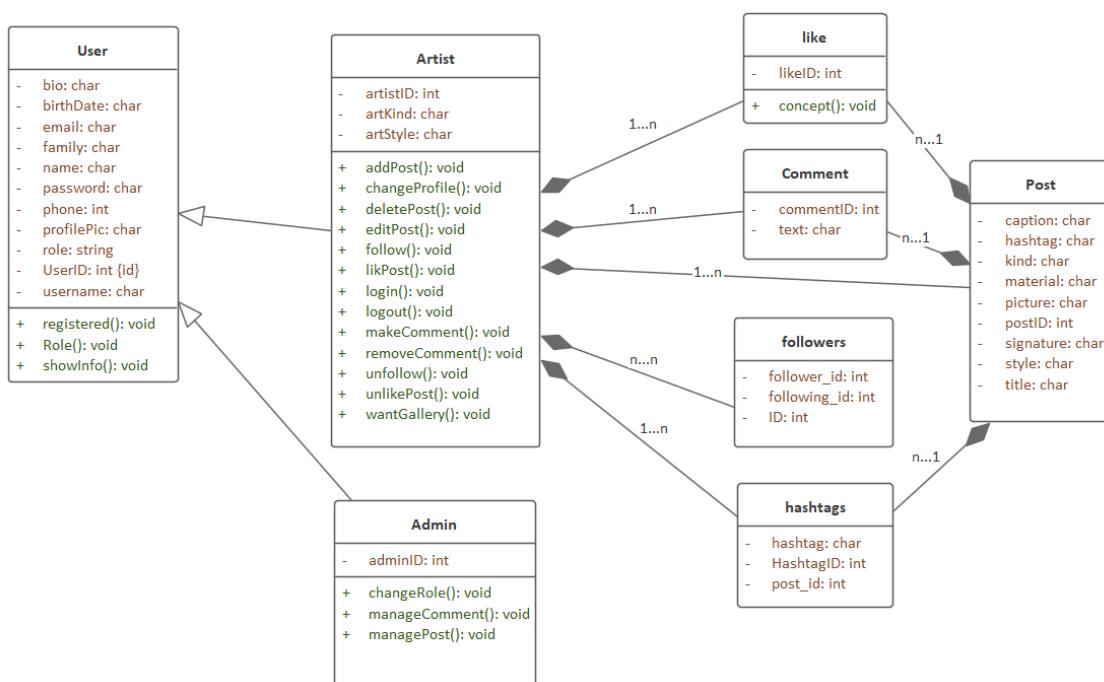


شکل 2-9. نمودار فعالیت جستجو

username	1
hashtag	2
title	3
category	4

## Class - 2- نمودار

نمودار کلاس خصیصه<sup>1</sup> ها و عملیات<sup>2</sup> یک کلاس و همچنین محدودیت<sup>3</sup> های اعمال شده بر روی سیستم را توصیف می‌کند. نمودارهای کلاس به طور گسترده به منظور مدل سازی سیستم‌های شی گرا<sup>4</sup> بکار گرفته می‌شوند، زیرا نمودار کلاس تنها دیاگرامی است که می‌تواند توسط زبان‌های شی گرا نگاشت<sup>5</sup> شود.



شکل 2-10. نمودار کلاس

نمودار کلاس موجودیت‌های سیستم را نشان میدهد:

- موجودیت User اطلاعات اولیه هر کاربر را نشان میدهد.

attribute	1
operation	2
constraint	3
Object oriented	4
map	5

- موجودیت Artist که از موجودیت User ارث میبرد، یعنی تمام ویژگی‌های User را دارد اما ویژگی‌هایی اضافه بر آن را نیز دارا میباشد.
- موجودیت Admin که از موجودیت User ارث میبرد، یعنی تمام ویژگی‌های User را دارد اما ویژگی‌هایی اضافه بر آن را نیز دارا میباشد.
- موجودیت like با Artist رابطه یک به چند دارد یعنی یک Artist میتواند صفر تا چندین like داشته باشد اما یک like تنها متعلق به یک Artist میباشد. لایک نشان دهنده علاقه مندی به یک Post میباشد. موجودیت like با Post رابطه یک به چند دارد یعنی یک Post میتواند صفر تا چندین like داشته باشد اما یک like تنها متعلق به یک Post میباشد.
- موجودیت comment با Artist رابطه یک به چند دارد یعنی یک Artist میتواند صفر تا چندین comment داشته باشد اما یک comment تنها متعلق به یک Artist میباشد. نشان دهنده نظر Artist به یک Post میباشد. موجودیت comment با Post رابطه یک به چند دارد یعنی یک Post میتواند صفر تا چندین comment داشته باشد اما یک comment تنها متعلق به یک Post میباشد.
- موجودیت followers با Artist رابطه چند به چند دارد یعنی یک Artist متواند صفر تا چندین follower و following داشته باشد. باید توجه داشت که id و follower\_id هر دو یک Artist هستند. که نشان میدهد Artist از طریق followers با خودش ارتباط دارد.
- موجودیت hashtag با Artist رابطه یک به چند دارد یعنی یک Artist میتواند صفر تا چندین hashtag داشته باشد اما یک hashtag تنها متعلق به یک Artist میباشد. نشان دهنده نشان گذاری Artist بر روی یک Post میباشد که برای جستجو یک Post به کار برده میشود. موجودیت hashtag با Post رابطه یک به چند دارد یعنی یک Post میتواند صفر تا چندین hashtag داشته باشد اما یک hashtag تنها متعلق به یک Post میباشد.
- موجودیت Post با Artist رابطه یک به چند دارد یعنی یک Artist میتواند صفر تا چندین Post داشته باشد اما یک Post تنها متعلق به یک Artist میباشد. Post نشان دهنده اثر یک Artist در سایت میباشد که اصلی ترین موجودیت بعد از کاربر است. رابطه موجودیت Post با موجودیت‌های comment، hashtag و like در بالا بررسی شد.

## 2-2- نمودار Sequence

نمودار ترتیب<sup>۱</sup> رفتار سیستم را مدل می‌کند. تاکید در این نمودار بر زمان و ترتیب ارسال پیام‌ها است.

### 2-2- نمودار ترتیب اضافه کردن پست یا عکس

ترتیب اضافه کردن پست را میتوانید در نمودار ترتیب شکل 11-2 مشاهده کنید.

مرحله 1) درخواست نمایش صفحه ایجاد پست به سیستم

مرحله 2) نمایش صفحه ایجاد پست توسط سیستم

مرحله 3) پر کردن فیلد های لازم

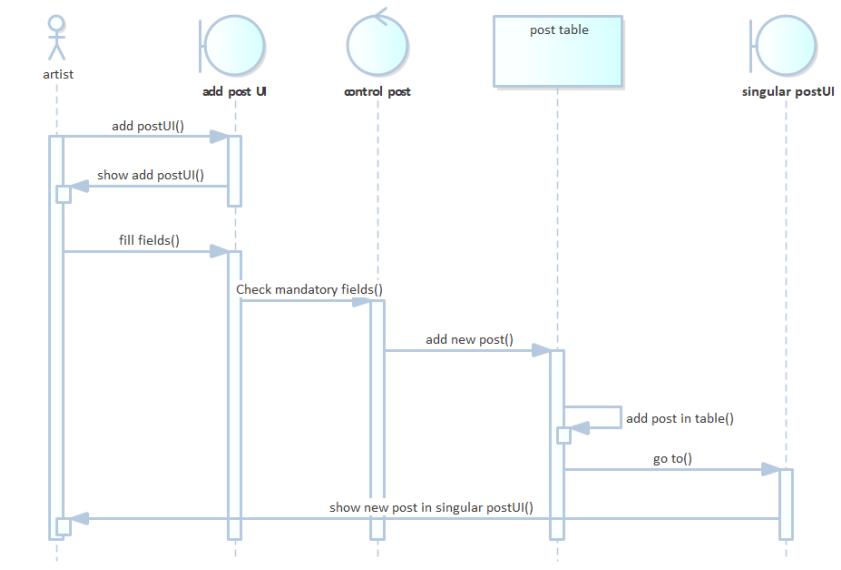
مرحله 4) چک کردن پر بودن فیلد های اجباری توسط سیستم

مرحله 5) درخواست اضافه کردن پست جدید به جدول مربوطه در پایگاه داده

مرحله 6) انجام درخواست اضافه کردن پست جدید به جدول مربوطه در پایگاه داده

مرحله 7) درخواست نمایش صفحه ی پست جدید توسط سیستم

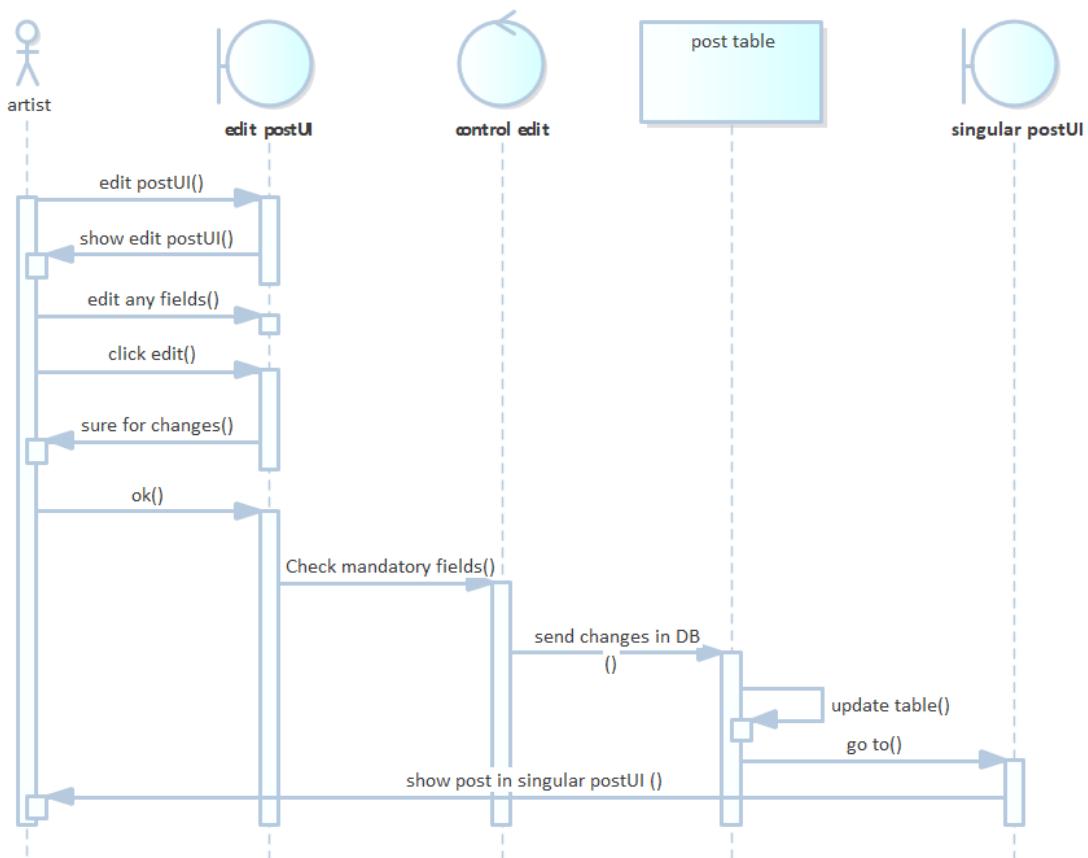
مرحله 8) مشاهده پست جدید توسط کاربر



شکل 11-2. نمودار ترتیب اضافه کردن پست یا عکس

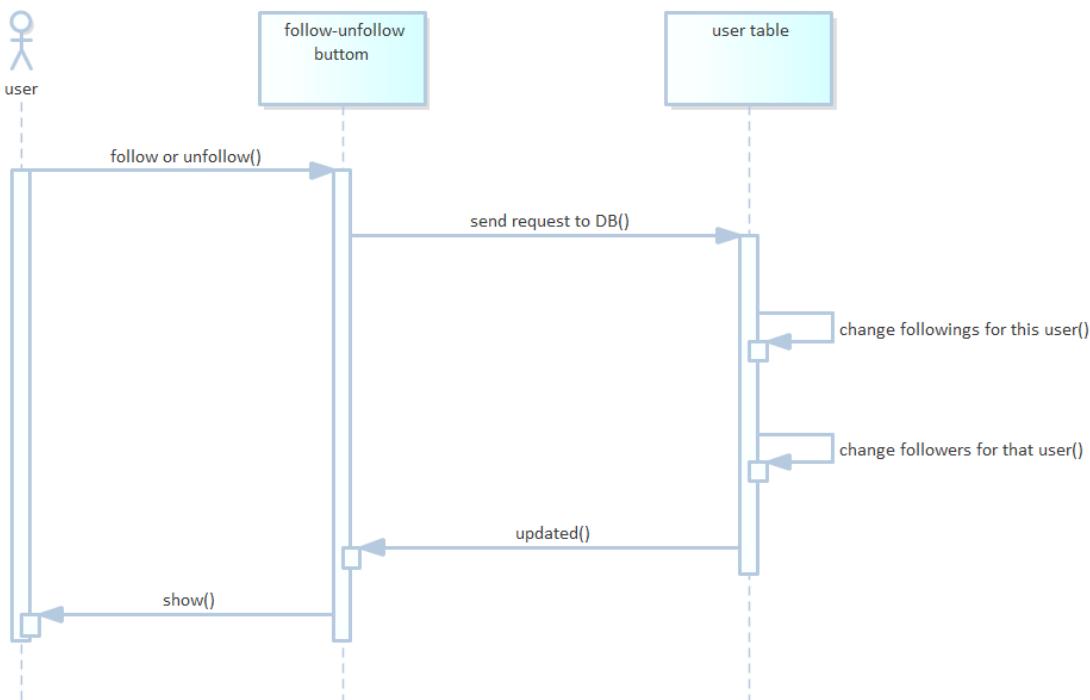
## 2-6-2- نمودار ترتیب ویرایش پست

نمودار ترتیب ویرایش پست مشابه نمودار ایجاد پست با اندکی تغییر است که در شکل 12-2 میتوانید آن را مشاهده کنید.



شکل 12-2. نمودار ترتیب ویرایش پست

## 6-2-3- نمودار ترتیب اضافه یا حذف دنبال کردن

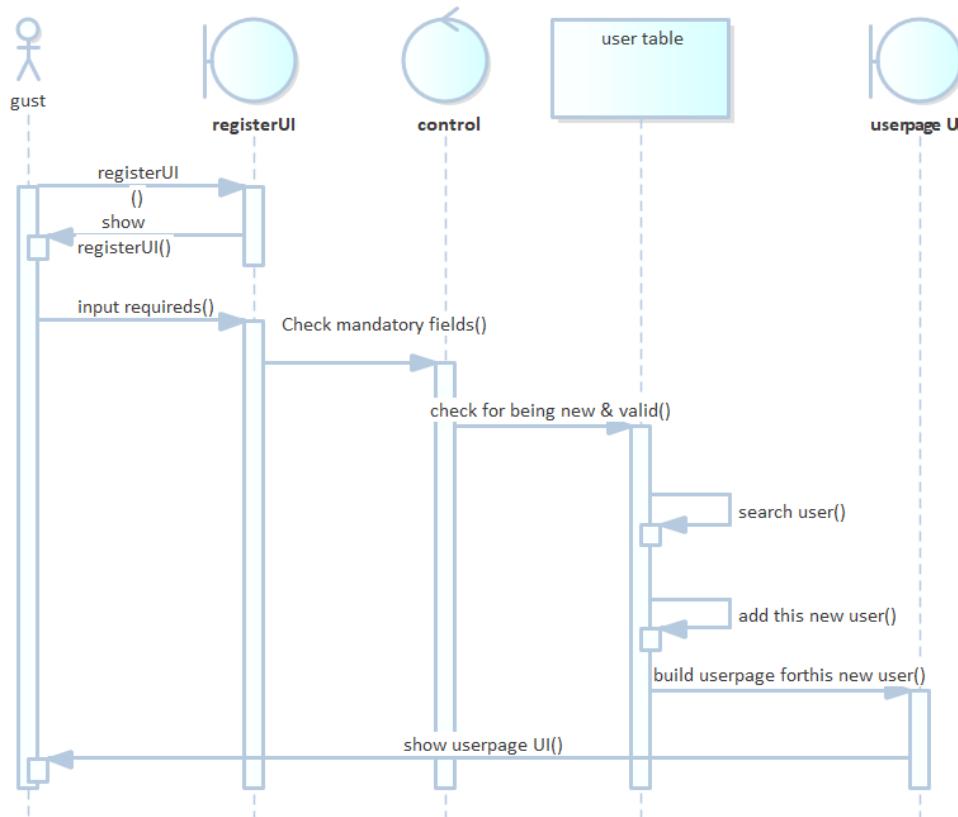


شکل 13-2. نمودار ترتیب اضافه یا حذف دنبال کردن

در شکل 13-2 کاملا مشهود است که کاربر با زدن یک دکمه در خواستی به سیستم میدهد و سیستم پایگاه داده را با توجه به این درخواست به روز رسانی میکند و سپس نتیجه برای کاربر قابل مشاهده خواهد بود. این در خواست در صورت درست<sup>1</sup> بودن follow و در صورت اشتباه<sup>2</sup> بودن unfollow خواهد بود.

