

دانشگاه تهران

دانشکده‌ی برق و کامپیوتر

پروژه‌ی کارشناسی

توسعه پورتال پژوهشی آزمایشگاه روش‌های صوری

دانشجو: نوشین ضمیری

استاد راهنما: دکتر فاطمه قاسمی

تابستان ۹۵

۵.....	چکیده.....
۶.....	مقدمه.....
۷.....	دلایل انتخاب فریمورک Ruby on Rails.....
۹.....	دلایل انتخاب مدل معماری MVC.....
۱۱.....	روش توسعه‌ی چابک.....
۱۳.....	سیر گام به گام پیاده سازی.....
۱۷.....	نیازمندی های پروژه.....
۱۸.....	مدل دامنه.....
۱۹.....	مورد کاربرد.....
۲۱.....	نموداری توالی.....
۲۶.....	معرفی سایت.....
۳۳.....	آزمون نرم افزار.....
۳۴.....	آزمون واحد.....
۳۵.....	مقدمه‌ای بر Rspec.....
۳۷.....	نتیجه گیری.....
۳۸.....	منابع.....

۱۰.....	مدل معماری MVC
۱۸.....	مدل دامنه
۱۹.....	مورد کاربرد ۱- کاربران سیستم
۲۰.....	مورد کاربرد ۲- ادمین سیستم
۲۱.....	نمودار توالی ثبت نام کاربر در سیستم
۲۲.....	نمودار توالی مشاهده تمامی مقالات
۲۳.....	نمودار توالی ایجاد مقاله ی جدید
۲۴.....	نمودار توالی نمایش مقاله
۲۵.....	نمودار توالی جست و جوی مقاله
۲۶.....	صفحه ی اول سیستم
۲۷.....	ورود به سایت
۲۷.....	پروژه های موجود در سایت
۲۸.....	ایجاد پروژه ی جدید
۲۸.....	صفحه ی مقالات
۲۹.....	ایجاد مقاله ی جدید
۲۹.....	اعضای سایت
۳۰.....	جزئیات یک کاربر
۳۰.....	تب مخصوص به ادمین
۳۱.....	ایجاد یک کاربر جدید
۳۱.....	ایجاد یک خبر جدید
۳۲.....	صفحه ی پروفایل
۳۵.....	آزمون نرم افزار
۳۶.....	استفاده از before و after در آزمون نرم افزار

این پروژه، توسعه پورتال پویا بر اساس اطلاعات ثبت شده توسط اعضای عضو در پورتال طبق روش های مهندسی توسعه نرم افزار می باشد. برای مثال صفحه ی شخصی افراد بر اساس پروژه های تعریف شده و افراد درگیر در این پروژه، مقالات ثبت شده در سیستم و اطلاعات شخصی وارد شده توسط خود فرد به صورت پویا<sup>1</sup> ایجاد می گردد. به همین ترتیب لیست مقالات آزمایشگاه و پروژه های انجام شده و یا در حال انجام و افراد درگیر در هر کدام به صورت پویا تولید می شود و در یک صفحه ی کاربر پسند نشان داده می شود.

از آنجایی که نگهداری و به روز رسانی صفحات پورتال به صورت استاتیک امکان پذیر نیست، این پروژه کمک می کند که اطلاعات صفحات به صورت پویا تولید شود. در نتیجه صفحات پژوهشی پورتال همیشه به روز خواهد بود.

از روش های توسعه نرم افزار به صورت تکراری<sup>2</sup> استفاده شده است و همچنین مدل معماری آن بر اساس مدل ام وی سی<sup>3</sup> می باشد. توسعه در محیط ریلز<sup>3</sup> انجام شده و از امکانات همین محیط برای آزمون نرم افزار استفاده شده است.

کلمات کلیدی: مدل ام وی سی، ریلز

---

<sup>1</sup> Dynamic

<sup>2</sup> Iterative

<sup>3</sup> Rails

استفاده از یک پورتال پویا به منظور مدیریت اطلاعات کاربران روشی مطمئن برای جلوگیری از خطا و صرفه‌جویی در زمان است. در این پورتال کاربران عضو، اعضای مهمان، مقالات و پروژه‌ها نگهداری و مدیریت می‌شوند. برای آزمایشگاه روش‌های صوری نیاز به این پورتال وجود داشت. پورتالی که در آن صفحات همیشه به‌روز هستند به این صورت که به عنوان مثال هنگام ایجاد یک پروژه جدید پس از ثبت افراد، اطلاعات این پروژه در پروفایل کاربران درگیر، وارد شود.

برای پیاده‌سازی این سایت از محیط ریلز و امکانات فراهم شده توسط آن استفاده شده است. به این دلیل که بسیاری از امکانات مورد نظر برای این پورتال توسط ابزارهای مخصوص ریلز ایجاد شده است و کافی است که شخصی سازی شوند. از طرفی دیگر با استفاده از امکانات همین محیط می‌توان آزمون‌های مورد نیاز را پیاده سازی کرد. دلایل انتخاب مدل معماری و فریمورک استفاده شده، به طور کامل بررسی شده است.

از مهم‌ترین این دلایل می‌توان به ساختار تمیز، امکان پیاده‌سازی آسان، امکانات وسیع، انعطاف پذیری بالا و آزمون نرم افزار اشاره کرد.

برای پیاده سازی این سیستم از نظر سخت‌افزاری نیاز به یک سیستم کامپیوتری با سیستم عامل لینوکس یا مک او اس است. برای استفاده از چارچوب روبی آن ریلز<sup>4</sup>، باید این سیستم کامپیوتری نرم‌افزارهای روبی، ریلز، پایگاه داده ی مناسب را نصب داشته باشد.

---

<sup>4</sup> Ruby on Rails

## دلایل انتخاب فریمورک روبی آن ریلز

در طی ده سال گذشته روبی آن ریلز به عنوان یکی از محبوب‌ترین ابزارها برای ساخت وب اپلیکیشن‌ها تبدیل شده‌است.

در حقیقت روبی آن ریلز یک ابزار طراحی وب اپلیکیشن است؛ یک فریمورک بر اساس زبان روبی که ساختاری تمیز و یک‌پارچه را برای کد فراهم می‌آورد. برنامه نویسان و طراحان وب بدین دلیل RoR را انتخاب می‌کنند که به آنها امکان انجام کارهای تکراری و دشوار را به آسان‌ترین صورت ممکن ارائه می‌کند. از دیگر دلایل انتخاب این فریمورک می‌توان به موارد زیر اشاره کرد:

ریلز می‌تواند پروسه‌ی طراحی را به میزان قابل توجهی سرعت بخشد. کدها در RoR بسیار تمیز و مرتب بوده و همچنین می‌توان از کامپوننت‌های<sup>5</sup> آن بارها و بارها استفاده کرد، بدون آن‌که نیاز باشد آنها را از صفر نوشت. علاوه بر این‌ها به کمک پلاگین‌هایی به نام Gems Ruby می‌توان قابلیت‌های متعددی را به سایت افزود که اگر این Gemها نبودند، باید آنها را از ابتدا می‌نوشتیم. این قابلیت‌ها و امکانات دست در دست هم ریلز را به یکی از سریع‌ترین و جالب‌ترین فریم ورک‌ها تبدیل کرده است.

از طرفی دیگر هرچه سرعت پیشرفت پروژه بالاتر برود، هزینه آن پایین‌تر خواهد آمد به همین دلیل با استفاده از ریلز نه تنها زمان بلکه هزینه را نیز حفظ خواهیم کرد. RoR به صورت خودکار بسیاری از کارها را انجام داده و به برنامه‌نویس کمک می‌کند که بخش‌های حجیم پروژه را به راحتی اجرا نماید. بدین شکل به جای کارهای روتین و حوصله سر بر، تنها بر روی مشکلات و باگ‌های اساسی تمرکز می‌کنیم. با سرعت اجرا و انعطاف‌پذیری بالای خود، RoR به کمپانی‌ها این امکان را می‌دهد که به سادگی با جریان‌ات به‌روز دنیا همراه شوند و پروژه‌های خود را سرعت بخشند! پروژه‌هایی که در حالت عادی زمان بسیار زیادی برای سرهم بندی و اجرا نیاز خواهند داشت. دلیل دیگر خود زبان روبی می‌باشد. روبی نه تنها بهترین مشخصه‌های زبان‌های داینامیک را در خود جای داده، بلکه بهترین راهکارهای زبان‌های استاتیک را نیز در اختیار برنامه نویسان قرار می‌دهد. از آنجایی که روبی بر پایه‌ی برنامه نویسی شیء گرا طراحی شده است، به کمک این زبان می‌توان کارها را در کوتاه‌ترین و بهترین شکل ممکن انجام داد.

دلیل دیگر متن باز بودن است. این یکی از مهمترین ویژگی‌های روبی است؛ هم فریمورک و هم بخش عمده‌ای از کتابخانه‌های ریلز به صورت متن باز در اختیار همگان قرار دارد. RoR بر روی سیستم عامل لینوکس اجرا می‌شود که این سیستم عامل نیز متن باز است. این بدان معناست که پروژه‌ای که بر اساس روبی و ریلز باشد به هیچ گونه هزینه‌ای جهت Licensing و خرید مجوز برنامه‌ها نیاز نخواهد داشت که یکی دیگر از استراتژی‌های صرفه‌جویی در هزینه است. از طرفی دیگر، متن باز بودن بدان معناست که یک مجموعه از برنامه نویسان و متخصصان همواره ریلز را پشتیبانی می‌کنند؛ پس اگر در جایی به مشکل برخوردیم، تنها لازم است سوال خود را در انجمن RoR مطرح نماییم.

---

<sup>5</sup> component

انعطاف پذیری نیز یکی دیگر از دلایل مهم است. پس از پایان پروژه و اجرای وب اپلیکیشن خود بر پایه‌ی ریلز می‌توان به راحتی آن را تغییر داد. افزودن امکانات جدید، اصلاح ماژول‌های دیتا و دیگر تغییرات در کوتاه‌ترین زمان ممکن و به آسانی تمام قابل اجرا است که باز هم به قدرت RoR در صرفه‌جویی زمان و هزینه دلالت دارد.

درک و فهم آسان، فریم ورک ریلز فعالیت‌های خود را ثبت می‌نماید. منظور آن است که دیگر برنامه نویسان می‌توانند به راحتی از اواسط یک پروژه وارد شده و تمامی اقداماتی که تا آن زمان انجام گرفته را مشاهده کنند و به راحتی پروژه را ادامه دهند. کدها در RoR بسیار قابل فهم و خوانا هستند (یک قابلیت دیگر که در زمان تغییر تیم برنامه نویسی به کمک کمپانی‌ها می‌آید). در کل باید اقرار کرد که ریلز راه کار بسیار خوبی است که آینده روشنی نیز در پیش رو دارد. در چند سال آینده شاهد توانایی‌های منحصر به فرد RoR در راه‌اندازی پروژه‌هایی که زمان کم و سرمایه‌ی محدود دارند، خواهیم بود.



## دلایل انتخاب مدل معماری MVC

نرم افزارهای وب معمولاً اطلاعات را از یک منبع اطلاعاتی گرفته و به کاربر نمایش می‌دهند و کاربر می‌تواند آنها را تغییر داده یا اطلاعات جدید ایجاد کند. که در نهایت توسط سیستم در بانک اطلاعاتی ذخیره خواهند شد. می‌توان عمده فعالیت یک نرم افزار وب را در این عملیات خلاصه کرد.

برای جداکردن بخش‌های مختلف نرم افزار این دلایل را داریم:

معمولاً تغییرات بر روی ظاهر برنامه نسبت به کدهای پردازش و منطق آن بسیار بیشتر است. ادغام این دو لایه باعث میشود کوچک‌ترین تغییرات بر روی ظاهر، کدهای بخش پردازش و منطق نرم افزار را نیز تحت تاثیر قرار دهد. این مشکل را می‌توان با جدا کردن این دو بخش و انتقال اطلاعات بین این دو، حل کرد. اصطلاحاً *thin-client* ایجاد می‌کنیم. به این معنی که کدهای مربوط به پردازش و منطق نرم افزار را از کدهای واسط کاربری (ظاهر سایت) جدا کنیم.

