

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



دانشکده مهندسی

پایان نامه‌ی کارشناسی

گرایش نرم افزار

پیاده سازی سیستم منشی دیجیتال

نگارش:

سینا ابراهیمی

استاد راهنما:

جناب آقای مهندس داوود محمدپور

پاییز ۹۵

تقدیم به

مقدس‌ترین واژه‌ها در لغت نامه دلم، مادر مهربانم که زندگیم را مدیون مهر و عطوفت آن می‌دانم.

پدر، مهربانی مشفق، بردبار و حامی.

خواهر و برادرم همراهان همیشگی و پشتوانه‌های زندگیم.

تمامی دوستانم که در مراحل مختلف زندگی حامی من بوده‌اند.

و با تشکر فراوان از استاد گرامیم، جناب مهندس محمدپور

استادی که سپیدی را بر تخته سیاه زندگیم نگاشت.

چکیده

سیستم منشی دیجیتال با هدف گرامیداشت و ارزش نهادن به مفهوم زمان ایده پردازی شده است. هدف اصلی این سیستم توانایی برنامه ریزی و تبلیغ غیر مستقیم برای صاحبان مشاغل است. مشتری ها در این سیستم توانایی جستجوی مشاغل مختلف را دارند و می توانند از آنها بر حسب نیاز خودشان وقت بگیرند. در این سیستم نقش های صاحب شغل، همکار و کاربر عادی تعریف شده اند که هر یک سطح دسترسی خاصی به سیستم دارند. سعی بر این بود تا بتوانم سیستمی پیاده سازی نمایم که برای کاربران نهایی بسیار ساده بوده و قابلیت اعلان نیز بصورت داخلی و اتوماتیک در این سیستم پیاده سازی شود.

فصل اول

مفاهیم اولیه و تکنولوژی

۱.۱: مقدمه

نسل جدید زبان‌های برنامه‌نویسی به این دلیل تولید می‌شود که زبان‌های قدیمی‌تر دارای امکانات محدود بوده و یا قدرت استفاده از تکنولوژی‌های فعلی را بصورت مطلوب ندارند. ASP.Net نسل جدیدی از ASP بوده که توسط شرکت Microsoft عرضه گردیده‌است. ASP.Net اولین سیستم جامع برای برنامه‌نویسی تحت وب (اینترنت) است که از مراحل سطح پایین آن یعنی زبان ماشین تا بالاترین سطح آن که برنامه‌نویسی ویژوال می‌باشد برای استفاده در اینترنت و شبکه‌های محلی طراحی شده‌است. بزرگترین مزیت آن در برابر سیستم‌های دیگر، امکانات اینترنت آن است. از ASP.Net می‌توان در طراحی و تولید سایت‌های وب اینترنت کوچک یک شرکت و نیز سایت‌های وب تجاری خیلی بزرگ استفاده نمود.

مهمترین نکته‌ای که در طراحی این محصول در نظر گرفته شده‌است، استفاده‌ی آسان، کارآیی بالا و نیز قابلیت فوق‌العاده این نرم‌افزار است. ASP.NET MVC بخشی از چارچوب کاری برنامه‌های وب ASP.NET application framework است، MVC یکی از دو نوع مدل برنامه‌سازی ASP.NET است.

در واقع ASP.NET MVC پیاده‌سازی مدل MVC به کمک ابزارهای توسعه نرم‌افزاری Microsoft در بستر وب (ASP.NET) است.

۱.۲: معرفی ASP.NET

زمانی که شرکت مایکروسافت نسخه‌ی اولیه‌ی .Net Framework را به نمایش گذاشت، برای اکثر طراحان و پیاده‌کنندگان مشخص شد که می‌بایست در انتظار تحولات چشمگیری در این عرصه باشند. ASP3 امکانات گسترده و انعطاف لازم را به منظور ایجاد سایت‌های پویا در اختیار علاقه‌مندان قرار می‌داد. تاکنون صدها کتاب و مقاله با موضوع ASP نوشته شده‌است. حاصل تمامی تلاش‌های انجام شده در این رابطه، تسهیل در امر طراحی و پیاده‌سازی وب‌سایت‌های پویا و برنامه‌های وب بود. چیزی که ASP نداشت، یک Framework برنامه‌نویسی بود. هر چیزی که در ASP انجام می‌شد با رویکرد کدنویسی بود و برنامه‌نویسان برای انجام هر کاری ملزم به نوشتن کدهای مورد نیاز بودند. ASP.Net با هدف غلبه بر محدودیت فوق طراحی شده‌است. ASP.Net یک Framework برنامه‌نویسی است که بر روی CLR3 ایجاد می‌شود و می‌توان از آن به منظور ایجاد برنامه‌های قدرتمند وب استفاده کرد. یکی از اهداف اساسی طراحی ASP.Net برنامه‌نویسی

ساده‌تر و با سرعت بیشتر از طریق کاهش حجم کدهای مورد نیازی است که برنامه‌نویسان ملزم به نوشتن آن‌ها می‌باشند. برنامه‌نویسی اعلانی، مجموعه‌ای گسترده از کنترل‌های سرویس‌دهنده به همراه رویدادهای مربوطه، یک کتابخانه کلاس بزرگ و ابزارهای پیاده‌سازی کاملاً حرفه‌ای نظیر Visual Studio.Net، از جمله ویژگی‌های شاخص ASP.Net محسوب می‌شود. کنترل‌های سرویس‌دهنده، معماری Postback، حمایت از حالت دید^۱ و استفاده از کد ترجمه شده و مدل برنامه‌نویسی مبتنی بر رویداد از مهم‌ترین دست‌آوردهای ASP.Net محسوب می‌شوند که نوید ایجاد و اشکال‌زدایی سریع برنامه‌های وب را در اختیار پیاده‌کنندگان قرار می‌دهد.

• مقایسه ASP.Net و ASP Classic

ASP.Net نسل بعدی ASP Classic است. اما این یک پیشرفت تکاملی است بطوریکه این دو فناوری تقریباً از یکدیگر متفاوت‌اند. صفحات ASP با زبان‌های دستورالعمل‌نویسی^۲ مانند VB Script یا Java Script ایجاد می‌شوند.

اما در ASP.Net ما یک فرایند کامل برنامه‌نویسی با زبان‌های Visual Basic یا C# داریم. همچنین در ASP Classic تنها پنج کلاس استاندارد Session، Response، Request، Server و Application وجود دارد. حال آن که ASP.Net می‌تواند از بیش از 4500 کلاس استاندارد موجود در بدنه‌ی .Net بهره‌جست. همچنین علی‌رغم قدرت و امکانات زیاد و متعدد ASP.Net استفاده از آن در مقایسه با ASP Classic بسیار آسان‌تر است. به عنوان مثال با استفاده از چند ابزار در یک صفحه‌ی ASP.Net، می‌توان یک صفحه بسیار پیچیده HTML به دست‌آورد که ساخت آن در ASP Classic نیاز به چند روز کار دارد.

۱.۲.۱: معرفی IIS

سرویس Internet Information Service یا به اختصار همان IIS وب سرور قدرتمند، قابل انعطاف، امن و با قابلیت کاربری آسان و محصول شرکت Microsoft می‌باشد که امکانات بسیار زیادی از قبیل Media Streaming، میزبانی وب، Application و ... را در اختیار وب‌سایت‌های مربوطه قرار می‌دهد.

¹ View State

² Scripting Languages

وب سرور IIS برای ایجاد، مدیریت و میزبانی وب^۳ مورد استفاده قرار می‌گیرد. این نرم‌افزار از امکاناتی مانند HTTP، HTTPS، SMTP، FTP، SFTP و NNTP به خوبی پشتیبانی می‌نماید و به صورت کامل با ویندوز سرور سازگار می‌باشد.

نسخه‌های قدیمی‌تر IIS دارای آسیب پذیری‌های بسیار زیادی بودند، که معروف‌ترین آن‌ها مربوط به کدهای مخرب Code Red Worm در سال ۲۰۰۱ می‌باشد. گرچه تا به حال هیچ‌گونه گزارشی مبنی بر وجود حفره‌های امنیتی در نسخه‌های ۶ و ۷ این نرم‌افزار منتشر نشده‌است اما Microsoft برای اطمینان هرچه بیشتر کاربران به طور کلی سعی در تغییر ساختار امنیتی نرم‌افزار سرویس‌دهنده وب خود در نسخه ۷،۵ نموده است.

یکی از این امکانات، قابلیت تحت عنوان Web Service Extension از نسخه IIS 6.0 اضافه گردیده که این نرم‌افزار را از اجرای هر برنامه ثالثی بدون اجازه سرویس‌دهنده منع می‌نماید.

تمامی اجزای نرم‌افزار مذکور در نسخه IIS 7 به صورت پودمانی^۴ تغییر یافته است، یعنی هر یک از قابلیت‌های آن می‌توانند به صورت منحصر بفرد نصب و یا حذف شوند.

برای برنامه‌نویسی وب در Visual Studio، ابتدا باید IIS را نصب کرد و بعد از آن اقدام به نصب Visual Studio شود. البته فارغ از این که روی چه سیستم عاملی از Microsoft Visual Studio نصب شده باشد، خود ابزار Visual Studio امکانات IIS را فراهم خواهد کرد. لازم به ذکر است که Microsoft با استفاده از زبان C++ این سرویس‌دهنده وب را توسعه داده است.

در پروژه ذیل از IIS 10.0 استفاده شده‌است که لازم به ذکر است این نسخه از IIS به همراه Windows 10 و Windows Server 2016 در تابستان ۲۰۱۵ عرضه شده‌است.

۱.۲.۲: معرفی Framework

Framework مجموعه منسجم از کلاس‌ها و توابع از پیش تعریف شده‌است که قابلیت‌های بالقوه گوناگون از یک زبان برنامه‌نویسی را در خود دارد و بدین ترتیب کاربر نهایی را قادر می‌سازد که از امکانات یک زبان استفاده

³ Web Hosting

⁴ modular

کند بدون اینکه درگیر مسائل پیچیده و وقت گیر آن شود، لذا همان طور که یک کلاس یا تابع با هدف جلوگیری از تکرار و افزایش سرعت کار، تعریف می‌شود، Framework از این هم فراتر رفته و علاوه بر افزایش سرعت، مواردی مثل توسعه‌پذیری و ساده‌سازی را هم مدنظر دارد.

NET Framework. از دو بخش اصلی، کتابخانه .net^۵ و CLR^۶ تشکیل شده‌است.

کتابخانه Net. مجموعه‌ای از کلاس‌های آماده از پیش نوشته شده‌است که در تمامی زبان‌های مبتنی بر Net. قابل استفاده هستند. اما CLR وظیفه اجرا و مدیریت برنامه‌های تحت Net. را دارد. کنترل و مدیریت اجرای برنامه‌ها، مدیریت حافظه و کنترل و مدیریت امنیت در برنامه‌های Net. از وظایف CLR هست.

Frameworkها در واقع یک هسته اصلی می‌باشند که هرگونه اضافه کردن، توسعه یا حذف اشیا می‌تواند به راحتی و در مقابل فایل‌های کتابخانه‌ای و وابسته به هسته اصلی این Framework در نظر گرفته شود.

هم‌چنین ایجاد یک برنامه کاربردی تحت‌وب بر مبنای یک Framework باعث می‌گردد تا توسعه‌دهندگان و برنامه‌نویسان علاوه بر تجارب خود بتوانند از تجارب سایر توسعه‌دهندگان نیز سود جسته و به کار بندند؛ مشخص بودن نوع معماری به کار رفته نیز باعث می‌گردد تا توسعه‌دهندگان مختلف بتوانند به راحتی سیستم خود را توسعه دهند.

۱.۳: معرفی معماری MVC

نام MVC از Model-View-Controller گرفته شده و هدف اصلی آن جدا سازی اجزای تشکیل دهنده برنامه است، بخصوص برنامه‌هایی که دارای واسط کاربری گرافیکی^۷ هستند.

M در MVC معادل مدل^۸ است و حاوی اطلاعاتی است که نهایتاً در اختیار کاربر قرار خواهد گرفت. Model وظیفه کار با پایگاه داده و دیگر اشیا را بر عهده دارد.

^۵ .NET Framework Class Library

^۶ Common Language Runtime

^۷ Graphical User Interface (GUI)

^۸ Model

V در MVC معادل دید^۹ است و حاوی نتیجه‌ای است که کاربر نهایتاً در مرورگر خواهد دید. view در واقع User Interface برنامه است و وظیفه ارتباط با کاربر نهایی را بر عهده دارد.

C در MVC معادل کنترل‌گر^{۱۰} است وظیفه کنترل View و Model و نحوه ارتباط آن دو را با هم بر عهده دارد.

در واقع MVC بر روی معماری‌های چندلایه‌ای، جهت جداسازی قسمت‌های مختلف برنامه و به طور دقیق تر، جدا کردن بخش‌های منطقی برنامه اعم از داده‌ها^{۱۱}، مجوزهای دسترسی^{۱۲}، چک کردن صحت داده‌ها و... از لایه نمایش^{۱۳} یا در واقع همان لایه‌ای که مستقیماً با کاربر نهایی در ارتباط است (مانند فرم‌ها، اجزا و...) قرار می‌گیرد. الگوی طراحی MVC در لایه نمایش در معماری سه لایه استفاده می‌شود و هدف نهایی آن جداسازی مفاهیم در لایه نمایش به منظور خواناتر کردن کد و بالا بردن قابلیت نگهداری^{۱۴} آن است.

این الگوی طراحی در سایر Frameworkها مثل Ruby on Rails و Django و Zend هم به کار برده شده‌است.

۱.۳.۱: معرفی MVC 5

چارچوب ASP.NET MVC یک چارچوب نرم افزاری تحت وب است که الگوی MVC (Model, View, Controller) را پیاده‌سازی می‌کند.

مهم‌ترین ویژگی‌های MVC 5 که در این پروژه مورد استفاده قرار دادیم:

- استفاده از سیستم Routing موجود در زیرساخت ASP.NET برای نمایش URLهایی بدون پسوند. برای مثال به جای این که آدرس localhost/Account/SignIn.cshtml را داشته باشیم آدرس localhost/Account/SignIn را خواهیم داشت.

⁹ View

¹⁰ Controller

¹¹ Data

¹² Permission

¹³ Presentation Layer

¹⁴ Maintainability

- مدیریت بهتر قسمت‌های مختلف سایت در پوشه‌های جداگانه: برای مثال View های هر مدل به طور اتوماتیک در پوشه مخصوص خود جای می‌گیرد و این امر ذاتاً باعث زیباتر شدن پروژه و قابلیت خوانایی بیشتر می‌شود.
- مقداری خودکار مدل متناظر با یک View در ASP.NET MVC به این صورت که در Controller مربوطه متد مربوط به آن View به صورت اتوماتیک ایجاد می‌شود.
- کنترل بهتر بر روی اعتبارسنجی اطلاعات دریافتی
- امکان استفاده از فرم ها و View های Razor به جای موتور وب فرم ها
- سازگاری کامل با JQuery Ajax و JQuery و انواع Framework های JavaScript
- امکانات فشرده سازی CSS و JS
- استفاده از سیستم کدسازی خودکار به نام Scaffolding

۱.۳.۲: معرفی Razor

Razor یک Syntax برنامه نویسی ASP.NET برای ساختن صفحات وب پویا (همان فرمت cshtml) به کمک C# و یا Visual Basic .NET است. به این صورت که یک صفحه وب برای مرورگر نوشته شده است، کدهای مبتنی بر سرور می‌توانند محتوای پویا ایجاد کنند. وقتی صفحه وبی فراخوانی می‌شود، سرور کدهای سمت سرور داخل صفحه را قبل از برگرداندن صفحه به مرورگر اجرا می‌کند.

این کدها عملیات پیچیده‌ای را، مانند دستیابی به پایگاه داده، انجام می‌دهند.

Razor بر مبنای ASP.NET است، و برای ایجاد کاربردهای وب طراحی گردیده است و قابلیت‌های علامت گذاری ASP.NET سنتی را دارد، اما استفاده از آن و یادگیری آن آسان تر است. Razor اولین بار در Visual Studio 2010 عرضه شد.

Razor یک View Engine با Syntax ساده است که اولین بار به عنوان بخشی از MVC 3 عرضه شد. لازم به ذکر است که فایل‌های تولید شده توسط این View Engine با فرمت cshtml ذخیره می‌شوند.

برای مثال اگر در قسمت کنترلر یک لیست از پایگاه داده بخوانیم و آن را داخل یک ViewBag بریزیم، در داخل View خود می‌توانیم آن ViewBag را توسط یک foreach فراخوانی کنیم و لیست مورد نظرمان

را به کاربر نشان بدهیم. توانایی ترجمه دستوراتی مانند `foreach` و `if` در فایل `html` که اکثراً هم با نماد `@` شروع می‌شوند از جمله قابلیت‌های موتور `Razor` می‌باشد.

۱.۳.۳: معرفی Entity Framework

Entity Framework با ارائه یک مدل مفهومی که با پایگاه داده و برنامه کار می‌کند، به عنوان یک واسطه عمل می‌کند و به توسعه دهنده این قابلیت را می‌دهد که با داده‌های `Database` هم‌چون `Object`‌هایی در زبان `C#` یا `VB` کار کند و به جای درگیر شدن مستقیم با ساختار پایگاه داده و برنامه‌نویسی مستقیم بر روی داده‌های آن، بر روی تعدادی موجودیت^{۱۵} که از روی پایگاه داده ساخته شده‌اند، پرس‌وجو^{۱۶} بنویسد، عملیات `CRUD` (`Create, Read, Update, Delete`) را انجام دهد، رابطه‌های بین موجودیت‌ها را مدیریت کند و حتی از روابط ارث‌بری بین موجودیت‌ها بهره ببرد. مجموعه این موجودیت‌ها و روابط بین آن‌ها، مدل مفهومی ما را در Entity Framework تشکیل می‌دهند که با نام `EDM (Entity Data Model)` شناخته می‌شود. لازم به ذکر است که Entity Framework یک ابزار `ORM`^{۱۷} می‌باشد.

به‌طور کلی در `EF` سه حالت مدل سازی موجود است:

- **Database First Modeling**: در این روش مدل ما از روی یک پایگاه داده موجود ایجاد می‌شود و می‌توان از طریق `Entity Data Model Designer` در `Visual Studio` تغییرات لازم را بر روی مدل انجام داد و یا در پایگاه داده تغییرات را اعمال کرده و در `Model Designer` مدل خود را `Update` کنیم.
- **Model First Modeling**: در این روش ابتدا مدل توسط برنامه نویس در محیط `Entity Data Model Designer` ایجاد می‌شود، سپس به‌طور خودکار پایگاه داده و کدها و اسکریپت‌های موردنیاز از روی مدل ساخته می‌شوند.

¹⁵ Entity

¹⁶ Query

¹⁷ Object-Relational Mapping

- **Code First Modeling**: در این روش کلاس‌های معادل موجودیت‌ها (جداول) توسط برنامه‌نویس نوشته می‌شوند (این کلاس‌ها POCO-Plain OLD CLR Objects نامیده می‌شوند) سپس EF بطور خودکار پایگاه داده و مدل را از روی این کلاس‌ها می‌سازد.

در واقع در Entity Framework با پایگاه داده و جداول آن مانند اشیائی برخورد می‌کنیم که مکانیزم‌های زیر برایشان فراهم شده است:

- انجام عملگرهای پایه (Create, Read, Update, Delete) CRUD
- مدیریت آسان رابطه‌های یک به یک، یک به چند و چند به چند
- قابلیت داشتن روابط ارث‌بری بین Entity ها
- قابلیت تبدیل اشیاء پایگاه داده به کلاس‌ها (مدل‌ها)

مزایای استفاده از Entity Framework به شرح زیر است:

- دسترسی به داده‌ها در یک زبان سطح بالا با استفاده از دستوراتی مانند Select و Where.
- مدل Conceptual را می‌توان با استفاده از روابط بین Entity ها بیان نمود. (تبدیل به مدل)
- مدیریت آسان تر داده‌ها، مانند افزودن، حذف و به‌روز رسانی
- هم‌چنین در ASP.NET و Microsoft MVC 5 Framework مکانیزم‌هایی برای آسان‌سازی ایجاد CRUD در IDE فراهم نموده است که می‌توان با چند کلیک به سادگی برای یک مدل خود Controller و چند View مربوط به عملیات CRUD را ایجاد نمود و در وقت صرفه‌جویی کرد. (این ویژگی را Scaffolding می‌نامند).