## 6-2-4- نمودار ترتیب ثبت نام در سایت

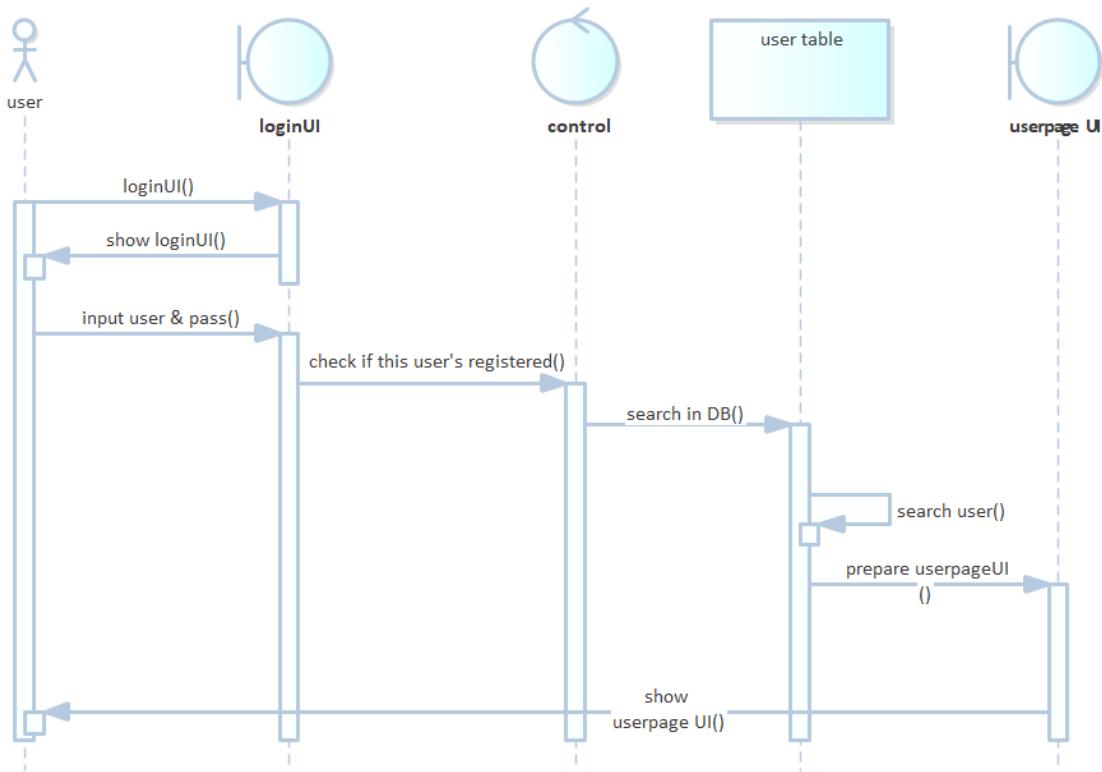
برای ثبت نام اولیه در سایت مواردی را لازم است که سیستم چک کنم مانند پر بودن فیلد های اجباری مانند نام، نام خانوادگی، نام کاربری، ایمیل و ... و چک کردن این که کاربری دو بار با یک ایمیل ثبت نام نکند. تمام این موارد میتوانند در نمودار ترتیب گویای روند کار سیستم باشند.



شکل 14-2. نمودار ثبت نام در سایت

## 5-2-6-2- نمودار ترتیب ورود به سایت

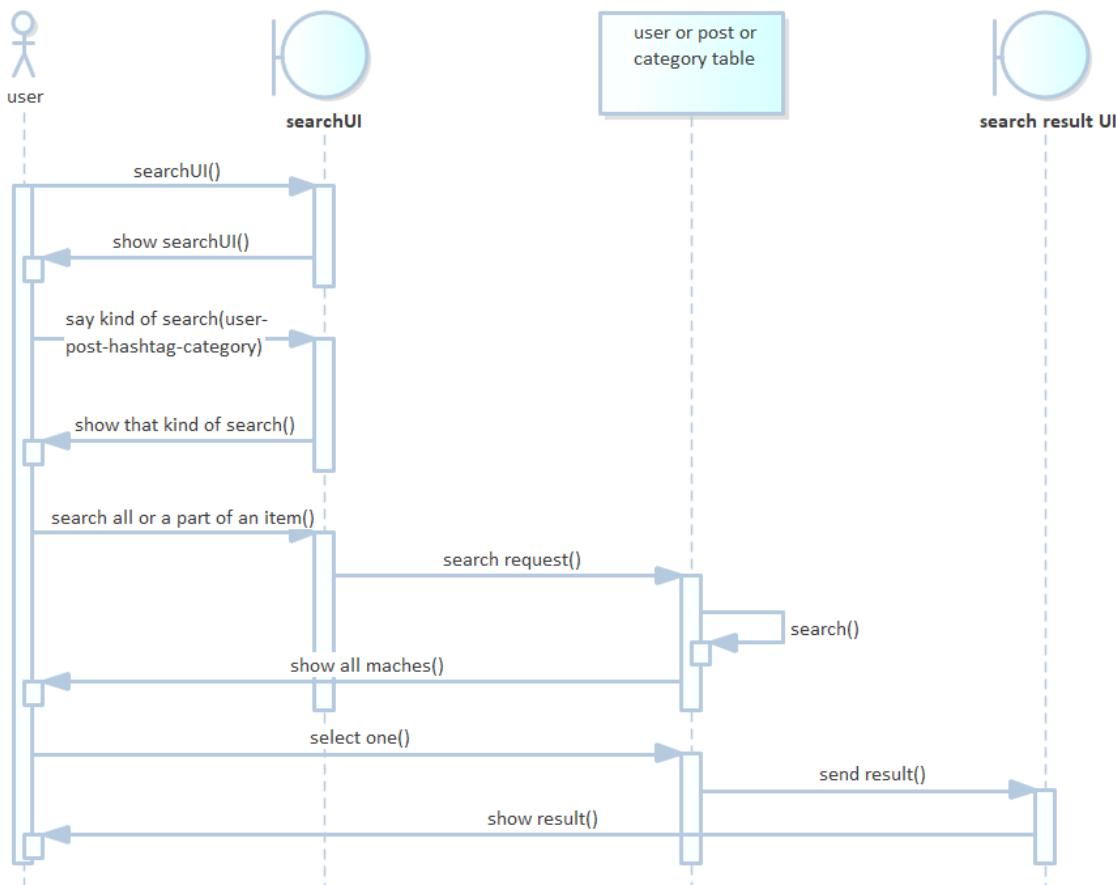
برای ورود به سایت سیستم ابتدا چک میکند که کاربر آیا این کاربر قبلا در سایت ثبت نام کرده(مثلا با چک کردن موجود بودن ایمیل در جدول کاربران) سپس پسورد شخص را چک میکند و اگر اطلاعات وارد شده درست بود به صفحه شخصی کاربر وارد میشود و آن را نمایش میدهد.



شکل 15-2. نمودار ترتیب ورود به سایت

## 6-2-2- نمودار ترتیب جستجو

برای جستجو کاربر ابتدا نوع جستجو را مشخص می‌کند و سپس در قسمت مربوط به آن جستجو را انجام میدهد. سیستم نیز با توجه به بخش انتخاب شده برای جستجو، جستجو را در همان بخش انجام میدهد و لیست مطالب یافت شده را به کاربر نشان میدهد. کاربر روی یک مورد از آن لیست کلیک می‌کند و نتیجه مورد نظر خود را مشاهده می‌کند. ترتیب مراحل را به طور واضح تر در نمودار ترتیب جستجو شکل 2-16 مشاهده کنید.

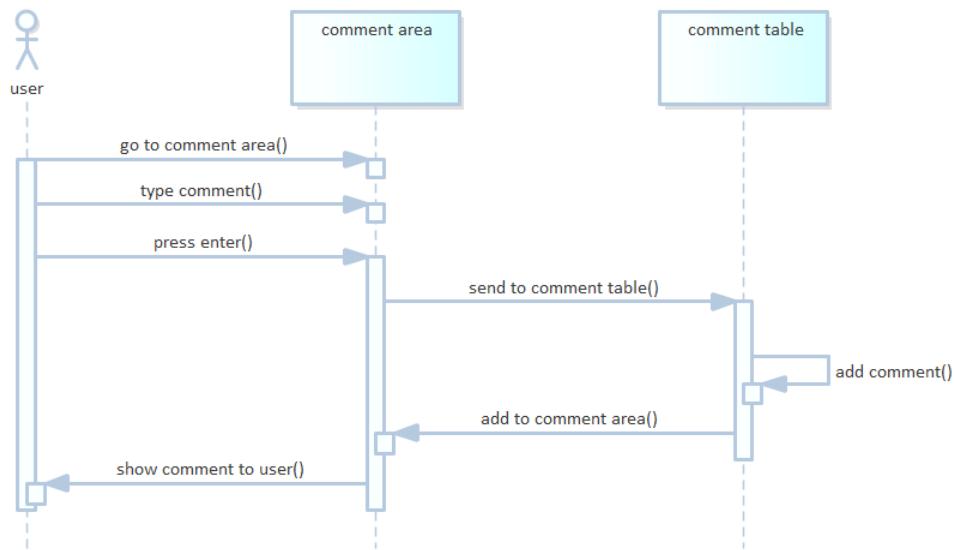


شکل 16-2. نمودار ترتیب جستجو

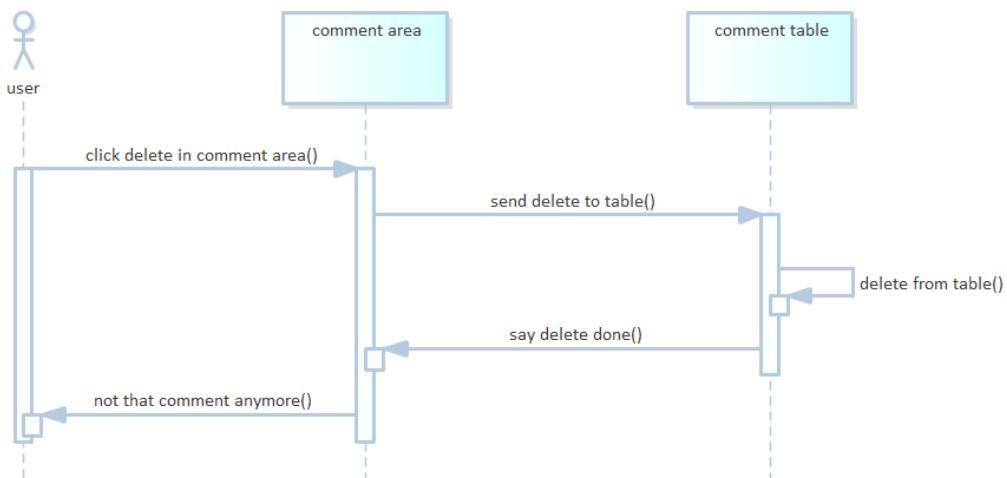
## 6-2- نمودار ترتیب اضافه و حذف کردن کامنت<sup>1</sup>

نمودار ترتیب اضافه و حذف کامنت یا همان نظر در سایت نیز بسیار ساده و گویا میباشد که به ترتیب میتوانید آن ها را در شکل های 17-2 و 18-2 مشاهده کنید.

<sup>1</sup> comment



شکل 2-17. نمودار ترتیب اضافه کردن کامنت



شکل 2-18. نمودار ترتیب حذف کامنت

## 7-2- نتیجه‌گیری

در این فصل با نمودار UML پروژه را مدلسازی کرده و بخش‌های مختلف آن را بررسی و تحلیل کردیم. یک تحلیل نسبی قبل از شروع هر پروژه‌ای لازم و ضروری است. از این نمودارها در بخش‌های مختلف پیاده سازی بهره می‌گیریم. در بخش بعدی به بررسی پایگاه داده و بررسی جرئیبات جداول پایگاه داده می‌پردازیم.

### **فصل ۳:**

## **ساختار داده ها و بانک اطلاعات**

### 3-1- مقدمه

در این بخش به معرفی قسمت عای مختلف پایگاه داده به کاربرده شده در پروژه میپردازیم. شامل جزئیات و روش طراحی جدول‌های پایگاه داده، کلیدهای اصلی و خارجی، معرفی نرم-افزار مدیریت پایگاه داده مورد استفاده، شماره نسخه و غیره. در هر یروژه نرم افزاری نیاز به پایگاه داده مناسب انکار ناپذیر است. پایگاه داده بستره برای ذخیره اطلاعات و سازماندهی آنان توسط جداول و ایجاد ارتباط بین جداول توسط کلید های خارجی و همچنین به کاربر امکان داده ها را بازیابی، ویرایش، ارجاع، یا حذف کند.

### 3-2- طراحی

در این بخش ابتدا نمای کلی پایگاه داده را در شکل 1-3 مشاهده میکنید. سپس جزئیات جداول تک به تک بررسی خواهند شد.

databaseGallery	
<b>gallery3 followers</b>	<b>gallery3 category_posts</b>
↳ id : bigint(20) unsigned ↳ follower_id : int(10) unsigned ↳ following_id : int(10) unsigned 🕒 created_at : timestamp 🕒 updated_at : timestamp	↳ id : bigint(20) unsigned ↳ post_id : int(10) unsigned ↳ category_id : int(10) unsigned 🕒 created_at : timestamp 🕒 updated_at : timestamp
<b>gallery3 hashtags</b>	<b>gallery3 posts</b>
↳ id : bigint(20) unsigned ↳ post_id : int(10) unsigned ↳ hashtag : varchar(255) 🕒 created_at : timestamp 🕒 updated_at : timestamp	↳ id : bigint(20) unsigned ↳ title : varchar(255) ↳ description : longtext ↳ category_pic : longtext 🕒 created_at : timestamp 🕒 updated_at : timestamp 🕒 deleted_at : timestamp
<b>gallery3 migrations</b>	<b>gallery3 comments</b>
↳ id : int(10) unsigned ↳ migration : varchar(255) ↳ batch : int(11)	↳ id : bigint(20) unsigned ↳ user_id : int(10) unsigned ↳ commentable_id : bigint(20) unsigned ↳ user_id : int(10) unsigned ↳ color : varchar(255) 🕒 created_at : timestamp 🕒 updated_at : timestamp 🕒 deleted_at : timestamp
<b>gallery3 password_resets</b>	<b>gallery3 photos</b>
↳ email : varchar(255) ↳ token : varchar(255) 🕒 created_at : timestamp	↳ id : bigint(20) unsigned ↳ post_id : int(10) unsigned ↳ img_url : longtext 🕒 created_at : timestamp 🕒 updated_at : timestamp
<b>gallery3 failed_jobs</b>	<b>gallery3 interests</b>
	↳ id : bigint(20) unsigned ↳ connection : text ↳ queue : text ↳ payload : longtext ↳ exception : longtext 🕒 created_at : timestamp 🕒 updated_at : timestamp 🕒 deleted_at : timestamp
	<b>gallery3 users</b>
	↳ id : bigint(20) unsigned ↳ name : varchar(255) ↳ lastname : varchar(255) ↳ username : longtext ↳ phone : varchar(255) ↳ email : varchar(255) 🕒 email_verified_at : timestamp ↳ password : varchar(255) ↳ remember_token : varchar(100) ↳ role : enum('admin','artist','user','unartisted') ↳ bio : longtext ↳ birthdate : date ↳ profile_pic : longtext ↳ country : varchar(255) ↳ color : varchar(255) 🕒 created_at : timestamp 🕒 updated_at : timestamp 🕒 deleted_at : timestamp

شکل 1-3. نمای کلی جداول پایگاه داده

برای طراحی جداول پایگاه داده از migration در فریم ورک لاراول بهره گرفته شده است. به مثابه‌ی Migrations برای پایگاه داده ایفا نمودن نقش می‌کند و این امکان را برای تیم برنامه نویسی فراهم می‌کند تا به راحتی schema پایگاه داده ای اپلیکیشن را بین اعضای تیم به اشتراک بگذارند. به عبارت دیگر migrations به اعضای تیم کمک می‌کند تا

تغییرات پایگاه داده را تایید ثبت کنند و نیز همیشه یک نسخه‌ی بروز از پایگاه داده در اختیار داشته باشند.