نرم افزارهای زیادی لازم است که اطلاعات مشابه را در قالب‌ها و روش‌های مختلف نمایش دهند. برای مثال اطلاعاتی که به یک تحلیلگر اطلاعات نمایش داده میشود متفاوت از اطلاعاتی است که به یک مدیر نمایش داده می‌شود. تحلیلگر اطلاعات نیاز دارد ریز اطلاعات رو در ردیف‌هایی مشاهده کند. اما مدیر در دید اول فقط نمای نموداری از اطلاعات نیاز دارد. نرم افزارهای با واسط‌های گرافیکی قدرتمند امکان به روزرسانی آنی و در لحظه اطلاعات در نماهای مختلف رو فراهم میکنند.

در طراحی وبسایت، طراحی واسط گرافیکی (UI)، به مجموعه مهارت‌های متفاوتی نسبت به توسعه کدهای منطق برنامه نیاز دارد. معمولاً افراد کمی هستند که مهارت‌های هر دو مجموعه را داشته باشند. وقتی این دو بخش از هم جدا شوند می‌توانیم کار توسعه نرم افزار رو در قالب یک تیم پیش ببریم و از مزایای داشتن تیم بهره مند شویم.

عملیات‌هایی که در UI انجام میشوند معمولاً در دو حوزه قرار دارند: نمایش اطلاعات و ذخیره سازی اطلاعات. هنگام نمایش اطلاعات بخش منطق<sup>۶</sup> نرم افزار اطلاعات مورد نیاز را به واسط گرافیکی انتقال می‌دهد. و در زمان ذخیره سازی، اطلاعات از بخش نمایش به بخش منطق برنامه انتقال داده میشود و ذخیره سازی اتفاق می‌فتد و بدنبال آن بخش UI مجدداً بروزرسانی می‌شود.

طراحی واسط کاربری، با توجه به نوع دستگاهی که وبسایت بر روی آن نمایش داده می‌شود (کامپیوتر، تلفن‌های هوشمند و تبلت‌ها و ...) ممکن است کمی متفاوت باشند. اگر نیاز شود نرم افزار وب را برای دستگاه دیگری بازنویسی کنیم، تغییرات در UI باعث خواهد شد بخش زیادی از کدهای پردازش و منطق برنامه هم تغییر کند. این موضوع باعث ایجاد خطاهای زیادی خواهد شد. اما اگر این دو بخش به شکل خوش تعریفی از یکدیگر جدا شوند، هزینه و زمان برای طراحی بسترهای جدید نیز کاهش پیدا می‌کند.

تولید تست اتوماتیک برای واسط گرافیکی در حال حاضر کار بسیار دشوار و زمانبری می‌باشد. این موضوع در مورد کدهای پردازش و منطق، با ابزارهایی که امروز داریم بسیار ساده تر شده است. بنابراین جداکردن این دو بخش باعث افزایش امکان تست پذیری نرم افزار نیز می‌شود.

---

<sup>۶</sup> Logic

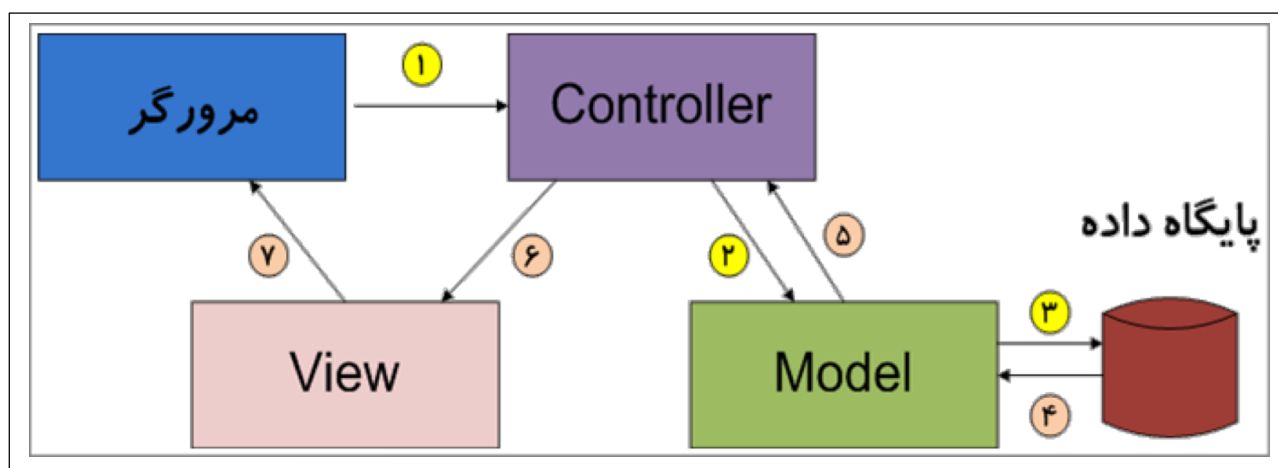
الگوی معماری MVC بخش‌های مدل‌سازی دامنه<sup>۷</sup>، نمایش<sup>۸</sup> و منطق تجاری<sup>۹</sup> رو به سه بخش تقسیم می‌کند. مدل (Model): روند انتقال داده را مدیریت میکند، درخواست‌هایی که در مورد وضعیت مدل، مثل تغییر فیلدی از یه رکورد، وجود دارد را پاسخ می‌دهد (معمولا از سمت view) و عملیات‌هایی که موجب تغییر وضعیت خودش می‌شود را انجام می‌دهد (معمولا از سمت کنترلر).

نمایش (View): نمایش اطلاعات به کاربر را مدیریت می‌کند.

کنترلر (Controller): بخش ورودی‌های کاربر را مدیریت می‌کند. و مدل و View مربوط به آن را فراخوانی و داده‌ها را از مدل به لایه نمایش اطلاعات (View) انتقال می‌دهد.

در این تعریف هر دو بخش کنترلر و view به مدل وابسته هستند. اما در مورد مدل این وابستگی وجود ندارد و از هر دو عنصر دیگر مستقل عمل می‌کند. این ویژگی می‌تواند به تولید بخش مستقل که قابلیت تست بالایی دارد منجر شود. در نرم افزارهای وب این تفکیک و جداسازی به خوبی دیده می‌شود و قابل لمس است. اما در نرم افزارهایی که طراحی UI و کدهای پردازش و منطق در هم تنیدگی دارند یا اصطلاحا rich-client ها این موضوع دیده نمی‌شود. MVC یا Model-View-Controller یک الگوی طراحی پایه برای جداسازی بخش واسط کاربری UI از قسمت‌های پردازشی و منطقی برنامه می‌باشد.

تست پذیری: وقتی از معماری MVC استفاده می‌کنیم تست پذیری بخش Back-end به شدت افزایش پیدا می‌کند. هر چند ایجاد تست‌های اتوماتیک برای اجزا و عناصر واسط گرافیکی یا front-end هنوز کار دشواری است، اما این جداکردن اجزا و بخش‌ها می‌تواند کمک بزرگی برای افزایش تست پذیری باشد. برای انجام تست‌های مختلف ابزارهای کاربردی زیادی هم به وجود آمده‌اند که زمان تست رو به شدت کاهش داده و میزان اعتماد به بخش‌های مختلف نرم افزار رو بالاتر می‌برند.



شکل (۱) مدل معماری MVC

<sup>7</sup> Domain model

<sup>8</sup> Presentation

<sup>9</sup> Bussines logic

## روش توسعه‌ی چابک<sup>۱۰</sup>:

در اغلب اوقات، روند پیاده‌سازی پروژه‌های نرم‌افزاری به صورت یک فعالیت بدون ساختار و فاقد نظام مندی پیش می‌رود که در آن طرح ریزی اصولی برای پیاده‌سازی نرم‌افزار صورت نمی‌گیرد. با رشد و گسترش سیستم، اشکالات این شیوه گسترش می‌یابد. راه حل ارائه مفهوم متدولوژی است. متدولوژی توسعه نرم‌افزار چارچوبی است که برای طرح ریزی، سازماندهی و کنترل فرایند توسعه سیستم به کار می‌رود. هر چارچوب متدولوژی توسعه نرم‌افزار به عنوان شالوده‌ای برای پیاده‌سازی دیدگاهی ویژه در جهت توسعه و نگهداری نرم‌افزار به کار می‌رود. از جمله این دیدگاه‌ها می‌توان به روش آبشاری، روش افزایشی، الگویی و روش تکراری اشاره نمود. اما از جمله چارچوب‌ها یا روش‌های سازمان یافته برای توسعه نرم‌افزار که در حال حاضر متداول و پرکاربرد است روش چابک می‌باشد.

توسعه‌ی چابک نگرشی برای تولید نرم‌افزار است که در آن نرم‌افزار به صورت مرحله به مرحله و تکاملی تحویل سفارش دهنده می‌گردد و با ارتباط تنگاتنگ با او سعی می‌شود که رضایتش جلب شود. به عبارتی دیگر توسعه‌ی چابک مجموعه‌ای از ارزش‌ها و اصول جهت توسعه‌ی نرم‌افزارهای کارا توسط تیم‌های خود سازمانده می‌باشد. ضعف‌های موجود در روش‌های سنتی باعث ایجاد این روش توسعه شد.

مشکلات روش‌های سنتی: صرف زمان زیاد و در واقع اتلاف زمان برای طراحی اولیه دقیق در فاز اولیه پروژه و همچنین مشخص نبودن نیازمندی‌ها از بزرگترین این مشکلات است. که بالا بودن هزینه‌ی تغییرات را در پی دارد. همچنین از مشکلات دیگر به طول انجامیدن پروژه و گذر از زمان تعیین شده و محصولات بدون کیفیت که زمانی برای آزمون آنها وجود نداشته است، می‌باشد.

تفکر چابک، فرصت مناسبی برای ایجاد یک بستر انطباق‌سازی<sup>۱۱</sup> به وجود آورده است. بدین صورت که محصول به صورت تکاملی ساخته می‌شود و در هر مرحله در اختیار مشتری قرار می‌گیرد. علاوه بر این پروسه‌ی تست نیز به صورت یکپارچه با توسعه‌ی بخش‌ها انجام می‌شود. این کار هم هزینه‌ی نگهداری هر بخش را کاهش می‌دهد و هم محصول نهایی با کیفیت می‌شود و هم تغییرات قابل اجرا خواهند بود.

ارزش‌های توسعه نرم‌افزار چابک:

- افراد و تعاملات بالاتر از فرایندها و ابزارها
- نرم‌افزار کارا بالاتر از مستند سازی جامع
- همکاری مشتری بالاتر از قرارداد کار
- جوابگویی به تغییرات بالاتر از پیروی یک طرح
- قوانینی که زیر بنای بیانیه توسعه نرم‌افزار چابک را شکل می‌دهند، به قرار زیر است:
- جلب رضایت مشتری با ارائه سریع نرم‌افزار کارا
- پذیرش تغییرات در نیازمندی‌های مشتری و اعمال آن حتی در اواخر زمان توسعه
- ارائه مکرر نرم‌افزار کارا

<sup>10</sup> Agile programming

<sup>11</sup> Adaption

- عامل اصلی سنجش پیشرفت پروژه، نرم‌افزار کارا است.
  - میزانی از سرعت توسعه که بتوان آن را با گام‌های مداوم و ثابت، حفظ نمود.
  - همکاری نزدیک و روزمره بین صاحبان تجارت و توسعه دهندگان سیستم
  - گفتگو رو در رو و تعاملات شفاهی، بهترین شکل ارتباطات است.
  - پروژه‌ها با افرادی که دارای انگیزه و قابل اعتماد هستند، پیش می‌روند.
  - توجه مداوم به طراحی خوب و کیفیت بالا از نظر فنی
  - سادگی
  - تیم‌هایی که اعضای آن دارای ویژگی خود سازماندهی و خود راهبری هستند.
  - قابلیت انطباق با شرایط در حال تغییر
- متدهای چابک از برنامه ریزی کل پروژه و در نظرگیری تمامی جزئیات آن در ابتدای کار، اجتناب نموده و پروژه را به بخش‌های کوچک تقسیم می‌کنند و سپس با برنامه‌ریزی هر بخش و تکمیل آن و افزودن آن به سایر بخش‌ها پروژه را پیش می‌برند. این بخش‌ها غالباً در بازه‌های زمانی کوتاه مدت توسعه می‌یابند. هر بازه زمانی یک چرخه توسعه نرم‌افزار کامل است که شامل برنامه‌ریزی، آنالیز نیازمندی‌ها، طراحی، برنامه‌نویسی، یونیت تست و تست نهایی است. این روش ریسک کلی پروژه را کاهش می‌دهد و امکان تطابق با تغییرات را نیز به سرعت فراهم می‌کند. هدف آن است که در انتهای هر بازه زمانی بخش قابل ارائه‌ای از نرم‌افزار با حداقل میزان مشکلات تولید شود. اما این بخش لزوماً شامل عملکردهای کافی نیست و ممکن است برای ارائه‌ی یک محصول و حتی یک مشخصه جدید چندین بازه زمانی لازم باشد. کارهایی که در هر بازه زمانی انجام می‌گیرد، بر اساس اولویت بندی صورت گرفته در جلسات است.