لازم به ذکر است که در این پروژه ابتدا پایگاه داده طراحی شد و سپس با استفاده از روش Database First دسترسی به پایگاه داده‌ها را با استفاده از Entity Framework فراهم نمودیم.

۱.۴: معرفی تکنولوژی Bootstrap

افزایش روزافزون استفاده از ابزارهای مختلف در طراحی سایت و همچنین تلاش طراحان سایت برای ایجاد سایتی کاربرپسند و زیبا موجب این شده است که استفاده از طراحی‌های پیش ساخته در طراحی سایت امروزه بیشتر متداول شود. از جمله امکانات Bootstrap می‌توان به چارچوب آن اشاره نمود؛ این چارچوب یا Framework با تمامی مرورگرهای استاندارد همخوانی داشته و حتی در نسخه‌های قدیمی‌تر مانند اینترنت اکسپلورر ۸ نیز ظاهر زیبای خود را حفظ میکند. از نسخه دوم Bootstrap به بعد طراحی واکنش‌گرا یا responsive نیز در آن لحاظ شد که موجب نمایش مناسب صفحه در تلفن‌های هوشمند و تبلت‌ها می‌گردد. همچنین می‌توان به متن باز بودن آن اشاره نمود. تا به اینجا نیز نسخه سوم این ابزار کاربردی منتشر شده است و ما در این پروژه از نسخه‌ی Bootstrap 3.3.6 استفاده کرده‌ایم.

۱.۴.۱: Bootstrap چیست؟

Bootstrap مجموعه‌ای از ابزارهای رایگان برای ایجاد صفحات وب و نرم افزارهای تحت وب است که شامل دستورات CSS، HTML و توابع JavaScript جهت تولید و نمایش فرم‌ها، دکمه‌ها، سربرگ‌ها، ستون‌ها و سایر المان‌های مورد نیاز طراحی وب می‌باشد.

Bootstrap در ابتدا توسط Mark Otto و Jacob Thornton در جهت ایجاد یک چارچوب ظاهری مشخص و یکسان در ابزارهای توییت‌ر طراحی و نوشته شد. قبل از شروع این پروژه نمونه‌های زیادی با همین رویکرد ایجاد شده بود که همگی با سرنوشتی مشابه و عدم استقبال طراحان وب دنیا مواجه شده بودند. به دلیل وجود مشکلات اساسی در نمونه‌های دیگر، سازنده اصلی توییت‌ر یا همان مارک اتو تصمیم به ساخت یک سیستم داخلی و قدرتمند برای خود را با نام Bootstrap گرفت.

۱.۴.۲: Bootstrap به زبان ساده

بزرگترین مشکل طراحان وب و کدنویسان قدیمی، ایجاد ظاهری زیبا و مناسب است. اصول کدنویسی و ایجاد زیربنا و ساختار مناسب برای یک سایت بسیار مهم و پیچیده است ولی نمایش صحیح خروجی کار و ایجاد یک فضای کارپسند نیز اهمیت بسیار بالایی خواهد داشت. Bootstrap قصد دارد که خلا میان طراحی و کدنویسی را از میان برداشته و کدنویسان را ترغیب به استفاده از طراحی‌های پیش فرض و استاندارد نماید. به همین منظور دستورات CSS و توابع jQuery مورد نیاز را برای شما فراهم کرده است تا شما بتوانید با استفاده از دستورات پیشفرض و رعایت اصول متناسب با طراحی Bootstrap زمان راه‌اندازی یک پروژه را تا حد زیادی کاهش داده و خروجی آن را متناسب با استانداردهای روز دنیا پیش ببرید.

۱.۴.۳: مزایای Bootstrap

بزرگترین مزیت Bootstrap این است که دارای مجموعه رایگانی از ابزارها برای ایجاد صفحات وب انعطاف پذیر و responsive می باشد.

به علاوه، با استفاده از اطلاعات رابط برنامه نویسی (API) در Bootstrap می توانید اجزاء واسط پیشرفته مانند scroll spy و تکمیل کننده خودکار کلمات (type ahead) را بدون نیاز به نوشتن حتی یک خط کد جاوا اسکریپت ایجاد نمایید. اما مزایای دیگری دارد که در زیر به آن اشاره می نماییم:

Mobile First: در طراحی Bootstrap از رویکرد موبایل در الویت استفاده می شود. یعنی در طراحی سایت قبل از اینکه به رایانه فکر کنیم ابتدا سایت را برای موبایل طراحی کنیم. در واقع دو نوع رویکرد برای طراحی سایت داریم: رویکرد Desktop محور، رویکرد Mobile First.

صرفه جویی در زمان: می توان با استفاده از قالب ها و کلاس های از پیش طراحی شده Bootstrap زمان و انرژی کمتری برای طراحی صرف کرده و بیش تر بر روی جنبه های دیگر پروژه متمرکز شد.

ویژگی های responsive : با استفاده از Bootstrap می توان به راحتی طراحی های responsive ایجاد کرد. ویژگی های responsive باعث می شوند که صفحات وب در دستگاه های مختلف و وضوح تصویر متفاوت به درستی و به صورت مناسب و بدون نیاز به هیچ گونه تغییر در کدگذاری، نمایش داده شوند.

طراحی منسجم و یکپارچه : تمامی مؤلفه های Bootstrap از قالب های طراحی مشترک از طریق یک کتابخانه مرکزی استفاده می کنند. بنابراین طرح و پیکربندی صفحات وب در طول توسعه و طراحی، ثابت و یکپارچه باقی می ماند.

سهولت استفاده: استفاده از Bootstrap بسیار ساده است به طوری که هر شخص با دانش و اطلاعات اولیه و پایه ای از HTML و CSS می تواند از آن استفاده کند.

سازگار با مرورگرها : Bootstrap با کلیه مرورگرهای پیشرفته و جدید مانند Mozilla Firefox، Google Chrome، Safari، Internet Explorer و Opera سازگار است.

۱.۵: معرفی زبان JavaScript

زبان برنامه‌نویسی اسکریپت مبتنی بر اشیاء است که توسط Netscape تولید شده‌است. این زبان، یک زبان شی‌گراست.

این زبان می‌تواند هم به‌صورت ساخت‌یافته و هم به‌صورت شی‌گرا مورد استفاده قرار گیرد. در این زبان اشیاء با اضافه شدن متدها و خصوصیات پویا به اشیاء خالی ساخته می‌شوند.

کاربرد گسترده این زبان در سایت‌ها و صفحات اینترنتی می‌باشد و به کمک این زبان می‌توان به اشیاء داخل صفحات HTML دسترسی پیدا کرد و آن‌ها را تغییر داد. به همین علت برای پویانمایی در سمت کاربر، از این زبان استفاده می‌شود.

JavaScript به یکی از زبان‌های برنامه‌نویسی پر طرفدار در وب تبدیل شده‌است. هر چند ابتدا بسیاری از برنامه‌نویسان حرفه‌ای این زبان را کم ارزش تلقی می‌کردند چون مخاطبین آن نویسندگان صفحات وب و آماتورهای این‌چنینی بودند. ظهور Ajax بار دیگر JavaScript را در معرض توجه قرار داد و برنامه‌نویسان حرفه‌ای بیشتری را به خود جذب نمود. نتیجه‌ی این تغییر، ازدیاد Framework و کتابخانه‌های جامعی در این زمینه (مانند jQuery و...)، بهبود شیوه‌های رایج برنامه‌نویسی در JavaScript و افزایش کاربرد JavaScript خارج از وب است.

۱.۵.۱: معرفی jQuery

jQuery یک کتابخانه سبک وزن چند مرورگری است که برای ساده سازی نوشتن اسکریپت‌های سمت کاربر (Client) در Html طراحی شده و امروزه محبوب‌ترین کتابخانه‌ی JavaScript در حال استفاده است.

jQuery نرم افزاری متن باز و رایگان است. زبان jQuery به گونه ای طراحی شده است که عمل هدایت به پرونده را آسان تر کرده باشد. می توان با آن حرکات انیمیشن ایجاد کرده و از رویدادهای صفحه استفاده کرد و مهم تر از همه می توان نرم افزارهایی مبتنی بر Ajax را ایجاد نموده و توسعه داد.

jQuery همچنین این اختیار را به برنامه نویسان می دهد که افزونه هایی برای کتابخانه JavaScript

ایجاد کنند. جدا از این ها، jQuery به توسعه دهندگان این اختیار را می دهد که تکه برنامه های سطح پایین مبادله ای (مانند ارتباط مرورگر با کاربر) و یا انیمیشن و حتی افکت های پیشرفته و سطح بالا و اشیاء فرضی را ایجاد کنند. به کارگیری همه ی این اجزای jQuery کمک می کند تا صفحات وب قدرتمند و پویا، ساده تر ایجاد شوند.

jQuery شامل ویژگی های زیر است:

- دسترسی به عناصر موجود در پرونده ها و تغییر در آن ها.
- کنترل آسان و قدرتمندتر رویدادها (Events).
- تغییر در CSS.
- ایجاد افکت و حرکات انیمیشنی.
- توسعه ی افزونه ها.
- تولید برنامه های کوچک سودمند.

۱.۵.۲: معرفی دیگر ابزارهای JavaScript استفاده شده

در این پروژه برای زیباسازی و همچنین سهولت دسترسی برای کاربر از چند ابزار JavaScript متن باز که توسط توسعه دهندگان دیگر توسعه داده شده بود، استفاده کردیم که در ذیل معرفی می شوند:

• Clockpicker

زمانی که از کاربر می خواهیم زمان رزرو وقت را وارد کند یا زمانی که صاحب شغل یا همکار زمان کاری خودش را مشخص کند، از این ابزار استفاده می کنیم تا زیبایی و دسترسی پذیری سایت بالا برود.

• PersianDatePicker

زمانی که صاحب شغل یا همکار می‌خواهند روز خاصی را در تقویم کاری خود قرار بدهند یا کاربر تاریخ رزرو خود را مشخص می‌کند هم از این ابزار استفاده می‌کنیم و تاریخ شمسی را از کاربر می‌گیریم ولی در Backend آن تاریخ را به تاریخ میلادی (مشخصاً نوع DateTime در .Net) تبدیل می‌کنیم تا به همان صورت در پایگاه داده آن را ذخیره کنیم.

• Select2

این ابزار را وقتی که صاحب شغل یا همکار می‌خواهد روزهای هفتگی یک ماه آینده خود را به صورت دسته‌ای وارد کند استفاده کرده‌ایم. بدین صورت که او روزهای هفته‌ی دلخواه خود را انتخاب می‌کند و ما توسط این ابزار آن را به شکل یک Multiple ListBox به کاربر نشان می‌دهیم و مقادیر آن را در Backend می‌خوانیم تا برای ذخیره در پایگاه داده آنها را آماده کنیم.

هم‌چنین در پروژه های وب MVC بصورت پیش فرض از برخی ابزارهای اسکریپتی مانند respond.js، jQuery.Validate، modernizr.js و... استفاده می‌شود که بیشترین کاربردهای آنها اعتبارسنجی فرم های ورودی توسط کاربر (نشان دادن هشدار و...) و نمایش صحیح HTML5 و CSS3 در مرورگرهای مختلف است.

۱.۶: Json چیست؟

JSON مخفف JavaScript Object Notation می‌باشد. اجازه دهید با یک مثال بیشتر به توضیح JSON بپردازیم. در مثال زیر شیء user دارای سه رکورد اطلاعات است که شامل نام و نام خانوادگی افراد می‌باشد.

```
{ "user": [
  { "firstName": "John" , "lastName": "Malkovich"},
  { "firstName": "Peter" , "lastName": "Dinklage"},
  { "firstName": "Anna" , "lastName": "Karina"} ] }
```

JSON جهت نگهداری و انتقال اطلاعات متنی به کار می‌رود، تقریباً شبیه XML است ولی نسبت به XML کم‌حجم‌تر و در نتیجه سریع‌تر است و با آن راحت‌تر می‌توان کار کرد.

عموماً از JSON در جاوا اسکریپت استفاده می‌شود ولی از آن جایی که ساختار خوب و نسبتاً کم‌حجمی دارد در بسیاری از زبان‌های برنامه‌نویسی دیگر و مخصوصاً در وب سرویس‌ها جهت تبادل اطلاعات به‌صورت متنی استفاده می‌گردد.

یکی از بزرگترین مزیت‌های JSON این است که می‌توانیم به‌صورت مجموعه‌ای از اشیاء در کدهای JavaScript به اشیاء داخلی آن دسترسی پیدا کنیم.

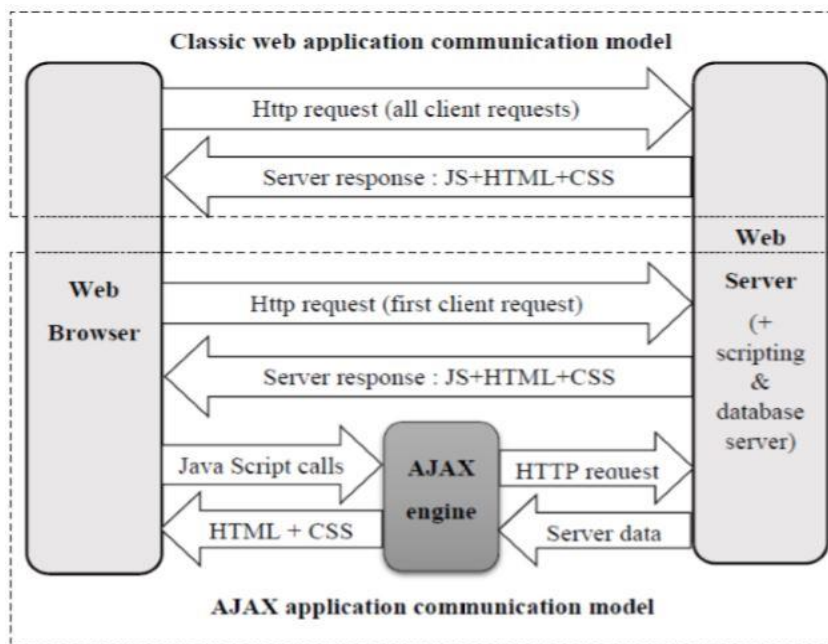
۱.۷: تکنولوژی AJAX

واژه AJAX به معنی ترکیب نامتقارن JavaScript و XML است. ماهیت صفحات وب و پروتکل Http به گونه‌ای است که به‌طور معمول وقتی در حال وب‌گردی هستیم، به ازای هر کنش و واکنش میان ما و سایتی که در حال کار کردن با آن هستیم، کل صفحه وب از نو به‌روزرسانی (refresh) می‌شود. Ajax فناوری جدیدی است که تغییر محسوسی را در این سناریو به‌وجود می‌آورد، به این ترتیب که به‌جای بارگذاری مجدد کل صفحه، فقط قسمتی تغییر می‌کند که قرار است اطلاعات جدید را به نمایش درآورد و کلیه عملیات ارسال اطلاعات و دریافت نتایج در پشت صحنه انجام می‌شود. در نتیجه هیچ‌گاه صفحه سفید و خالی وب در فواصل کنش و واکنش‌های هنگام کار با مرورگر دیده نمی‌شود و احساسی مشابه تجربه کار با یک نرم‌افزار Desktop به کاربر دست می‌دهد. جادوی AJAX چیزی نیست جز یک فکر بکر و آن هم ترکیب JavaScript و XML در قالب یک موجود نرم‌افزاری جدید. در این پروژه برای مثال در قسمت رزرو وقت از این تکنولوژی استفاده شده است.

۱.۷.۱: چگونگی کارکرد AJAX

در برنامه‌های کلاسیک تحت وب ارتباط بین مرورگر و سرور به‌صورت مستقیم و با استفاده از درخواست Http برقرار می‌شود. در مدل کلاسیک با هر درخواست صفحه از طرف کاربر، سرور کلیه کدهای HTML و CSS را یک مرتبه به سمت مرورگر ارسال می‌کند. پس از اینکه کاربر فرم را پر و آن را به سمت مرورگر برمی‌گرداند و این روند ادامه پیدا می‌کند. اما در مدلی که از AJAX استفاده می‌شود و آن هم در پاسخ به اولین درخواست است، تعدادی فایل JavaScript هم که همان موتور AJAX است به همراه کدهای HTML و CSS که ساختار صفحه را تشکیل می‌دهند، بارگذاری می‌شوند. کلیه درخواست‌های بعدی به‌صورت JavaScript بوده و به

موتور **AJAX** ارسال می‌شود. این موتور درخواست اطلاعات را به صورت آسنکرون (نامتقارن) به سرور می‌فرستد. بنابراین تنها بخشی از صفحه که مورد نیاز کاربر است مبادله می‌شود. در نهایت موتور **AJAX** اطلاعات را بدون بارگذاری کل صفحه نمایش می‌دهد. انتقال تنها اطلاعات ضروری به جای کل صفحه بین سرور و مرورگر باعث می‌گردد که میزان پاسخ‌گویی رابط کاربری بالا رود. در شکل زیر به خوبی ارتباط بین مرورگر و سرور و همچنین تفاوت‌های موجود در دو مدل ذکر شده نمایش داده شده است.



۱.۷.۲: مشکلات AJAX

گرچه این روش مزایای بسیاری دارد و شکاف بین برنامه‌های تحت ویندوز و برنامه‌های تحت وب بسیار کمتر کرده است، اما مشکلاتی نیز دارد که هنگام کار به این روش، باید به آنها توجه کرد. یکی از مشکلاتی که در مورد **AJAX** عنوان می‌شود، این است که کاربرد کلید **Back** در مرورگرها را مختل کرده و این برخلاف عادت کاربران در محیط وب است. البته برای حل مسئله راه‌حلهایی پیشنهاد شده که مجال بحث در مورد آنها نیست.

مشکل دیگری که در مورد سیستم‌های بر اساس **AJAX** وجود دارد استفاده **AJAX** از **JavaScript** است و این مسئله که کاربر می‌تواند اجازه‌ی اجرای **JavaScript** در مرورگر خود را ندهد. پس باید پیش از بارگذاری موتور **AJAX** از اجازه داشتن اجرای **JavaScript** روی مرورگر کاربر اطمینان پیدا کرد. مسئله‌ی دیگری که باید به آن توجه داشت، این است که کدهای **JavaScript** برنامه را هرکسی به راحتی می‌تواند ببیند. در نتیجه

بهتر است قسمت‌هایی از برنامه را که مربوط به امنیت یا منطق پردازشی برنامه می‌آشود حتما سمت سرور نگه داشت.

به نظر می‌رسد ساخت سیستمی که ترکیب متوازی از پردازش‌های سمت سرور و یک موتور AJAX قوی در طرف کاربر (Client) باشد، می‌تواند راه حل بسیار مناسبی باشد.

۱.۸: Font Awesome چیست؟

فونتی است که دارای آیکون‌های بسیار زیادی می‌باشد که با فراخوانی آن در قالب می‌توانید از این آیکون‌ها بهره ببرید. از ویژگی‌های مهم این آیکون‌ها می‌توان به سرعت بارگذاری زیاد سایت و در نتیجه افزایش سرعت و عدم نیاز به sprite اشاره کرد. استفاده از این فونت دو روش دارد:

روش اول این است که کد زیر را در قالب مورد نظر فراخوانی کنید و سپس کد آیکون‌ها را در قسمت‌های مورد نظر قرار بدهید.