### 3-2-3- جزئیات جدول users

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	<b>id</b>	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT
2	<b>name</b>	varchar(255)	utf8mb4_unicode_ci		No	None		
3	<b>lastname</b>	varchar(255)	utf8mb4_unicode_ci		No	None		
4	<b>username</b>	longtext	utf8mb4_unicode_ci		No	None		
5	<b>phone</b>	varchar(255)	utf8mb4_unicode_ci		Yes	NULL		
6	<b>email</b>	varchar(255)	utf8mb4_unicode_ci		No	None		
7	<b>email_verified_at</b>	timestamp			Yes	NULL		
8	<b>password</b>	varchar(255)	utf8mb4_unicode_ci		No	None		
9	<b>remember_token</b>	varchar(100)	utf8mb4_unicode_ci		Yes	NULL		
10	<b>role</b>	enum('admin', 'artist', 'user', 'unartisted')	utf8mb4_unicode_ci		No	user		
11	<b>bio</b>	longtext	utf8mb4_unicode_ci		Yes	NULL		
12	<b>birthdate</b>	date			Yes	NULL		
13	<b>profile_pic</b>	longtext	utf8mb4_unicode_ci		Yes	NULL		
14	<b>country</b>	varchar(255)	utf8mb4_unicode_ci		Yes	NULL		
15	<b>color</b>	varchar(255)	utf8mb4_unicode_ci		Yes	NULL		
16	<b>created_at</b>	timestamp			Yes	NULL		
17	<b>updated_at</b>	timestamp			Yes	NULL		
18	<b>deleted_at</b>	timestamp			Yes	NULL		

شکل 2-3. جزئیات جدول users

جدول users جزییات مربوط به تمامی کاربران (ادمین یا هنرمند) را در خود نگهداری می‌کند. کاربر هنگام ثبت نام ملزم به پر کردن فیلد های اجباری است تا رکورد آن کاربر در جدول ایجاد شود.

فیلد های اجباری جدول users عبارتند از: id , name , lastname , username , email , password , role البته id به صورت خودکار پر می‌شود و کاربر آن را وارد نمی‌کند و در جدول های دیگر نیز به همین صورت است.

فیلد role هم به صورت خودکار artist قرار داده شده اما در پنل ادمین ، ادمین اصلی سایت میتواند کاربران دیگر را ادمین کند. فیلد username و Email هم باید در جدول برای هر کاربر منحصر به فرد باشد.

### 3-2-3- جزئیات جدول posts

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	<b>id</b> 	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT
2	<b>title</b>	varchar(255)	utf8mb4_unicode_ci		No	None		
3	<b>description</b>	longtext	utf8mb4_unicode_ci		Yes	NULL		
4	<b>gallery_id</b>	int(10)		UNSIGNED	Yes	NULL		
5	<b>user_id</b>	int(10)		UNSIGNED	No	None		
6	<b>color</b>	varchar(255)	utf8mb4_unicode_ci		Yes	NULL		
7	<b>created_at</b>	timestamp			Yes	NULL		
8	<b>updated_at</b>	timestamp			Yes	NULL		
9	<b>deleted_at</b>	timestamp			Yes	NULL		

شکل 3-3. جزئیات جدول posts

این جدول پست هایی که هر کاربر artist می‌داند به سایت اضافه کند را در خود ذخیره می‌کند. می‌بینید که id و user\_id و gallery\_id کلید خارجی های این جدول هستند. gallery\_id در این پروژه کاربردی ندارد و تنها برای توسعه قرار داده جده است که در پایان مستندات بیشتر به آن اشاره خواهیم کرد. و اما user\_id نشان میدهد که این پست را کدام کاربر ایجاد کرده. این یک رابطه oneToMany را نشان میدهد به این معنی که، هر کاربر میتواند تعداد زیادی(many) پست داشته باشد اما هر پست تنها متعلق به یک(one) کاربر است.

این یک رابطه یا relation را بین جداول نشان میدهد. روابط دیگری هم وجود دارد مانند manyToMany ، oneToMany ، oneToOne

جدول پست با جداول دیگری نیز رابطه دارد از جمله:

Photos, categories, hashtags, likes, comments

این ها همه ویژگی هایی هستند که یک پست میتواند داشته باشد، یک پست علاوه بر ویژگی های جدول شکل 3-3 مانند title و description میتواند عکس، دسته بندی، هشتگ، لایک و نظر هم داشته باشد که مورد بررسی قرار خواهد گرفت.

### 3-2-4- جزئیات جدول photos

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	<b>id</b> 📁	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT
2	<b>post_id</b>	int(10)		UNSIGNED	No	None		
3	<b>img_url</b>	longtext	utf8mb4_unicode_ci		No	None		
4	<b>created_at</b>	timestamp			Yes	NULL		
5	<b>updated_at</b>	timestamp			Yes	NULL		

شکل 4-3. جزئیات جدول photos

با توجه به فیلد posts\_id جدول photos رابطه oneToMany با جدول posts دارد. به این معنی که هر post میتواند چندین photo یا همان عکس داشته باشد. اگر میخواستیم پست های ما تنها یک عکس داشته باشند فقط کافی بود فیلد img\_url را به جدول posts اضافه کنیم. اما پست ها این سایت قابلیت داشتن چندین عکس را دارند بنابراین جدول جداگانه ای برای عکس ها در نظر گرفته ایم.

### 3-2-5- جزئیات جدول categories

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	<b>id</b> 🔑	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT
2	<b>title</b>	varchar(255)	utf8mb4_unicode_ci		No	None		
3	<b>description</b>	longtext	utf8mb4_unicode_ci		No	None		
4	<b>category_pic</b>	longtext	utf8mb4_unicode_ci		No	None		
5	<b>created_at</b>	timestamp			Yes	NULL		
6	<b>updated_at</b>	timestamp			Yes	NULL		
7	<b>deleted_at</b>	timestamp			Yes	NULL		

شکل 5-3. جزئیات جدول categories

جدول categories نیز با جدول posts رابطه دارد. اما هیچ کلید خارجی را در این جدول مشاهده نمیکنیم! این بدین خاطر است که رابطه categories با posts به صورت manyToMany است. به این معنی که هر پست میتواند جزء تعداد زیادی دسته بندی باشد و همینطور هر دسته بندی میتواند شامل تعداد زیادی پست باشد که متعلق به آن دسته است. برای این منظور به یک

جدول میانی احتیاج خواهیم داشت که اسم آن را category\_posts گذاشته ایم که در قسمت بعد به آن اشاره خواهد شد.

در نتیجه در یک رابطه manyToMany سه جدول درگیر خواهند بود. در اینجا این سه جدول در نتیجه در یک رابطه manyToMany سه جدول درگیر خواهند بود. در اینجا این سه جدول category\_posts , categories

### 3-2-6- جزئیات جدول category\_posts

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	<b>id</b> 📜	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT
2	<b>post_id</b>	int(10)		UNSIGNED	No	None		
3	<b>category_id</b>	int(10)		UNSIGNED	No	None		
4	<b>created_at</b>	timestamp			Yes	NULL		
5	<b>updated_at</b>	timestamp			Yes	NULL		

شكل 6. جزئیات جدول category\_posts

این جدول ، جدول میانی posts و categories است که category\_posts نام گرفته. این جدول id های دو جدول دیگر را برای به وجود آوردن رابطه manyToMany نگهداری میکند.

### 3-2-7- جزئیات جدول hashtags

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	<b>id</b> 📜	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT
2	<b>post_id</b>	int(10)		UNSIGNED	No	None		
3	<b>hashtag</b>	varchar(255)	utf8mb4_unicode_ci		No	None		
4	<b>created_at</b>	timestamp			Yes	NULL		
5	<b>updated_at</b>	timestamp			Yes	NULL		

شكل 7. جزئیات جدول hashtags

با توجه به فیلد posts\_id جدول hashtags نیز رابطه oneToMany posts با جدول hashtags دارد. به این معنی که هر post میتواند چندین hashtag یا همان برچسب داشته باشد.

Hashtag نیز میتواند نوعی دسته بندی فرعی برای پست‌ها باشد که هر کاربر هنگام ایجاد پست برای همان پست تعدادی هشتگ تعیین میکند. برای مثال با جستجو کردن هشتگ "نقاشی" تمام پست‌هایی که هشتگ "نقاشی" برای آن در نظر گرفته شده باشد فراخوانی میشوند.

### 3-2-8. جزئیات جدول likes

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	<b>id</b>	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT
2	<b>post_id</b>	int(10)		UNSIGNED	No	None		
3	<b>user_id</b>	int(10)		UNSIGNED	No	None		
4	<b>like</b>	tinyint(1)			Yes	NULL		
5	<b>visited</b>	tinyint(1)			No	0		
6	<b>created_at</b>	timestamp			Yes	NULL		
7	<b>updated_at</b>	timestamp			Yes	NULL		

شکل 3-8. جزئیات جدول likes

اشتباه نکنید! چون در این جدول دو کلید خارجی میبینید به این معنی نیست که این یک جدول میانی است و رابطه‌ی manyToMany شکل گرفته. با توجه به مفهوم لايك کردن پست‌ها توسط کاربران این نکته قابل فهم است که جدول لايك یک رابطه users oneToMany با جدول posts دارد. به این معنی که یک لايك متعلق به یک پست و یک کاربر خاص است و اما یک کاربر میتواند تعداد زیادی پست را لايك کند و یک پست میتواند تعداد زیادی کاربر لايك شده باشد.

### 3-2-9. جزئیات جدول comments

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	<b>id</b>	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT
2	<b>user_id</b>	int(10)		UNSIGNED	No	None		
3	<b>commentable_id</b>	bigint(20)		UNSIGNED	No	None		
4	<b>commentable_type</b>	varchar(255)	utf8mb4_unicode_ci		No	None		
5	<b>parent_id</b>	bigint(20)		UNSIGNED	No	0		
6	<b>approved</b>	tinyint(1)			No	0		
7	<b>comment</b>	longtext	utf8mb4_unicode_ci		No	None		
8	<b>visited</b>	tinyint(1)			No	0		
9	<b>created_at</b>	timestamp			Yes	NULL		
10	<b>updated_at</b>	timestamp			Yes	NULL		

شکل 3-9. جزئیات جدول comments

جدول کامنت یا همان نظرات به صورت توسعه پذیر تری طراحی شده و از رابطه morphOne و morphMany در طراحی آن استفاده شده است. این قابلیت در پروژه های بزرگ میتواند خیلی کمک کننده باشد. morphMany و Comment را در مدل morphOne و Comment.php میخواهیم قابلیت "comment" گذاری را داشته باشد میگذاریم که در این پروژه تنها در رابطه با مدل Post به کار گرفته شده است. عکس مربوط به این مدل ها را دی ادامه مشاهده میکنید.

```

File Edit View Navigate Code Refactor Run Tools VCS Window Help Gallery - Comment.php
Gallery / app / Comment.php
Project ▾
  - Comment.php
  - Category.php
  - CategoryPost.php
  - Comment.php
  - Follower.php
  - Hashtag.php
  - Interest.php
  - Like.php
  - Photo.php
  - Post.php
  - User.php
  - bootstrap
  - config
  - database
  - node_modules library root
  - public
  - resources
    - js
    - lang
    - sass
  - views
Comment.php
8  {
9     protected $fillable=['comment','approved','parent_id','commentable_id'];
10
11     //-----//relation call like post()
12
13     // relation one to many revers
14     public function post(){
15         return $this->belongsTo( related: 'App\Post' )->withTrashed();
16     }
17     // relation one to many revers
18     public function user(){
19         return $this->belongsTo( related: 'App\User' )->withTrashed();
20     }
21     //to return post within using $comment->commentable
22     public function commentable(){
23         return $this->morphTo();
24     }
25     // a relation between parent comments and it's replies(childs)
26     public function child(){
27         return $this->hasMany( related: 'App\Comment' , foreignKey: 'parent_id' , localKey: 'id' );
28     }
29     public function parent(){
30         return $this->hasOne( related: 'App\Comment' , foreignKey: 'id' , localKey: 'parent_id' );
31     }
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46

```

شکل 3-10. مدل Comment و رابطه polymorphic

```

File Edit View Navigate Code Refactor Run Tools VCS Window Help Gallery - Post.php
Gallery / app / Post.php
Project ▾
  - Post.php
  - Category.php
  - CategoryPost.php
  - Comment.php
  - Follower.php
  - Hashtag.php
  - Interest.php
  - Like.php
  - Photo.php
  - Post.php
  - User.php
  - bootstrap
  - config
  - database
  - node_modules library root
  - public
  - resources
    - js
    - lang
    - sass
  - views
Post.php
19     // relation one to many
20     // public function comments(){
21     //     return $this->hasMany('App\Comment');
22     //}
23
24     //instead of above we use polymorphic for comments //copy paste this code for other commentable models
25     public function comments(){
26         return $this->morphMany( related: 'App\Comment' , name: 'commentable' );
27     }
28     // relation one to many
29     public function likes(){
30         // i thought it's hasMany
31         return $this->hasMany( related: 'App\Like' );
32     }
33     // relation one to many
34     public function hashtags(){
35         return $this->hasMany( related: 'App\Hashtag' );
36     }
37     // relation many to many ...
38     public function categories(){
39         return $this->belongsToMany( related: Category::class , table: 'category_posts' );
40     }
41     // relation one to many revers
42     public function user(){
43         return $this->belongsTo( related: 'App\User' )->withTrashed();
44     }
45
46

```

شکل 3-11. مدل Post و رابطه polymorphic

فیلد های commentable\_id و commentable\_type نشان میدهد که چه مدلی دارد از قابلیت کامنت استفاده میکند. مثلا در اینجا که مدل مورد نظر ما پست است، commentable\_id ما پستیرا نشان میدهد که در جدول posts قرار دارد و برای آن کامنت گذاشته شده و commentable\_type نیز مشخص میکند که این کامنت مربوط به مدل و جدول پست است و آدرس مدل پست App/Post باید در این فیلد جای گیرد.

برای قابلیت پاسخ دهی یا همان ریپلای<sup>1</sup> به نظرات کاربرد دارد. اگر 0 باشد کامنت Parent\_id اصلی است و reply نیست و گرنگ id کامنت مربوطه از همین جدول در این فیلد قرار میگیرد. Approved برای تایید کامنت از طرف ادمین به کار می رود. به صورت پیشفرض 0 است تا وقتی که ادمین آن را تایید کند تا مقدارش 1 شود و در سایت نمایش داده شود.

هم وقتی یک مشاهده کاربر صفحه نویسیکشن مربوط به کامنت پست خود را مشاهده کند.