## سیر گام به گام پیاده سازی :

برای برنامه نویسی، نیاز به محیطی داریم که امکان دسترسی به همه فایل های پروژه را امکان پذیر سازد. یک ادیتور مناسب برای روبی، نرم افزار سابلایم<sup>۱۲</sup> است. هنگام برنامه نویسی، تمام فایل هایی که باید بر روی مرورگر نمایش داده شوند در فولدر ویو و همه اشیای سیستم که برای دسترسی به داده به آنها نیاز داریم در فولدر مدل وجود دارند. در فولدر کنترلر، همه اشیا و سایر کنترلر ها که برای هدایت و کنترل سیستم به آنها نیاز داریم، قرار دارد.

ایجاد پروژه :

برای ساخت پروژه وارد محیط ترمینال می شویم. در ابتدا باید به دایرکتوری ای که می خواهیم پروژه را در آن ایجاد کنیم برویم. برای ایجاد یک پروژه روبی دستور زیر را وارد می کنیم:

**Rails new project name**

داخل پروژه فایلی به اسم **gemfile** وجود دارد. باید در این فایل تمامی **gem** هایی که قصد استفاده از آنها را داریم وارد کنیم. همچنین فولدر **app** داریم که در آن فولدرهای **models, controller, assets** و **view** وجود دارند. فولدر **assets** شامل **image** و **java scripts, style sheets, fonts** می شود. فولدر **image** شامل تمامی عکس هاییست که در طول پروژه مورد استفاده قرار می گیرد. همچنین فولدر **style sheets** شامل فایل های **java scripts** و **css** است. فولدر دیگری به اسم **db** وجود دارد که شامل تمامی فایل های مربوط به پایگاه داده و جداول و شمای مربوط به پروژه است.

برای شروع پیاده سازی، ابتدا باید اشیا مورد نیاز سیستم و مشخصه های هرکدام را برای سیستم مان مشخص کنیم. اشیا در لایه مدل قرار می گیرند و هنگامی که درخواستی از مرورگر می آید، برای پاسخ به پایگاه داده می رویم و اطلاعات شی مورد نظر را بازیابی می کنیم. این اشیا امکان ذخیره سازی مدل خود و بازیابی آن از پایگاه داده را فراهم می کند.

در ابتدا باید یوزر را بسازیم. به کمک **devise gem** این کار را انجام می دهیم. با این صورت می توانیم امکان ثبت نام و ورود به سیستم را فراهم کنیم. جهت نصب **devise** ابتدا از فولدر پروژه **gemfile** را باز کرده و در انتهای آن **gem devise** را وارد می کنیم. سپس دستور **bundle install** را می زنیم که **gem** ذخیره شود. سپس دستور زیر را در ترمینال وارد می کنیم.

**Rails g devise:install**

سپس برای ایجاد مدل یوزر دستور زیر را در ترمینال وارد می کنیم.

**Rails g devise User**

حالا در فولدر **model** فایل **user.rb** ایجاد شده است.

---

<sup>12</sup> sublime

با ساختن هر شی در فولدر پایگاه داده، یک فایل به نام شی ای که ساخته شده است در فولدر db/migrate ایجاد می گردد. در ابتدای نام این فایل، تاریخ و زمان ساخت آن شی می آید و سپس نام آن که شامل مشخصه های آن شی، نوع هر مشخصه و ... است.

تا در schema ذخیره شود. سپس فایل routes.rb از فولدر config را باز کرده و devise\_for :users را به آن اضافه می کنیم. هر شی باید rout شود تا در url قابل استفاده باشد.

برای بالا آوردن سرور باید دستور rails s را در ترمینال وارد کنیم، سپس در url مرورگر خود خط زیر را وارد کنیم: <http://localhost:3000>

با وارد کردن این دستور صفحه اصلی پروژه می آید، برای دانستن مسیرهای مختلف url دستور <http://localhost:3000/users> را وارد می کنیم.

در این پروژه، کاربری که در سیستم ثبت نام می کند باید تعیین نقش شود. برای طراحی چنین سیستمی، به کاربران دسترسی های متفاوتی داده می شود، برای مثال ادمین افراد را در سیستم ثبت نام می کند و شناسه کاربری و رمز عبور را به افراد می دهد که بتوانند به سیستم وارد شوند. برای ثبت کاربران در سیستم در قسمت user\_controller اطلاعات وارد شده توسط تابع addUser در سیستم ذخیره می شود. این کنترل را پس از نصب devise gem باید به صورت دستی در پروژ وارد کنیم، زیرا این جم تنها مدل user را ایجاد می کند و اگر کاربر نیاز به ایجاد کنترل مربوطه داشته باشد باید آن را خود ایجاد کند.

ایجاد اشیا در سیستم:

پس از ایجاد کاربران و نقش ها در سیستم حال باید سایر اشیا مورد نیاز را تعریف کنیم. برای ساختن اشیا به ترمینال می رویم و دستور زیر را برای هر شی وارد می کنیم.

```
rails g model model_name
```

توجه داشته باشید که حرف اول اسم مدل باید بزرگ باشد.

سپس برای ثبت شی در شما دستور rake db:migrate را در ترمینال وارد می کنیم.

### کنترلر و ویو:

پس از ساختن مدل های همه اشیا سیستم، به سراغ ساخت ویو و کنترلر برای هر شی می رویم. به ترمینال رفته و برای هر شی به طور جداگانه دستور زیر را اجرا می کنیم:

```
rails g controller model-name
```

این دستور برای هر شی یک فایل به نام modelname\_controller.rb در فولدر کنترلر (مانند: posts\_controller.rb

) و یک فولدر با نام اسم مدل در قسمت ویو می سازد. در هر کنترلر باید توابع مورد نیاز هر شی (-new-index

show-delete-create-destroy-update) تعریف کنیم که وظیفه کنترل کردن مدل و نمایش آن را دارند.

در فولدر ویو باید برای هر کدام از توابع که می خواهیم در سیستممان نمایش دهیم یک ویو تعریف

کنیم. این فایل ویو به صورت html نوشته می شود و چون در محیط روبی هستیم پسوند html.erb می گیرد.

### ارتباط اشیا در ریلز:

پس از تعریف این ویوی اولیه و کنترل‌ها، باید روابط بین اشیا که در مدل دامنه آمده است را در اینجا ایجاد کنیم و اگر نیاز است دو شی به هم مرتبط باشند باید کلید خارجی ای برای برقراری این ارتباط قرار دهیم.

در ریلز برای برقراری ارتباط میان اشیا، انواع مختلفی از روابط تعریف شده است. در اینجا به بررسی انواع آن می‌پردازیم.

#### 1-ارتباط تعلق دارد:

اولین ارتباط یک رابطه یک به یک بین دو شی را نشان می‌دهد. برای مشخص نمودن این شی در آن عبارت belongs\_to را وارد می‌کنیم.

#### 2-ارتباط یکی دارد:

این ارتباط، یک رابطه یک به یک را نشان می‌دهد. اما در اینجا رابطه اشاره دارد به شی ای که مالکیت شی دیگر را بر عهده دارد

#### 3-ارتباط خیلی دارد:

رابطه خیلی دارد یک رابطه one to n را بین دو شی تعریف می‌کند. در طرف مقابل این رابطه، یک تعلق دارد، قرار دارد. این رابطه می‌گوید که یک شی با صفر یا چند مورد از شی دیگر در ارتباط است.

#### 4-ارتباط خیلی دارد: (از طریق)

این رابطه نیز یک رابطه n to n را با مدل میانی بیان می‌کند. یعنی برای بیان این رابطه بین دو شی احتیاج به تعریف کردن مدل جدیدی است تا دو شی را به هم مرتبط سازد. در این سیستم رابطه بین شی role و user از این نوع است. بین این دو شی مدل میانی role-user وجود دارد. در این سیستم هر کاربر می‌تواند چندین نقش داشته باشد و هر نقش می‌تواند متعلق به چندین کاربر باشد.

### استفاده از آواتار<sup>13</sup>:

بعد از تعریف ارتباطات بین اشیاء، می‌خواهیم هر کاربر امکان این را داشته باشد تا برای خود عکس پرسنلی بگذارد. برای این کار از `paperclip gem` استفاده می‌کنیم. به `gemfile` می‌رویم و در انتهای آن `gem "paperclip"` را اضافه می‌کنیم. سپس به ترمینال رفته و `bundle install` را اجرا می‌کنیم. حالا می‌خواهیم برای افراد این امکان را فراهم کنیم تا عکس خود را در سیستم بارگذاری کنند. دستور زیر را در ترمینال می‌زنیم:

```
rails paperclip model-name avatar
```

برای مثال داریم:

```
rails g paperclip user avatar
```

پس از این دستور `rake db:migrate` را اضافه می‌کنیم. حالا اگر به فولدر `db/migration` برویم می‌بینیم یک فایل به اسم `add_attachment_avatar_to_user` به آن اضافه شده است. این فایل مشخصه‌های لازم جهت بارگذاری عکس را به فایل `db/schema` اضافه می‌کند. سپس باید تنظیماتی نیز در مدل به وجود بیاوریم. این تنظیمات مشخص می‌کند که ابعاد و سایز و ... تصویر به چه صورتی باید باشد. پس از این‌ها در قسمت `show` باید کدی را وارد کنیم که کاربر بتواند عکس آپلود کند.

---

<sup>13</sup> Avatar



## نیازمندی‌های<sup>۱۴</sup> پروژه :

از نیازمندی‌های سیستم می‌توان به این موارد اشاره کرد که باید کاربر پسند بوده و امکان استفاده‌ی اسان را برای کاربران فراهم سازد.

کاربران بر اساس نقششان دسته‌بندی می‌شوند. به این صورت که امکان تخصیص نقش به کاربران توسط ادمین سایت داده می‌شود. نقش‌هایی مانند ادمین سایت، اساتید، دانشجویان وجود دارند.

سطح دسترسی کاربران باید مشخص باشد. به این صورت که به عنوان مثال دانشجویان قادر به حذف یک پروژه از سایت نخواهند بود.

کاربران توسط ادمین سایت در سیستم ثبت‌نام می‌شوند و سپس اجازه‌ی ورود به سیستم و ویرایش پروفایل خود را خواهند داشت.