نمونه کد:

```
<link rel="stylesheet" href="~/Content/css/font-awesome.min.css">
<i class="fa fa-plus-circle"></i>
```

روش دوم استفاده از CSS است که برای این کار نیاز دارید فونت‌ها را به قالب CSS خود اضافه کنید. لازم به ذکر است که ما در این پروژه از روش اول و از نسخه‌ی Font Awesome 4.6.1 استفاده کرده‌ایم. این تکنولوژی بُرداری است و از کیفیت آن کاسته نمی‌شود.

۱.۹: معرفی IDE

محیط یکپارچه توسعه نرم‌افزار (Integrated Development Environment) عبارت‌است از محیطی عمدتاً گرافیکی که تمام یا شماری از ابزارهای لازم برای توسعه نرم‌افزار (بخش‌هایی یا تمام زنجیره ابزار توسعه) را خود دارد. در IDE دسترسی به ابزارها و اعمال آن‌ها در پروژه جاری تسهیل شده‌است.

امکاناتی که به طور معمول در IDE ها وجود دارد:

- I. ویرایش و نوشتن کد به صورت پیشرفته با استفاده از امکانات پیشنهاد دهنده اتوماتیک که با نوشتن حرف اول یک دستور نام کامل دستورهایی که وجود دارد لیست می شود.
- II. نمایش کدها به صورت رنگی و امکان تمییز متغیرها و کامنت ها از Syntax زبان.
- III. کمک به رفع عیب های نرم افزار و حل مشکلات آن (Debugging).

در این پروژه هم ما از Visual Studio 2015 استفاده نمودیم تا بتوانیم از امکاناتی هم چون MVC 5، Entity Framework و همچنین Razor Syntax در پروژه بهره مند شویم.

۱.۹.۱: معرفی Visual Studio

Visual Studio نام مجموعه برنامه نویسی شرکت Microsoft است که دارای چند زبان برنامه نویسی است. نرم افزار Visual Studio ، نرم افزاری توسعه یافته برای برنامه نویسان رایانه است که توسط شرکت Microsoft تولید شده است. تمرکز اصلی این نرم افزار از اولین نسخه های آن تاکنون بر روی خصوصیت IDE بودن آن است که به برنامه نویس اجازه می دهد تا برنامه های کاربردی مستقل، وبسایت، برنامه های کاربردی وب و یا سرویس های وب را که بر روی تعدادی از Platform های پشتیبانی شده توسط .NET. Microsoft Framework (البته برای تمام نسخه های بعد از Visual Studio 6) همچنین Platform هایی مانند Microsoft Windows servers and workstations, Pocket PC Smartphones و World Wide Web browsers اجرا می شوند را به راحتی ایجاد نماید. از جمله قابلیت های اضافه شده که در آخرین ویرایش ارائه شده است می توان به Silverlight اشاره کرد که یک Web Application framework بسیار پیشرفته جهت توسعه نرم افزارهای کاربردی تحت وب می باشد.

Visual Studio یک مجموعه از برنامه هایی است که ارتباط بسیار نزدیک باهم دارند که Microsoft آن را به توسعه دهندگان و برنامه نویسان برنامه های کاربردی اهدا نمود تا آن ها را وادار نماید در محیطی توسعه یافته بر روی Platform های ویندوز و .Net. به ساخت برنامه های خود بپردازند. Visual Studio می تواند برای نوشتن برنامه های کنسولی، ویندوزی، سرویس های ویندوز، برنامه های کاربردی موبایل، برنامه های کاربردی ASP.NET

و سرویس‌های وب ASP.NET بنا به انتخاب شما همراه با زبان‌هایی مانند J, VB.NET, C#, C++ استفاده شود.

با Visual Studio واقعاً چه کارهایی می‌توان انجام داد؟ در زیر تعدادی از کاربردهایی را که برای تولید آن‌ها می‌توان از Visual Studio استفاده نمود معرفی گردیده‌اند:

Console Applications: این کاربرد برای اجرای خطوط دستور البته بدون محیط گرافیکی استفاده می‌شود که از این کاربرد برای برخی از ابزارهای کوچک یا برای اجرا شدن کدها توسط دیگر کاربردها استفاده می‌شود. این دستورها در خط فرمان اجرا می‌شود.

Windows Forms Applications: برای برنامه‌های کاربردی ویندوزی که با استفاده از .NET Framework نوشته می‌شوند.

Windows Services: سرویس‌ها برنامه‌های کاربردی هستند که در پس زمینه ویندوز اجرا می‌شوند.

ASP.NET Applications: یک تکنولوژی قدرتمند که برای طراحی و ساخت صفحات وب پویا استفاده می‌شود. (در این پروژه از این نوع Application استفاده نموده ایم).

ASP.NET Web Services: مدل سرویس‌های وب را بطور کامل فراهم نموده تا شما به راحتی و با سرعت سرویس‌های وب را تولید نمایید.

Windows Mobile Applications: که می‌تواند بر روی ابزارهایی که شامل Framework هستند مانند Pocket PC ها و همچنین Cell Phone هایی که Microsoft Smartphone Platform بر روی آن‌ها اجرا می‌شود، اجرا گردد.

MFC/ATL/Win32 applications: شما هم‌چنان می‌توانید برنامه‌های سنتی MFC، ATL یا برنامه‌های Win32 را با استفاده از C++ ایجاد نمایید. این برنامه‌ها برای اجرا به .NET Framework نیاز ندارند اما نمی‌توانند از مزایای .NET Framework. نیز بهره‌ای ببرند.

Visual Studio Add-ins: شما می‌توانید از خود Visual Studio برای ساخت توابعی جدید و قابل اضافه شدن به خود Visual Studio استفاده نمایید.

کاربردهای دیگر: Visual Studio هم‌چنین شامل پروژه‌هایی برای توسعه برنامه‌های کاربردی شما، کار با Database ها، ساخت گزارش‌ها و... می‌باشد.

لازم به ذکر است در این پروژه از ویرایش Visual Studio Enterprise استفاده شد که در ادامه به برخی ویژگی‌های آن می‌پردازیم.

۱.۹.۲: معرفی Visual Studio Enterprise

نسخه Enterprise تمامی قابلیت‌های نسخه Professional را دارد و البته امکانات بیشتری از جمله قابلیت توسعه‌دهی پایگاه‌داده، همکاری تیمی، قابلیت معماری سیستماتیک، سیستم متریک (شامل میزان بهره‌وری CPU، RAM و Network و...)، ابزارهای تست و Report را در داخل خود جای داده‌است.

لازم به ذکر است نسخه Professional تمامی قابلیت‌های ویرایش Standard را دارد و از Remote Debugging، SQL Server Developer Edition، برنامه‌نویسی موبایل، Server Explorer و... برخوردار است. از نسخه ۲۰۰۸ به بعد امکان توسعه‌ی برنامه‌های Office نیز در IDE گنجانده شده‌است.

۱.۱۰: معرفی پایگاه‌داده مورد استفاده

از آنجایی که شرکت Microsoft سعی در تولید یک Platform برنامه‌نویسی نموده است، ما بر آن شدیم تا از این Platform به بهترین شکل ممکن استفاده کنیم و با توجه به اینکه پروژه توسط ASP.NET طراحی و اجرا می‌شد، از پایگاه‌داده Microsoft SQL Server استفاده کردیم.

۱.۱۰.۱: معرفی Microsoft SQL Server

Microsoft SQL Server یک نرم‌افزار سیستم مدیریت بانک‌های اطلاعاتی است که توسط شرکت Microsoft توسعه داده می‌شود. برخی از ویژگی‌های این سیستم مدیریت پایگاه‌داده‌ها به این شرح است:

- بانک اطلاعاتی رابطه‌ای
- امکان استفاده از Triggerها، Viewها و Stored Procedureها

- پشتیبانی از XML
- بسیار قدرتمند و بدون محدودیت حجم و تعداد رکورد
- پشتیبانی از Full Text Search برای سرعت در بازیابی اطلاعات و استفاده از زبان طبیعی در جستجوها (Query ها)

لازم به ذکر است که در این پروژه از MSSQL Server 2016 بهره بردیم.

۱.۱۱: ساختار پوشه پروژه

هرکدام از اجزای MVC Framework در پوشه‌ای جداگانه قرار گرفته‌است که این امر سادگی جداسازی بخش‌های مختلف و همچنین زیبایی بصری و خوانایی بیشتر را امکان پذیر می‌سازد. پوشه پروژه در ASP.NET MVC شامل پوشه‌های زیر است:

App_Data: در این پوشه که یک پوشه بسیار محافظت شده از لحاظ امنیتی می‌باشد فایل‌های بسیار مهمی نظیر فایل mdf و ldf. بانک اطلاعاتی قرار می‌گیرند.

App_Start: در این پوشه فایل‌هایی نظیر RouteConfig که برای تنظیم آدرس‌دهی در پروژه استفاده می‌شود و یا فایل BundleConfig که برای آدرس‌دهی Component های مختلف Javascript مانند JQuery و CSS مانند Bootstrap استفاده می‌شود، نگهداری می‌شوند.

bin: در این پوشه تمامی فایل‌های اجرایی Visual Studio و ASP.NET قرار دارند.

Content: این پوشه نیز حاوی فایل‌های CSS و همچنین تصاویر مورد نیاز در پروژه است.

Controllers: در این پوشه تمامی Controller های مبتنی بر مدل طراحی MVC نگهداری می‌شوند. لازم به ذکر است پسوند فایل‌های Controller در ASP.NET، cs می‌باشد.

Fonts: در این پوشه فایل‌های فونت‌های مختلف استفاده شده در پروژه از جمله Font Awesome و IRAN Sans قرار گرفته‌است.

Models: در این پوشه تمامی Model های مبتنی بر مدل طراحی MVC نگهداری می شوند. لازم به ذکر است پسوند فایل های Model در ASP.NET، cs. می باشد. با توجه به استفاده ما از Entity Framework و تبدیل اشیاء بانک اطلاعاتی به اشیاء قابل ترجمه در MVC توسط قابلیت های Entity Framework هر جدول پایگاه داده ما اینجا به یک مدل (یک فایل cs.) تبدیل شده است به علاوه مدل هایی که بعدها برای ساده سازی به پروژه افزوده ایم. هم چنین در این پوشه یک فایل edmx. هم وجود دارد که نمودار رابطه ای اشیاء پایگاه داده است و می توانیم در آن به پایگاه داده برای تغییرات دسترسی پیدا کنیم.

obj: در این پوشه هم اطلاعاتی نظیر log های اجرای پروژه و فایل های Cache برای سرعت بخشیدن به اجرای پروژه نگهداری می شوند.

Scripts: در این پوشه فایل های JavaScript مورد استفاده در پروژه که با پسوند js. ذخیره شده اند قرار می گیرند.

Security: در این پوشه برخی Controller های از پیش آماده MVC برای مدیریت امنیت وبسایت قرار گرفته است. برای مثال SessionPersister وظیفه ذخیره کردن Session ها در هر زمانیکه برای تغییر یا مقداردهی فراخوانی می شوند را بر عهده دارد.

Utilities: در این پوشه که بعداً به پروژه اضافه کردیم توابعی برای تبدیل تاریخ شمسی ای که از ورودی دریافت می کنیم به نوع استاندارد DateTime و بالعکس پیاده سازی شده است که برای سهولت هرچه بیشتر دسترسی و ذخیره نوع DateTime استفاده شده اند.

Views: در این پوشه تمامی View های مبتنی بر مدل طراحی MVC نگهداری می شوند. لازم به ذکر است پسوند فایل های View در ASP.NET، cshtml. می باشد و به ازای هر Controller یک پوشه درست می شود که View های مربوط به آن Controller را در خود جای داده است. هم چنین در پوشه Shared و در فایل layout.cshtml نمای کلی View های وبسایت را به کمک قابلیت های بسیار Razor، ساخته شده است. فایل Header.cshtml هم برای مدیریت View در قسمت بالایی صفحات ایجاد شده است. اصولاً هر زمان که می خواهیم از قسمتی از View در چند جای پروژه استفاده کنیم در پوشه Shared یک Partial View می سازیم و در آینده به راحتی از آن در بقیه View های خود استفاده می کنیم. برای مثال برای نشان دادن Input مناسب برای گرفتن تاریخ شمسی از کاربر در پوشه EditorTemplates یک Partial View به نام DateTime.cshtml ساختیم تا هر جا که برای نوع DateTime برای کاربر یک Input نشان دادیم

این Partial View را ببیند. هم چنین برای نمایش تاریخ شمسی به کاربر هم همین رویه را در پوشه DisplayTemplates اجرا کردیم تا هر زمان به کاربر خواستیم نوع DateTime را نشان بدهیم تاریخ میلادی داخل سرور را به تاریخ شمسی تبدیل کرده و به صورت شمسی به کاربر نمایش دهیم.

فایل **Web.Config**: تنظیمات پیکربندی مربوط به پایگاه داده و فایل های پیکربندی هسته در این فایل نگهداری می شوند.

فایل **Appointer.sln**: این فایل، همان فایل اجرایی Visual Studio است که با اجرای آن به محیط Visual Studio رفته و می توانیم تغییرات خود را در پروژه ایجاد نماییم.

پوشه **packages**: در این پوشه تمامی تکنولوژی هایی که خود MVC در پروژه به صورت خودکار استفاده می کند وجود دارند. فایل های پکیج های nuget و فایل های dll و ... از جمله فایل هایی است که به صورت خودکار در این پوشه توسط MVC ذخیره شده اند.

۱.۱۰: خلاصه

در این فصل به معرفی ASP.net، تکنولوژی Bootstrap و زبان JavaScript پرداختیم. در ادامه به معرفی معماری MVC و مفاهیم اصلی آن پرداخته شد و همچنین توضیح مختصری در رابطه با Font Awesome و IDE و پایگاه داده مورد استفاده در پروژه شرح داده شد. همچنین توضیحات کوتاهی در مورد پیکربندی پوشه‌ی حاوی فایل‌های مختلف پروژه دادیم.

فصل دوم

لزوم انجام پروژه

۲.۱: مقدمه

زمان کمیاب ترین و ارزشمندترین دارایی انسان است که غیرقابل جایگزینی می‌باشد. از بدو تولد، شمارش معکوس مصرف جبران ناپذیر دارایی زمان (عمر انسان) آغاز می‌شود. عقب ماندن از جهان را فقط با توقف زمان می‌توان جبران کرد، که البته امری غیر ممکن است.

در مشاغل امروزی هم زمان صاحب شغل اهمیت دارد و هم زمان مشتری‌ها. این حق مشتری است که بتواند در وقتی که می‌خواهد به او خدمت‌رسانی شود، منوط به این که صاحب شغل در حال خدمت‌رسانی به مشتری دیگری نباشد. صاحبان مشاغل می‌دانند که یکی از عوامل مهم در جذب و نگه داشتن مشتری‌ها به خصوص در اولین برخورد، ارزش نهادن به وقت آن‌هاست. ضمناً در تئوری‌های فروش مدرن همیشه گفته شده که "حق، همیشه با مشتری است"، چرا که او کسی است که کالا یا خدمتی را می‌خواهد و قرار است پول بپردازد. یکی از مهم‌ترین حقوق مشتری خدمت‌رسانی به او در مدت از پیش تعیین شده می‌باشد. اگر در مورد زمان از قبل تعیین شده بدقولی صورت گیرد مطمئناً مشتری حداقل قسمتی از اعتماد خود را از دست خواهد داد.

۲.۲: راهکارهای سیستم منشی دیجیتال برای زمان‌بندی

برای شرح لزوم انجام این پروژه لازم است مثالی را مطرح کنیم. فرض کنید شما یک آرایشگر هستید. در زمان‌های نه چندان دور (و چه بسا در حال حاضر در بسیاری از آرایشگاه‌ها) نوبت‌دهی به مشتری‌ها از طریق صف (همان الگوریتم FCFS^{۱۸}) صورت می‌پذیرفت ولی در حال حاضر بسیاری از آرایشگاه‌ها به مشتریان خود وقت قبلی می‌دهند تا وقت مشتریان در صف اصلاح مو تلف نشود! البته به کسانی که بدون نوبت هم می‌خواهند به اصلاح مویشان پرداخته شود نیز در صورت داشتن وقت قبلی مشتری دیگر، متذکر می‌شوند که باید بیشتر منتظر بمانند که همین فرایند باعث می‌شود آن مشتری دفعه‌ی بعدی که به آرایشگاه می‌آید از قبل وقت بگیرد تا وقتش تلف نشود. البته روال معمول آرایشگاه‌ها بدین گونه است که کسانی که وقت قبلی می‌خواهند بگیرند قبلاً با آرایشگر تماس می‌گیرند و سپس آرایشگر در یک دفتر زمان درخواستی مشتری را چک می‌کند تا اگر مشتری دیگری در آن زمان وقت نگرفته باشد، به مشتری پشت تلفن می‌گوید که وقتش را در دفتر ثبت کرده و در آن زمان منتظر او خواهد بود. احتمالاً شما هم به غیر مدرن بودن این روش خرده خواهید گرفت. وقتی می‌توان با استفاده از یک سیستم تمام این عملیات را پیاده‌سازی کرد، چرا باید یک آرایشگر (یا مشاغل خدماتی دیگر) وقت خود را صرف نوشتن قرار ملاقات در دفتر و چک کردن بقیه قرارها از طریق دفتر باشد؟

¹⁸ First Come First Served

در سیستم منشی دیجیتال تمامی این نیازها در نظر گرفته شده است. مشاغل بسیاری می‌توانند از این سیستم استفاده نمایند. اکثر مشاغل خدماتی که با ماهیت زمان سر و کار دارند و نوبت‌دهی در آن‌ها اهمیت دارد قابلیت استفاده از سیستم را خواهند داشت.

مشاغلی مانند انواع مشاغل پزشکی، مشاوره‌ای، خدماتی (مانند پیرایش، نظافت و ...) و حتی برخی از مشاغل تجاری نیز می‌توانند از این سیستم استفاده نمایند.

صاحبان مشاغل پس از ثبت نام در سیستم باید شغل خود را تعریف کنند. پس از آن برای این که مشتریان بتوانند وقت رزرو کنند باید ابتدا زمان‌های کاری خود را اضافه کنند و سپس خدمات خود را بیفزایند. دوباره مثال آرایشگر را در نظر بگیرید. مثلاً او به‌جز جمعه در تمام روزهای هفته و از ساعات ۹ تا ۱۳ صبح و ۱۶ تا ۲۱ شب کار می‌کند و سه خدمت اصلاح معمولی، اصلاح با شستشوی سر و مرتب کردن مو را به ترتیب با زمان‌های ۲۵، ۴۰ و ۱۰ دقیقه انجام می‌دهد.

پس از تعریف خدمات و تعیین زمان‌های کاری او می‌تواند به مشتریان خود آدرس سایت و عنوان شغل خود در سایت را بگوید تا آن‌ها از این به بعد از طریق سایت وقت بگیرند. حتی اگر یک مشتری دوباره تلفنی به آرایشگر مراجعه کند، آرایشگر به اسم خودش یک وقت از خودش می‌گیرد و در قسمت توضیحات مشخصات مشتری را می‌نویسد.

زمانی که مشتری‌ها وقت گرفته باشند، صاحب شغل می‌تواند لیست قرارهای کاری خودش را به راحتی مشاهده کند.

ممکن است این سوال به ذهن خواننده خطور کند که اگر یک صاحب شغل همکار دیگری هم داشته باشد چه می‌شود؟ در سیستم منشی دیجیتال به این سوال نیز پاسخ داده شده است. کافی است صاحب شغل به قسمت ویرایش شغل رفته و یک "کلید ثبت نام برای همکاران" تعریف نماید. حال او این کلید را به همکار(های) خود می‌گوید و آن‌ها در زمان ثبت نام در سایت پس از انتخاب نقش کاربری همکار باید این کلید را بزنند تا در این شغل به عنوان یک همکار ثبت شوند. یک همکار هم پس از ثبت نام مانند یک صاحب شغل باید زمان‌های کاری و خدمات خود را در سیستم تعریف کند تا کاربران بتوانند از او وقت بگیرند. لازم به ذکر است که همکاران تقریباً تمامی کارهای صاحب شغل به‌جز ویرایش شغل و مدیریت همکاران را می‌توانند انجام دهند.