### 3-2-10 جزئیات جدول followers

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	<b>id</b>	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT
2	<b>follower_id</b>	int(10)		UNSIGNED	No	None		
3	<b>following_id</b>	int(10)		UNSIGNED	No	None		
4	<b>created_at</b>	timestamp			Yes	NULL		
5	<b>updated_at</b>	timestamp			Yes	NULL		

شکل 3-12. جزئیات جدول followers

هر follower یک کاربر است که به جدول users برمیگردد. با توجه به مفهوم دنبال کردن یا همان فالو کردن یک کاربر follower\_id مربوط به id کسی است که دارد عمل فالو کردن کسی را انجام میدهد و following\_id نیز مربوط به id کسی است که توسط آن شخص اول فالو شده. و هر دو id در جدول users موجود می باشند. رابطه هایی که بین منظور باید در مدل user نوشته شود در شکل 3-13 مشاهده میکنید.(در مدل Follower لازم به نوشتن رابطه ای بین منظور نمی باشد)

```

User.php
public function following() {
    return $this->belongsToMany(related:User::class, table: 'followers', foreignPivotKey: 'follower_id', relatedPivotKey: 'following_id');
}

public function followers() {
    return $this->belongsToMany(related:User::class, table: 'followers', foreignPivotKey: 'following_id', relatedPivotKey: 'follower_id');
}

```

شکل 3-13. مدل User و رابطه با جدول followers

### 3-3- نرم افزار مدیریت پایگاه داده‌ها

پایگاه داده استفاده شده در این پروژه MySQL است. MySQL یکی از مشهورترین سیستم‌های مدیریت دیتابیس است که قابلیت اجرا شدن در بیش از 20 نوع پلت فرم مختلف شامل Linux، Windows را دارد و اغلب به صورت ترکیبی با زبان برنامه‌نویسی PHP استفاده می‌شود. MySQL متن باز یا Open Source است. این مورد بدان معنی است که شما می‌توانید بدون آنکه هزینه‌ای به شخص یا سازمانی پرداخت نمایید، از آن استفاده کنید. شماره نسخه استفاده شده MariaDB-10.4.11 می‌باشد. در این پروژه از نرم افزار XAMPP استفاده شده است. در این نرم افزار سرویس دهنده‌های Apache و MySQL و Perl و PHP نیز به صورت اتوماتیک در هنگام نصب نرم افزار Xampp به سادگی نصب و پیکربندی می‌شوند. نرم افزار Xampp، یک شبیه ساز سرور به صورت لوکال می‌باشد که بسیاری از مشکلات برنامه نویسان و طراحان وب را حل کرده است و به آن‌ها اجازه می‌دهد که بدون نیاز به دسترسی به اینترنت کارهای خود را روی کامپیوتر خود آزمایش کنند. در واقع نرم افزار Xampp کدهای php را به صورت localhost در سیستم شما اجرا می‌کند و با این کار می‌توانید قبل از آپلود کردن کارهای خود روی هاست اصلی، آن‌ها را روی کامپیوتر شخصی خود تست کنید. در قسمت منابع بخش [1] لینک دانلود نرم افزار XAMPP قرار داده شده است.

### 3-4- نتیجه‌گیری

در این فصل جزئیات کامل جداول مورد استفاده در پروژه و روابط بین آن‌ها، کلید‌های اصلی و کلید‌های خارجی و کاربرد فیلد‌های مختلف تعریف شده بررسی شد. و همچنین پایگاه داده مورد استفاده تعیین شد. در فصل بعدی قسمت پیاده سازی پروژه مورد بررسی قرار می‌گیرد و کد نویسی قسمت‌های مختلف با معماری MVC قرار داده شده در لاراول و فولدر بندی‌های در نظر گرفته شده بازگو خواهد شد.

# فصل ۴: پیاده‌سازی

## 4-1- مقدمه

این فصل حاوی جزئیات پیاده‌سازی، ابزارها، نرم‌افزارها، چارچوب‌ها، زبان‌ها و تنظیمات است. همچنین تصاویری از اجراهای نمونه و رخدادها می‌باشد.

## 4-2- کدنویسی

ابزار پیاده‌سازی این پروژه محیط برنامه نویسی PhpStorm<sup>1</sup> می‌باشد. در این محیط کد‌ها با زبان PHP و فریم‌ورک<sup>2</sup> استفاده شده برای PHP فریم‌ورک Laravel نام دارد. لاراول با رعایت معماری MVC طراحی شده است.

فریم‌ورک (FrameWork) که در اصطلاح یک چارچوب نرم افزاری گفته می‌شود مجموعه ای است از کتابخانه‌های برنامه نویسی و مجموعه‌ای از قوانین برای برنامه نویسان است.

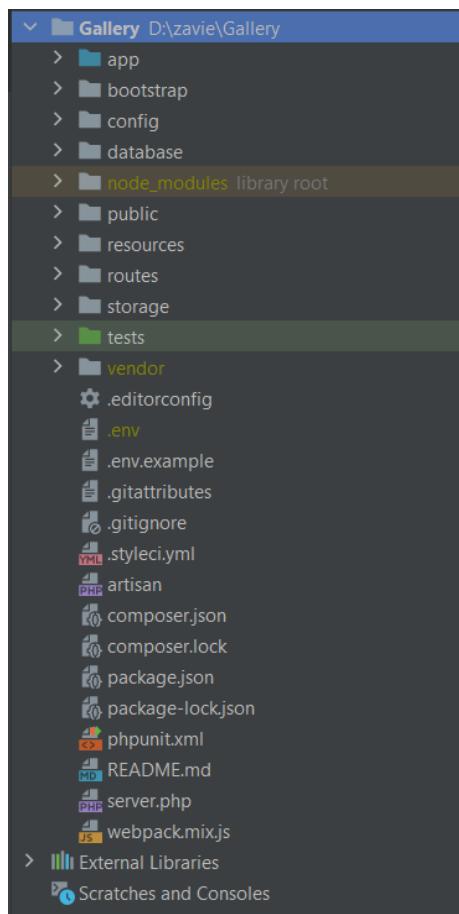
فریم‌ورک یا چهارچوب به برنامه نویسان کمک می‌کند کدهای کمتری را بنویسند و در زمان کمتری به بهترین نتیجه برسند. نسخه لاراول استفاده شده در این پروژه 7 می‌باشد.

عبارت MVC یا Model View Controller و از این است که در هنگام کار کردن با فریم‌ورک ها مورد استفاده قرار می‌گیرد. این عبارت یک شکل خاص از ترکیب کدهای برنامه نویسی است. در این ساختار کدهای برنامه به سه بخش تقسیم می‌شوند. با تقسیم و طبقه‌بندی کدها، برنامه به صورت سه لایه مجزا از هم در می‌آید. برنامه نویسی لایه ای مزایای بسیاری دارد و در این مقاله نمی‌گنجد. تنها نکته‌ای که می‌توان در اینجا بیان کرد این است که در برنامه نویسی MVC بخش Model کار ذخیره و بازیابی اطلاعات را بر عهده دارد و بخش View وظیفه‌ی نمایش اطلاعات به کاربر را بر عهده دارد و بخش Controller دریافت اطلاعات از کاربر و پردازش را بر عهده دارد.

فریم‌ورک لاراول پس از نصب پوشه بندی‌هایی در اختیار شما می‌گذارد که کد نویسی را به خوبی ساختار بندی می‌کند. نام رایج‌تر پوشه در برنامه نویسی دایرکتوری می‌باشد. تصویر کلی ساختار لاراول را در شکل 4-1 مشاهده می‌کنید.

<sup>1</sup> <https://www.jetbrains.com/phpstorm/>

<sup>2</sup> framework

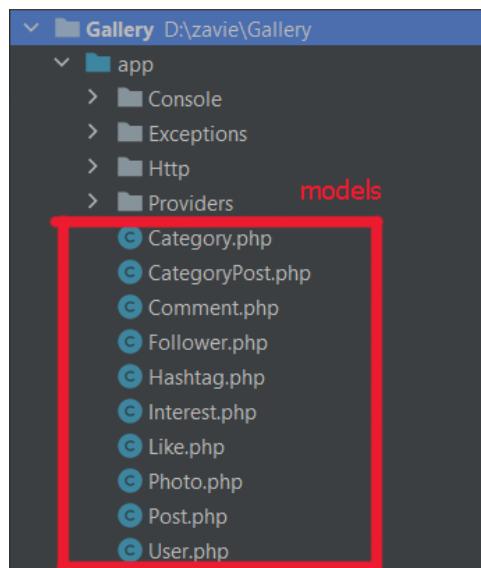


شکل 4-1. پوشه بنده لاراول

در ادامه به بررسی دایرکتوری های استفاده شده و اصلی تر در پروژه می پردازیم.

## 4-2-4. دایرکتوری app

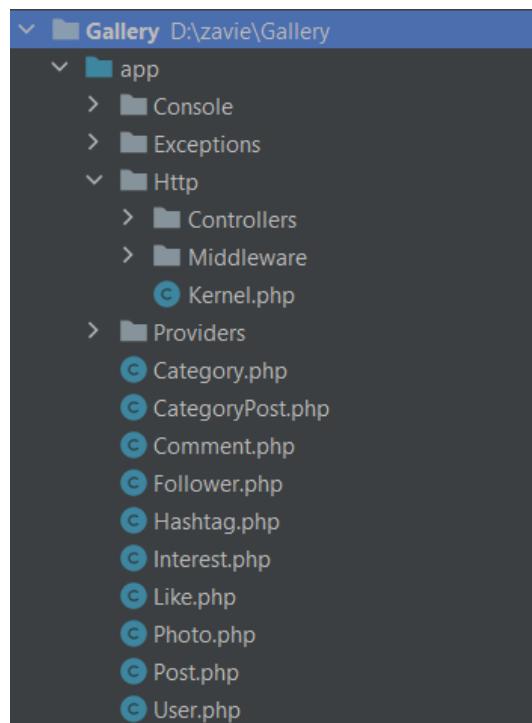
پوشه APP پوشه اصلی برنامه می باشد. و تقریبا تمام فایل های اصلی و هسته برنامه داخل این پوشه میباشد. بطور پیشفرض namespace ی که لاراول برای این دایرکتوری قرار داده است APP میباشد. این پوشه شامل پوشه های مختلف دیگری نیز هست که به بیان آن می پردازیم.



شکل 2-4. دایرکتوری app

ابتدا مشاهده می‌کنیم که دارکتوری app به طور مستقیم محل قرار گیری مدل‌های ساخته شده در برنامه است. البته می‌توان به طور دستی در همین محل پوشه‌ای به اسم models تعریف کرد و مدل‌ها را در آن قرار داد اما در این پروژه ضرورتی دیده نشده است.

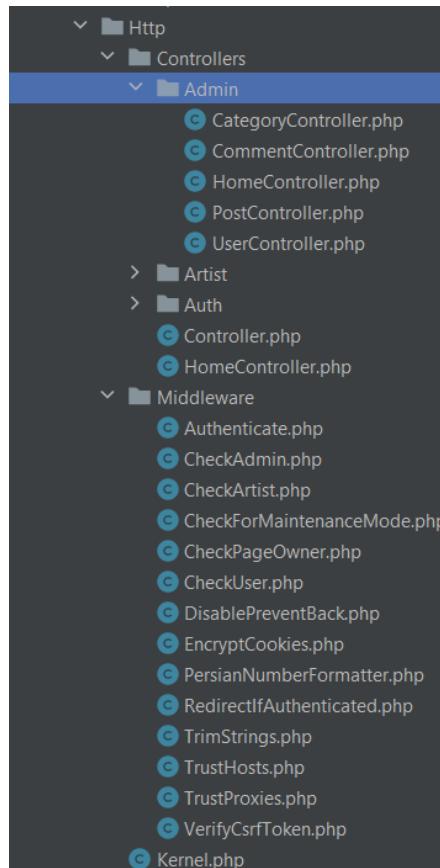
#### 4-2-2-2- دایرکتوری Http واقع در دایرکتوری app



شکل 4-3. دایرکتوری Http

پوشه Http شامل پوشه های controller ها و درخواست های اپلیکیشن است. پوشه middleware یک میان افزار است که مکانیزم فیلتر و ارتباط بین درخواست ها و پاسخ را شامل می شود.

در شکل 4-4 میتوانید جزئیات دایرکتوری های controller و middleware را مشاهده کنید.



شکل 4-4. دایرکتوری Controllers و Middleware

#### 4-2-3- دایرکتوری config

این پوشه شامل تمام فایل های پیکر بندی و تنظیمات و پارامتر های مربوط به کارکرد صحیح یک برنامه لاراول می باشد. در تصویر زیر می توانید فایلهای موجود در این دایرکتوری را ببینید. قابل ذکر است که نام فایلها با عملکرد آنها مرتبط میباشد.

#### 4-2-4- دایرکتوری database

همانطور که از آن پیداست این دایرکتوری شامل پارامتر های مختلف برای مدیریت پایگاه داده

است. که شامل سه زیر پوشه است :

- Seeds شامل کلاس هایی برای تست دیتابیس میباشد.

- Migrations بوسیله مایگریشن ها میتوان برای جدول های دیتابیس خود را طراحی کرده و آنها را مدیریت کنید.

- Factories بوسیله factory ها میتوان برای داده های ساختگی برای مدل های برنامه ایجاد کرد.

از migration در این قسمت بهره ای بیشتری گرفته شده که در شکل 4-5 میتوانید های ساخته شده و همچنین migration جدول users را به عنوان نمونه مشاهده کنید.

```

class CreateUsersTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('lastname');
            $table->longText('username');
            $table->string('phone')->nullable();
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->rememberToken();
            $table->enum('role',['admin','artist','user','unregistered'])->default('user');
            $table->longText('bio')->nullable();
            $table->date('birthday')->nullable();
            $table->longText('profile_pic')->nullable();
            $table->string('country')->nullable();
            $table->string('color')->nullable();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('users');
    }
}

```

شکل 4-5. دایرکتوری migrations –migration users

## 4-4-5. دایرکتوری public

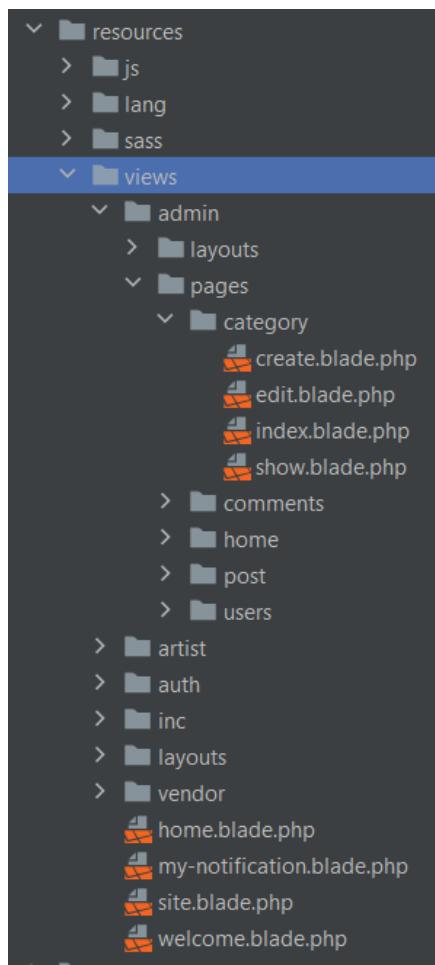
این پوشه شامل فایل ها و پوشه های زیر است:

- فایل htaccess، بوسیله این فایل یکسری تنظیمات به سرور مون اعمال میکنیم.
- CSS و JS در دایرکتوری public همچنین asset های برنامه مانند javascript و یا javascript قرار میگیرد.
- index.php این فایل برای راه اندازی برنامه ، لازم و حیاتی و اولین نقطه برای درخواستهای وارد شده به برنامه است.

## 4-2-6- دایرکتوری resources

این دایرکتوری برای بهبود هر چه بیشتر برنامه وب شما می‌باشد و شامل فولدر‌ها و فایل‌هایی هست که در اینجا بیان می‌کنیم:

- assets این پوشه حاوی فایل‌های کامپایل نشده LESS و SCSS بوده که برای استایل دهی برنامه مورد نیاز می‌باشند.
  - lang فایل‌های مر بوط به زبان برنامه در این دایرکتوری قرار می‌گیرد.
  - views ویو‌ها یکی از نقش‌های اصلی در معماری mvc را بازی می‌کنند که شامل فایلهای template و html می‌باشد که با کاربر تعامل ایجاد می‌کنند.
- در شکل 4-6 نمونه‌ای جزئیات بخش view‌ها را مشاهده می‌کنید.



شکل 6-4. دایرکتوری resources

## 4-2-4- دایرکتوری routes

در برنامه هایی که با فریمورک لاراول توسعه داده شده، درخواست های فرستاده شده از url را قسمت های خاصی از برنامه بررسی می کنند. اینکه کدام درخواست در کدام قسمت از برنامه باید بررسی بشود را روترا مشخص می کنند. در واقع مسیریابی درخواست ها را روترا انجام می دهند. اگر درخواستی داشته باشیم که مسیری برای آن وجود نداشته باشد با خطای 404 مواجه می شویم.