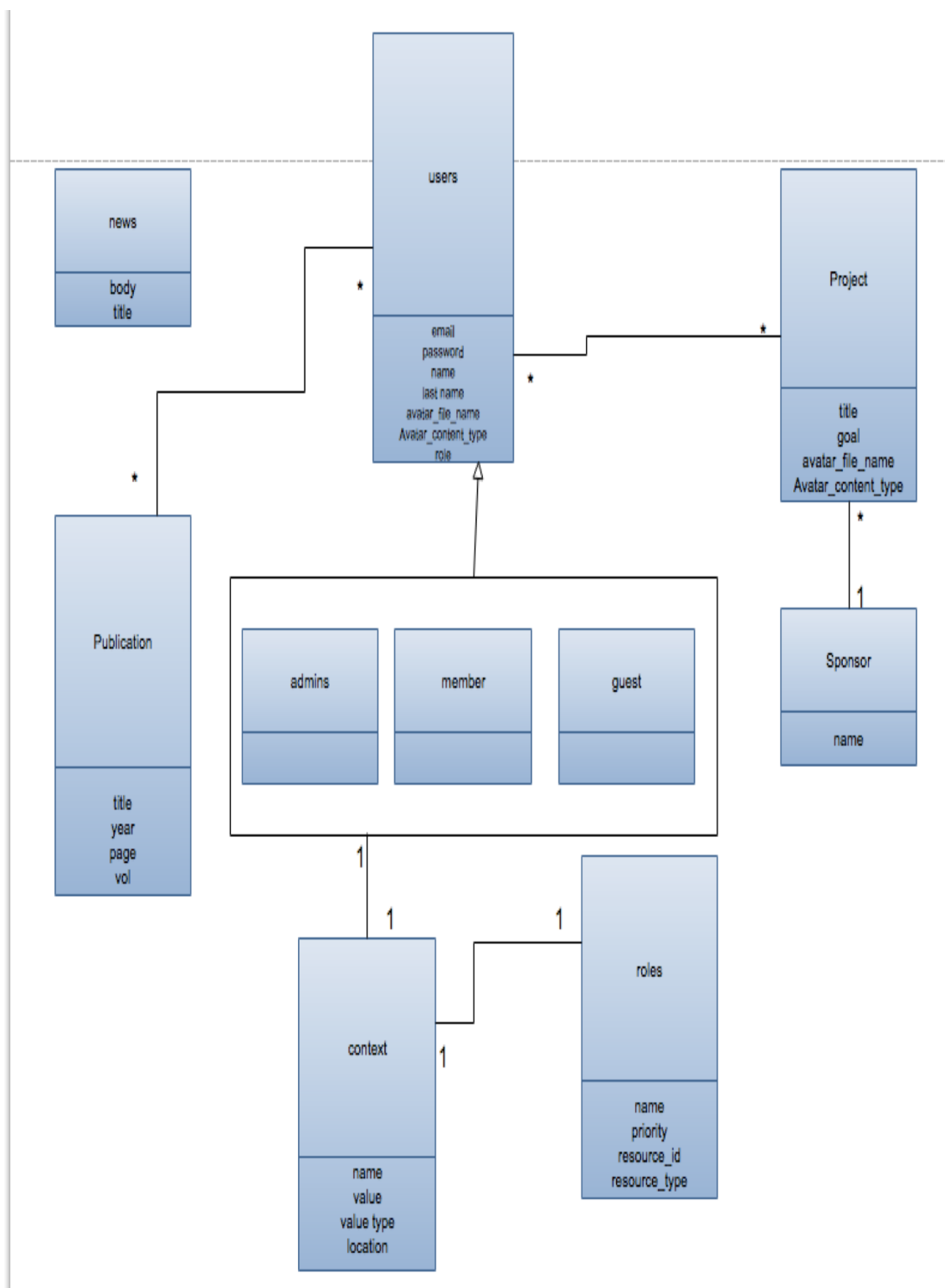
همچنین امکان جست‌وجوی اعضا، مقالات و پروژه‌ها فراهم شده‌است.

چند نمونه از مهم ترین نیازمندی های عملیاتی پروژه عبارت اند از :

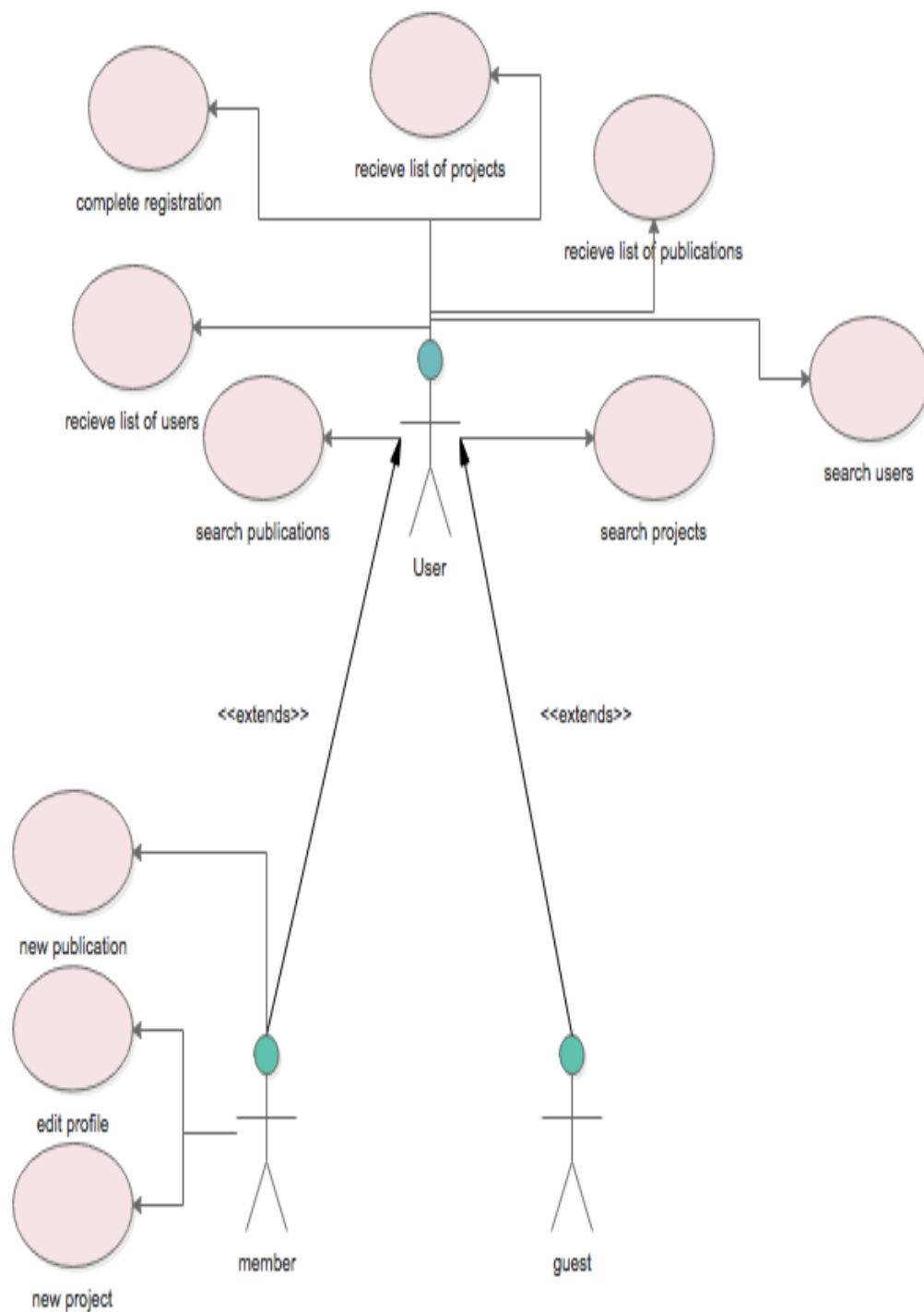
۱. ادمین می‌تواند افراد را در سایت ثبت‌نام کند.
۲. افراد می‌توانند ثبت‌نام خود را کامل کنند.
۳. افراد می‌توانند صفحه‌ی کاربری خود را ویرایش کنند.
۴. افراد می‌توانند مقاله-پروژه‌ی جدید ایجاد کنند.
۵. افراد می‌توانند مقاله-پروژه‌های خود را ویرایش کنند.
۶. افراد می‌توانند افراد-مقالات-پروژه‌های موجود در سایت را جست‌وجو کنند.
۷. ادمین می‌تواند افراد-پروژه-مقالات را از سیستم حذف کند.
۸. سیستم لیست اخبار را نشان می‌دهد.
۹. سیستم لیست آخرین مقالات را نمایش می‌دهد.
۱۰. سیستم لیست کاربران را نمایش می‌دهد.
۱۱. سیستم پس از اضافه شدن مقاله-پروژه‌ی جدید لینک آنها را در پروفایل کاربر به‌روز رسانی می‌کند.
۱۲. سیستم پس از حذف شدن مقاله-پروژه‌ی جدید پروفایل کاربر را به‌روز رسانی می‌کند.
۱۳. ادمین می‌تواند اسپانسر مقالات را ایجاد کند.
۱۴. ادمین می‌تواند به کاربران نقش اختصاص دهد.
۱۵. ادمین می‌تواند به کاربران اولویت اختصاص دهد.