حال ببینیم کاربر معمولی در این سیستم باید چه کار کند. پس از انجام ثبت نام کاربر لیستی از شغل‌ها را خواهد دید و می‌تواند از بین آن‌ها شغلی را انتخاب نماید. سپس او لیست همکاران و دکمه‌ای برای جزئیات بیشتر در مورد شغل مورد نظر را خواهد دید. پس از انتخاب یک همکار در صورتی که همکار مورد نظر خدمات و زمان‌های کاری خود را اضافه کرده باشد، کاربر می‌تواند زمان و خدمت مورد نظر خود را انتخاب کند و در ضمن ببیند که آیا کس دیگری در این زمان وقت گرفته است یا خیر. در آخر پس از فشردن دکمه رزرو وقت در صورتی که با بقیه وقت‌ها مغایرتی وجود نداشته باشد، کاربر با موفقیت از همکار مورد نظر وقت می‌گیرد. حال کاربر با دیدن لیست رزرواسیون وقت‌های خود می‌تواند ببیند از چه کسانی و در چه زمان‌هایی وقت گرفته است.

حال که اهمیت پیاده‌سازی سیستم منشی دیجیتال شرح داده شد و راهکار سیستم برای صرفه‌جویی در وقت صاحبان مشاغل و مشتریان به‌صورت کلی توضیح داده شد، نوبت آن است که روند کار^{۱۹} سیستم را در صفحات مختلف سایت بررسی کنیم.

۲.۳: روند کار سیستم منشی دیجیتال

حال با نمایش تصاویری از سیستم، روند کار آن را نمایش می‌دهیم.

۲.۳.۱: عملیات تعریف شده برای تمامی کاربران (قبل از ورود)

۱. صفحه اصلی سایت (Home/Index)

هر کاربری قبل از ورود به سیستم این صفحه را به عنوان اولین صفحه‌ی سایت مشاهده خواهد کرد. با استفاده از میانبرهای این صفحه، کاربر می‌تواند با انتخاب نوع شغل مورد نظر خود به قسمت جستجوی مشاغل برود. همچنین در قسمت سمت چپ و بالای صفحه (Header) دکمه‌های ثبت نام و ورود و همچنین جستجوی مشاغل قرار داده شده‌اند. در قسمت راست تعدادی از دکمه‌ها قرار دارند که در صورت ورود کاربر به سیستم و با توجه به نقش کاربر تغییر خواهند کرد.

۲. ثبت نام (Account/SignUp)

کاربران به وسیله این صفحه می‌توانند در سایت ثبت نام کنند. در این صفحه کاربر اطلاعات مورد نیاز را به همراه نقش کاربری خود وارد می‌کند و سپس به صفحه‌ای متناسب با نقش کاربری‌اش فرستاده می‌شود تا فرایند ثبت نامش سریع‌تر انجام شود. کاربران عادی به صفحه‌ی جستجوی مشاغل، صاحبان مشاغل به صفحه ایجاد شغل، و همکاران به صفحه ثبت کد شغلی فرستاده می‌شوند.

۳. ورود به سیستم (Account/SignIn)

کاربران به وسیله این صفحه می‌توانند به صفحه داشبورد خود مراجعه کنند. در این صفحه کاربر در فیلد بالا ایمیل یا نام و نام خانوادگی خود را وارد کرده و در فیلد دوم رمز عبور خود را وارد می‌کند تا وارد سیستم

شود. در صورتی که کاربر مشخصاتش را درست وارد کرده باشد، بعد از ورود به سیستم به صفحه‌ای متناسب فرستاده خواهد شد. کاربر معمولی در صورتی که قبلاً رزرو وقت کرده باشد و زمان رزرو بعد از زمان حال باشد، به صفحه رزرواسیون وقت خود فرستاده می‌شود. در غیر این صورت او را به صفحه جستجوی مشاغل خواهیم فرستاد.

صاحبان مشاغل و همکاران نیز در صورتی که زمان کاری‌ای اضافه نکرده باشند، به صفحه افزودن دسته‌ای زمان‌های کاری (بر اساس انتخاب روزهای هفته) فرستاده می‌شوند. اگر قبلاً سرویسی اضافه نکرده باشند، به صفحه‌ای افزودن سرویس جدید منتقل خواهند شد. اگر هر دو کار بالا را قبلاً انجام داده باشند، به صفحه‌ای لیست رزروهای کاربران از آن‌ها منتقل خواهند شد که اگر هنوز کاربری از آن‌ها وقت نگرفته باشد، پیامی به او خواهیم داد که آدرس سایت را به مشتریان بدهند.

بهتر است برای شرح گام به گام روند کار سیستم، صفحه‌های جستجوی مشاغل و مشاهده همکاران یک شغل را در قسمت عملیات تعریف شده برای کاربر بیاوریم، هرچند این دو مورد کاربرد جزو مواردی اند که کاربران قبل از ورود به سیستم نیز می‌توانند به آن‌ها دسترسی داشته باشند.

۲.۳.۲: عملیات تعریف شده برای صاحب شغل

۱. افزودن شغل (Jobs/Create)

پس از ثبت اطلاعات کاربری، صاحب شغل به این صفحه فرستاده می‌شود. او اطلاعات مختلف شغل را وارد می‌کند و پس از زدن دکمه‌ی ثبت، ابتدا به صفحه‌ی افزودن دسته‌ای زمان‌های کاری (بر اساس انتخاب روزهای هفته) فرستاده می‌شود. در گام بعدی هم او را به صفحه افزودن سرویس می‌فرستیم، تا در آینده مشکلی از نظر کار کردن با سایت نداشته باشد. به عبارت دیگر، به جای آموزش نحوه استفاده از سایت، خودمان در ابتدا با فرستادن او به صفحات مرتبط، او را آموزش می‌دهیم.

۲. ویرایش شغل (Jobs/Edit)

صاحبان مشاغل می‌توانند مشخصات شغل خودشان را در این صفحه تغییر دهند. نکته‌ی مهم این است که در صفحه افزودن شغل ما صاحب شغل را مجبور نکردیم برای ورود همکاران، کلید تعریف کند، ولی در قسمت ویرایش شغل این امکان را برای او گذاشته‌ایم تا بتواند برای افزودن همکاران کلید تعریف کند.

همان‌طور که بعداً توضیح خواهیم داد همکاران خواهند توانست با این کلید در شغل خود ثبت نام کنند. به عبارت دیگر داشتن همکاران در یک شغل انتخابی^{۲۰} است.

۱۱۱. لیست همکاران (Jobs/JobCorpsList)

صاحبان مشاغل می‌توانند پس از این که همکاران در شغل آن‌ها ثبت نام کردند، لیست آن‌ها را در این صفحه مشاهده کنند. همان‌طور که مشاهده می‌کنید در ستون سمت چپ هر سطر دکمه حذف همکار وجود دارد که با فشردن آن، صاحب شغل به صفحه‌ی بعدی می‌رود تا حذف همکار مورد نظر را تایید کند.

۱۱۲. حذف همکار (Jobs/DeleteJobCorp)

در این صفحه مشخصات همکار را به صاحب شغل نمایش می‌دهیم و او می‌تواند در صورت تمایل آن همکار را حذف کند.

بقیه‌ی عملیات صاحب شغل و همکار با هم مشترک است، لذا آن‌ها را در قسمت بعدی شرح می‌دهیم.

۲.۳.۳: عملیات تعریف شده برای همکاران

لازم به ذکر است که تمامی عملیات همکاران مشترک با صاحبان مشاغل است و تمامی عملیات ذیل برای صاحبان مشاغل نیز قابل دسترسی می‌باشند. پس در صورت استفاده از لفظ همکار، مقصود همکار یا صاحب شغل می‌باشد.

۱. عضویت در شغل (JCDashboard/EnrollJob)

تنها این صفحه مخصوص همکاران است و صاحبان مشاغل نیازی به مشاهده‌ی آن نخواهند داشت. در واقع هر وقت یک صاحب شغل به همکاران کلیدی برای ورود می‌دهد، همکاران پس از صفحه‌ی اولیه‌ی ثبت نام به این صفحه منتقل خواهند شد. در این صفحه کد مربوطه را می‌زنند، سپس با زدن دکمه بررسی کد، مشخصات شغل را خواهند دید و در صورت تایید می‌توانند به شغل مورد نظر به عنوان همکار افزوده شوند.

II. افزودن زمان کاری دسته‌ای (JCDashboard/AddWorkingTime)

در این صفحه همکار پس از انتخاب روزهای هفته‌ای که تمایل دارد در یک ماه آینده در آن‌ها کار کند، بازه‌های زمانی مورد نظر خود را انتخاب می‌کند. پس از فشردن دکمه ثبت زمان، همکار مورد نظر به صفحه لیست قرارهای کاری‌اش فرستاده خواهد شد.

III. افزودن زمان کاری تکی (JCDashboard/AddWorkingDate)

در این صفحه همکار می‌تواند برای یک تاریخ به خصوص، زمان کاری‌اش را ثبت کند. (مثلاً اگر او در یک ماه آینده بخواهد در یک هفته به خصوص یک روز بیشتر کار کند، باید به این صفحه مراجعه کند.) با انتخاب تاریخ مورد نظر و بازه‌ی زمانی مورد نظر، همکار دکمه ثبت زمان را می‌فشارد. پس از فشردن دکمه ثبت زمان، همکار مورد نظر به صفحه لیست قرارهای کاری‌اش فرستاده خواهد شد.

IV. ویرایش لیست زمان‌های کاری (JCDashboard/ModifyWorkingDate)

این صفحه یک لیست از زمان‌های کاری همکار به او نشان خواهد داد. در ستون سمت چپ هر زمان کاری دکمه حذف زمان کاری وجود دارد که همکار با فشردن آن، همکار می‌تواند زمان کاری مورد نظر را حذف کند.

V. لیست قرارهای کاری (JCDashboard/AppointmentList)

در این صفحه لیست قرارهای کاری همکار به او نشان داده خواهند شد. پیوند صفحه‌های قرارهای آتی و قرارهای گذشته هم در زیر این لیست قرار دارد و همکار در صورت تمایل می‌تواند آن‌ها را نیز مشاهده کند. همچنین در ستون سمت چپ هر سطر، کاربر می‌تواند با کلیک بر روی پیوند جزئیات، جزئیات بیشتری مربوط به آن قرار کاری مشاهده کند.

.VI جزئیات قرار کاری (JCDashboard/AppointmentDetails)

پس از انتخاب قرار کاری مورد نظر در لیست قرارهای کاری، همکار به این صفحه منتقل می‌شود و می‌تواند جزئیات آن قرار کاری را مشاهده کند. همچنین در پایین مشخصات قرار کاری، دکمه کنسل وجود دارد که همکار با فشردن آن به صفحه بعدی منتقل می‌شود.

.VII کنسل کردن قرار کاری (JCDashboard/AppointmentCancellation)

پس از زدن دکمه کنسل، همکار را به این صفحه می‌فرستیم و با نمایش جزئیات قرار کاری از او می‌خواهیم که در صورت تمایل قرار کاری مورد نظر را کنسل کند.

.VIII افزودن سرویس (Services/Create)

اگر یک همکار هنوز سرویسی برای کار خود اضافه نکرده باشد، به این صفحه منتقل خواهد شد. در این صفحه همکار، می‌تواند یک سرویس به سرویس‌های خود اضافه کند. همچنین پس از فشردن دکمه ثبت سرویس، او به لیست سرویس‌های خود منتقل خواهد شد و امکان ویرایش و حذف سرویس‌های قبلی را هم از طریق این صفحه خواهد داشت.

۲.۳.۴: عملیات تعریف شده برای کاربران عادی

لازم به ذکر است تمامی عملیات تعریف شده برای کاربران عادی برای همکاران و صاحبان مشاغل نیز قابل دسترسی می‌باشند.

.I ویرایش مشخصات کاربر (Account/Edit)

کاربران می‌توانند با کلیک بر روی نام خود در قسمت سمت چپ Header به قسمت ویرایش کاربر مراجعه کنند. در این صفحه آن‌ها می‌توانند مشخصات خود را تغییر دهند.

II. جستجوی مشاغل (Main/Jobs)

این صفحه را می‌توان صفحه اصلی کاربران دانست. پس از ورود به سیستم کاربران عادی به این صفحه فرستاده خواهند شد و قادر خواهند بود ضمن مشاهده لیست تمامی مشاغل، جستجو هم انجام دهند. با کلیک بر روی عنوان شغل مورد نظر، کاربر به صفحه‌ی لیست همکاران آن شغل منتقل خواهد شد.

III. لیست همکاران یک شغل (Main/JobCorpsList)

پس از انتخاب یک شغل، کاربران به این صفحه منتقل می‌شوند. در این صفحه لیست همکاران یک شغل آورده شده است و کاربر با فشردن دکمه رزرو وقت در ستون سمت چپ هر همکار به صفحه رزرو وقت او منتقل خواهد شد. هم‌چنین کاربران با کلیک کردن بر روی پیوند مشخصات شغل، مشخصات شغل مورد نظرشان را می‌توانند مشاهده کنند.

IV. رزرو وقت انتخابی (Main/ChooseReserve)

پس از انتخاب یک شغل و همکار مورد نظر، کاربر با فشردن دکمه رزرو وقت به این صفحه منتقل خواهد شد. در این صفحه با انتخاب تاریخ مورد نظر خود، اطلاعات آن زمان کاری همکار را به اضافه بقیه زمان‌های رزرو شده در آن زمان کاری را خواهد دید. سپس می‌تواند ساعت ملاقات و سرویس مورد نظرش را انتخاب کند. پس از فشردن دکمه رزرو وقت، اگر زمان انتخابی‌اش جزو زمان‌های کاری همکار باشد و فرد دیگری در آن زمان وقت نگرفته باشد، وقت او را رزرو خواهیم کرد و سپس او را به صفحه لیست قرارهای امروز و فردایش می‌فرستیم. اگر هم زمان انتخابی کاربر مناسب نباشد، پیام خطای مناسبی به او می‌دهیم و از او می‌خواهیم دقت بیشتری به خرج داده و زمان مناسبی انتخاب نماید. واضح است که اگر همکار مورد نظر هیچ سرویس یا هیچ زمان کاری‌ای اضافه نکرده باشد، کاربران قادر به رزرو وقت از او نخواهند بود.

V. لیست قرارهای امروز و فردای کاربر (Main/Reservations)

پس از رزرو وقت کاربران به این صفحه منتقل خواهند شد و می‌توانند قرارهای خود در دو روز آتی را مشاهده کنند. پیوند صفحه‌های قرارهای آتی و قرارهای گذشته هم در زیر این لیست قرار دارد و کاربر در صورت تمایل

می تواند آن ها را نیز مشاهده کند. هم چنین در ستون سمت چپ هر سطر، کاربر می تواند با کلیک بر روی پیوندهای جزئیات، ویرایش و کنسل، عملیات مربوطه به آن Reservation را انجام دهد.

IX. جزئیات قرار (Main/ReservationDetails)

پس از انتخاب قرار مورد نظر در لیست قرارها، کاربر به این صفحه منتقل می شود و می تواند جزئیات آن قرار را مشاهده کند. هم چنین در پایین مشخصات قرار کاری، دکمه های ویرایش و کنسل وجود دارد که همکار با فشردن آن ها به دو صفحه بعدی منتقل می شود.

X. کنسل کردن قرار (Main/ReserveCancellation)

پس از زدن دکمه کنسل، کاربر را به این صفحه می فرستیم و با نمایش جزئیات قرار از او می خواهیم که در صورت تمایل قرار مورد نظرش را کنسل کند.

XI. ویرایش قرار (Main/ReserveEdit)

پس از زدن دکمه ویرایش، کاربر را به این صفحه می فرستیم و کاربر می تواند با عوض کردن زمان یا سرویس مورد نظر خود، قرار خود را همکار مورد نظرش را ویرایش کند. پس از فشردن دکمه ویرایش، در صورت صحت اطلاعات وارد شده توسط کاربر او را به صفحه لیست قرارهای امروز و فردایش می فرستیم.

۲.۴: کارهای آینده و نتایج

۲.۴.۱: نتایج

هدف از این کار ساماندهی زمانبندی بین صاحبان مشاغل خدماتی و مشتریان آن ها بود که با یاری خدا این هدف را محقق نمودیم. در این کار، نهایت تلاش خود را کردیم تا کاربران به راحتی با سیستم ارتباط برقرار کنند^{۲۱} و به

²¹ User-Friendliness

منظور مشتری مداریِ صاحبان مشاغل، کاربرانِ عادی بتوانند با کمترین تعداد کلیک وقت رزرو کنند و از این که صاحب شغل به وقت آن‌ها اهمیت می‌دهد نهایت لذت را ببرند.

با توجه به کاربرد روز افزون گوشی‌های هوشمند و بقیه‌ی گجت‌های قابل حمل با صفحه نمایش‌های مختلف، سعی کردیم با استفاده‌ی هوشمندانه از تکنولوژی Bootstrap تجربه‌ی کاربری^{۲۲} خوبی به کاربرانی بدهیم که می‌خواهند به سرعت و با استفاده از گوشی هوشمندشان وقت بگیرند.

در پروژه‌ی مذکور ابتدا برخی نمونه‌های خارجی سیستم‌های رزرو وقت را مورد بررسی قرار دادیم. سپس پس از مطالعات کافی به این نتیجه رسیدیم مدل سه لایه‌ی معماری MVC برای پروژه مناسب است. با توجه به تجربه‌های کاری و تجربی قبلی بر آن شدیم تا از زبان تحت وب ASP.NET و چارچوب MVC آن برای پیش‌برد پروژه استفاده کنیم. ابزارهای میکروسافت هماهنگی بسیار خوب و کاملی با هم دارند و پس از یک یادگیری اولیه می‌توان کار پیاده‌سازی با استفاده از مجموعه‌ی این ابزارها آغاز نمود.

بهترین نتیجه‌ای که از انجام این پروژه گرفته شد یادگیری شخصی ابزارهای مختلف برنامه‌نویسی وب و البته بیشتر تکمیل یادگیری‌ها بود. هم‌چنین به زودی با راه‌اندازی سایت در اینترنت شاهد نتایج مثبت این پروژه خواهیم بود.

۲.۴.۲: کارهای آینده

پس از تلاش شخصی و البته کمک گرفتن از بسیاری از دوستان صاحب تجربه توانستیم این پروژه را به ثمر برسانیم. اولین کار در آینده‌ی نزدیک راه‌اندازی این سیستم در فضای اینترنت است تا بتواند کمی صاحبان مشاغل و مشتریان آن‌ها را از دغدغه‌ی زمان‌بندی و اتلاف وقت بیش از حد برباند. در آینده‌ی نزدیک برای صاحبان

مشاغل و مشتریان یک TimeTable تحت جاوااسکریپت به پروژه الحاق خواهیم کرد که با استفاده از یک نمای پویا و جامع سعی در حداکثر کردن رضایت کاربران از سیستم خواهد داشت.

در کارهای آتی، طراحی اپلیکیشن نسخه‌های موبایل iOS و اندروید در اولویت قرار دارد. با استفاده از اپلیکیشن موبایل دسترسی پذیری^{۲۳} سیستم به حد مطلوبی خواهد رسید و کاربران می‌توانند به راحتی با گوشی هوشمند خود از وقت‌های رزرو شده و دیگر امکانات سیستم بهره ببرند.

پیشنهاد دیگر که جزو اهداف آتی بلند مدت می‌باشد تغییر ظاهر سایت (و احتمالاً اپلیکیشن موبایل) می‌باشد تا کاربران از آن بیشتر لذت ببرند و کاربران بیشتری جذب سیستم شوند. هم‌چنین برای تبلیغات صاحبان مشاغل هم می‌توان ماژول‌های تبلیغاتی به سیستم افزود.

فصل سوم

تحلیل و طراحی

۳.۱: مقدمه

در این فصل به تحلیل و بررسی سیستم منشی دیجیتال می‌پردازیم.