برای routing در این پروژه از فایل web.php واقع در دایرکتوری routes استفاده شده است. نمونه ای از کد این قسمت را در شکل 7-4 مشاهده کنید.

```

24 //Route::get('/home', 'HomeController@index')->name('home');
25
26 //my routes
27
28 //-----artist
29
30 //Route::resource('artist/post','Artist\PostController');
31 //todo: add middleware auth & is artist
32 Route::prefix('artist')->namespace( value: 'Artist' )->name( value: 'artist.' )->middleware(['auth','isArtist'])
33
34 ->group(function (){
35     Route::resource( name: 'post', controller: 'PostController' );
36
37     //-----post
38     Route::patch( url: 'post/update-hashtag/{post}', action: 'PostController@updateHashtag' )->name( name: 'post.updateHashtag' );
39     Route::patch( url: 'post/update-category/{post}', action: 'PostController@updateCategory' )->name( name: 'post.updateCategory' );
40     Route::get( url: 'post/edit-profile/{user}', action: 'PostController@editProfile' )->name( name: 'post.editProfile' );
41     Route::patch( url: 'post/update-profile/{user}', action: 'PostController@updateProfile' )->name( name: 'post.updateProfile' );
42     Route::patch( url: 'post/update-profilePic/{user}', action: 'PostController@updateProfilePic' )->name( name: 'post.updateProfilePic' );
43
44     //-----home
45     Route::get( url: 'home/post', action: 'HomeController@index' )->name( name: 'home_index' );//show following users post or all
46     Route::get( url: 'home/post/{user}', action: 'HomeController@index_user' )->name( name: 'home_index_user' );//show following users post or all
47     Route::post( url: 'home/follow-unfollow/{follower}' ,{following}, action: 'HomeController@follow_unfollow' )->name( name: 'home_follow_unfollow' );
48     Route::get( url: 'home/show-followings/{user}', action: 'HomeController@showFollowings' )->name( name: 'home_showFollowings' );
49     Route::get( url: 'home/show-followers/{user}', action: 'HomeController@showFollowers' )->name( name: 'home_showFollowers' );
50     Route::get( url: 'home/popular', action: 'HomeController@index_popular' )->name( name: 'home_index_popular' );
51
52     //-----notifications
53

```

شکل 4-7. دایرکتوری routes

## 4-2-4. دایرکتوری vendor

این دایرکتوری تمام dependency ها یا وابستگی ها و پکیج های مربوط به کامپوزر<sup>1</sup> را در بر میگیرد.

## 4-3-4. اجرای نرم افزار

کاربر برای دسترسی به سایت ابتدا باید ورود یا ثبت نام کند. در این بخش فرایند های مهم سایت را مورد بررسی قرار میدهیم که ورود و ثبت نام یکی از آن موارد هستند.

<sup>1</sup> composer

### 4-3-4- فرایند های مهم سایت

#### 4-3-1-1 login-register فرایند



شکل 4-8. ظاهر صفحه register در سایت

در این شکل ظاهر صفحه ثبت نام را مشاهده میکنید. کاربر اگر به درستی فیلد های فوق را پر کند در سایت ثبت نام خواهد شد. در لاراول یک سیستم احراز حیث<sup>1</sup> از پیش تعیین شده نیز وجود دارد که ما با توجه به نیاز های سایت خود میتوانیم آن ها را تغییر بدھیم. در ادامه کد های مربوط به این بخش را میتوانید بررسی و مشاهده کنید.

<sup>1</sup> authentication

```

1 @extends('layouts.app')
2 @section('custom-css') <link href="{{url('assets/auth/css/auth_style.css')}}" rel="stylesheet" @endsection
3 @section('content')
4 <div class="container">
5   <div class="row justify-content-center">
6     <div class="col-md-8">
7       <div class="card card-background-style">
8         <div class="card-header"><h2 class="pt-1">ثبت نام</h2>
9         <p class="gray-comments">بپ جمع ما خوش آمدید!</p>
10        <div class="card-body">
11          <form method="POST" action="{{ route('register') }}>
12            @csrf
13            <div class="flex-main">
14              <div class="form-group row">
15                <div class="form-group row">
16                  <div class="form-group row">
17                    <div class="form-group row">
18                      <div class="form-group row last-form-input">
19                        <div class="form-group row">
20                          </div>
21                        </div>
22                      </div>
23                    </div>
24                  </div>
25                </div>
26              </div>
27            </form>
28          </div>
29        </div>
30      </div>
31    </div>
32  </div>
33 </div>
34 </div>
35 </div>
36 </div>
37 </div>
38 </div>
39 </div>
40 </div>
41 </div>
42 </div>
43 </div>
44 </div>
45 </div>
46 </div>
47 </div>
48 </div>
49 </div>
50 </div>
51 </div>
52 </div>
53 </div>
54 </div>
55 </div>
56 </div>
57 </div>
58 </div>
59 </div>
60 </div>
61 </div>
62 </div>
63 </div>
64 </div>
65 </div>
66 </div>
67 </div>
68 </div>
69 </div>
70 </div>
71 </div>
72 </div>
73 </div>
74 </div>
75 </div>
76 </div>
77 </div>
78 </div>
79 </div>
80 </div>
81 </div>
82 </div>
83 </div>
84 </div>
85 </div>
86 </div>
87 </div>
88 </div>
89 </div>
90 </div>
91 </div>
92 </div>
93 </div>
94 </div>
95 </div>
96 </div>
97 </div>
98 </div>
99 </div>
100 </div>
101 </div>
102 </div>
103 </div>
104 </div>
105 </div>
106 </div>
107 </div>
108 </div>
109 </div>
110 </div>
111 </div>
112 </div>
113 </div>
114 </div>

```

شکل 4-9 Register view codes .

کد های بالا مربوط به طراحی ظاهر فرم و قرار دادن فیلد ها و استایل دهی به آن ها همچنین خطای validation در صورت وجود می باشد. همه این ها در آخر ظاهر شکل 4-8 را به ما میدهد.

قسمت مهم این کد action فرم است. که ما را به (register) route هدایت میکند. این route در web.php معرفی شده است. البته شد ما از authentication لاراول استفاده میکنیم تمامی های مربوط به auth در

Auth::routes(['verify'=>true]);  
خلاصه شده اند.

در کنترولر register ابتدا کد های مربوط به validation را بررسی میکنیم.

```

protected function validator(array $data)
{
    return Validator::make($data, [
        'name' => ['required', 'alpha', 'max:255'],
        'email' => ['required', 'string', 'email', 'max:255',
        'unique:users', 'ends_with:.com'],
        'password' => ['required', 'string', 'min:8',
        'confirmed', 'not_regex:/^.*(?=.*[;,\'\']).*$/' ],
        'lastname' => ['required', 'alpha'],
        'username' => ['required', 'alpha_dash', 'unique:users'],
        'phone' => ['required', 'numeric', 'starts_with:09'],
    ]);
}

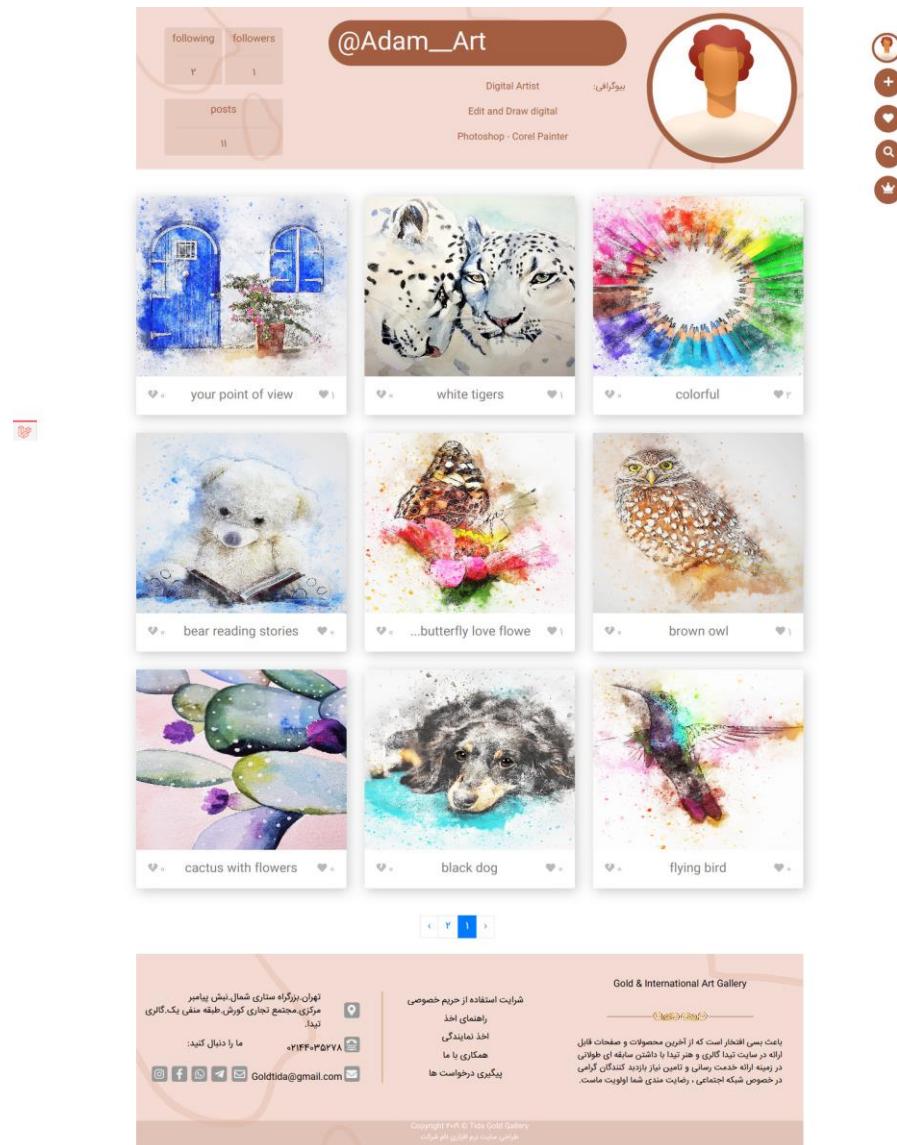
```

```
    ]) ;
}
```

سپس در قسمت بعد ساخت user جدید صورت می‌گیرد.

```
protected function create(array $data)
{
    return User::create([
        'name' => $data['name'],
        'email' => $data['email'],
        'password' => $data['password'],
        'lastname' => $data['lastname'],
        'username' => $data['username'],
        'phone' => $data['phone'],
        'role' => 'artist',
        //todo: add other fields
    ]);
}
```

سپس کاربر پس از ثبت نام به صفحه اصلی سایت هدایت می‌شود. ما نمونه‌ای از صفحه اصلی سایت را در شکل در ادامه نشان خواهیم داد برای کاربری که قبل از ثبت نام کرده و تعدادی پست گذاشته.



شکل ۱۰-۴. صفحه اصلی هنرمند سایت full page

### 4-3-1-2- قابلیت responsive بودن صفحات سایت

با توجه به گسترش روز افزون استفاده از موبایل و وبگردی کاربران از طریق موبایل، امر Bودن وبسایت ها بسیار مهم و ضروری میباشد. به سایت هایی Responsive یا واکنشگرا گویند که در دستگاههای با اندازه مختلف از جمله کامپیوتر ها، لپ تاپ، تبلت ها، همچنین موبایل و... نمایش مطلوب و با ظاهری مناسب داشته باشد.

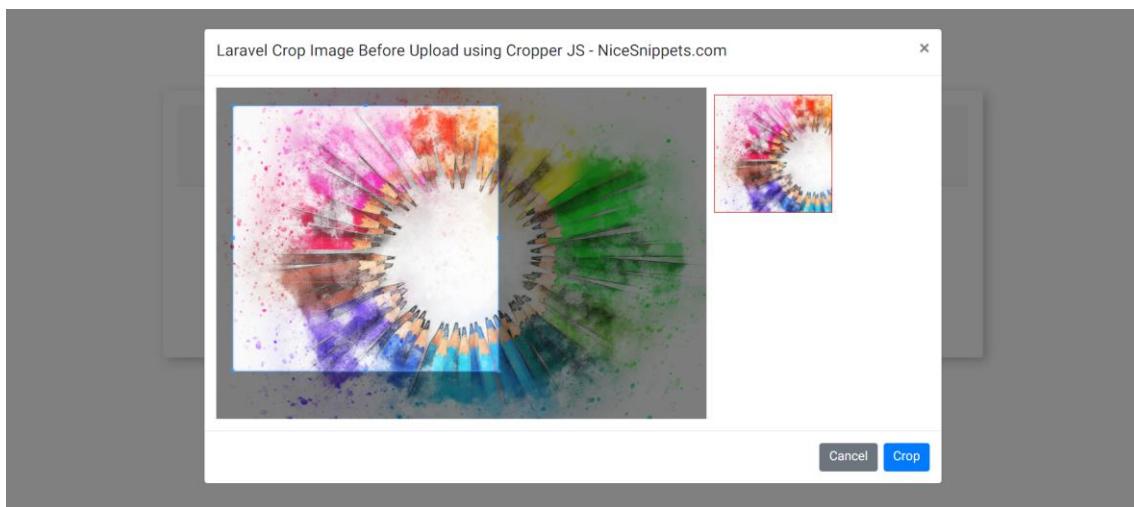
صفحات سایت گالری نیز به صورت responsive طراحی شده که نمونه لپ تاپ و موبایل آن در شکل ۱۱-۴ قابل مشاهده است.



شکل 4-11. طراحی responsive سایت

#### 4-3-1-3- استفاده از imageCropper در پروژه

برای اضافه کردن عکس پست برای هنرمند قابلیت crop عکس در نظر گرفته شده است. کد های مربوط به js و controller این قسمت برای شما قرار داده شده است.



شکل 4-12. Crop کردن عکس قبل از آپلود

```

use ...;

class ImageCropperController extends Controller
{
    public function index($id)
    {
        return view('artist.pages.posts.cropper', ['post_id' => $id]);
    }

    public function upload(Request $request, $id)
    {
        $folderPath = public_path('storage/postphotos/');

        $image_parts = explode( delimiter: ";base64,", $request->image);
        $image_type_aux = explode( delimiter: "image/", $image_parts[0]);
        $image_type = $image_type_aux[1];
        $image_base64 = base64_decode($image_parts[1]);
        $filename=uniqid() . '.jpeg';
        $file = $folderPath . $filename;

        file_put_contents($file, $image_base64);
        Photo::create([
            'post_id'=>$request->input('post_id'),
            'post_id'=>$id,
            'img_url'=>'storage/postphotos/' . $filename,
        ]);
        return response()->json(['success'=>'success']);
    }
}

```

\App\Http\Controllers\Artist

شکل ۴-۱۳. کد مربوط به controller imageCropper

```

<script>

var $modal = $('#modal');
var image = document.getElementById('image');
var cropper;

$( "body" ).on("change", ".image", function(e) {
    var files = e.target.files;
    var done = function (url) {
        image.src = url;
        $modal.modal('show');
    };
    var reader;
    var file;
    var url;

    if (files && files.length > 0) {
        file = files[0];

        if (URL) {
            done(URL.createObjectURL(file));
        } else if (FileReader) {
            reader = new FileReader();
            reader.onload = function (e) {
                done(reader.result);
            };
            reader.readAsDataURL(file);
        }
    }
});

```

```

        }
    });

$modal.on('shown.bs.modal', function () {
    cropper = new Cropper(image, {
        aspectRatio: 1,
        viewMode: 3,
        preview: '.preview'
    });
}).on('hidden.bs.modal', function () {
    cropper.destroy();
    cropper = null;
});

$("#crop").click(function() {
    canvas = cropper.getCroppedCanvas({
    });

    canvas.toBlob(function(blob) {
        url = URL.createObjectURL(blob);
        var reader = new FileReader();
        reader.readAsDataURL(blob);
        reader.onloadend = function() {
            var base64data = reader.result;

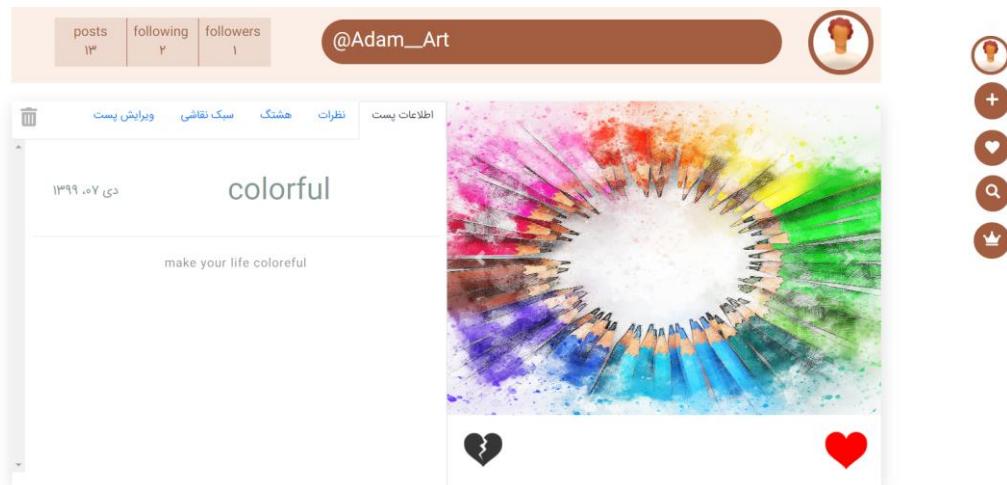
            $.ajax({
                type: "POST",
                dataType: "json",
                url: "upload/{{post_id}}",
                data: {'_token': '_token'],
            }).attr('content', 'image': base64data},
                success: function(data){
                    $modal.modal('hide');
                    // alert("آپلود موفقیت با شما عکس");
                    Swal.fire({
                        title: 'تبریک',
                        text: 'آپلود موفقیت با شما عکس',
                        icon: 'success',
                        confirmButtonText: 'باشه'
                    })
                }
            );
        }
    });
});

</script>

```

در بالا نیز کد script مربوط به cropper را مشاهده می‌کنید که با انتخاب عکس و کلیک بر روی دکمه crop عکس بدون refresh صفحه آپلود می‌شود و پیغام موقیت‌نمایش داده می‌شود.