---

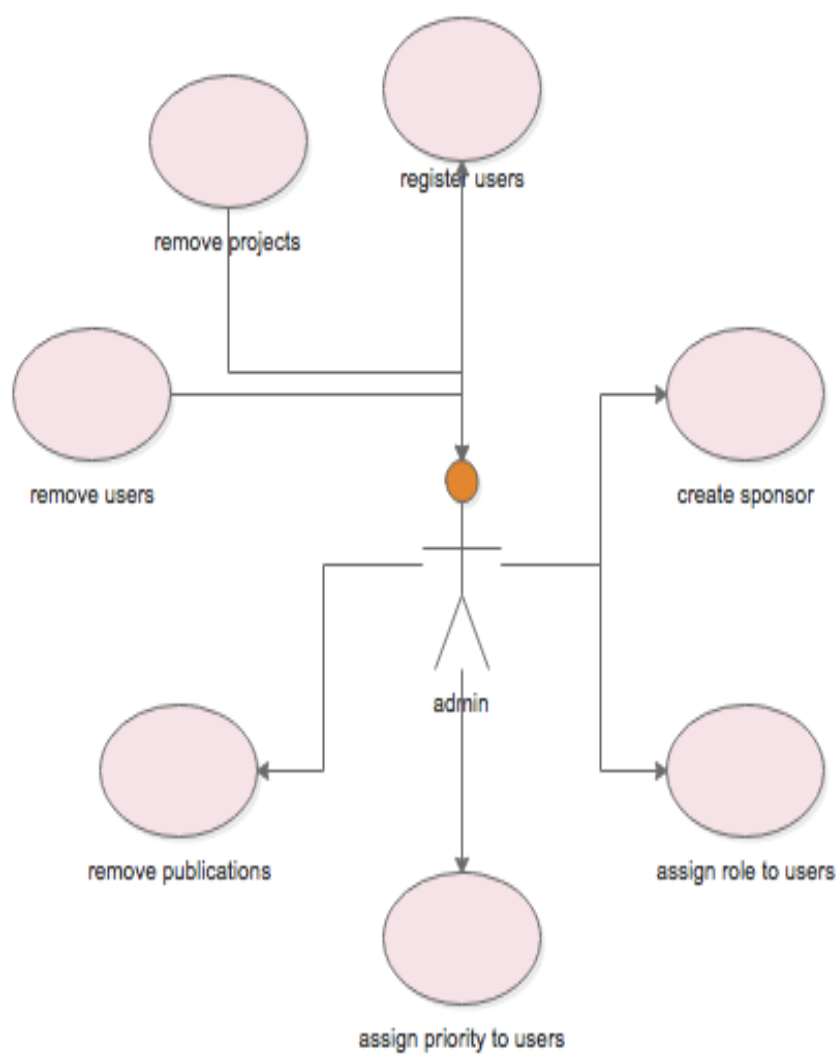
<sup>14</sup> Requirement



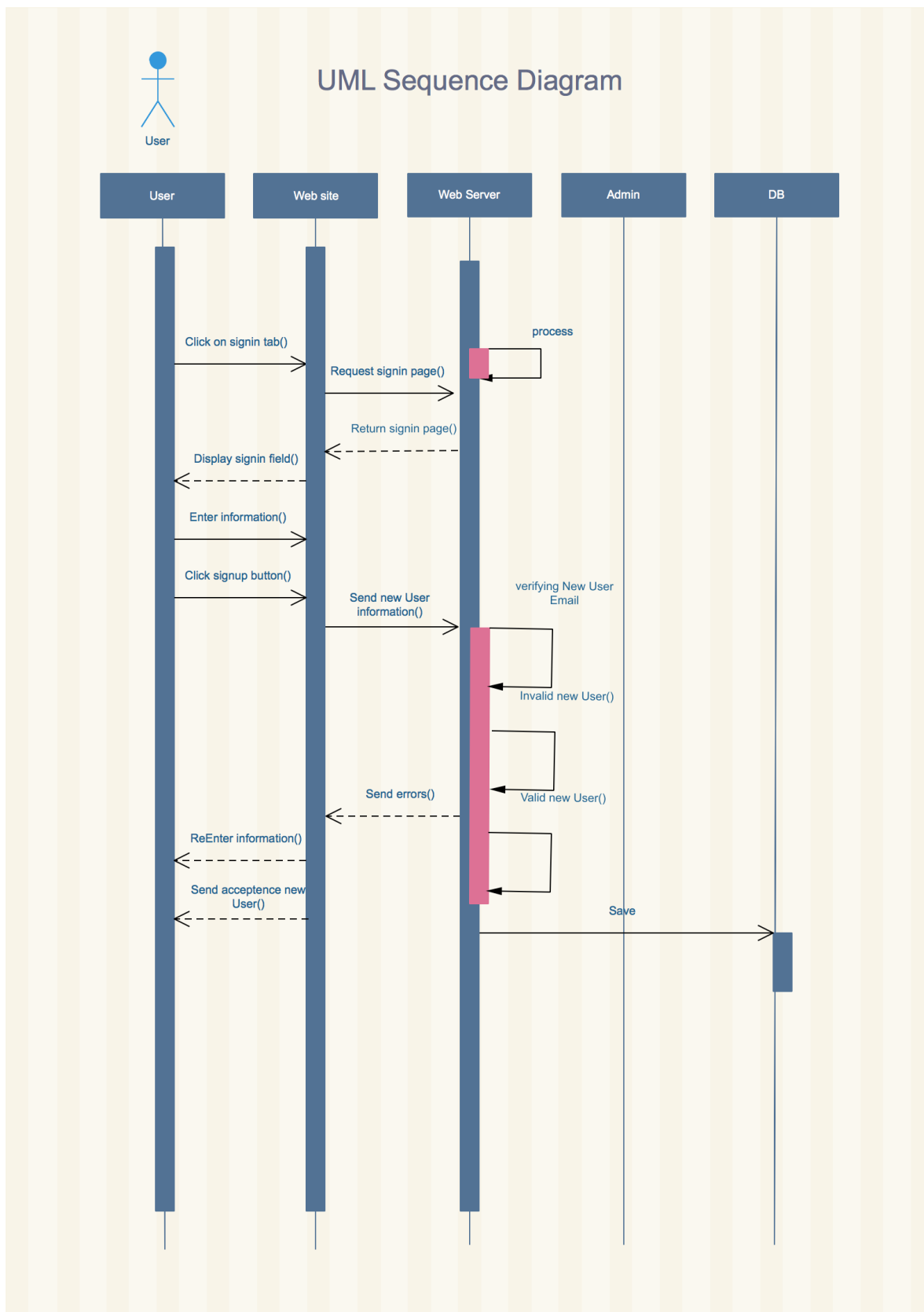
شکل ۲) مدل دامنه



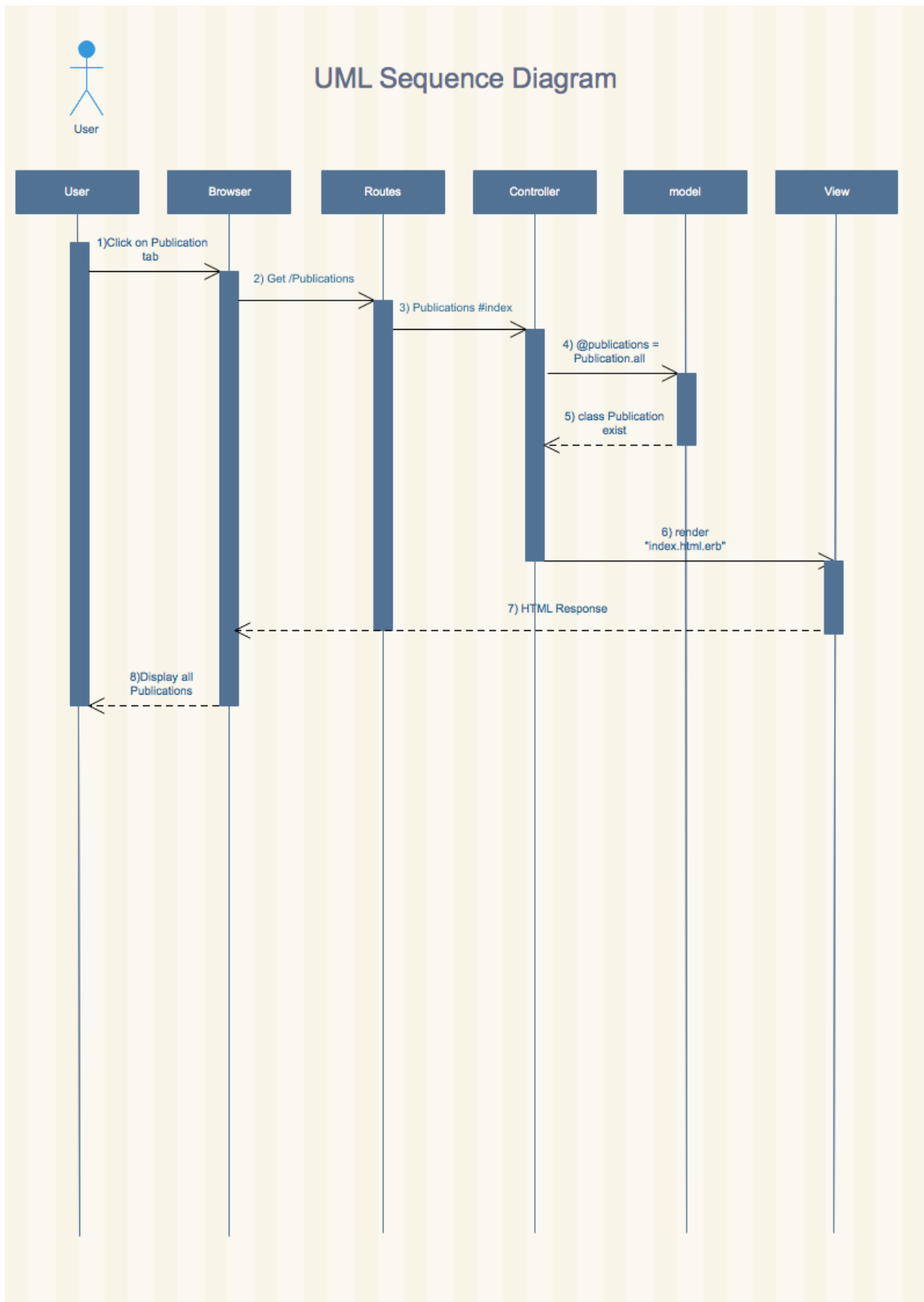
شکل ۳) مورد کاربرد ۱ - کاربران سیستم



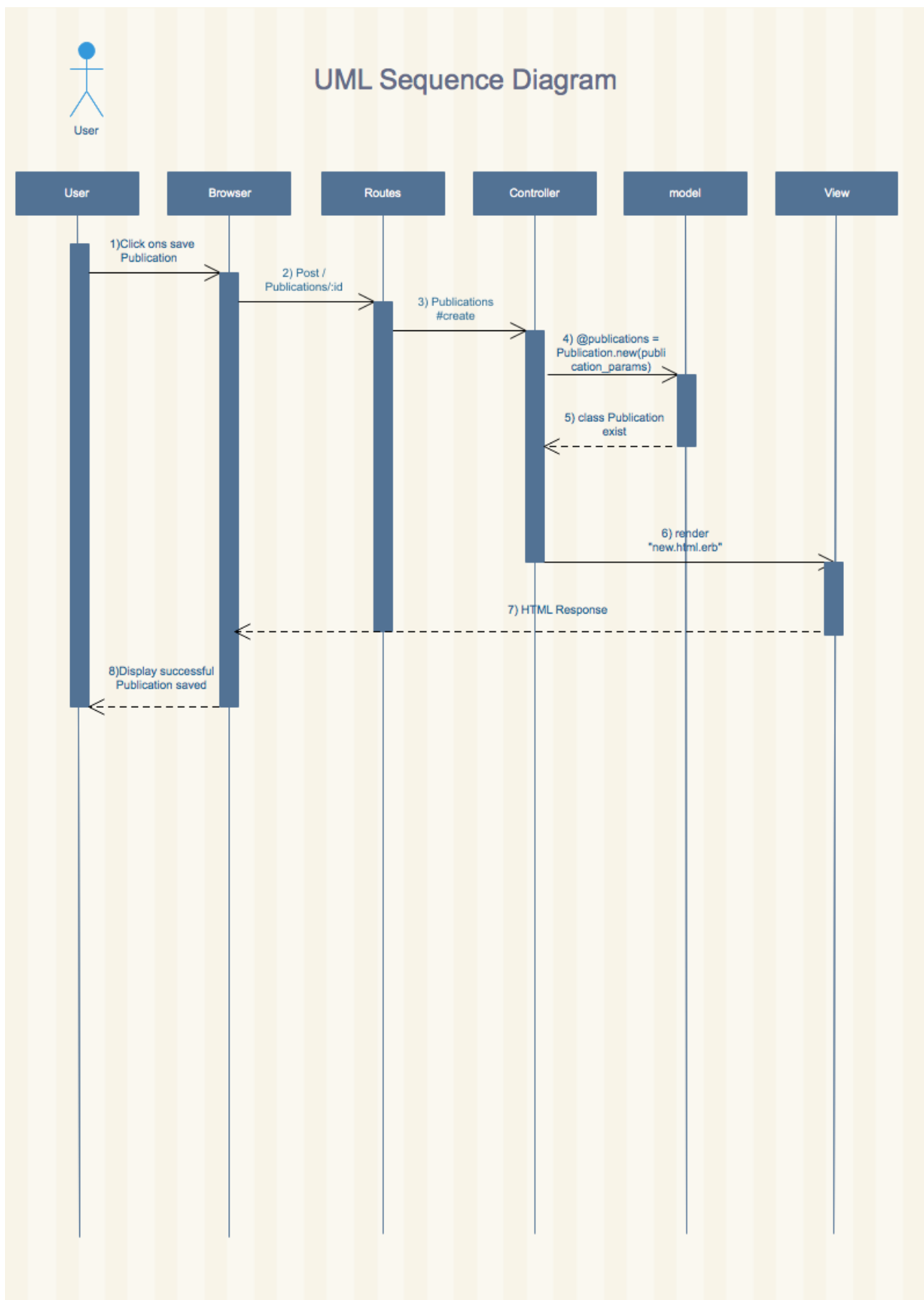
شکل ۴) مورد کاربرد ۲ - ادمین سیستم



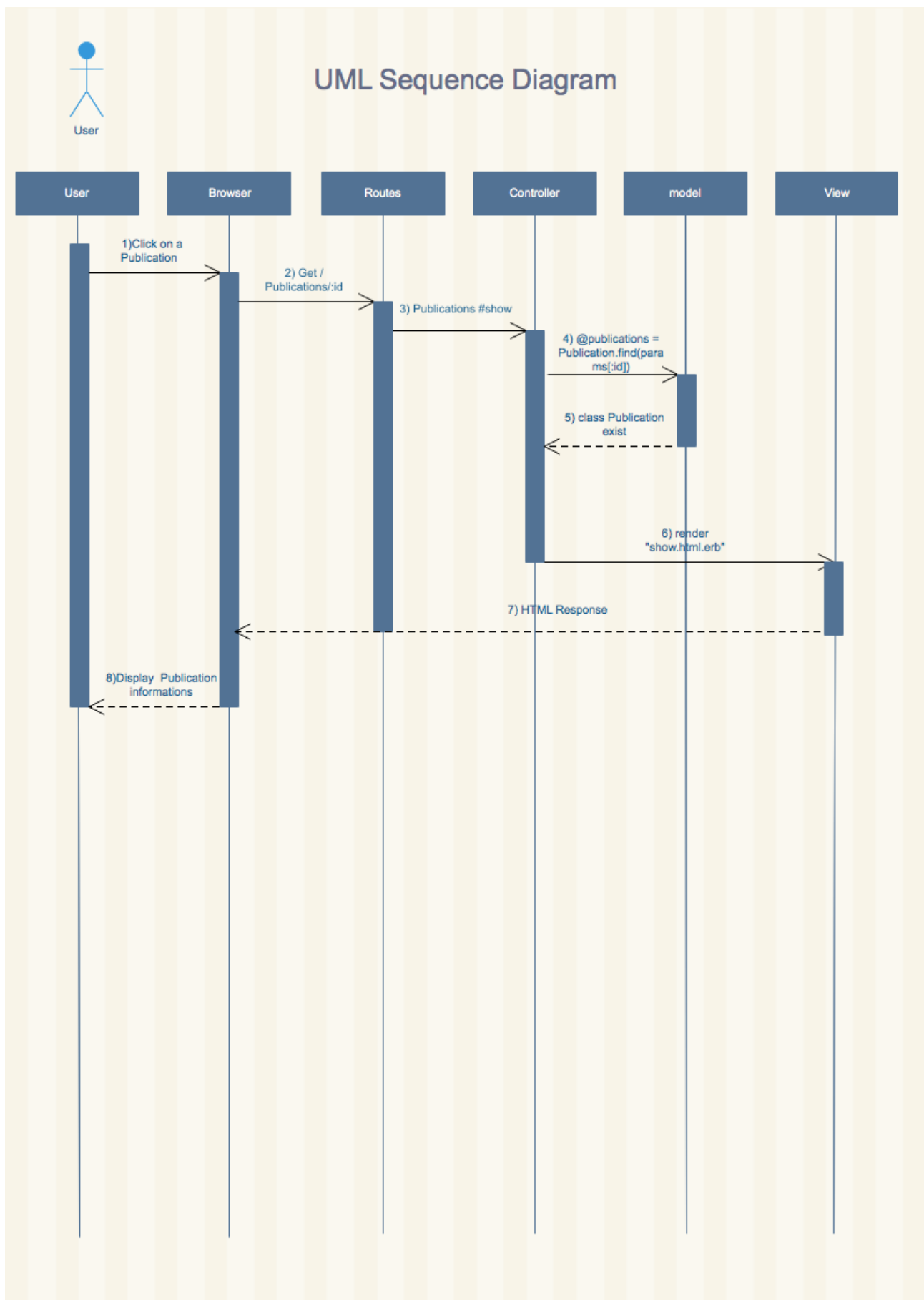
شکل ۵) نمودار توالی ثبت نام کاربر در سیستم



شکل ۶) نمودار توالی مشاهده تمامی مقالات

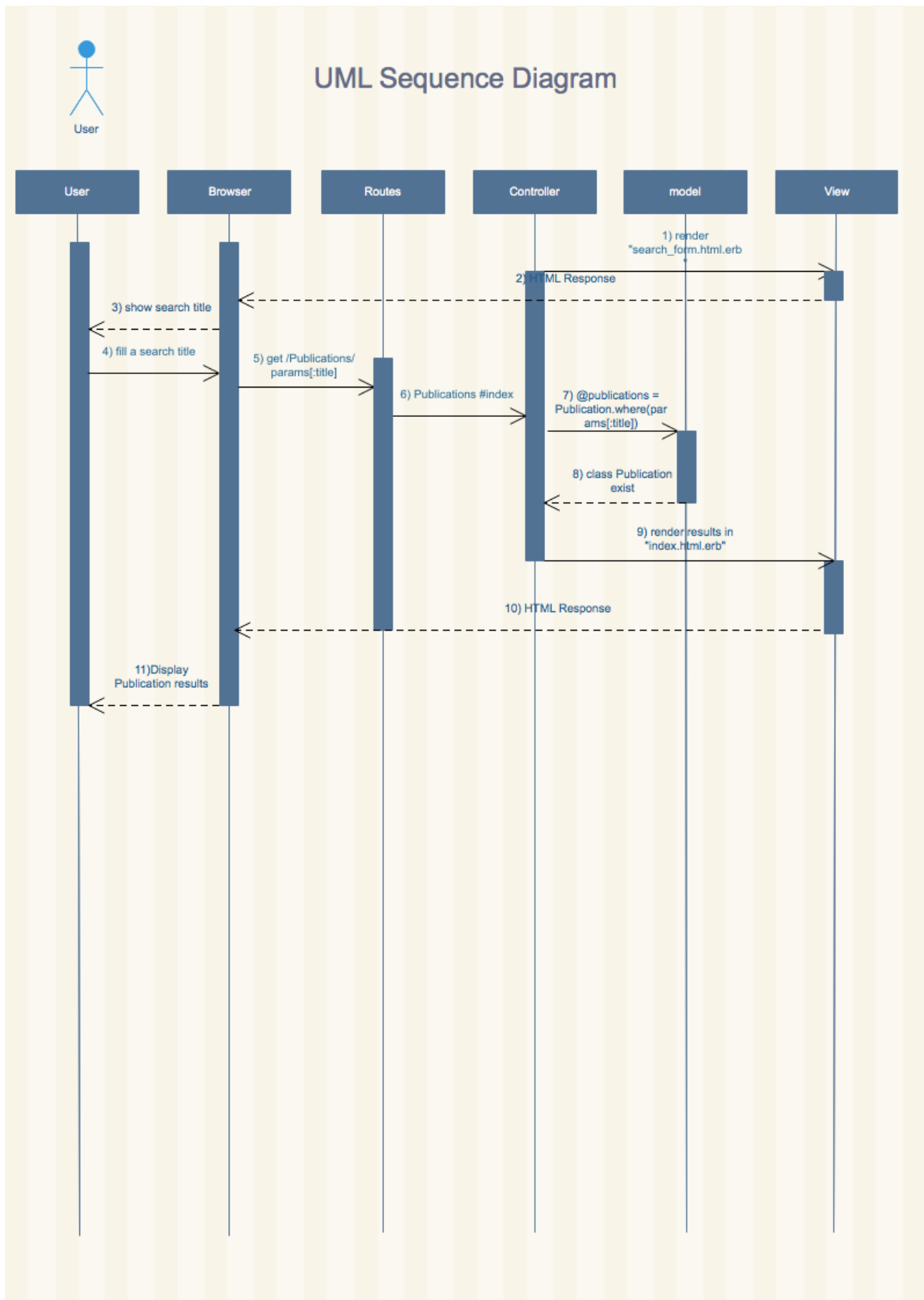


شکل ۷) نمودار توالی ایجاد مقاله ی جدید