در این تحلیل و بررسی، ابتدا نیازمندی‌های ابتدایی سیستم را از نظر گذرانده‌ایم. سپس موارد کاربردی سیستم و کاربران آن از طریق نمودار مورد کاربرد نشان داده شده است. علاوه بر این نمودار، ترتیب انجام عملیات سیستم توسط نمودار توالی کار برای نقش‌ها (بازیگران) مختلف سیستم ارائه شده است و در پایان نیز به معرفی کلی پایگاه داده سیستم منشی دیجیتال می‌پردازیم.

هر سیستمی می‌تواند دو جنبه داشته باشد: ایستا و پویا. یک سیستم تنها زمانی کامل محسوب می‌شود که هر دو جنبه را به‌طور کامل پوشش دهد.

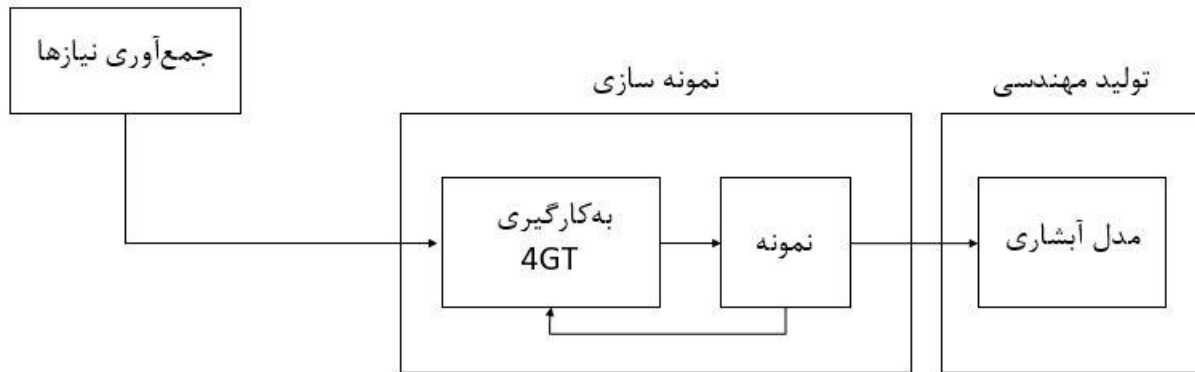
۳.۱.۱: مدل‌های فرایند نرم‌افزار^{۲۴}

فرایند نرم‌افزاری مجموعه‌ای از فعالیت‌هاست که هدف آن توسعه یا تکمیل نرم‌افزار است. بر اساس نوع نرم‌افزاری که قصد توسعه دادن آن را داریم و همچنین طبیعت پروژه و کاربرد آن مدل خاصی انتخاب می‌شود.

۳.۱.۲: ترکیب الگوها

بسته به کاربرد می‌توان برای توسعه یک نرم‌افزار از ترکیب مدل‌های قبلی استفاده نمود مثلاً اگر خصوصیات نرم‌افزار را نتوان به درستی شناخت ابتدا با روش 4GT یک نمونه‌سازی سریع انجام می‌گیرد. پس از مشخص شدن جزئیات برای داشتن یک محصول مطمئن و باکیفیت از مدل آبخاری استفاده شود زیرا این مدل شروع هر فاز با تست و اطمینان از فاز قبلی آغاز می‌شود).

شکل زیر فرایند دقیق این کار را نشان می‌دهد.



لازم به ذکر است در این پروژه ما از این مدل فرایند نرم افزار استفاده کردیم.

۳.۱.۳: نمودارهای رفتاری

این گونه نمودارها با استفاده از اجزای خود بر هم کنش اشیاء درون سیستم را نشان می دهند و رفتار سیستم را توصیف می کنند. در واقع نمودارهای کلاس یک دید ایستا از کلاس ها در سیستم فراهم می کنند ولی نمودارهای رفتاری یک دید پویا فراهم می نمایند و نشان می دهند که سیستم و کلاس های آن در آینده چگونه تغییر می کنند.

دید ایستا به تحلیل گر کمک می کند که با مشتری ارتباط برقرار کند. دید پویا کمک می کند که یک تحلیل گر با یک تیم از توسعه دهندگان ارتباط برقرار نماید و به توسعه دهندگان کمک می کند که برنامه ها را ایجاد نمایند. نمودارهای رفتاری تصویری از جنبه ی پویای سیستم ارائه می دهد. اگر بخواهیم جنبه ی پویای یک سیستم را دقیق تر توضیح دهیم، باید بگوییم که جنبه ی پویای سیستم همان بخش های در حال تغییر و حرکت سیستم می باشد.

به طور کلی، UML پنج نوع نمودار رفتاری ارائه می دهد که در زیر آنها را مشاهده می کنید:

۱. نمودار مورد کاربرد (Use case diagram)
۲. نمودار توالی (sequence diagram)
۳. نمودار همکاری (collaboration diagram)
۴. نمودار حالت (State diagram)
۵. نمودار فعالیت (Activity diagram)

به علت این که دو مورد از نمودارها بیشتر مورد توجه قرار می گیرند و ضمناً بقیه نمودارها نیز از روی آن ها قابل استخراج هستند در ادامه به بررسی مواردی از دو نمودار مورد کاربرد و توالی خواهیم پرداخت. برای مثال نمودار همکاری، همان نمودار توالی است بدون این که فاکتور زمان و ترتیب انجام کارها در آن اهمیت داشته باشد. یا مثلاً نمودار فعالیت از لحاظ تحلیلی ارزش چندانی ندارد چون نمودار توالی تقریباً آن را پوشش می دهد.

۳.۲: نیازمندی ها

خروجی فرایند مهندسی سیستم تعریفی از یک سیستم کامپیوتری یا محصول است. در این مرحله نیز این مشکل وجود دارد که چگونه مطمئن شویم که تعریف ارائه شده از سیستم نیازهای مشتری را برطرف می کند و انتظارات او را رفع می سازد. برای این منظور نیازمند به طی فرایند مهندسی محصول هستیم. این فرایند مکانیزم های مناسب را فراهم می آورد تا تشخیص دهیم مشتری چه می خواهد، نیازهای تحلیل چیست، یک راه حل معقول کدام است و ابهامات نیازمندی ها در کجاست.

مهندسی نیازمندی ها شامل مراحل زیر است:

استخراج نیازمندی ها: اهداف سیستم و یا محصول تعیین می گردد و نیز این که چه چیزی انجام گیرد، سیستم و یا محصول چگونه نیازهای تجاری را رفع می کند و بر اساس پایه ای روزانه کار می کند.

تحلیل و مذاکره ی نیازمندی ها: هنگامی که نیازمندیها جمع آوری شدند عمل تحلیل روی آن ها انجام می گیرد. تحلیل، نیازمندی ها را در زیر دسته هایی خاص طبقه بندی می کنند، ارتباط هر کدام را با دیگری بررسی نموده، جامعیت و ابهامات آن ها را تست و نیازمندی ها را بر اساس نیاز مشتری اولویت بندی می کند.

تعریف نیازمندی ها: قالب استاندارد برای نمایش نیازمندی ها که جامعیت آن ها حفظ شود ایجاد می گردد.

مدل سازی سیستم: بر اساس تعریف ایجاد شده از سیستم، یک مدل از آن ساخته می شود.

اعتبارسنجی نیازمندی ها: نیازمندی ها برای وجود ابهامات مورد آزمایش و بررسی دقیق قرار می گیرند.

مدیریت نیازمندی ها: مجموعه ای از فعالیت ها را تعریف می کند که باعث می شوند تیم پروژه بتواند تعیین، کنترل و ردگیری نیازمندی ها و تغییرات آن ها را در هر زمان مدیریت کند.

هنگامی که نیازمندی‌ها تعیین شدند، جدول ردیابی تشکیل می‌شود. این جدول هرکدام از نیازمندی‌های تعریف‌شده را به یک یا چند جنبه از سیستم یا محیط ربط می‌دهد. به گونه‌ای که نباید در آخر هیچ‌کدام از جنبه‌های سیستم بدون نیازمندی(های) متناظر بماند و همچنین نیازمندی‌ای باقی نماند که به جنبه‌ی سیستم متناظر متصل نشده باشد.

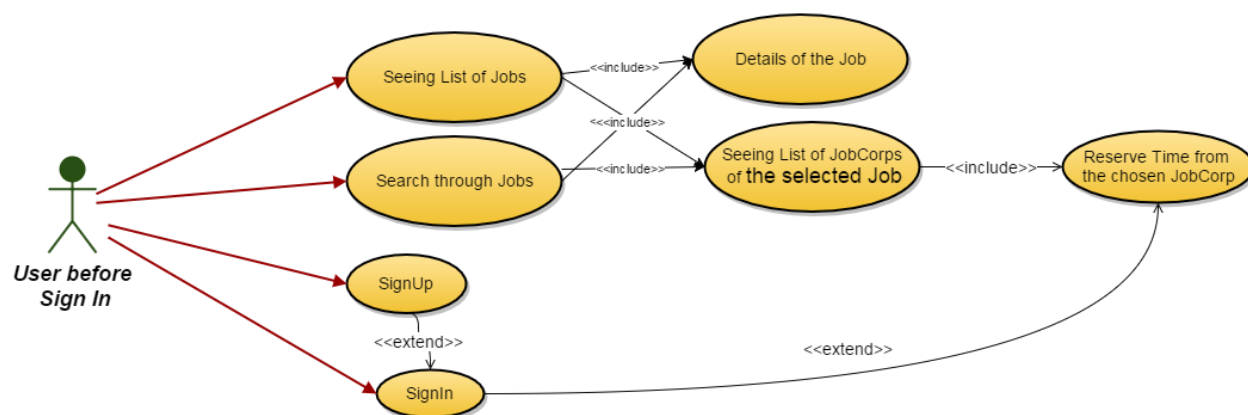
۳.۲.۱: نیازمندی‌های سیستم منشی دیجیتال

ردیف	نیازمندی
۱	سیستم باید دارای یک تقویم فارسی برای رزرو وقت باشد. (همگی زمان‌های نمایش داده شده به کاربران باید شمسی باشند)
۲	سیستم باید دارای صفحه‌ی ثبت نام مناسب باشد.
۳	سیستم باید قابلیت رزرو وقت برای کاربر از صاحب شغل در زمان‌های کاری صاحب شغل را داشته باشد. (بدون برخورد زمانی رزروهای کاربران با یکدیگر)
۴	سیستم باید دارای صفحه‌ی ورود کاربران باشد. (اهراز هویت)
۵	سیستم باید دارای قابلیت جستجوی شغل بر اساس عنوان شغل، نوع شغل و شهر را داشته باشد.
۶	سیستم باید طوری طراحی شود که برای کاربر ساده ترین شکل ممکن را داشته باشد. (UI و UX مناسب و درخور)
۷	سیستم باید قابلیت های پیشرفته‌ی وارد کردن زمان‌های کاری به‌صورت دسته ای یا تکی را به صاحبان مشاغل بدهد. (بدون افزودن زمان کاری مشتری قادر به رزرو وقت نخواهد بود)
۸	سیستم باید قابلیت ویرایش اطلاعات کاربری و تغییر نقش را به کاربر بدهد.
۹	سیستم باید قابلیت لغو ملاقات را هم برای مشتری و هم برای صاحبان مشاغل فراهم نماید.
۱۰	سیستم باید به مشتری این امکان را بدهد که اطلاعات مربوط به شغل مورد نظر و همچنین زمان‌های ملاقات صاحب شغل و همکاران آن شغل را مشاهده کند.
۱۱	سیستم باید به صاحب شغل این امکان را بدهد که با تعریف یک کد ثبت نام، آن را به همکاران خود بدهد تا آن‌ها با استفاده از آن بتوانند در سایت برای همکار شدن در آن شغل ثبت نام کنند.
۱۲	سیستم باید به صاحب شغل توانایی حذف همکاران را بدهد.

۱۳	سیستم باید به صاحب شغل و همکار این قابلیت را بدهد که بتوانند زمان‌های کاری‌ای که قبلاً به سیستم اضافه کرده اند را تک تک حذف (لغو) کنند.
۱۴	سیستم باید به صاحبان شغل و همکاران این امکان را بدهد که برای خود چندین سرویس (شامل نام سرویس، زمان سرویس‌دهی و شرح مختصر) تعریف نمایند تا مشتری‌ها به راحتی سرویس خود را انتخاب نمایند. (بدون افزودن سرویس مشتری قادر به رزرو وقت نخواهد بود)

۳.۳: کاربران سیستم

نمودار مورد کاربرد کاربران قبل از وارد شدن به سیستم به صورت زیر است:

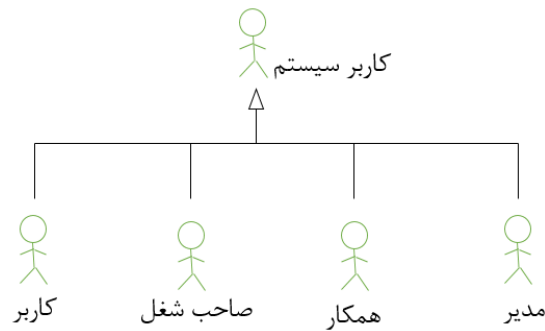


شرح موارد کاربرد مهم کاربران قبل از ورود به سیستم:

- مشاهده لیست مشاغل و قابلیت جستجوی آن‌ها: هر کاربری قبل از ورود به سیستم می‌تواند این قابلیت را داشته باشد.
- هم‌چنین در ادامه مورد کاربرد بالا، کاربران پس از انتخاب یک شغل، قادر به دیدن جزئیات بیشتر در مورد شغل مورد نظر و هم‌چنین مشاهده‌ی لیست همکاران در آن شغل می‌باشند.
- کاربران می‌توانند با استفاده از نام (یا ایمیل) و رمز عبور خود وارد سایت شوند. البته ورود به سایت منوط به ثبت نام در سایت می‌باشد.
- ورود به صفحه رزرو وقت از یک همکار منوط به ورود به سایت می‌باشد.

در این پروژه ۴ نوع کاربر (یا همان actor) داریم:

لازم به ذکر است که در طراحی شیء گرا رابطه ارث بری is-a بین actorها را می‌توان تعریف کرد. در شکل زیر این رابطه را با استفاده از پیکان‌هایی به سمت کاربر سیستم (که در بالای آن‌ها مثلث‌های توخالی وجود دارند) تعریف نموده‌ایم.

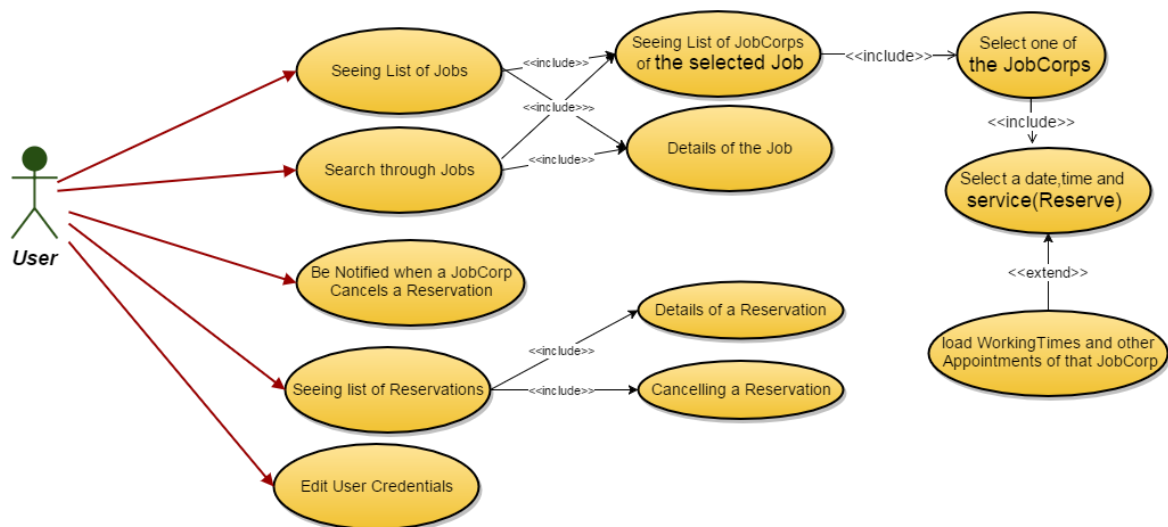


در قسمت سمت راست صفحه^{۲۵} در هر نوع View به نسبت نقش کاربر لینک‌هایی برای دسترسی کاربران قرار داده شده است. در پایان توضیح هر actor لینک‌هایی را که در اختیارش خواهیم گذاشت را ذکر خواهیم کرد.

کاربری که هنوز وارد سیستم نشده است، تنها به لینک‌های زیر دسترسی خواهد داشت:

۳.۳.۱: کاربر عادی (User)

محدوده عملیات کاربر عادی محدود به دو کنترلر HomeController و MainController می‌باشد. کاربر عادی می‌تواند از بین مشاغل جستجو کند و از JobCorp مورد نظر در آن شغل وقت بگیرد. همچنین می‌تواند لیست Reservationهای خودش را به راحتی مشاهده کند. همچنین می‌تواند از بین لیست Reservation خود یکی از آن‌ها را انتخاب نموده و آن را کنسل کند یا جزئیات آن را مشاهده کند.



شرح موارد کاربرد مهم کاربران عادی:

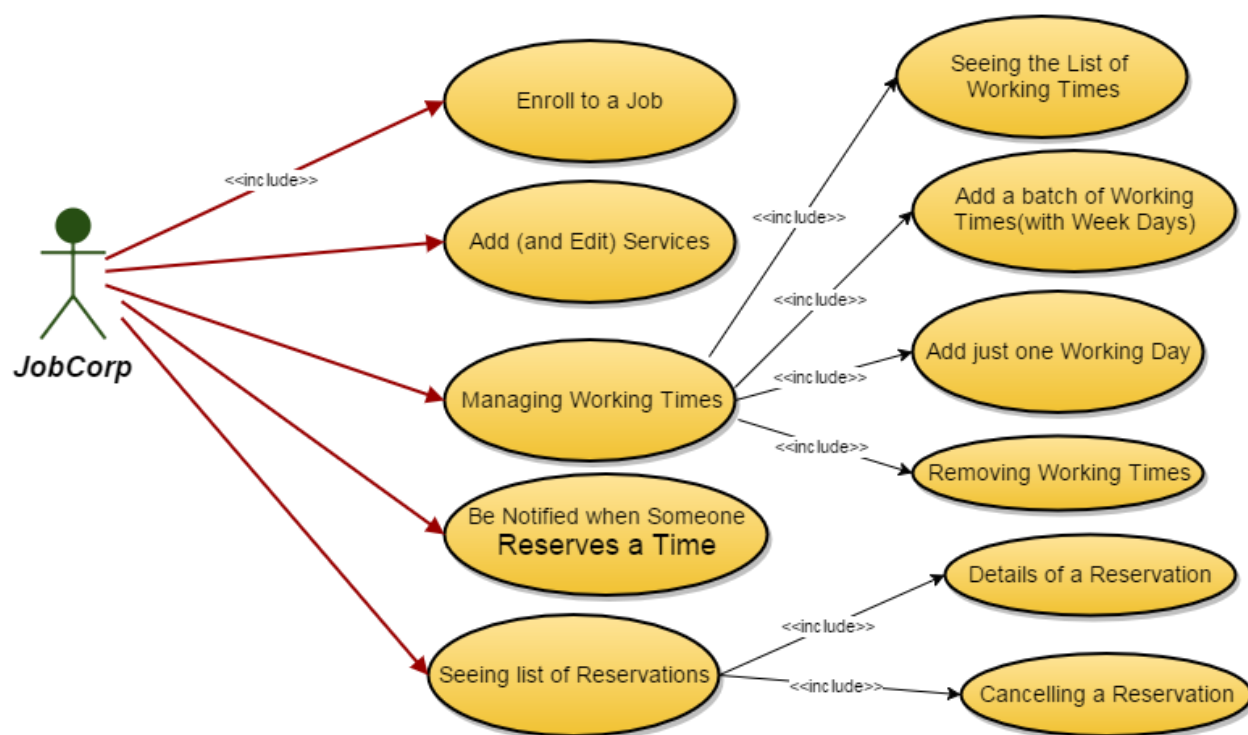
- زمانی که یک همکار، وقتی را کنسل می کند به آن ها اطلاع داده می شود.
- می توانند لیست رزرو وقت خود را با تفکیک قرارهای قبلی، بعدی و دو روز آینده مشاهده کنند.
- می توانند جزئیات وقت های گرفته شده خود را مشاهده کنند و همچنین توانایی کنسل کردن آن ها را نیز دارند.
- می توانند اطلاعات کاربری خود را ویرایش کنند.
- (برخلاف کاربران وارد نشده به سیستم) می توانند پس از انتخاب یک همکار، به صفحه رزرو رفته و زمان و سرویس خود را انتخاب کرده و وقت را رزرو نمایند.