#### 4-3-1-4- قابلیت refresh بدون like-dislike-unlike-undislike صفحه



شکل 4-14. قابلیت like در سایت

کد زیر در بخش ویو این صفحه قرار می‌گیرد.

```
<div class="interaction">
    <a id="likeBtn" href="#" class="btn like heart like_btn like_ui"
    {{Auth::user()->likes()->where('post_id', $post->id)->first() ? 
Auth::user()->likes()->where('post_id', $post->id)->first()->like == 1
? 'active' : 'deactive' : 'deactive'}}">
        @if(Auth::user()->likes()->where('post_id', $post->id)-
>first())
            @if(Auth::user()->likes()->where('post_id', $post->id)-
>first()->like == 1)
                You like this post
            @else
                Like
            @endif
        @else
            Like
        @endif
    </a>
    <a id="dislikeBtn" href="#" class="btn like broken-heart like_ui
dislike_btn" {{ Auth::user()->likes()->where('post_id', $post->id)-
>first() ? Auth::user()->likes()->where('post_id', $post->id)-
>first()->like == 0 ? 'active' : 'deactive' : 'deactive' }}">
        @if(Auth::user()->likes()->where('post_id', $post->id)-
>first())
            @if(Auth::user()->likes()->where('post_id', $post->id)-
>first()->like == 0)
                You do not like this post
            @else
                Dislike
            @endif
        @else
            Dislike
        @endif
    </a>
</div>
```

سپس این قسمت را به پایین کد blade یا همان ویو مربوطه اضافه می‌کنیم.

```
<script>
    var token = '{{ Session::token() }}';
    var urlLike = '{{ route('like') }}'; <!-- in web.php route like --
>
</script>
```

کد فوق به آدرس زیر در web.php اشاره می‌کند.

```
Route::post('/like', 'Artist\PostController@postLikePost')-
>name('like');
```

سپس کد مربوط به ajax را مشاهده خواهید کرد.

```
$('.like').on('click', function (event) {
    event.preventDefault();
    postId = event.target.parentNode.parentNode.dataset['postid'];
    var isLike = event.target.previousElementSibling == null ? true :
false;
    //send the ajax request
    $.ajax({
        method: 'POST',
        url: urlLike,
        data: {isLike: isLike, postId: postId, _token: token}
    })
        .done(function () {
            //change the page
            event.target.innerText = isLike ?
event.target.innerText == 'Like' ? 'You like this post' : 'Like' :
event.target.innerText == 'Dislike' ? 'You do not like this post' :
'Dislike';
            if (isLike) {
                event.target.nextElementSibling.innerText =
'Dislike';
            } else {
                event.target.previousElementSibling.innerText =
'Like';
            }
        });
    if ($(this).hasClass("active")){
        $(this).removeClass("active")
    }
    else{
        $(".like").removeClass("active");
        $(this).addClass("active")
    }
});
```

و در آخر کد controller این بخش را میتوانید مشاهده کنید.

```
public function postLikePost(Request $request){
    $post_id = $request['postId'];
    $is_like = $request['isLike'] === 'true';
    $update=false;
    $post=Post::find($post_id);

    if (!$post){
        return null;
    }
```

```

$user= Auth::user();
//get all the likes this user had
$like=$user->likes()->where('post_id', $post_id)->first();
if ($like) {
    //already liked this post
    $already_like=$like->like; //accessing the value
    $update=true;

    if ($already_like == $is_like){
        //undo the liking
        $like->delete();
        return null;
    }
} else{
    $like = new like();
}
$like->like = $is_like;
$like->user_id = $user->id;
$like->post_id = $post->id;
if ($update){
    $like->update();
} else{
    $like->save();
}
return null;
}

```

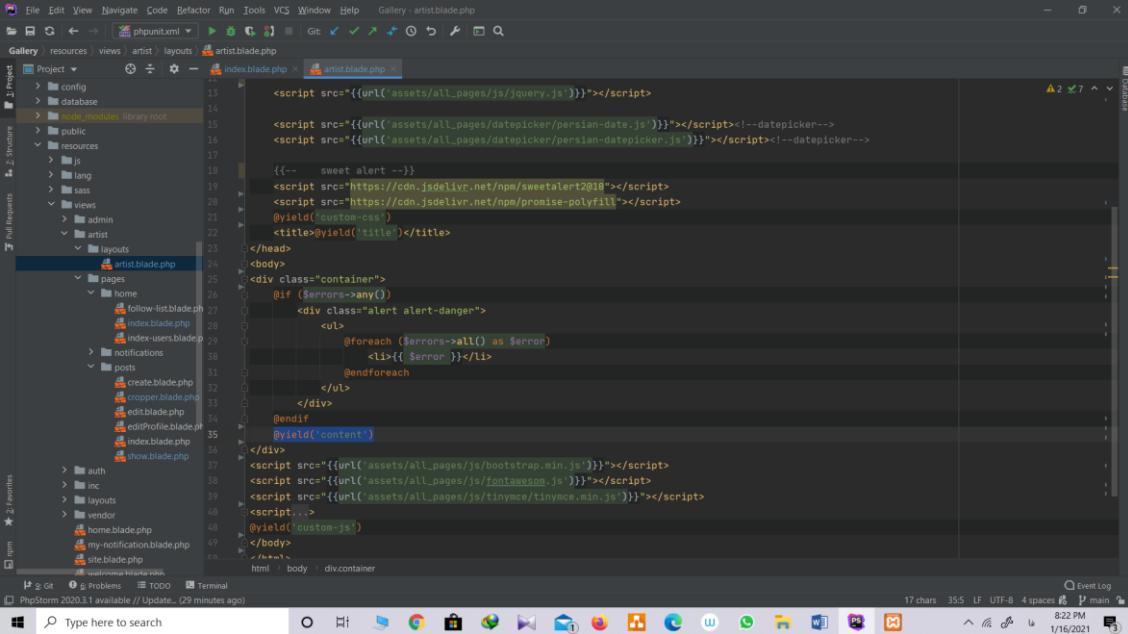
#### 4-3-1-5 جستجو

قسمت مربوط به جستجو سایت بخش های مختلفی برای جستجو دارد. در شکل 4-15 نیز میتوانید بخش های مختلف جستجو را مشاهده کنید. برا بیان بیشتر کاربر هنرمند میتوانست نlm کاربر را از جدول کاربران و یا پستی را بر اساس عنوان و یا هشتگ های استفاده شده در پست از جدول پست ها جستجو کند. همچنین میتواند در قسمت سبک اثر پستی را بر اساس دسته بندی اش از منوی دسته بندی های قرار داده شده در فرم جستجو یا همان فیلتر<sup>1</sup> کند.



شکل 4-15. بخش جستجو برای هنرمند

کد مربوط به این قسمت به صورت include در layout اصلی صفحه قرار داده شده. برای جداسازی تکه ای از کد برای استفاده بهینه از زمان و خوانا شدن کد و ننوشتن کد تکراری در متن به کار می‌رود. Layout خم تعریفی مشابه اما کاربردی متفاوت دارد. در include تکه ای از کد را برداشته و در صورت نیاز آن را در صفحات مختلف فراخوانی می‌کنیم. اما در layout صفحه ای html با دسترسی های مورد نیاز در همه صفحات مانند فراخوانی کتابخانه و استایل های مورد نیاز طراحی می‌کنیم و بخش تغییر پذیر سایت را در yield قرار میدهیم و صفحات مختلف را با های مختلف طراحی می‌کنیم. این بخش ها را نیز علاوه بر کد جستجو در ادامه مشاهده می‌کنید.



```

<?php
    @endphp
    <head>
        <script src="{{url('assets/all_pages/js/jquery.js')}}"></script>
        <script src="{{url('assets/all_pages/daterangepicker/persian-date.js')}}"></script><!--datepicker-->
        <script src="{{url('assets/all_pages/daterangepicker/persian-datepicker.js')}}"></script><!--datepicker-->
        {{-- sweet alert --}}
        <script src="https://cdn.jsdelivr.net/npm/sweetalert2@10"></script>
        <script src="https://cdn.jsdelivr.net/npm/.promise-polyfill"></script>
        @yield('custom-css')
        <title>@yield('title')</title>
    </head>
    <body>
        <div class="container">
            @if ($errors->any())
                <div class="alert alert-danger">
                    @foreach ($errors->all() as $error)
                        <li>{{ $error }}</li>
                    @endforeach
                </div>
            @endif
            @yield('content')
        </div>
        <script src="{{url('assets/all_pages/js/bootstrap.min.js')}}"></script>
        <script src="{{url('assets/all_pages/js/fontawesome.js')}}"></script>
        <script src="{{url('assets/all_pages/js/tinymce/tinymce.min.js')}}"></script>
        @yield('custom-js')
    </body>
<?php
    @endphp

```

شکل 4-16. اصلی کاربر هنرمند Layout

```

@extends('artist.layouts.artist')

@section('custom-css')
    <link rel="stylesheet" href="{{url('assets/artist/css/index_style.css')}}"><!--custom-->
@endsection

@section('title') artist homepage @endsection

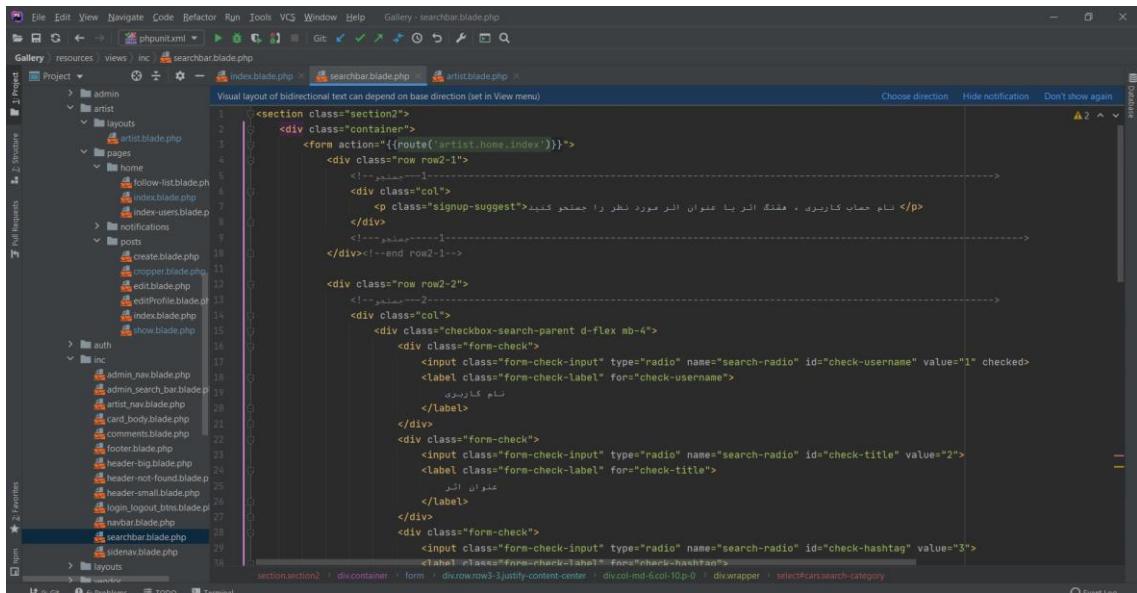
@section('content')
    @include('inc.sidenav')

    @include('inc.searchbar')


{{-- main --}}
<div class="main">
    <div class="row posts-box">
        @foreach($posts as $post)
            <div class="col-md-4" data-aos="fade-right" data-aos-duration="2000">
                <div class="tag-parent">
                    <a href="{{route('artist.post.show',['post'=>$post->id])}}" class="show-img-style">
                        <div class="card card-index" >
                            {{-- image--}}
                            @if($post->photos->first())
                                {{-- <div class="col-md-3 pl-0 pr-0 mr-0 ml-0">--}}
                                <div class="post-img-parent">
                                    photos->first()->img_name}}"/>
                            @endif
                        </div>
                </div>
            </div>
        @endforeach
    </div>
</div>

```

شکل 4-17. استفاده از include برای جستجو



شکل 4-18. کد مربوط به جستجو کاربر هنرمند بخش اول

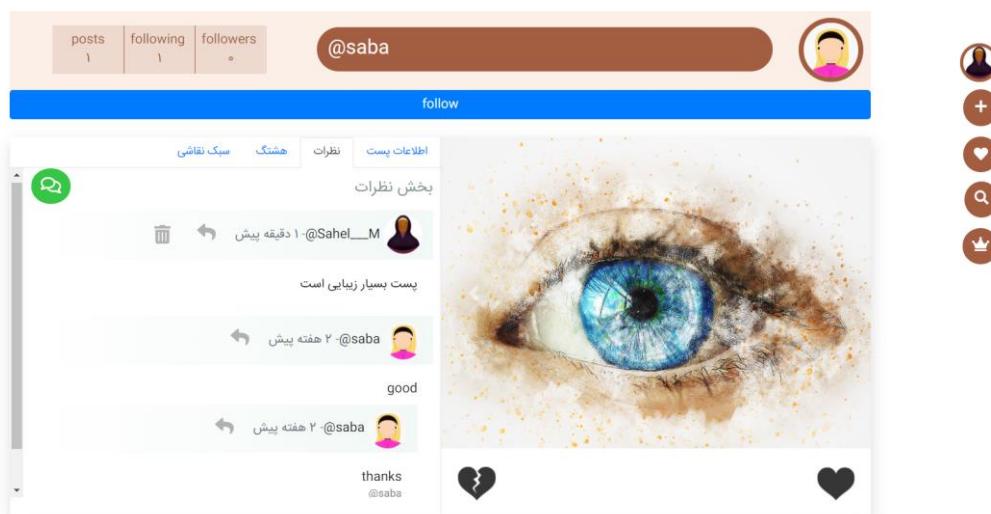
```
Gallery | resources | views | inc | seachbar.blade.php
index.blade.php | seachbar.blade.php | artist.blade.php

Visual layout of bidirectional text can depend on base direction (set in View menu)
Choose direction Hide notification Don't show again
1<-- Search btn -->
2<div class="btn-search-parent">
3    <button type="submit" class="btn btn-warning btn-block">i class="fas fa-search"</i></button>
4</div>
5<!-- p_d p_h -->
6</div><!-- end row2-3-->
7
8</form>
9
10<form action="{{route('artist.home.index')}}">
11<div class="row row3 justify-content-center">
12    <div class="col-md-6 col-10 p-0">
13        <div class="wrapper">
14            <select class="search-category" name="categories" id="cars" onfocus='this.size=5;' onblur='this.size=1;' onchange='this.size=1; this.blur(); this.form.<input type="text" value="Search" name="q" style="width: 100%; height: 100%; border: none; border-bottom: 1px solid #ccc; outline: none; font-size: 1em; font-weight: bold;">'>
15                <option value=""><!-- اسکرچ -->
16                @foreach( \App\Category::get() as $category )
17                    <option value="{{$category->id}}>{{($category->title)}}
18                @endforeach
19            </select>
20        </div>
21    </div>
22</div><!-- end row3-3-->
23
24</form>
25
26</div><!-- end container2-->
27</section><!-- end section2-->
```

شکل 19-4. کد مربوط به جستجو کاربر هنرمند بخش دوم

6-1-3-4-نظر دھی

کاربری که در سایت ثبت نام کرده باشد میتواند در بخش نظرات نظر ثبت کند. درخواست ثبت نظر به صورت ajax و در نتیجه بدون رفرش صفحه انجام میپذیرد اما در سایت نمایش داده نمیشود تا وقتی که ادمین سایت آن را تایید کند. کامنت ها قابل پاسخ دهی نیز هستند. صاحب اصلی پست میتواند کامنت های خود و دیگران را برای پست خود پاک کند. اما کاربران دیگر فقط کامنتی که خودشان گذاشته اند را میتوانند پاک کنند. کد مربوطه را در ادامه میتوانید مشاهده کنید.