شکل ۸) نمودار توالی نمایش مقاله

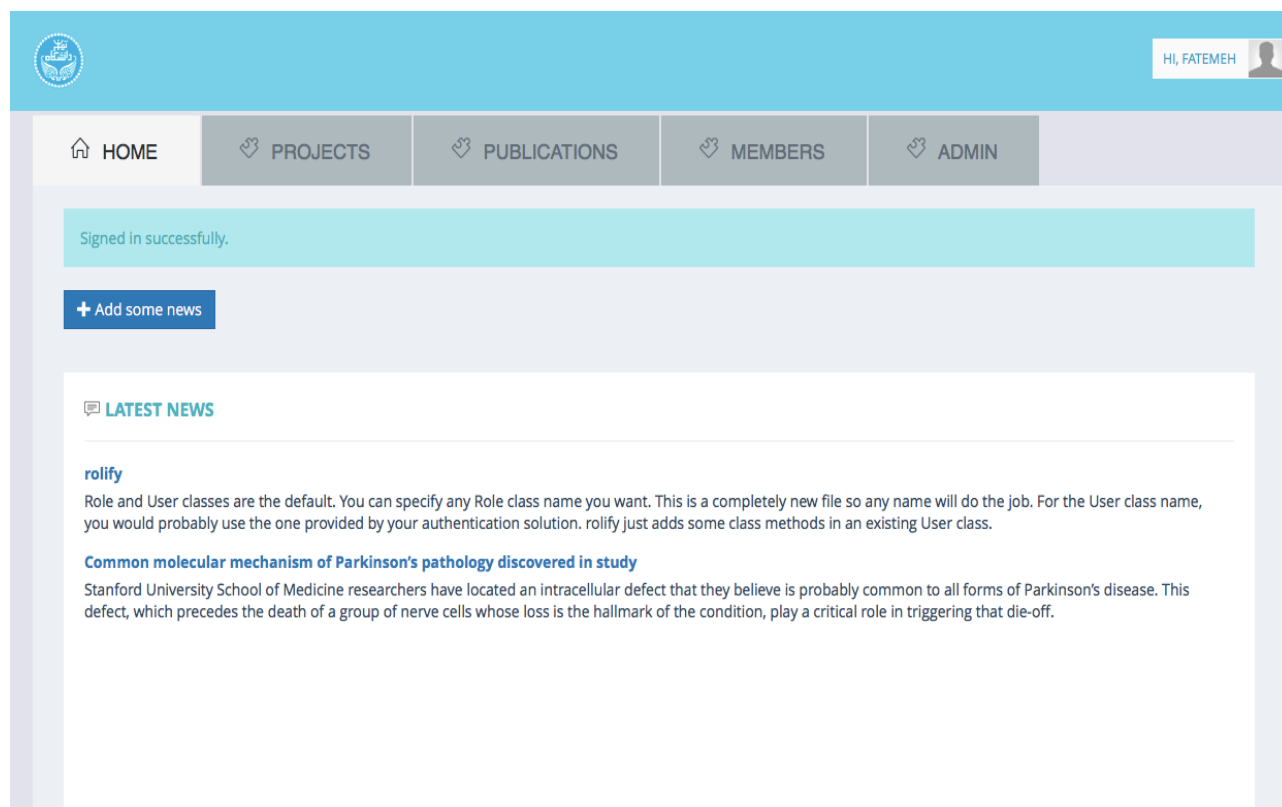




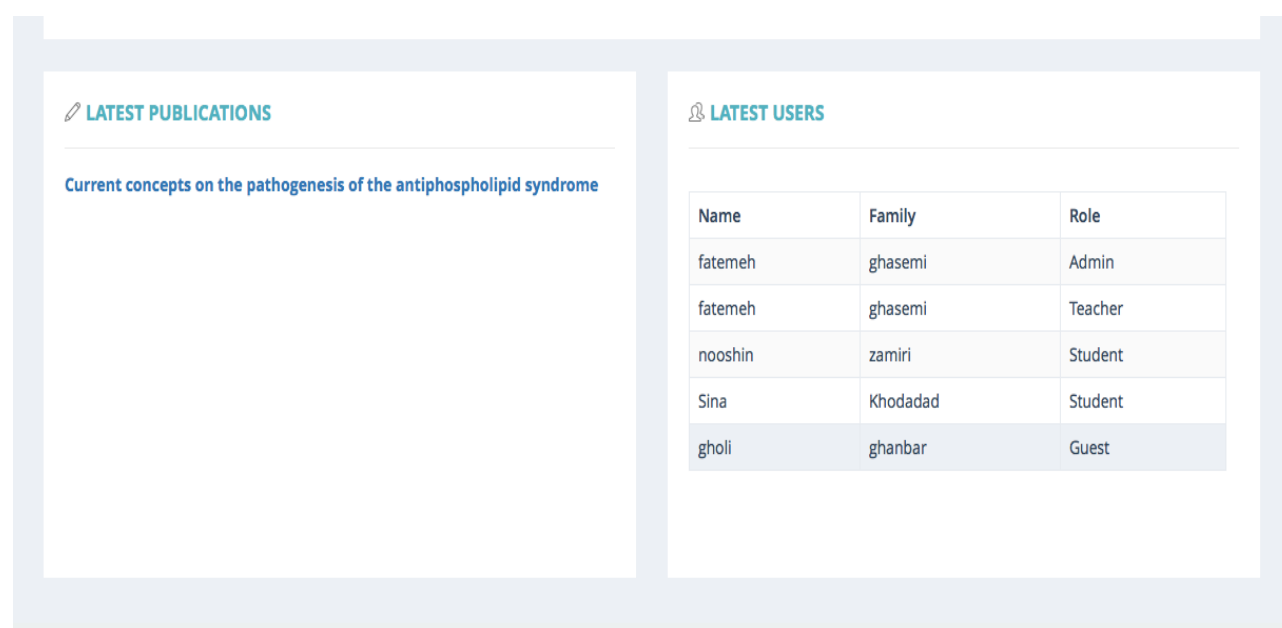
شکل ۹) نمودار توالی جستجوی مقاله

## معرفی سایت :

در این قسمت صفحات سایت را توضیح می‌دهیم. تصاویر ۱۰ نمایش دهنده ی صفحه ی اول سایت می‌باشد. در این تصاویر ادمین سایت وارد شده است. در صفحه ی اول آخرین اخبار، مقالات و کاربران نمایش داده شده است و همچنین ادمین می‌تواند خبر جدید ایجاد کند. در صورتی که کاربری با نقش دیگری وارد شود تب مخصوص ادمین نمایش داده نخواهد شد.