۳.۳.۲: همکار (JobCorp)

یک همکار تمامی امکانات یک کاربر عادی از جمله امکان رزرو و مشاهده لیست Reservation را خواهد داشت. همچنین می تواند زمان های کاری خودش را برای شغلی که در آن ثبت شده است را وارد کند و لیست Appointments (را که مخصوص قرارهای کاری اش است) را ببیند. همچنین توانایی کنسل کردن قرارهای کاری را دارد و می تواند به راحتی جزئیات آن را مشاهده کند.

شکل زیر نمودار مورد کاربرد مخصوص کارهایی است که اکتور همکار می تواند انجام دهد. در بالاترین پیکان سمت چپ منظور از include این است که شاید JobCorp صاحب شغل باشد. اگر JobCorp خود صاحب شغل

باشد نیازی به **Enroll** کردن ندارد. هم‌چنین چند قابلیت دیگر نیز دارد که در نمودار بعدی فقط آن‌ها را نشان خواهیم داد. تمامی کارهای این نمودار جز **Enroll** کردن، توسط صاحب شغل هم قابل انجام است.



شرح موارد کاربرد مهم همکاران:

- قبل از انجام هر کار مربوط به همکاران در سیستم باید در شغلی ثبت نام کنند. (برای شلوغ نشدن نمودار مورد کاربرد، دیگر از سمت **Enroll to a Job** به سمت بقیه موارد کاربرد پیکان **extend** نکشیده‌ایم.
- آن‌ها قابلیت مشاهده، افزودن و حذف سرویس‌های خود را دارند.
- می‌توانند زمان‌های کاری خود را مدیریت کنند. متدهای مختلفی برای این کار برای آن‌ها در نظر گرفته شده است. مثلاً می‌توانند به صورت دسته‌ای و با تعریف روزهای هفته، زمان کاری خود را تا یک ماه آینده مشخص کنند. هم‌چنین می‌توانند تنها یک روز انتخاب نمایند و به زمان کاری خود اضافه کنند (استثنا اضافه کنند). و هم‌چنین قابلیت حذف زمان‌های کاری خود را خواهند داشت. (استثنا حذف کنند).
- زمانی که کاربری از آن‌ها وقت می‌گیرد به آن‌ها اطلاع داده شود.
- لیست رزرو وقت خود را با تفکیک قرارهای قبلی، بعدی و دو روز آینده را ببینند. (یعنی لیست زمان‌هایی که کاربران از آن‌ها رزرو کرده اند). البته همانند کاربران عادی، همکاران هم توانایی مشاهده لیست رزرو وقت خود با شغل‌های دیگر را هم دارند.

- می‌توانند جزئیات وقت‌های رزرو شده توسط کاربران را مشاهده کنند و همچنین توانایی کنسل کردن آن‌ها را نیز دارند.

۳.۳.۳: صاحب شغل (JobOwner)

صاحب شغل تمام امکانات نقش‌های همکار و کاربر عادی را دارد. همچنین توانایی عوض کردن مشخصات شغل مورد نظر و مدیریت همکاران (از جمله حذف همکاران و تعریف کلید ورود برای همکاران جدید) را داراست.

شکل زیر نمودار مورد کاربرد مخصوص کارهایی است که اکتور صاحب شغل می‌تواند انجام دهد. البته این کارها به جز کارهایی است که در نمودار بالا معرفی شده‌اند. همان‌طور که پیش‌تر گفتیم صاحب شغل تمامی عملیات همکار را می‌تواند انجام دهد. وجود **include** قبل از "مشاهده لیست همکاران" نشانگر این است که اگر صاحب شغل هنوز همکاری نداشته باشد نمی‌تواند لیستی را هم مشاهده کند.



شرح موارد کاربرد مهم صاحبان مشاغل:

- توانایی تعریف کردن کلید ورود برای ثبت نام همکاران دیگر در شغل
- توانایی ویرایش مشخصات شغل
- توانایی مشاهده لیست همکاران و همچنین توانایی حذف همکاران

۳.۳.۴: مدیر (Admin)

مدیر نقش نظارت بر سایت را برعهده دارد و می‌تواند نوع شغل و شهرهای جدید تعریف کند.

شرح موارد کاربرد مهم مدیر:

- توانایی تعریف نوع شغل‌های جدید و هم‌چنین ویرایش نوع شغل‌های قبلی
- توانایی تعریف شهرهای جدید و هم‌چنین ویرایش شهرهای قبلی

در این سیستم نقش‌ها سلسله مراتب دارند و برای مثال یک JobCorp می‌تواند مانند یک کاربر عادی از JobCorp دیگری وقت رزرو کند ولی کاربر عادی مثلاً نمی‌تواند وقت کاری خود (جزو شرح وظایف JobCorp) را به سیستم بیفزاید. شکل زیر بهتر مفهوم این سلسله مراتب را تبیین می‌نماید. ضمناً نماد پیکان با مثلث توخالی همان رابطه is-a می‌باشد. یعنی برای مثال یک JobCorp یک User می‌باشد و تمام ویژگی‌ها و قابلیت‌های یک User را دارد.

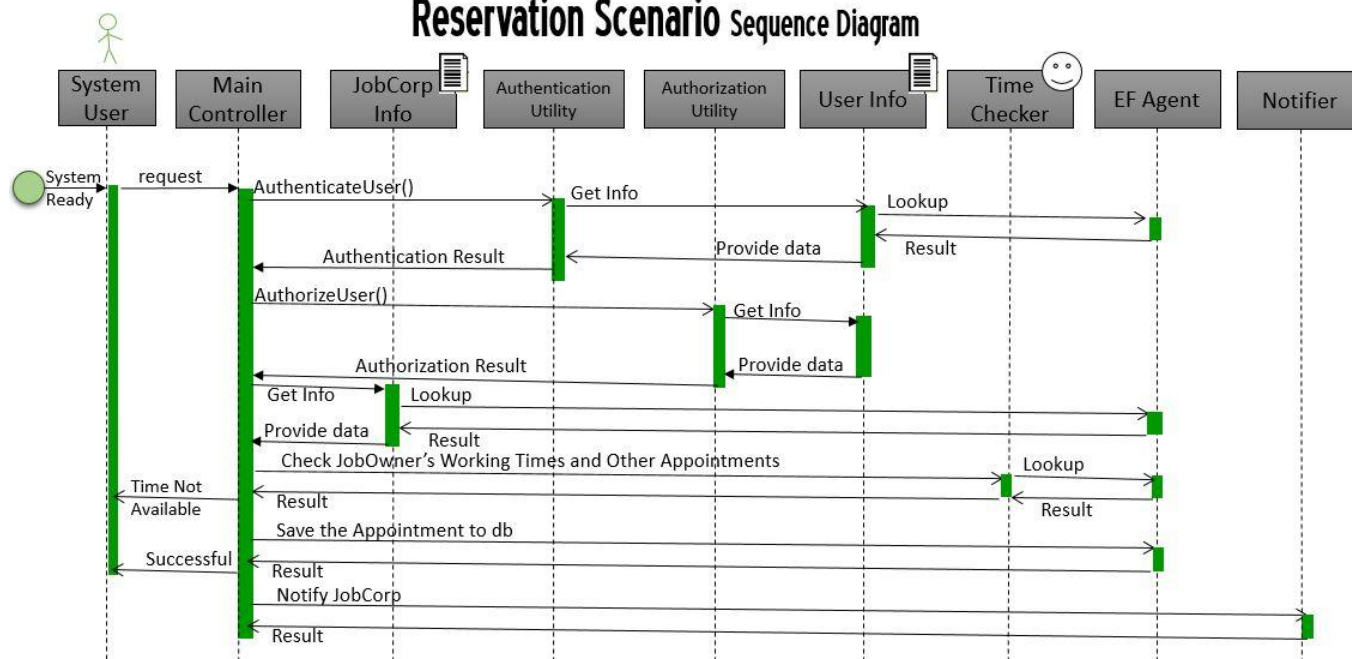


۳.۴: چند نمودار UML دیگر

در ادامه چند نمودار UML که به فهم بیشتر سیستم کمک می‌کنند را مورد بررسی قرار می‌دهیم. ابتدا چند نمودار توالی را بررسی می‌کنیم.

نمودار زیر نمودار توالی رزرو کردن وقت توسط کاربر است که با جزئیات به جریان یافتن داده‌ها و اعمال مختلف در سیستم هنگام رزرو وقت توسط کاربر می‌پردازد.

Reservation Scenario Sequence Diagram



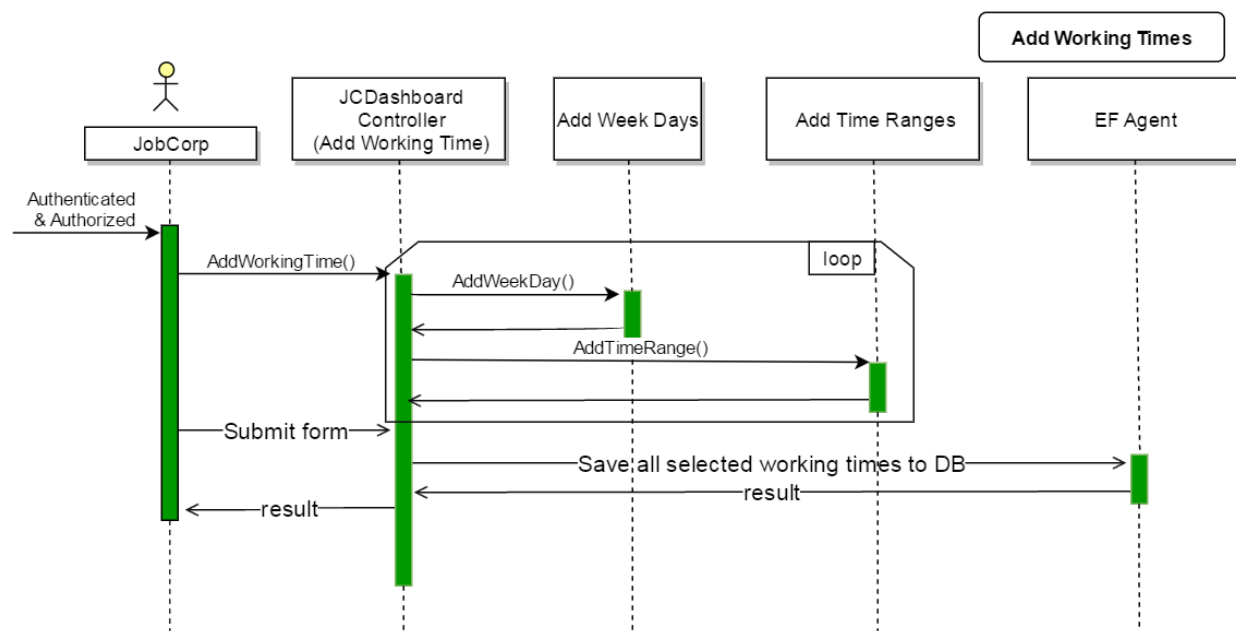
منظور از **System User** هر نوع کاربر سیستم است حتی کاربری که هنوز ثبت نام نکرده است. ابتدا با درخواست رزرو وقت از یک همکار در یک شغل، یک درخواست به **Main Controller** می‌فرستد، سپس سیستم چک می‌کند که آیا او وارد سیستم شده است یا خیر (**Authentication Utility**). سپس در صورتی که کاربر در سیستم وارد شده باشد نقش کاربری او را چک می‌کنیم. (**Authorization Utility**) سپس با پرس و جو از واسط سیستم با پایگاه داده (**Entity Framework Agent**) سرویس‌ها و زمان‌های کاری همکار مورد نظر در **MainController** لود می‌شوند. به محض انتخاب زمان و سرویس توسط کاربر، زمان‌های همکار مورد نظر از طریق یک موجودیت **worker-type** به نام **Time Checker** که از پایگاه داده اطلاعات را می‌گیرد، بررسی می‌شوند و جواب این بررسی به **Main Controller** بازمی‌گردد. سپس در صورت عدم تداخل زمانی، **Main Controller** تمامی اطلاعات مربوط به **Reservation** را به پایگاه داده می‌فرستد تا آنجا ذخیره شوند. در گام آخر نیز **Main Controller** توسط **Notifier** قرار کاری جدید را از طریق ایمیل یا پیامک به همکار مورد نظر اطلاع‌رسانی می‌کند.

لازم به ذکر است تنها اکتور **System User** در این دیاگرام وجود داشت. البته خود پایگاه داده را نیز می‌توان یک اکتور سیستم در نظر گرفت ولی برای اختصار تنها واسط پایگاه داده یعنی **EF Agent** را در این نمودار نشان دادیم. در مدل استاندارد **EF Agent** در هر تراکنش با پایگاه داده ارتباط برقرار خواهد کرد و این باعث شلوغی بیش از حد نمودار می‌شود. هم‌چنین در مدل استاندارد باید اکتور **System User** را جدا در نظر

می‌گرفتیم و هر کلیک او را از طریق موجودیت مرزی **Interface Page** نشان می‌دادیم که باز هم به دلیل شلوغ شدن بیش از حد نمودار از نمایش آن پرهیز کرده ایم.

برای این که دیگر نیازی به نمودار دامنه نباشد بالای موجودیت‌ها با علامت‌هایی نوع آن‌ها را مشخص کرده ایم. برای مثال موجودیت **Time Checker** مفهوم **worker** بودن یک موجودیت را در بردارد، بدین معنی که مسئول چک کردن زمان از پایگاه داده می‌باشد و باید نتیجه را به **Main Controller** اعلام کند. هم چنین موجودیت‌های **User Info** و **JobCorp Info** نیز مفهوم **thing** بودن یک موجودیت را در بردارند و با استفاده از این موجودیت‌ها در کارهای بعدی دیگر نیازی به مراجعه به پایگاه داده نخواهد بود. هم چنین تمامی موجودیت‌هایی که علامتی کنار آن‌ها نیست اجزای مرزی^{۲۶} سیستم هستند، یعنی با اکتورها یا اجزای خارج از سیستم (که البته در بیشتر موارد آن‌ها را هم جزو اکتورها در نظر می‌گیرند) سر و کار دارند.

نمودار زیر نمودار توالی افزودن زمان‌های کاری به صورت دسته‌ای توسط همکاران (یا صاحبان مشاغل) است که با جزئیات به جریان یافتن داده‌ها و اعمال مختلف در سیستم هنگام افزودن زمان‌های کاری می‌پردازد:



برای اختصار دیگر نخواهیم گفت همکار یا صاحب شغل، ذکر کلمه **JobCorp** برای نمایندگی از هر دو actor است. **JobCorp** با انتخاب گزینه «روزهای کاری هفتگی» به صفحه مربوطه وارد می‌شود.

با انتخاب گزینه «افزودن روزهای هفته» می تواند به تعداد مورد نیازش روز هفته اضافه کند.

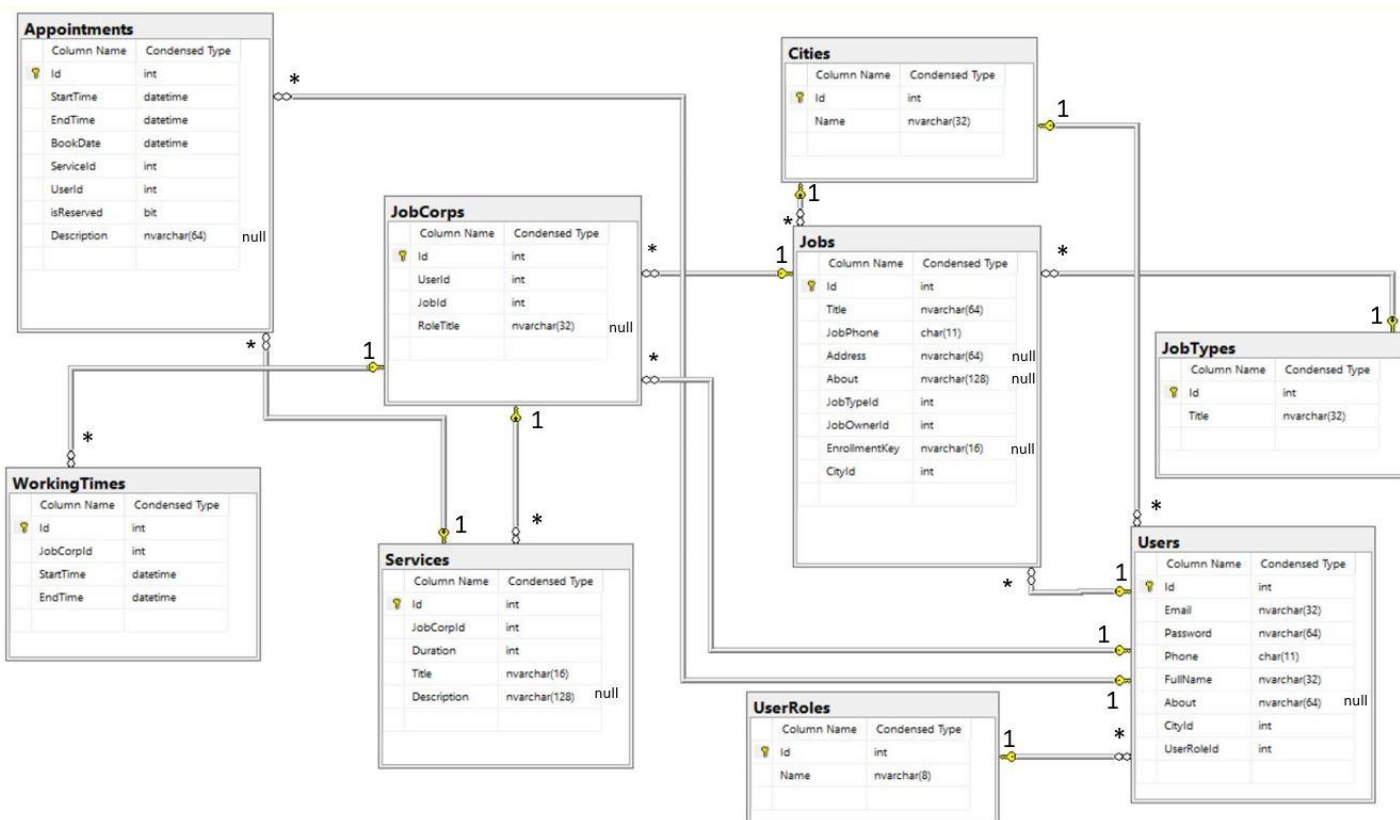
سپس با انتخاب گزینه «ساعت شروع و پایان»، به تعداد مورد نیاز خود، بازه ی زمانی برای روزهای انتخاب شده اضافه نماید.

پس از انجام این کارها با فشردن گزینه «ثبت زمان»، زمان های کاری مورد نظر JobCorp در پایگاه داده ثبت خواهند شد و سپس او به صفحه «لیست قرارهای کاری» خود فرستاده می شود.

از آن جا که این پروژه را به صورت Database-First پیش بردیم و اکثر مدل ها در نمودار ER که در بخش بعد خواهیم آورد، به همراه روابطشان نمایش داده شده اند، نیازی نمی بینیم کلاس دیاگرام پروژه را در اینجا بیاوریم.