شکل 4-20. ظاهر بخش نظردهی سایت

```

<script>
{{-- for ajax comment box --}}
$('#sendComment').on('show.bs.modal', function (event) {
    var button = $(event.relatedTarget) // Button that triggered the modal
    let parent_id = button.data('id')
    console.log(parent_id)

    // If necessary, you could initiate an AJAX request here (and then do the updating in a callback).
    // Update the modal's content. We'll use jquery here, but you could use a data binding library or other methods instead.
    var modal = $(this)
    modal.find('input[name="parent_id"]').val(parent_id)
})
//for ajax form request
//java script
document.querySelector('#sendCommentForm').addEventListener('submit',function (event){
    event.preventDefault(); //form data doesn't send and stop here with submit
    let target=event.target;//target is our form
    // console.log(target.querySelector('input[name=commentable_id]'))
    let data={
        commentable_id : target.querySelector('input[name=commentable_id]').value,
        commentable_type : target.querySelector('input[name=commentable_type]').value,
        parent_id : target.querySelector('input[name=parent_id]').value,
        comment : target.querySelector('textarea[name=comment]').value,
    }
    //validation
    // if(data.comment.length < 2){
    //     console.error('plz enter comment more than 2 char')
    //     return ;
    // }
    // ajax request with jquery library not axios
})
script

```

شکل 4-21. کد script مربوط به بخش نظرات بخش اول

```

// ajax request with jquery library not axios
// access in layout blade meta tag
$.ajaxSetup({
    headers:{
        'X-CSRF-TOKEN' : document.head.querySelector('meta[name="csrf-token"]').content,
        'Content-Type' : 'application/json'
    }
})

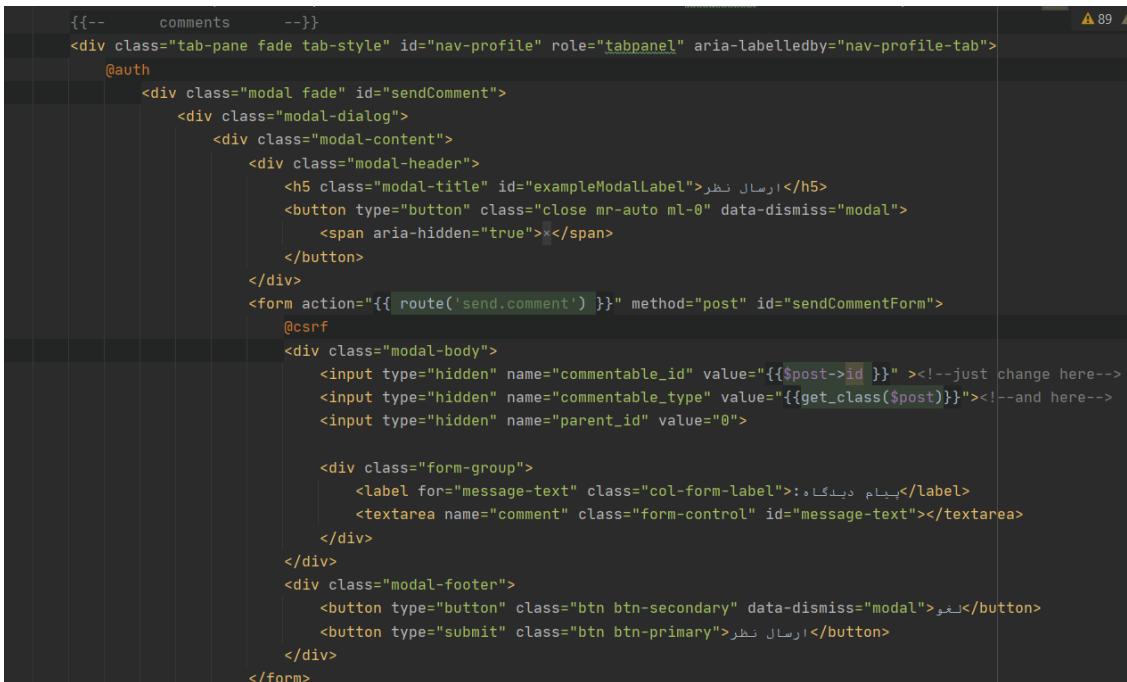
$.ajax({
    type:'POST',
    url:'/comments',
    data:JSON.stringify(data),//our data is an object we should send it as json
    success:function (data){
        // console.log(data)
        // alert("عکس شما با موفقیت آپلود شد");
        Swal.fire({
            title: 'با تشکر',
            text: 'متن شما با موفقیت ارسال شد. پس از تایید در سایت نمایش داده خواهد شد',
            icon: 'success',
            confirmButtonText: 'باشه'
        })
    }
})

```

شکل 4-22. کد script مربوط به بخش نظرات بخش دوم

کارایی کد های script این بخش برای ارسال پیام ایجاد کردن نظر بدون رفرش صفحه در نظر

گرفته شده است.



```

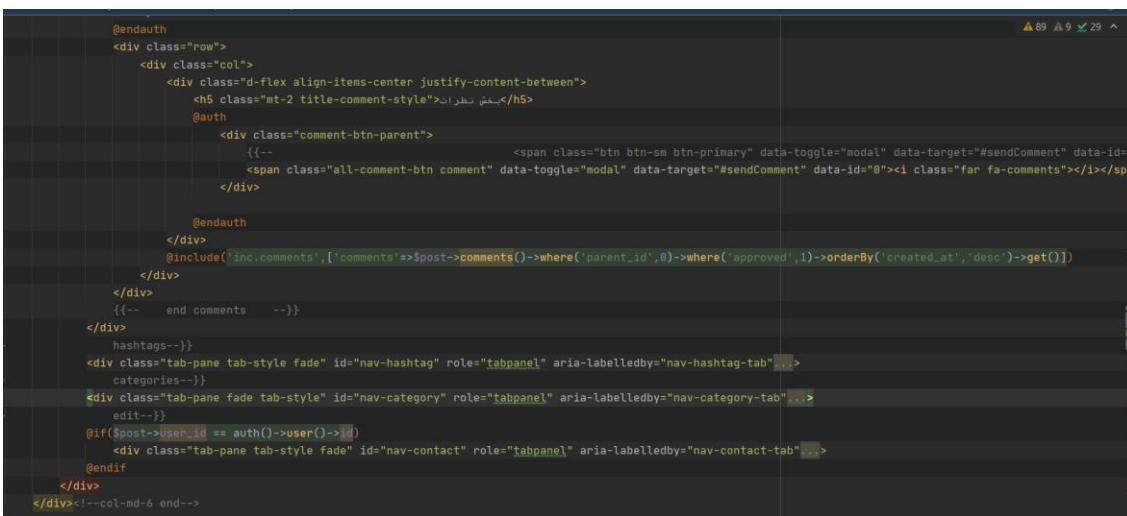
{{-- comments --}}


@auth
    <div class="modal fade" id="sendComment">
        <div class="modal-dialog">
            <div class="modal-content">
                <div class="modal-header">
                    <h5 class="modal-title" id="exampleModalLabel">رسال نظر</h5>
                    <button type="button" class="close mr-auto ml-0" data-dismiss="modal">
                        <span aria-hidden="true">×</span>
                    </button>
                </div>
                <form action="{{ route('send.comment') }}" method="post" id="sendCommentForm">
                    @csrf
                    <div class="modal-body">
                        <input type="hidden" name="commentable_id" value="{{ $post->id }}" ><!--just change here-->
                        <input type="hidden" name="commentable_type" value="{{ get_class($post) }}" ><!--and here-->
                        <input type="hidden" name="parent_id" value="0" >

                        <div class="form-group">
                            <label for="message-text" class="col-form-label">متن دیدگام</label>
                            <textarea name="comment" class="form-control" id="message-text"></textarea>
                        </div>
                    </div>
                    <div class="modal-footer">
                        <button type="button" class="btn btn-secondary" data-dismiss="modal">غیر</button>
                        <button type="submit" class="btn btn-primary">ارسال نظر</button>
                    </div>
                </form>
            </div>
        </div>
    </div>


```

شکل 4-23. کد نوشته شده در comment blade برای بخش اول



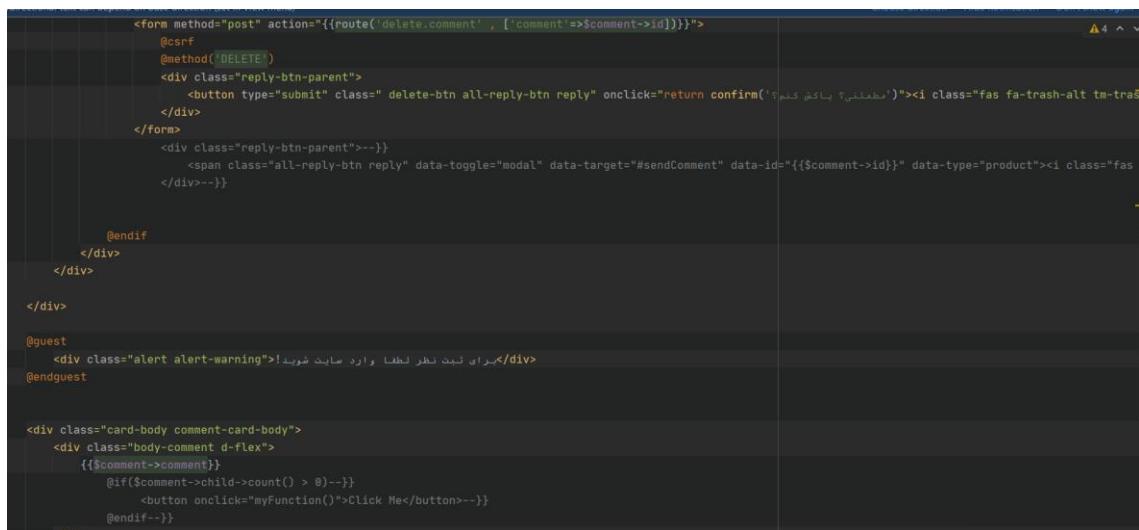
```

@endauth
<div class="row">
    <div class="col">
        <div class="d-flex align-items-center justify-content-between">
            <h5 class="mt-2 title-comment-style">نمای نظرات</h5>
            @auth
                <div class="comment-btn-parent">
                    {{-- <span class="btn btn-sm btn-primary" data-toggle="modal" data-target="#sendComment" data-id="{{$post->id}}>
                    <span class="all-comment-btn comment" data-toggle="modal" data-target="#sendComment" data-id="0"><i class="far fa-comments"></i></span>
                </div>
            @endauth
        </div>
        @include('inc.comments',[ 'comments'=>$post->comments()->where('parent_id',0)->where('approved',1)->orderBy('created_at','desc')->get()])
    </div>
    {{-- end comments --}}
</div>
hashtags-->
<div class="tab-pane tab-style fade" id="nav-hashtag" role="tabpanel" aria-labelledby="nav-hashtag-tab">
    categories-->
<div class="tab-pane tab-style fade" id="nav-category" role="tabpanel" aria-labelledby="nav-category-tab">
    edit-->
    @if($post->user_id == auth()->user())
        <div class="tab-pane tab-style fade" id="nav-contact" role="tabpanel" aria-labelledby="nav-contact-tab">
    @endif
</div>
</div><!--col-md-6 end-->

```

شکل 4-24. کد نوشته شده در comment blade برای بخش دوم

کد بخش blade برای نشان داده ضاهر فرم کامنت برای بخش نظرات میباشد.



```

<form method="post" action="{{route('delete.comment', ['comment'=>$comment->id])}}>
    @csrf
    @method('DELETE')
    <div class="reply-btn-parent">
        <button type="submit" class=" delete-btn all-reply-btn reply" onclick="return confirm('برای حذف کلمه مطابقت با محتوا را بحذف کنید')"><i class="fas fa-trash-alt tm-trash"></i></button>
    </div>
</form>
<div class="reply-btn-parent">-->
    <span class="all-reply-btn reply" data-toggle="modal" data-target="#sendComment" data-id="{{$comment->id}}" data-type="product"><i class="fas fa-comment-dots tm-comment"></i>-->
</div>

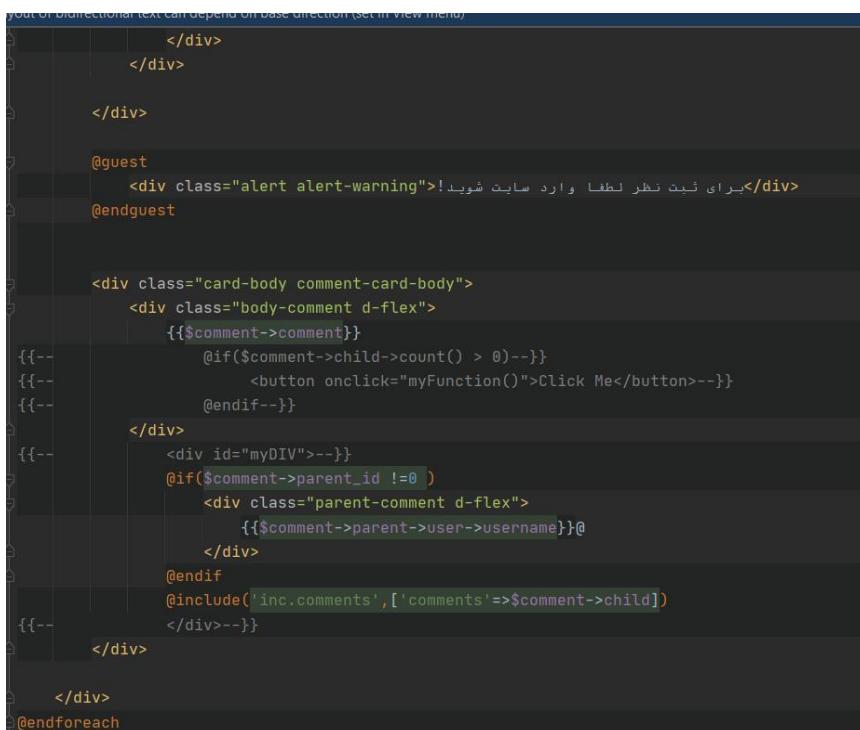
@endif
</div>
</div>

@guest
    <div class="alert alert-warning">برای ثبت نظر لطفا وارد سایت شوید!</div>
@endguest

<div class="card-body comment-card-body">
    <div class="body-comment d-flex">
        {{$comment->comment}}
        {{$comment->child->count() > 0)-->}}
            <button onclick="myFunction()">Click Me</button>-->
        @endif-->
    </div>
</div>

```

شکل 4-25. کد مربوط به comment include بخش اول



```

<!-- Out of bidirectional text can depend on base direction (set in view menu) -->
</div>
</div>

</div>

@guest
    <div class="alert alert-warning">برای ثبت نظر لطفا وارد سایت شوید!</div>
@endguest

<div class="card-body comment-card-body">
    <div class="body-comment d-flex">
        {{$comment->comment}}
        {{$comment->child->count() > 0)-->}}
            <button onclick="myFunction()">Click Me</button>-->
        @endif-->
    </div>
    <div id="myDIV"-->-->
        @if($comment->parent_id != 0 )
            <div class="parent-comment d-flex">
                {{$comment->parent->user->username}}@<br>
            </div>
        @endif
        @include(['inc.comments', ['comments'=>$comment->child]])
    </div>-->
</div>
</div>
@endforeach

```

شکل 4-26. کد مربوط به comment include بخش دوم

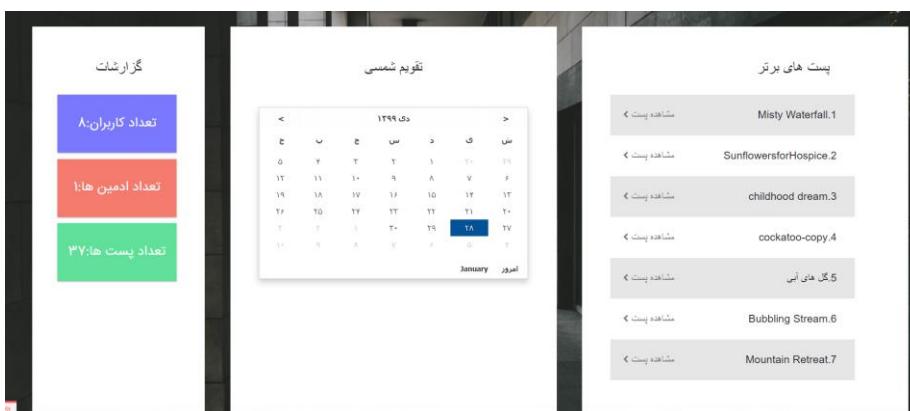
میبینید که در شکل 4-26 از include comment در قسمت include دوباره استفاده شده این یک حالت تو در تو به include میدهد که به روش بازگشتی در برنامه نویسی معروف است. که برای بخش پاسخ های تو در تو (پاسخ به پاسخ) به کار برده میشود.