شکل (۱۰) صفحه ی اول سیستم



شکل (۱۱) صفحه ی اول سیستم

تصویر ۱۲ صفحه لاگین سایت را نمایش می‌دهد. کاربران برای ورود احتیاج به آدرس ایمیل و پسورد دارند. توجه شود که این آدرس ایمیل را در ابتدا ادمین هنگام ثبت نام افراد وارد می‌کند و از آن پس توسط همان آدرس ایمیل می‌توان وارد سایت شد. همچنین در صورت فراموشی رمز می‌تواند رمز خود را بازیابی کند.

شکل ۱۲) ورود به سایت

تصویر ۱۳ صفحه نمایش پروژه‌های موجود در سایت. ادمین می‌تواند پروژه جدید ایجاد کند، پروژه‌های ساخته شده را حذف کند (این امکان برای سایر کاربران وجود ندارد) و می‌تواند ویرایش انجام دهد. امکان حذف پروژه‌های موجود برای سایر کاربران وجود ندارد. همانطور که در تصویر نشان داده شده است، اطلاعاتی از پروژه قابل نمایش است مانند هدف پروژه، اسپانسر پروژه و تاریخ شروع و اتمام پروژه. همچنین می‌توان پروژه را ویرایش کرد و جزئیات بیشتر آن را مشاهده کرد.

Title	Sponsor Name	Goal	Start Date	End Date			
System Verification Lab	Sharif University	Managment	2012-09-06	2016-09-06	Show	Edit	Destroy
System Verification Lab master	university of tehran	Improvison and utilizing official managements	2015-09-06	2018-09-06	Show	Edit	Destroy
water	university of tehran	step by step	2012-09-10	2016-09-10	Show	Edit	Destroy

شکل ۱۳) پروژه‌های موجود در سایت

تصویر ۱۴ صفحه‌ی ایجاد یک پروژه‌ی جدید را نمایش می‌دهد. می‌توان افراد موجود در سایت را به پروژه اضافه کرد که با این کار پس از ایجاد پروژه، پروفایل کاربران به‌روز می‌شود و این پروژه در پروفایل آنان وجود خواهد داشت.

Back

## New project

Title

Goal

Start date

End time

Sponsor

users

Create Project

شکل ۱۴ ایجاد پروژه‌ی جدید

تصویر ۱۵ نمایش دهنده‌ی صفحه‌ی مقالات سایت می‌باشد. همچنین می‌توان مقاله‌ی جدید ایجاد کرد و در لیست مقالات جست و جو انجام داد. در اینجا نیز امکان حذف یک مقاله تنها برای ادمین وجود دارد.

+ New Publication

## Listing publications

Publication Title:

publication search

Title	Year	Volume	Pages	Journal_Conference			
Current concepts on the pathogenesis of the antiphospholipid syndrome	2016	44	12 2	International Journal	Show	Edit	Destroy

شکل ۱۵ صفحه‌ی مقالات

تصویر ۱۶ صفحه‌ی ساخت مقاله‌ی جدید است. در این قسمت یک مقاله‌ی جدید ایجاد می‌شود و مانند قسمت ایجاد پروژه کاربران در مقاله شرکت داده شده سپس پروفایل آنها به‌روز رسانی می‌شود. همچنین فایل با فرمت پی دی اف قابل آپلود کردن در این قسمت می‌باشد.

Back

## New publication

Title

Year

Volume

Page from:  from  to:  to

Authors  Nothing selected

Journal\_Conference  Select a Journal\_Conference

Documents    no file selected

شکل ۱۶ ایجاد مقاله‌ی جدید

تصویر ۱۷ افراد سایت را نمایش می‌دهد. می‌توان با جزییات بیشتر افراد را مشاهده کرد و همچنین جست و جو انجام داد.

## Listing profiles

First name:

Last name:

profile search

Name	Family	email	
ali	hosseini	ali@ali.com	<a href="#">show</a>
fatemeh	ghasemi	fatemeh@ghasemi.com	<a href="#">show</a>
nooshin	zamiri	nooshin@zamiri.com	<a href="#">show</a>
gholi	ghanbar	gholi@ghanbar.com	<a href="#">show</a>
ramtin	khosravi	ramtinkhosravi@gmail.com	<a href="#">show</a>

شکل ۱۷ اعضای سایت

به عنوان مثال با انتخاب یکی از کاربران لیست پروژه‌های کاربر و مقالات او نمایش داده می‌شود.

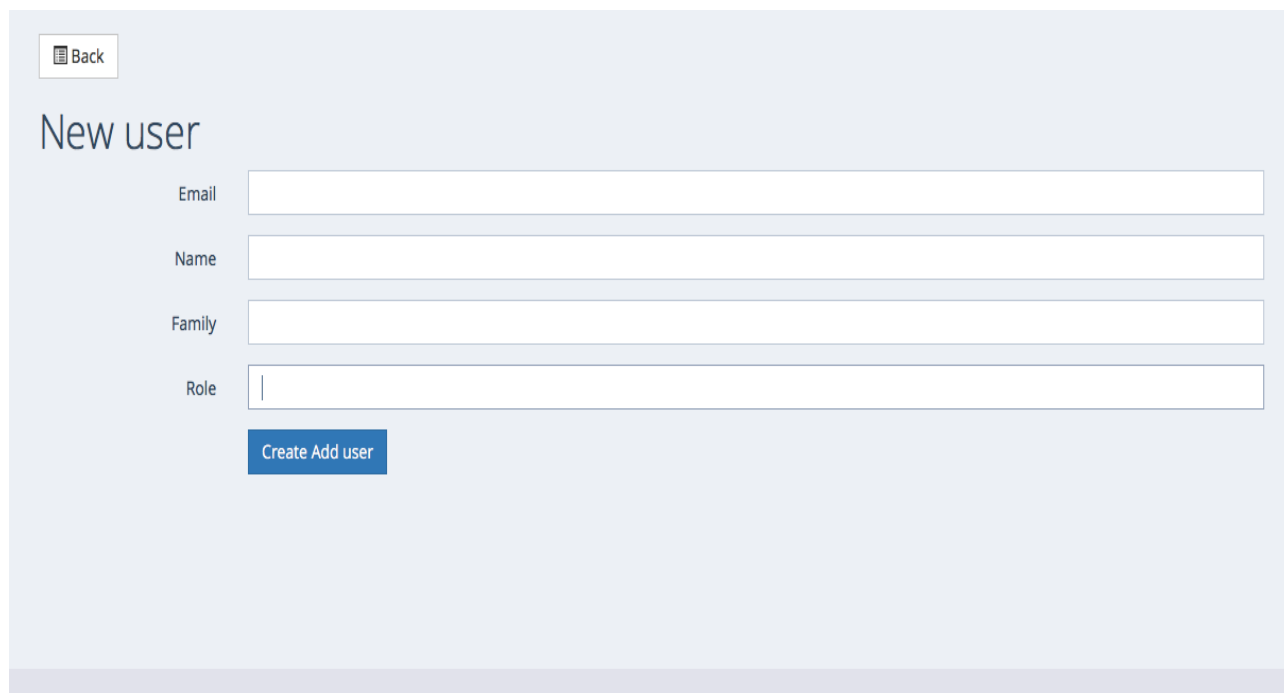
شکل ۱۸) جزئیات یک کاربر

تصویر ۱۹ تب مخصوص به ادمین است که به سایر کاربران نمایش داده نمی‌شود. در این قسمت ادمین لیست تمامی افراد را مشاهده می‌کند و امکان ویرایش آنها را دارد. همچنین ادمین می‌تواند اسپانسر جدید ایجاد کند و کابر جدید اضافه کند.

MEMBERS CONTEXT ADD USER NEW SPONSORS			
First Name	Last Name	Email	
ali	hosseini	ali@ali.com	Edit
fatemeh	ghasemi	fatemeh@ghasemi.com	Edit
nooshin	zamiri	nooshin@zamiri.com	Edit
gholi	ghanbar	gholi@ghanbar.com	Edit
ramtin	khosravi	ramtinkhosravi@gmail.com	Edit
Sina	Khodadad	sinakhodadad@yahoo.com	Edit

شکل ۱۹) تب مخصوص به ادمین

در تصویر ۲۰ ادمین کاربر جدید ایجاد می‌کند. پس از این قسمت کاربران می‌توانند ثبت‌نام خود را تکمیل کرده و پروفایل خود را ویرایش کنند.



Back

## New user

Email

Name

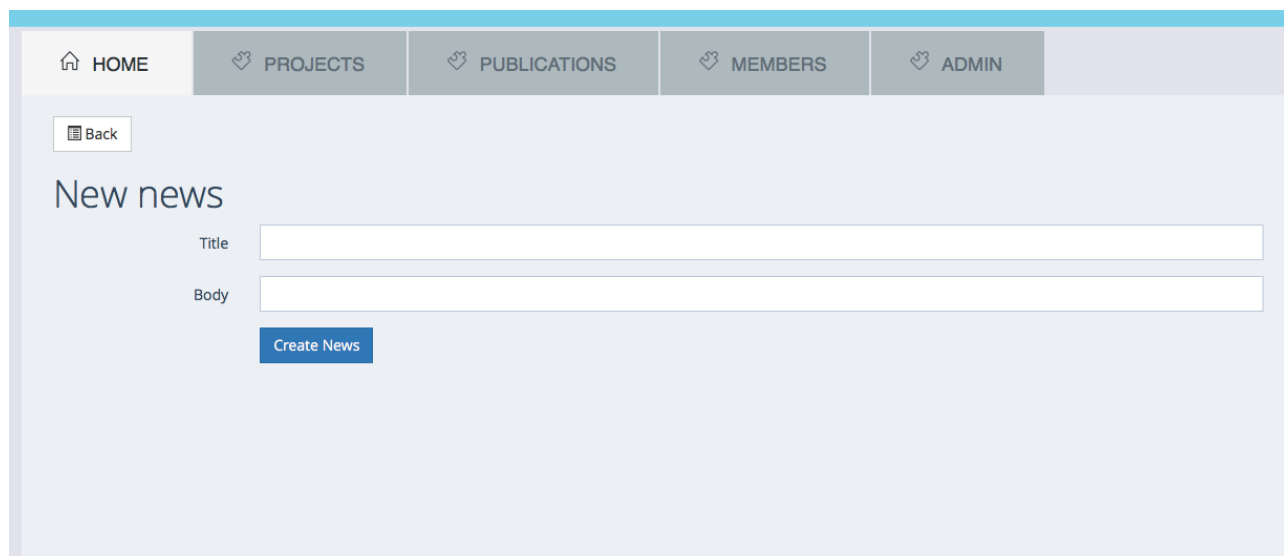
Family

Role

Create Add user

شکل ۲۰) ایجاد یک کاربر جدید

تصویر ۲۱ صفحه‌ی ساخت یک خبر جدید توسط ادمین است.