۳.۵: پایگاه داده

در زیر نمودار ER^{۲۷} پایگاه داده به شکل مبسوط آورده شده است:



۳.۵.۱: جدول‌های پایگاه داده

حال تمامی جدول‌های واقع شده در پایگاه داده را مورد بررسی قرار می‌دهیم:

۱. جدول UserRoles:

در قسمت Name، اسم نقش کاربر نوشته می‌شود. ضمناً این جدول تنها شامل چهار مقدار User، JobOwner، JobCorp و Admin می‌باشد. در جدول Users کلید خارجی این جدول به نام UserRoleId وجود دارد.

UserRoles			
	Column Name	Condensed Type	Nullable
🔑	Id	int	No
	Name	nvarchar(8)	No

۲. جدول Cities:

در قسمت Name، نام شهر نوشته می‌شود. در جدول‌های Users و Jobs کلید خارجی این جدول به نام CityId وجود دارد.

Cities			
	Column Name	Condensed Type	Nullable
🔑	Id	int	No
	Name	nvarchar(32)	No

۳. جدول Users:

در قسمت Email، ایمیل کاربر، در قسمت Password رمز عبور کاربر، در قسمت Phone شماره موبایل کاربر، در قسمت FullName نام و نام خانوادگی کاربر، در قسمت About کاربر می‌تواند توضیحاتی در مورد خود نیز بدهد (فیلد Nullable می‌باشد). هم‌چنین دو کلید خارجی CityId و UserRoleId نیز به ترتیب به دو جدول Cities و UserRoles متصل می‌شوند. جدول‌های Jobs، JobCorps و Appointments هم با کلید خارجی UserId به این جدول متصل می‌شوند.

Users			
	Column Name	Condensed Type	Nullable
🔑	Id	int	No
	Email	nvarchar(32)	No
	Password	nvarchar(64)	No
	Phone	char(11)	No
	FullName	nvarchar(32)	No
	About	nvarchar(64)	Yes
✓	CityId	int	No
✓	UserRoleId	int	No

IV. جدول JobTypes

در قسمت Title عنوان نوع شغل مورد نظر قرار می‌گیرد. (برای مثال پیرایش، تجارت یا پزشکی) در جدول Jobs کلید خارجی این جدول به نام JobTypeId وجود دارد.

JobTypes			
	Column Name	Condensed Type	Nullable
🔑	Id	int	No
	Title	nvarchar(32)	No

V. جدول Jobs

در قسمت Title عنوان شغل، در قسمت JobPhone شماره تلفن محل کار، در بخش Address که Nullable هم می‌باشد آدرس محل کار، در بخش About شرح مختصری درباره‌ی شغل مورد نظر می‌تواند داده شود. (Nullable) همچنین EnrollmentKey نیز دربردارنده رمز ورود برای همکاران دیگر در شغل است که ابتدا به صورت پیش فرض Null فرض می‌شود و سپس صاحب شغل می‌تواند در صورت تمایل آن را اضافه کند.

ضمناً سه کلید خارجی CityId و JobOwnerId و JobTypeId نیز به ترتیب به دو جدول Cities و Users و JobTypes متصل می‌شوند. هر موجودیت Job تنها یک صاحب شغل دارد که شناسه کلید خارجی JobOwnerId که به جدول Users متصل شده آن را تعیین می‌کند. جدول JobCorps هم با کلید خارجی JobId به این جدول متصل می‌شوند.




Jobs			
	Column Name	Condensed Type	Nullable
🔑	Id	int	No
	Title	nvarchar(64)	No
	JobPhone	char(11)	No
	Address	nvarchar(64)	Yes
	About	nvarchar(128)	Yes
✓	JobTypeId	int	No
✓	JobOwnerId	int	No
	EnrollmentKey	nvarchar(16)	Yes
✓	CityId	int	No

VI. جدول JobCorps

در قسمت RoleTitle همکار می‌تواند نقشش را بنویسد. به صورت پیش فرض به JobOwner در RoleTitle مقدار صاحب شغل و برای JobCorp لای بعداً اضافه شده مقدار همکار در نظر گرفته می‌شود.



همچنین دو کلید خارجی JobId و UserId نیز به ترتیب به دو جدول Jobs و Users متصل می‌شوند.

جدول‌های Services و WorkingTimes هم با کلید خارجی JobCorpId به این جدول متصل می‌شوند.

JobCorps			
	Column Name	Condensed Type	Nullable
	Id	int	No
	UserId	int	No
	JobId	int	No
	RoleTitle	nvarchar(32)	Yes

VII. جدول Services:

در قسمت Title همکار مورد نظر عنوان خدمت خود را می‌نویسد و در قسمت Description که Nullable هم هست می‌تواند درباره آن توضیح دهد (قیمت بگذارد و ...). در قسمت Duration هم وقتی که برای آن خدمت خواهد گذاشت را می‌نویسد (به دقیقه). هم‌چنین این جدول با کلید خارجی JobCorpld نیز به جدول JobCorps متصل می‌شود. جدول Appointments هم با کلید خارجی Serviceld به این جدول متصل می‌شود.

Services			
	Column Name	Condensed Type	Nullable
	Id	int	No
	JobCorpld	int	No
	Duration	int	No
	Title	nvarchar(16)	No
	Description	nvarchar(128)	Yes

VIII. جدول WorkingTimes:

در قسمت StartTime که نوع datetime نیز دارد شروع زمان کاری یک روز قرار می‌گیرد. (لازم به ذکر است که در یک روز می‌توان چند زمان کاری هم داشت.) (مثال به زبان ساده: دوشنبه ۲۷ آذر ۱۳۹۵ ساعت ۱۰:۰۰) و قسمت EndTime هم به همین گونه است با این تفاوت که پایان زمان کاری را مشخص می‌کند. (ادامه مثال بالا: دوشنبه ۲۷ آذر ۱۳۹۵ ساعت ۱۴:۰۰) این جدول با کلید خارجی JobCorpld نیز به جدول JobCorps متصل می‌شود.

WorkingTimes			
	Column Name	Condensed Type	Nullable
🔑	Id	int	No
	JobCorpId	int	No
	StartTime	datetime	No
	EndTime	datetime	No

IX. جدول Appointments:

در قسمت StartTime کاربر زمان شروع ملاقاتش را ثبت می کند. زمان EndTime را بعدا در کنترلر خود مساوی StartTime+Service.Duration می کنیم. BookDate زمان رزرو وقت توسط کاربر است. بیت isReserved هر وقت ۰ شود بدین معناست که قرار کنسل شده است. (موقع کنسل شدن قرار آن را از رکوردهای Appointments حذف نمی کنیم.) در غیر این صورت بیت isReserved یک خواهد بود. قسمت Description نیز Nullable است و کاربر می تواند توضیحاتی به همکاری که از او وقت گرفته بدهد.

همچنین دو کلید خارجی ServiceId و UserId نیز به ترتیب به دو جدول Services و Users متصل می شوند. (این که User از کدام JobCorp وقت گرفته به راحتی از Service انتخابی قابل تشخیص خواهد بود.)

Appointments			
	Column Name	Condensed Type	Nullable
🔑	Id	int	No
	StartTime	datetime	No
	EndTime	datetime	No
	BookDate	datetime	No
	ServiceId	int	No
	UserId	int	No
	isReserved	bit	No
	Description	nvarchar(64)	Yes

فصل چهارم

پیاده سازی

۴.۱: مقدمه

با توجه به این که در طراحی سیستم منشی دیجیتال از معماری MVC استفاده شد و در پیاده سازی آن از ASP.NET MVC 6 بهره جستیم، در بخش پیاده سازی به تشریح سه قسمت معماری MVC یعنی مدل‌ها، کنترلرها و View ها می‌پردازیم.

لازم به ذکر است که در این پروژه ابتدا پایگاه داده طراحی شد و سپس با استفاده از روش Database First دسترسی به پایگاه داده ها را با استفاده از Entity Framework فراهم نمودیم.

۴.۲: بخش مدل‌ها

نظر به این که در این پروژه از Entity Framework و روش Database First استفاده نموده ایم دو نوع مدل را اینجا تشریح خواهیم کرد:

- مدل‌های استخراج شده از پایگاه داده
- مدل‌های دیگر

ابتدا به بررسی مدل‌هایی می‌پردازیم که از پایگاه داده استخراج شده‌اند:

۴.۲.۱: مدل‌های استخراج شده از پایگاه داده

در این پروژه سعی کردیم با استفاده از قابلیت های Entity Framework پایگاه داده ای را که برای سیستم قبل از پیاده‌سازی طراحی کرده بودیم را به سادگی روی پروژه پیاده‌سازی نماییم. در این عملیات هر جدول در پایگاه‌داده تبدیل به یک کلاس مدل می‌شود و هر فیلد جدول پایگاه‌داده هم به یک attribute به همراه setter و getterهایش در همان کلاس تبدیل می‌شود.

ضمناً همه این کلاس ها زیرمجموعه فایل Appointer.edmx می‌شوند که وقتی می‌خواستیم پایگاه‌داده را به مدل تبدیل کنیم توسط Entity Framework ساخته شده‌بود. ضمناً یک فایل دیگر به اسم Appointer.edmx.sql هم داریم که دستورات sql درون آن قرار دارند.

برای رعایت اصل ایجاز این کلاس‌ها و **Attribute**‌های آن‌ها را در اینجا تشریح نمی‌کنیم چون در فصل قبل و در طراحی پایگاه داده به آن‌ها پرداخته بودیم.

در اینجا تنها به یک مثال از این کلاس‌ها می‌پردازیم و آن را تشریح می‌کنیم:

کلاس **User.cs** که از جدول **Users** استخراج شده است به صورت زیر توسط **Entity Framework** تولید شده است:

```
namespace Appointer.Models
{
    using System;
    using System.Collections.Generic;
    using System.ComponentModel.DataAnnotations;
    using System.Web.Mvc;

    public partial class User
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
        "CA2214:DoNotCallOverridableMethodsInConstructors")]
        public User()
        {
            this.Appointments = new HashSet<Appointment>();
            this.JobCorps = new HashSet<JobCorp>();
            this.Jobs = new HashSet<Job>();
        }

        public int Id { get; set; }

        [Required(ErrorMessage = "وارد کردن ایمیل الزامی است")]
        [EmailAddress(ErrorMessage = "ایمیل درست وارد کنید")]
        [StringLength(32)]
        [DataType(DataType.EmailAddress)]
        [Display(Name = "ایمیل")]
        public string Email { get; set; }

        [Required(ErrorMessage = "وارد کردن رمز عبور اجباری است")]
        [DataType(DataType.Password)]
        [Display(Name = "رمز عبور")]
        [StringLength(64, ErrorMessage = "رمز عبور باید حداقل ۶ کاراکتر باشد", MinimumLength = 6)]
        public string Password { get; set; }
    }
}
```

```

[DataType(DataType.PhoneNumber)]
[Display(Name = "شماره موبایل")]

[Required(ErrorMessage = "وارد کردن شماره موبایل الزامی است")]
[StringLength(11)]
[RegularExpression(@"^09(1[0-9]|3[1-9]|2[1-9])-[0-9]{7}$", ErrorMessage = "شماره موبایل خود را با فرمت زیر وارد کنید: 09127404062")]
public string Phone { get; set; }

[Required(ErrorMessage = "این فیلد باید پر شود")]
[Display(Name = "نام و نام خانوادگی")]
public string FullName { get; set; }

[Display(Name = "درباره شما")]
public string About { get; set; }

[Display(Name = "شهر")]
public int CityId { get; set; }

[Display(Name = "وظیفه")]
public int UserRoleId { get; set; }

public String RoleName
{
    get
    {
        return Enum.GetName(typeof(MyUserRole), (MyUserRole)UserRoleId);
    }
}

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
public virtual ICollection<Appointment> Appointments { get; set; }
public virtual City City { get; set; }
[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
public virtual ICollection<JobCorp> JobCorps { get; set; }
[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage",
"CA2227:CollectionPropertiesShouldBeReadOnly")]
public virtual ICollection<Job> Jobs { get; set; }
public virtual UserRole UserRole { get; set; }
}
}

```


در همان ابتدای تعریف کلاس و در تابع `public User()` به معرفی دیگر کلاس‌هایی پرداخته شده که با استفاده از کلید خارجی به کلاس `User` متصل هستند. برای مثال در خط اول `this.Appointments = new` `(HashSet<Appointment>())` به معرفی کلاس `Appointment` پرداخته شده تا اگر در آینده این کلاس از کلاس `User` استفاده نمود با استفاده از این خط به این کلاس دسترسی پیدا کند. همچنین وجود سطر زیر که در آخر کلاس آورده شده هم برای این امر ضروری است:

```
public virtual ICollection<Appointment> Appointments { get; set; }
```

همچنین در آخر کلاس به تعریف جدول‌هایی پرداخته می‌شود که در جدول `User` به سمت آن‌ها کلید خارجی وجود دارد. برای مثال سطر:

```
public virtual UserRole UserRole { get; set; }
```

کلاس `UserRole` را به کلاس `User` معرفی می‌کند تا در آینده بتوان از محتویات آن کلاس در کلاس `User` استفاده کرد.

سپس به معرفی فیلدهای مختلف جدول `Users` پرداخته شده است. در بالای تعریف بعضی از فیلدها نوعی حاشیه نویسی انجام شده است که بعدها در طراحی `View`‌ها به ما کمک شایانی خواهند کرد. برای مثال در بالای فیلد `Phone` این حاشیه‌نویسی‌ها انجام شده‌اند که تک تک به معرفی آن‌ها می‌پردازیم:

```
[DataType(DataType.PhoneNumber)]
```

این حاشیه‌نویسی باعث می‌شود تا کامپایلر بفهمد این فیلد را باید از نوع شماره تلفن بگیرد و نوع دیگری را نپذیرد تا در آیند و در ذخیره‌ی آن در پایگاه داده به مشکل نخوریم.

```
[Display(Name = "شماره موبایل")]
```

این حاشیه‌نویسی باعث می‌شود تا هر وقت در `View` از دستور `@Html.LabelFor(model => model.Phone)` استفاده کردیم به کاربر نهایی عبارت "شماره موبایل" نشان داده شود.

```
[Required(ErrorMessage = "وارد کردن شماره موبایل الزامی است")]
```

این حاشیه‌نویسی باعث می‌شود تا هر وقت کاربر برای مثال در فرم ثبت نام این فیلد را خالی بگذارد و بخواهد فرم را `submit` کند با این خطا مواجه شود.

```
[StringLength(11)]
```

این حاشیه‌نویسی باعث می‌شود تا حداکثر طول رشته‌ای که از کاربر گرفته می‌شود را مشخص کنیم. با توجه به این که در ایران شماره‌های موبایل ۱۱ رقمی هستند و نیاز نیست برای این فیلد بیشتر جا در نظر بگیریم تصمیم گرفتیم کاربر نتواند بیشتر از این تعداد رقم را نتواند وارد کند تا در ثبت مشخصات خود سهل انگاری نکند.

```
[RegularExpression(@"^09(1[0-9]|3[1-9]|2[1-9])-[0-9]{7}$", ErrorMessage = "لطفا شماره خود را این  
گونه وارد کنید: ۰۹۱۲۷۴۰۴۰۶۲")]
```

با استفاده از این حاشیه نویسی از عبارات منظم استفاده کرده‌ایم بدین صورت که چون در ایران همه شماره‌های موبایل با ۰۹ آغاز می‌شوند و پس از این دو رقم ۲،۱ یا ۳ داریم که بعد از آنها یک رقم دلخواه می‌تواند بیاید (رقم چهارم از چپ) و بعد از این رقم‌ها که برای بررسی شماره موبایل ضروری بودند ۷ رقم دیگر داریم که دلخواه هستند. این عبارت منظم این‌گونه پیاده سازی می‌شود: `^09(1[0-9]|3[1-9]|2[1-9])-[0-9]{7}$`

(لازم به ذکر است که این عبارات منظم استاندارد شده باید با کاراکتر `^` آغاز شده و با کاراکتر `$` پایان یابند.)

در قسمت دوم این حاشیه نویسی (ErrorMessage) اگر کاربر رشته مورد نظر را طوری که ما خواسته بودیم وارد نکند با استفاده از توابع `jquery` ساخت‌یافته خود MVC پس از رفتن به فیلدهای دیگر به کاربر خطا را نشان می‌دهد تا شماره موبایل را درست وارد کند.

لازم به ذکر است تمام این حاشیه نویسی‌ها که کار ما را در بخش `View` بسیار ساده‌تر می‌کنند از کتابخانه‌ی `System.ComponentModel.DataAnnotations` استفاده می‌کنند که در همان ابتدا آن را به کلاس‌ها افزوده‌ایم. هم‌چنین به علت پشتیبانی نشدن زبان فارسی در `Encoding` پیش‌فرض `Visual Studio` باید این فایل را پس از انجام این تغییرات با `Encoding` زیر ذخیره نمود:

Unicode (UTF-8 without signature) - Codepage 65001

در این قسمت می‌توانیم برای کلاس صفات دیگری (به جز فیلدهای جدول پایگاه داده) را نیز تعریف کنیم تا در پیاده‌سازی به ما کمک کنند. برای مثال در کلاس `Appointment.cs` صفت زیر را تعریف کردیم تا بعداً در `View` بتوانیم به راحتی از تقویم فارسی استفاده نماییم:

```
[Display(Name = "تاریخ")]  
public DateTime myDate { get; set; }
```

بعداً در کنترلر می‌توانیم روی چنین صفاتی اعمال مختلف انجام دهیم تا آنها را به شکلی که می‌خواهیم در یکی (یا چند تا) از فیلدهای پایگاه داده ذخیره کنیم.

نکات کافی در مورد این نوع از مدل‌ها گفته شد. حال به بررسی دیگر مدل‌ها خواهیم پرداخت:

۴.۲.۲: مدل های اصلی

۱. مدل حساب کاربری (Account Model)

این مدل شامل تمامی اطلاعات مربوط به حساب کاربری است. در این مدل از اطلاعات مدل `User.cs` استفاده کرده ایم و با استفاده از متدهایی نظیر `findAll` (که برای تایید کاربران توسط مدیر گروه مورد استفاده قرار می گیرد) سعی در راحت تر کردن کار `AccountController` داشته ایم.

کلاس `AccountModel` مانند یک `Interface` عمل می کند چون تابع `Constructor` آن خالی است و از تعدادی متد به شرح زیر تشکیل شده است:

- `public User find(string email)`: این تابع وظیفه یافتن کاربران بر اساس ایمیل آنها را بر عهده دارد.

- `public User findById(int id)`: این تابع وظیفه یافتن کاربران بر اساس `id` آنها را بر عهده دارد.

- `public User signIn(string email, string fullname, string password)`

این تابع دو متغیر وارد شده در صفحه `SignIn` را می گیرد و بررسی می کند که آیا کاربری با این مشخصات وجود دارد یا نه. در فیلد اول صفحه `SignIn` کاربر می تواند ایمیل یا نام خود را وارد کند که در صورت برابری با رمز عبور وارد شده در رکورد جدول `Users` در پایگاه داده می تواند وارد شود. با استفاده از یک `Query` ساده ی `Entity Framework` می توانیم این کار را انجام دهیم:

```
return db.Users.Where(acc => (acc.Email.Equals(email) || acc.FullName.Equals(fullname)) && acc.Password.Equals(password)).FirstOrDefault();
```

- `public User signup(User user)`: این تابع کاربر را به سیستم اضافه کرده و در پایگاه داده ذخیره می کند.

- `public User EditUser(User user)`: در این تابع ویژگی هایی را که کاربر تغییر داده را در پایگاه داده تغییر می دهیم و در آن ذخیره می کنیم.

- `public User DeleteUser(User user)`: در این تابع کل یک سطر مربوط به یک کاربر را از پایگاه داده حذف می کنیم و وضعیت پایگاه داده را ذخیره می کنیم.

- `public List<User> findAll()`: این تابع لیست تمام کاربران را برمی گرداند.