#### 4-3-1-7- بخش ادمین سایت

بخش مدیریت سایت نیز در بخش ادمین قرار دارد. ادمین به صفحات هنرمندان دسترسی ندارند و هنرمند نیز به ادمین دسترسی ندارد. ادمین قادر است اطلاعات مختلف از جمله کاربران، دسته بندی ها، پست ها و نظرات را ایجاد، ویرایش، حذف و یا بازگردانی کند. همچنین ایجاد دسته بندی های جدید به عهده ای ادمین است. همچنین در داشبورد ادمین نمودار های گزارشگیری نیز قرار داده شده اند تا اطلاعاتی را از پایگاه داده برای ما به نمایش درآورند.



شکل 4-27. صفحه اصلی داشبورد ادمین بخش اول



شکل 4-28. صفحه اصلی داشبورد ادمین بخش دوم

#### 4-4- نتیجه‌گیری

در این فصل کد های اساسی و صفحات مهم سایت را نمایش و توضیح دادیم. و نمونه هایی از

صفحات اجرا شده را نمایش دادیم. توضیحات این فصل حاوی جزئیات پیاده‌سازی، ابزارها، نرم‌افزارها، چارچوب‌ها، زبان‌ها و تنظیمات بود. در فصل بعد جمع‌بندی کلی و پیشنهادات کار‌های آتی را خواهیم آورد.

## فصل 5:

# جمع‌بندی و پیشنهاد‌ها

## 5-5- نتیجه گیری

در این پایان نامه یک سیستم مبتنی بر وب گالری عکس های هنری را در تحلیل، طراحی و پیاده سازی کرده ایم. این سایت آنلاین شبکه اجتماعی مزایای بسیاری را میتواند در بر داشته باشد. پلتفرم PHP برای توسعه برنامه های سازمانی مناسب است. سرعت توسعه برنامه در آن بالا بوده و در عین حال از سرعت و کیفیت بالایی برخوردار است. همچنین استفاده از معماری MVC، کمک زیادی به جداسازی بخش‌های مختلف سیستم کرده که باعث پاسخگویی بهتری به تغییرات می شود. هم اکنون اینستاگرام یکی از محبوب ترین شبکه های اجتماعی می باشد. که بخش های اساسی یک سایت اجتماعی از جمله گذاشتن پست ، اعمال تغییرات ، نوشتن بیوگرافی و فالو و آفالو کردن دیگران همچنین لایک و نظر دهی به پست ها و دریافت نوتیفیکیشن آن ها و جستجو پست ها ، کاربران و هشتگ ها از جمله نیاز های اساسی است که در این پروژه رعایت شده است.

## 5-5- پیشنهادهایی برای کارهای آتی

کارهایی که فرصت نشد در این پایان‌نامه انجام شود و در ادامه قابل ادامه‌دهی است می‌توان گفت شامل موارد زیر است که به صورت تیتر وار آورده شده است.

- گذاشتن بخش خرید و فروش آثار هنری
- گذاشتن بخش مزایده قبل از فروش برای تعیین قیمت و محبوبیت یک اثر
- اضافه کردن جواوا اسکریپت و ajax بیشتر برای کمتر رفرش شدن صفات
- نشان دادن کاربر های آنلاین
- ثبت activity کاربران برای ادمین
- اضافه کردن برگزاری گاری هنری آنلاین برای معرفی اثار هنری یک سبک
- گذاشتن استوری (پست غیر داعمی) مانند سایت اینساگرام
- قابلیت نگهداری اثار محبوب
- قابلیت چت خصوصی با کاربران
- و بسیاری موارد دیگر در صورت احساس نیاز

همچنین با تست بیشتر برنامه قسمت های مختلف را میتوان اشغال گیری کرد یا قابلیت هایی را تغییر داد و یا اضافه کرد.

# مِنَابِع

## فهرست منابع

لينك دانلود نرم افزار XAMPP

[1] <https://www.apachefriends.org/download.html>

مستندات لاراول 7

[1] <https://laravel.com/docs/7.x>

سایت بوت استرپ

[1] <https://getbootstrap.com/>

پکیج موریلاگ جلالب برای تقویم شمسی

[1] <https://github.com/morilog/jalali>

لينك گیت هاب برای بک آپ پروژه

[1] <https://github.com/EliFaveen/gallery>

کد به کار رفته برای ایجاد هشتگ

[1] [https://codepen.io/nikolett\\_codes/pen/daWxe](https://codepen.io/nikolett_codes/pen/daWxe)

کد به کار رفته برای کراپ عکس

<https://www.nicesnippets.com/blog/laravel-crop-image-before-upload-using-cropper-js>

[1]

لينك کمکی برای آموزش لاراول

[1] <https://www.tutsmake.com/laravel-7-6-tutorial-from-scratch-laravel-step-by-step/>

دوره مقدماتی آموزش لاراول

<https://www.youtube.com/watch?v=EU7PRmCpx-0&list=PLlIlgF->

[1] RfqbYhQsN5WMXY6VsDMKGadrJ-

دوره پیشرفته آموزش لاراول

[1] <https://roocket.ir/series/laravel-projects>

سامانه شبیه ساز دریافت ایمیل های پروژه local mailtrap

[1] <https://mailtrap.io/>

# پیوست‌ها

## پیوست الف (راه اندازی پروژه)

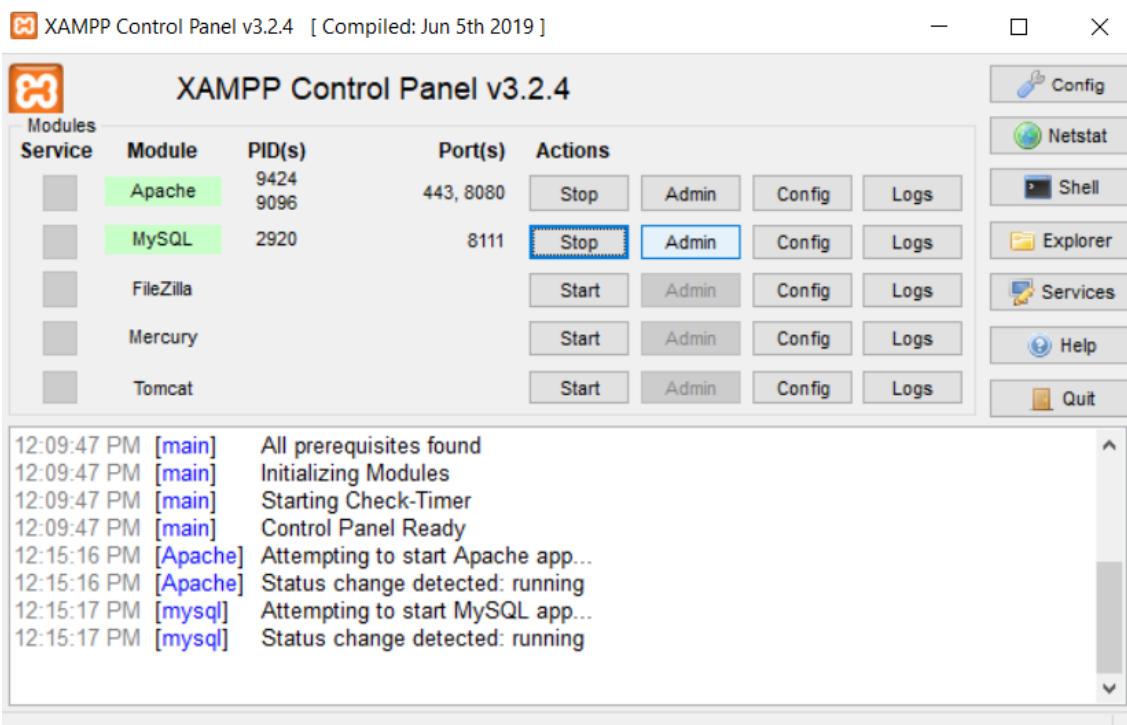
برای راه اندازی پروژه شما نیاز خواهید داشت که ابتدا برنامه XAMPP که یک پکیج کامل از apache و mysql، php هست را نصب کنید. با این کار دیگر نیاز نیست برنامه دیگری را نصب کنید. File اصلی پروژه را در سیستم خود قرار دهید و با Command-line (CMD) موجود در ویندوز خود را به پوشه یا همان دایرکتوری پروژه برسانید. 2 دستور لازم است که اجرا شود:

– برای نصب کردن پکیج های به کاربرده شده در پروژه که در دایرکتوری vendor قرار میگیرند.

– برای اجرای پروژه در یک سرور local Php artisan serve اما قبل از نوشتن این 2 دستور در cmd فایل دیتابیس که به همراه پروژه قرار دارد را در محلی کهxampp را نصب کردید بروید و در آدرس زیر قرار دهید.

Xampp/mysql/data/gallery3

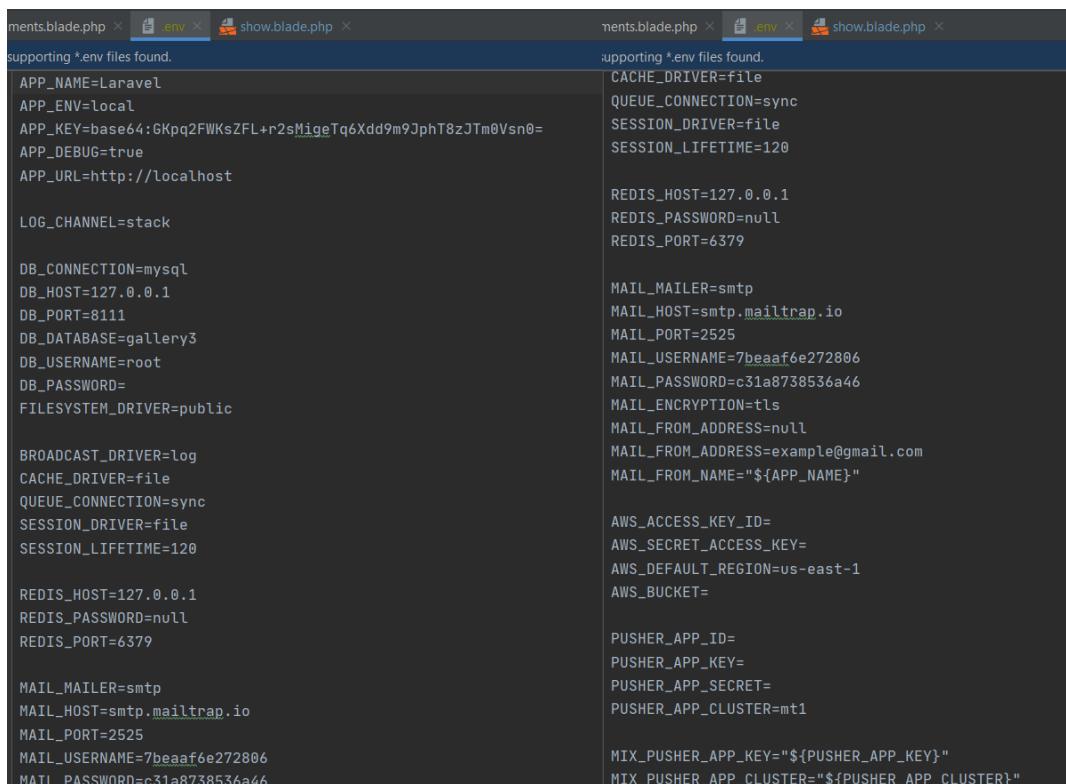
در واقع باید فایل gallery3 را کپی و در پوشه data قرار دهید. سپس باید mysql و apache را در XAMPP اجرا کنید. با زدن دکمه admin در جلوی mysql به phpMyAdmin دسترسی خواهید داشت و میتوانید فایل و اطلاعات galerrey3 را مشاهده کنید.



شکل 1-5. محیط برنامهxampp

بعد از زدن دستور `php artisan serve` در cmd به مرورگر خود رفته و `localhost:8000` را در نوار مرورگر تایپ کنید و دکمه `enter` را بزنید. حال باید بتوانید صفحه اصلی سایت گالری را مشاهده کنید.

لازم به ذکر است فایل `.env`. واقع در پروژه اصلی تنظیمات بیشتری را شامل میشوند که قابل تغییر هستند. مثلا اگر شما برای `xampp` خود `username` و `password` دیگری قرار داده اید باید در این بخش آن را وارد کنید. در ادامه فایل `.env` را مشاهده میکنید.



```
ments.blade.php × env × show.blade.php ×
supporting *.env files found.

APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:GKpq2FWKsZFL+r2sMigeTq6Xdd9m9JphT8zJTm0Vsn0=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=gallery3
DB_USERNAME=root
DB_PASSWORD=
FILESYSTEM_DRIVER=public

BROADCAST_DRIVER=log
CACHE_DRIVER=file
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_MAILER=smtp
MAIL_HOST=smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=7beaaaf6e272806
MAIL_PASSWORD=c31a8738536a46
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS=null
MAIL_FROM_ADDRESS=example@gmail.com
MAIL_FROM_NAME="${APP_NAME}"

AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=

PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_APP_CLUSTER=mt1

MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
```

شکل 2-5. فایل `.env`

## Abstract

This site is a design of a social networking site in order to create a connection between artists around the world and to present the works of the artists of the "Tida Gold Art Gallery" site.

This site is developed with php scripting language and by laravel framework with MVC architecture.

The site has various capabilities such as placing likes and comments for the desired works, the ability to create posts and crop photos, as well as the admin section for managing users, posts, creating categories for works of art and managing comments are the main features of the site admin section.

In addition to having the basic features of a social site, this site has the ability to be developed in various sections, including holding an online art gallery, the ability to save popular posts, the ability to rate points or report abusive users, buy and sell works and many more Other depending on the need.

**Keywords:** Social networking site, Laravel framework, MVC architecture, art gallery



**Technical and Vocational University  
Shariaty Technical College**

A Dissertation Submitted in Partial Fulfillment for the Degree of Bachelor  
of Software Engineering

**Analysis, design and implementation of "Tida  
Art Gallery" social network site using PHP  
(MVC)**

By:

**Elham Hosseini**

Supervisor:

**Seyed Hossein Ahmadpanah**

**December 2020**