HOME PROJECTS PUBLICATIONS MEMBERS ADMIN

Back

## New news

Title

Body

Create News

شکل ۲۱) ایجاد یک خبر جدید

تصویر ۲۲ صفحه ی پروفایل یک فرد را نمایش می دهد. در این صفحه مقالات و پروژه های کاربر نمایش داده شده است. با کلیک بر روی نام مقاله فایل پی دی اف مقاله بارگزاری می شود. و همچنین با انتخاب نام پروژه ها فرد به صفحه ی پروژه ی مربوطه هدایت می شود.

HI, FATEMEH

HOME PROJECTS PUBLICATIONS MEMBERS ADMIN

**fatemeh ghasemi**  
ADMIN

3 PROJECTS 1 PUBLICATIONS

Email:  
fatemeh@ghasemi.com

Bio:

**PUBLICATIONS**

ghasemi & zamiri, Current concepts on the pathogenesis of the antiphospholipid syndrome , Vol. 2, No. 7, pp. 12-44, 2016

**PROJECTS**

System Verification Lab, The goal for this project is: Managment, Duration for implementing this project is from 2012-09-06 to 2016-09-06.

System Verification Lab master, The goal for this project is: Improvison and utilizing official managements, Duration for implementing this project is from 2015-09-06 to 2018-09-06.

water, The goal for this project is: step by step, Duration for implementing this project is from 2012-09-10 to 2016-09-10.

شکل ۲۲) صفحه ی پروفایل



## آزمون نرم افزار :

آزمون نرم افزار را می توان یکی از زیرمجموعه های مبحث کیفیت نرم افزار با نام تضمین کیفیت در نظر گرفت. آزمون به دنبال خطایابی محصول نرم افزاری قبل از تحویل به مشتری است.

تست نرم افزار را می توان به صورت های زیر معنا کرد:

- تلاش هایی در جهت عیب یابی و رفع آن، نه تلاش در جهت اثبات کامل صحت نرم افزار زیرا این قضیه با ماهیت تست تفاوت دارد.
- تحقیق بر روی کیفیت یک محصول
- تست ها همیشه شامل سوال و جواب هایی هستند که نرم افزار را با آن امتحان می کنیم در حالی که از برنامه انتظار داریم با توجه به ورودی هایی که با استفاده از سوالات وارد می کنیم، جواب های صحیحی را به عنوان خروجی به دست دهد.
- تست نرم افزار جزئی از بازبینی و اعتبارسنجی نرم افزار می باشد. بازبینی بررسی می کند که آیا نرم افزار از مشخصاتش پیروی می کند یا خیر. اعتبارسنجی باید تضمین کند که نرم افزار انتظارات مشتری را برآورده می سازد یا نه.

در واقع هدف فرایند تست در یک جمله ایجاد اعتماد نسبت به سیستم است.

چه زمانی تست شروع می شود؟

در طول چرخه عمر توسعه نرم افزار، تست آغاز شده و تا استقرار نرم افزار به طول می انجامد. با این حال تمام تست ها بستگی به مدا توسعه ای دارد که شرکت ها انجام می دهند. به طور مثال در مدل آبشاری، تست در مرحله تولید نرم افزار انجام می شود اما در مدل افزایشی، تست در پایان هر افزایش یا تغییر، تکرار می شود و در پایان تولید نرم افزار هم دوباره تست انجام می شود.

انجام تست در طول این چرخه مزایای زیر را در بر دارد:

- کاهش زمان تولید
- کاهش هزینه ها
- کاهش زمان دوباره کاری ها
- کاهش خطاهای نرم افزاری
- افزایش بازدهی
- افزایش کیفیت نرم افزار
- افزایش رضایت مشتری

چه زمانی تست پایان می یابد؟

تعیین زمان پایان تست بسیار دشوار است. تست فرایندی بی پایان می باشد و تعیین زمانی برای توقف آن بسیار دشوار است.

## آزمایش واحد<sup>۱۷</sup>

آزمایش واحد، بررسی صحت عملکرد قطعه‌ای از کد، به وسیله‌ی کدهای دیگری است که توسط برنامه نویسنده نوشته خواهند شد. عموماً این آزمایش‌ها جهت بررسی یک متد تهیه می‌شوند. در این مرحله باید در نظر داشت که هدف، بررسی کارایی نرم‌افزار نیست. هدف این است که بررسی کنیم آیا قطعه کد جدیدی که به برنامه اضافه شده است درست کار می‌کند و آیا هدف اصلی از توسعه‌ی آن را برآورده می‌سازد؟

اهمیت و مزایای آزمایش واحد :

نوشتن آزمایش‌های واحد به تولید کدهایی با کیفیت بالا در دراز مدت منجر خواهد شد. برای مثال لازم است قابلیت جدیدی را به برنامه‌ای که قبلاً نوشته‌ایم اضافه کنیم و برای اعمال این تغییرات نیاز است تا قسمتی از کدهای موجود تغییر کند، همچنین کلاس‌ها و متدهای دیگری نیز به برنامه افزوده گردند. حال چگونه اطمینان حاصل کنیم که قسمت‌های پیشین سیستم که درست کار می‌کنند، هم اکنون نیز مانند قبل کار می‌کنند؟ حجم کدهای نوشته شده بالا است و آزمایش دستی از نظر زمانی ممکن است مقدور نباشد. آزمایش واحد روشی برای حل این مشکل است. همچنین روال‌های آزمایشات صورت گرفته در آینده تبدیل به مرجع مهمی جهت درک چگونگی عملکرد قسمت‌های مختلف سیستم خواهند شد. با استفاده از آزمایش‌های واحد، بدترین حالات ممکن را قبل از وقوع می‌توان در نظر گرفت و بررسی کرد. از دیگر مزایای آن این است که برنامه‌نویس را وادار می‌کند کار خود را به واحدهای کوچکتری که قابلیت بررسی مستقلی دارند بشکند. اجرای این آزمایشات نیز به صورت خودکار صورت می‌گیرد.

خصوصیات یک آزمون مناسب :

- باید خودکار و قابل تکرار باشد. به عبارتی هر زمان که نیاز بود بتوان آن را دوباره اجرا کرد.
- اجرا و اعمال آن آسان باشد.
- یک بار نوشته شده و در آینده فقط اجرا شود.
- توسط هر فردی قابل اجرا باشد، زیرا ممکن است پروژه گروهی باشد لذا هر توسعه دهنده‌ی دیگری باید بتواند آن را اجرا کند.
- زمان تست کوتاه باشد.
- تست‌ها باید قابلیت ادغام با یکدیگر را داشته باشند زیرا نرم‌افزار ممکن است از هزاران واحد کوچک تشکیل شده باشد.

---

<sup>17</sup> Unit test

## مقدمه‌ای بر RSpec

در دنیای رویی، کتابخانه‌های تست متعددی وجود دارد. یکی از آنان که دارای محبوبیت زیادی است RSpec می‌باشد. این کتابخانه از روش کمی متفاوتی برای آزمون یک نرم‌افزار، از طریق آزمون یک رفتار بجای آزمون متدهای مشخص، استفاده کرده است.

RSpec این امکان را فراهم می‌سازد که چیزی را که تست می‌کنیم در بلاک `describe` کپسوله سازی کنیم. همین‌طور برای اینکه تست‌ها را به صورت اتوماتیک انجام دهیم از `guard` استفاده می‌کنیم که این بخش با توجه به اینکه ما کدام فایل را تغییر می‌دهیم تست‌های مربوط به آن را اجرا می‌کنید و باعث می‌شود که در زمان ما صرفه جویی شود.

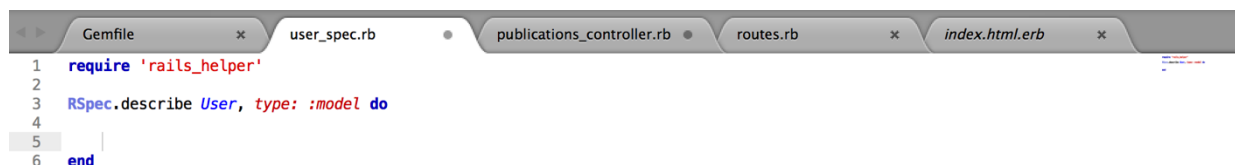
در یونیت تست از `describe` برای توضیح رفتار یک کلاس استفاده می‌کنیم:



```
Gemfile x user_spec.rb publications_controller.rb routes.rb index.html.erb x
1 require 'rails_helper'
2
3 RSpec.describe User, type: :model do
4
5
6 end
```

شکل (۲۳) آزمون نرم افزار

تست ها با استفاده از بلاک `it` نوشته می‌شوند.



```
Gemfile x user_spec.rb publications_controller.rb routes.rb index.html.erb x
1 require 'rails_helper'
2
3 RSpec.describe User, type: :model do
4
5
6 end
```

شکل (۲۴) آزمون نرم افزار

برای استفاده از RSpec لازم است که جم آن را نصب کنیم و برای این کار دستور `gem install rspec` را استفاده می‌کنیم.

حالت های متفاوت برای تست را با استفاده از `before` و `after` پیاده سازی می کنیم.

```
require 'rails_helper'

RSpec.describe News, type: :model do

  before do
    @news = News.new({:title => 'Service 101: Thousands of Bruins volunteer across Los Angeles', :body => 'The sleeping giant that is Volunteer Day began rousing before dawn; by 8 a.m. a slow-moving human "river" forr Amid upbeat greetings of "Happy Volunteer Day!" from residence hall staff, the new students from near and far – from distar'})
  end

  it {should validate_presence_of(:title)}
  it {should validate_presence_of(:body)}
end
```

شکل ۲۵) استفاده از `before` و `after` در نوشتن آزمون نرم افزار

نتیجه گیری:

در این پروژه با استفاده از محیط روبی آن ریلز که محیطی نسبتاً جدید، گرافیکی و کاربر پسندتر است به پیاده سازی سیستم مورد نظر پرداختیم. برای پیاده سازی از معماری سه لایه ام وی سی استفاده کردیم. همچنین از gem های devise، rolify، paperclip برای ایجاد کاربر، تعیین نقش ها، بارگذاری عکس و ایجاد دسترسی استفاده کردیم. همچنین آشنایی با مفاهیم آزمون نرم افزار و آزمایش واحد به دست آمد که موفق شدیم آن را در پروژه پیاده سازی کنیم. از دیگر دستاوردهای این پروژه مرور و مطالعه ی دقیق تر روش های توسعه ی نرم افزار و نمودارهای کمک کننده در توسعه می باشد.

مراجع:

[1] J. R. Michael Bachle, "Ruby on Rails," 2008.

[2] D. H. H. Dave Thomas, Agile Web Development with Rails, 2010.

- [3] Hansson, David Heinemeier. "Ruby on rails." Website. Projektseite: <http://www.rubyonrails.org> (2009).
- [4] Agile software development methods (By Pekka Abrahamson, Outi Salo Jussi Ronkainen & Juhani Warsta).
- [5] Principles behind the Agile Manifesto (<http://agilemanifesto.org>) (<http://agilemanifesto.org/iso/pr/>)
- [6] Rolify Gem, (2016), GitHub repository, <https://github.com/RolifyCommunity/rolify>
- [7] RSpec Gem, (2016), Github repository, <https://github.com/rspec/rspec-rails>
- [8] platformatic Devise Gem, (2016), GitHub repository, <https://github.com/plataformatec/devise>
- [9] thoughtbot Paperclip Gem, (2016), GitHub repository, <https://github.com/thoughtbot/paperclip>