- `public List<City> GetCityList()` : این تابع با یک Query ساده نام شهرها را از جدول Cities به صورت لیست برمی گرداند. از این تابع مثلا در صفحه SignUp استفاده می کنیم تا لیست شهرها را در قالب یک DropDown List به کاربر نمایش دهیم.
- `public List<UserRolePersian> GetUserRolesList()` : این تابع لیست تمام نقش های مجاز کاربران را برمی گرداند. بدیهی است که نباید نقش مدیر هنگام ثبت نام به کاربری نشان داده شود.

II. مدل HourMinute

این مدل با داشتن تنها یک Attribute به نام `public TimeSpan hm {set; get;}` دارد و از آن تنها در کنترلر داشبورد JobCorp و در اکشن `AddWorkingTime` که برای ذخیره ی دسته ای زمان های کاری صاحبان مشاغل و همکاران به کار می رود استفاده شده است تا بتوانیم لیستی از زمان ها را در اختیار داشته باشیم. لازم به ذکر است نوع `TimeSpan` برای ذخیره زمان به کار برده می شود. مثلا رشته `23:30` را گرفته و آن را به ساعت و دقیقه دلخواه .NET ترجمه می کند تا بعدا در نوع `DateTime` نیز به راحتی بتوانیم آن را به تاریخ مورد نظرمون بیفزاییم.

III. مدل ClockRangeModel

این مدل با داشتن تنها دو Attribute ساعت شروع و ساعت پایان، برای ذخیره ی زمان هایی که صاحب شغل یا همکار به صورت دسته ای برای زمان های کاری خود وارد می کند به کار می رود. به این صورت که در مدل اصلی `WorkingTime` یک لیست از این مدل تعریف کرده ایم تا بازه های زمانی که کاربر تعریف می کند را داخل این لیست نگه داریم و سپس در کنترلر آن را به تاریخ (یا تاریخ های) انتخاب شده الحاق کرده و به صورت استاندارد خودمان در پایگاه داده آن را ذخیره می کنیم.

IV. مدل LittleWorkingTime

این مدل همانطور که از نامش برمی آید زیرمجموعه ای از مدل `WorkingTime` است که جزو مدل هایی است که Entity Framework از روی پایگاه داده ساخته است. از این مدل زمانی استفاده می شود که کاربر وارد صفحه رزرو وقت از یک صاحب شغل بشود. وقتی کاربر تاریخ مورد نظرش را انتخاب می کند از طریق JSON تاریخ را گرفته و پردازش می کنیم تا زمان های کاری ای که صاحب شغل در آن روز خاص ثبت کرده به کاربر نشان

دهیم. هم‌چنین لیست دیگر ملاقات‌های آن روز را به کاربر نشان می‌دهیم تا زمانی را انتخاب نکند که با بقیه زمان‌ها تداخل داشته باشد.

V. مدل LittleJob

این مدل همانطور که از نامش برمی‌آید زیرمجموعه‌ای از مدل Job است که جزو مدل‌هایی است که Entity Framework از روی پایگاه داده ساخته است. از این مدل زمانی استفاده می‌شود که موقع ثبت نام کاربر نقش خود را همکار انتخاب می‌کند. سپس ما او را به صفحه‌ای هدایت می‌کنیم که او کدی که صاحب شغل به او داده را وارد کند. در زیر این فیلد متنی دکمه‌ای به نام "بررسی" گذاشته‌ایم که کاربر با فشردن آن کد شغلی را به‌صورت JSON و با استفاده از AJAX به سمت سرور می‌فرستد. سرور هم با استفاده از JSON شیء LittleJob را که اطلاعات حیاتی شغلی که EnrollmentKey آن برابر با کد ارسالی از طرف کاربر باشد را روی صفحه نمایش می‌دهد. اگر هم که کد وارد شده اشتباه باشد به کاربر اطلاع می‌دهیم. در صورت مطابقت EnrollmentKey با کد وارد شده کاربر می‌تواند به عنوان همکار آن شغل در سیستم ثبت نام نماید.

VI. مدل MyUserRole

این مدل حاوی یک Enum است که نقش‌های کاربران در پایگاه داده را در آن تعریف کرده و به هریک عددی اختصاص داده ایم. با توجه به این که در پایگاه داده به همین صورت (انگلیسی) نام این نقش‌ها را ذخیره کرده‌ایم، این مدل را صرفاً برای ایجاد ارتباط بین پایگاه داده و View کاربر تعریف کرده ایم و با استفاده از مدل بعدی این اعداد که به هر نقش تخصیص داده شده را به اسامی فارسی تخصیص می‌دهیم. در View با استفاده از یک Cast این تبدیل را انجام می‌دهیم و در DropDown List نقش کاربر (در صفحه ثبت نام) نقش فارسی را به کاربر نمایش می‌دهیم که البته با گذاشتن یک شرط در کنترلر نقش مدیر یا Admin را از اسامی نمایش داده شده خارج می‌کنیم. Enum داخل این مدل به‌صورت زیر تعریف می‌شود:

```
public enum MyUserRole
{
    User = 1,
    JobOwner = 2,
    JobCorp = 3,
    Admin = 4
}
```

۷.۷. UserRole_Persian مدل

این مدل را در بالا توضیح دادیم. Enum داخل این مدل به صورت زیر تعریف می شود:

```
public enum UserRole_Persian
{
    کاربر = 1,
    کارفرما = 2,
    همکار = 3,
    مدیر = 4
}
```

در این قسمت، مدل های سیستم را یک بررسی اجمالی نمودیم. مدل ها قلب تپنده ی پروژه هستند. حال نوبت بخش کنترلر است که وظیفه ی اصلی آن ایجاد ارتباط بین بخش مدل و View می باشد:

۴.۳: بخش کنترلرها

حال به بررسی کنترلرها می پردازیم:

۴.۳.۱: Account Controller

۱. اکشن SignUp:

در قسمت GET این اکشن لیست شهرها، لیست انواع شغل ها و لیست نقش های کاربر را در ViewBag های مربوطه می گذاریم تا در View بتوانیم آن ها را فراخوانی کنیم. در قسمت POST این اکشن نیز اطلاعات را در جدول Users پایگاه داده ذخیره می کنیم و سپس با توجه به نقش انتخابی کاربر او را به صفحه بعدی هدایت می کنیم.

۲. اکشن SignIn:

در قسمت POST این اکشن ابتدا چک میکنیم که اطلاعات با هم مطابقت داشته باشند و در صورت مطابقت چند شرط را چک میکنیم تا مسیر هدایت کاربر را مشخص کنیم. ابتدا با استفاده از Session نقش کاربر، نقشش را می فهمیم. سپس با استفاده از Query های زیر و استفاده کردن از آن ها در شرط ها کاربر را به صفحاتی هدایت می کنیم که بهترین صفحه ی پیشنهادی سیستم برای او هستند. مثلاً اگر یک JobCorp هنوز سرویسی اضافه نکرده باشد او را به کنترلر Services اکشن Create هدایت می کنیم تا

برای خود سرویسی بسازد تا مشتری بتواند از او وقت بگیرد. یا مثلاً کاربر عادی اگر قبلاً وقتی گرفته باشد او را به صفحه‌ی Reservations از کنترلر Main می‌فرستیم تا ببیند چه وقت‌هایی را رزرو کرده است. اگر هنوز وقتی نگرفته باشد، او را به صفحه جستجوی مشاغل (Main/Jobs) هدایت می‌کنیم.

```
var hasReservationsToday = db.Appointments.Any(p => p.UserId == SessionPersister.UserId  
&& p.StartTime >= DateTime.Now && p.StartTime <= dt);
```

```
var hasReservations = db.Appointments.Any(p => p.UserId == SessionPersister.UserId);
```

```
var hasAppointmentsToday = db.Appointments.Any(a => a.Service.JobCorp.UserId ==  
SessionPersister.UserId && a.StartTime >= DateTime.Now && a.StartTime <= dt);
```

```
var hasAppointments = db.Appointments.Any(p => p.Service.JobCorp.UserId ==  
SessionPersister.UserId);
```

```
var hasServices = db.Services.Any(p => p.JobCorp.UserId == SessionPersister.UserId);
```

```
var hasWorkingTimes = db.WorkingTimes.Any(p=>p.JobCorp.UserId==SessionPersister.UserId);
```

نام متغیرها برای درک کار آن‌ها کافی است. لازم به ذکر است دو Query اول برای چک کردن داشتن وقت ملاقات‌های کاربر عادی و چهار Query بعدی برای کارهای مختلف JobCorp‌ها طراحی شده‌اند.

III. اکشن SignOut

در این اکشن Session‌ها را برابر Null می‌کنیم و سپس دوباره کاربر را به صفحه SignIn می‌فرستیم.

IV. اکشن Edit کاربر:

بسیار شبیه به اکشن SignUp عمل می‌کند. با این تفاوت که در متد POST آن به‌جای ذخیره کردن یک رکورد در پایگاه داده رکورد مربوط به کاربر مورد نظر را تغییر می‌دهیم.

Home Controller :۴.۳.۲

این کنترلر برای صفحات اصلی وبسایت که در همه وبسایت‌ها موجود است **View** متناظرشان را برمی‌گرداند. کاربری که برای بار اول به وبسایت بیاید به صفحه **Index** این کنترلر هدایت خواهد شد. لازم به ذکر است در اکشن‌های این کنترلر به جز برگرداندن **View** متناظر کار دیگری صورت نمی‌گیرد.

۱. اکشن **Index** (صفحه اصلی وبسایت)

۲. اکشن **About**

۳. اکشن **Contact**

Job Controller :۴.۳.۳

این کنترلر را به صورت مستقیم و از طریق **Scaffolding** مدل **Job** که از جدول **Jobs** در پایگاه داده درست شده بود ساختیم. چون اکشن‌های این کنترلر همان **CRUD** هست دیگر آن‌ها را لازم به توضیح بیشتر نمی‌بینیم. ضمناً لازم به ذکر است که تمامی اکشن‌های این کنترلر تنها برای صاحب شغل قابل دسترسی بوده و دسترسی بقیه نقش‌ها به اکشن‌های این کنترلر مسدود شده است.

۱. اکشن **Index**

۲. اکشن **Create**

۳. اکشن **Edit**

۴. اکشن **Delete**

۷. اکشن **JobCorpsList**

در داخل یک **ViewBag** لیست همکاران شغل مورد نظر را می‌ریزیم و به **View** می‌فرستیم. لازم به ذکر است که در این اکشن هیچ کاربر دیگری به جز صاحبان مشاغل حق ورود ندارد و در همان ابتدای اکشن آن‌ها را به اکشن مناسبی هدایت می‌کنیم.

۶. اکشن **DeleteJobCorp**

در صورت انتخاب شدن یک همکار در View بالا و زدن دکمه حذف همکار، Id همکار را به این اکشن می‌فرستیم و سپس با توجه به id رکورد او را از جدول JobCorps در پایگاه داده حذف می‌کنیم و نقش کاربری او را هم به کاربر عادی تغییر می‌دهیم. البته عملیات تغییر در پایگاه داده پس از تایید کاربر انجام می‌شود، یعنی پس از اکشن بعدی!

VII. اکشن DeleteJobCorpConfirmed

همانطور که توضیح داده شد در این اکشن رکورد این همکار را از جدول JobCorps در پایگاه داده حذف می‌کنیم و نقش کاربری او را هم به کاربر عادی تغییر می‌دهیم.

۴.۳.۴: Service Controller

این کنترلر را نیز به صورت مستقیم و از طریق Scaffolding مدل Service که از جدول Services در پایگاه داده درست شده بود ساختیم. چون اکشن‌های این کنترلر همان CRUD هست دیگر آن‌ها را لازم به توضیح بیشتر نمی‌بینیم.

I. اکشن Index

II. اکشن Create

III. اکشن Edit

IV. اکشن Delete

۴.۳.۵: JCDashboard Controller

نام این کنترلر مخفف JobCorp Dashboard است. در واقع چون بیشتر کارهایی که همکار و صاحب شغل انجام می‌دهند یکسان است، لذا تصمیم گرفتیم برای صرفه‌جویی در زمان و رعایت مولفه‌های مهندسی نرم‌افزار فعالیت‌های مشترک این دو نقش را در این کنترلر پیاده‌سازی نماییم.

اکشن‌های این کنترلر عبارتند از:

IV. اکشن Index:

در این اکشن قرار ملاقات‌های صاحب کار (یا همکار) را از پایگاه داده خوانده و به View متناظر می‌فرستیم.

V. اکشن EnrollJob:

وقتی کاربری در صفحه ثبت نام گزینه همکار را انتخاب می‌کند او را به این View هدایت خواهیم کرد. در این اکشن اگر کاربر کد صحیحی وارد کند او را به عنوان همکار به شغل مربوطه اضافه خواهیم کرد و سپس او را به اکشن AddWorkingTime از همین کنترلر هدایت می‌کنیم.

VI. اکشن AddWorkingTime:

در قسمت HttpGet این اکشن زمان‌های کاری جاری کاربر را از پایگاه داده می‌خوانیم و به View می‌فرستیم. و در قسمت HttpPost این اکشن سعی در ذخیره کردن زمان‌های انتخابی کاربر داریم. هسته این اکشن از یک حلقه for تشکیل شده است. با توجه به این که در View از کاربر روزهای هفته به صورت دسته‌ای دریافت می‌شوند و بعد از آن کاربر امکان وارد کردن ساعت شروع و پایان روزهای کاری را نیز به صورت دسته‌ای دارد، باید در داخل این اکشن با ساز و کاری اطلاعات دریافتی را مدیریت کنیم. این حلقه به این صورت پیاده‌سازی می‌شود:

```
for (int i = 0; i < 31; i++)
{
    t = DateTime.Now;
    t = t.AddDays(i);
    DoW = (int)t.DayOfWeek;
    if (WeekDayId.Contains(DoW))
    {
        if (i == 0)
        {
            if (ts <= DateTime.Now.TimeOfDay)
            {
                t = t.AddDays(7);
                for (int j = 0; j < wd.Range.Count; j++)
                {
                    stimestlist.Add(new HourMinute{
                        hm = TimeSpan.Parse(wd.Range[j].StartHour) });
                    etimestlist.Add(new HourMinute{
```

```

        hm = TimeSpan.Parse(wd.Range[j].EndHour) });
list.Add(new WorkingTime{
    JobCorpId = jc.Id,
    StartTime = t.Date + stimelist[j].hm,
    EndTime = t.Date + etimelist[j].hm });
    }
    continue;
}
else //when the start hour is valid
{
    for (int j = 0; j < wd.Range.Count; j++)
    {
        stimelist.Add(new HourMinute{
            hm = TimeSpan.Parse(wd.Range[j].StartHour) });
        etimelist.Add(new HourMinute{
            hm = TimeSpan.Parse(wd.Range[j].EndHour) });
        list.Add(new WorkingTime{
            JobCorpId = jc.Id,
            StartTime = t.Date + stimelist[j].hm,
            EndTime = t.Date + etimelist[j].hm });
    }
}

}
if ((i == 7) && (ts <= DateTime.Now.TimeOfDay)) {
    continue;
}
if ((i == 0) && (ts >= DateTime.Now.TimeOfDay)) {
    continue;
}
for (int j=0; j<wd.Range.Count; j++){ //****
    stimelist.Add(new HourMinute{
        hm = TimeSpan.Parse(wd.Range[j].StartHour) });
    etimelist.Add(new HourMinute{
        hm = TimeSpan.Parse(wd.Range[j].EndHour) });
    list.Add(new WorkingTime{
        JobCorpId = jc.Id,
        StartTime = t.Date + stimelist[j].hm,
        EndTime = t.Date + etimelist[j].hm });
}
}
}
}

```

حال به شرح این قطعه کد کلیدی می‌پردازیم. ابتدا زمان حال را در متغیر `t` که از نوع `DateTime` است قرار می‌دهیم. سپس (در هر بار تکرار حلقه) `i` روز به `t` اضافه می‌کنیم. در متغیر `DoW` که مخفف روزهای هفته می‌باشد شکل عددی روز هفته متغیر `t` را قرار می‌دهیم. در اولین شرط خود چک

می‌کنیم که آیا DoW در این تکرار از حلقه ددر آرایه‌ای که از سمت View به کنترلر پاس داده شده (آرایه‌ای شامل اعداد ۰ تا ۶ که نشانگر روزهای هفته هستند و کاربر انتخاب کرده است) وجود دارد یا نه. حال استثناها را بررسی می‌کنیم. اگر i با صفر برابر باشد بدین معنی است که همین امروز هم جزو انتخاب‌های روز هفته‌ی کاربر می‌باشد. سپس اگر ساعت انتخابی کاربر قبل از ساعت فعلی امروز باشد ۷ روز به t اضافه می‌کنیم و سپس عمل ذخیره را در list خود که از نوع WorkingTime می‌باشد، انجام می‌دهیم و با دستور continue حلقه را از ابتدا فراخوانی می‌کنیم. در غیر این صورت (اگر ساعت انتخابی کاربر بعد از ساعت فعلی امروز باشد) بدون مشکل t را ذخیره می‌کنیم. سپس با دو شرط چک می‌کنیم اگر قبلاً این دو شرط را اعمال کرده باشیم با دستور continue به ابتدای حلقه برویم تا دیگر نگران افزونگی داده در پایگاه داده نباشیم.

در نهایت به for آخر می‌رسیم که با **** مشخص شده است. این حلقه کار اصلی ذخیره را بر عهده دارد. پس آن را با دقت بیشتری بررسی می‌کنیم. در شرط wd.Range.Count < j در داخل for چک می‌کنیم که تعداد تکرارهای حلقه بیش از تعداد بازه‌های زمانی که کاربر وارد کرده نشود. لیست stimelist شامل زمان‌هایی خواهد بود که زمان شروع انتخابی توسط کاربر هستند. همینطور لیست etimelist هم شامل زمان‌های پایان انتخابی توسط کاربر می‌باشد. هم‌چنین قبلاً متذکر شدیم که لیست list شامل یک لیست از نوع WorkingTime می‌باشد. (در نهایت همین لیست را به پایگاه داده ارسال خواهیم کرد تا به صورت دسته‌ای ذخیره شود). با توجه به جدول WorkingTime در پایگاه داده برای هر رکورد در لیست list متغیرهای JobCorpld، StartTime و EndTime را ذخیره می‌کنیم. برای مثال با دستور StartTime = t.Date + stimelist[j].hm تاریخ t را به زمان شروع شماره j انتخابی کاربر الحاق می‌کنیم.

VII. اکشن AddWorkingDate:

در قسمت HttpGet این اکشن زمان‌های کاری جاری کاربر را از پایگاه داده می‌خوانیم و به View می‌فرستیم. و در قسمت HttpPost این اکشن سعی در ذخیره کردن زمان‌های انتخابی کاربر داریم. این اکشن را کاربر زمانی استفاده می‌کند که به جای انتخاب زمان کاری دسته‌ای بخواهد تنها یک روز را به زمان‌های کاری‌اش بیفزاید. بنابراین در View یک تقویم فارسی به کاربر نشان می‌دهیم و پس از این که کاربر اطلاعات را به سمت سرور فرستاد سعی می‌کنیم آن روز را به ساعت ابتدا و انتهای انتخاب شده توسط

کاربر الحاق کنیم. دقیقاً همانطور که در اکشن قبلی این کار را کردیم با این تفاوت که دیگر لازم نیست از لیستی استفاده کنیم و تنها با چند متغیر کار خود را پیش می‌بریم:

```
wd.JobCorpId = jc.Id;  
TimeSpan ts = TimeSpan.Parse(wd.start);  
wd.StartTime = wd.myDate.Date + ts;  
TimeSpan es = TimeSpan.Parse(wd.end);  
wd.EndTime = wd.myDate.Date + es;
```

wd همان مخفف **WorkingDate** است و قرار است آن را در پایگاه داده ذخیره کنیم. ابتدا **JobCorpId** آن را ست می‌کنیم. در متغیرهای **ts** و **es** به ترتیب زمان انتخابی ابتدایی و انتهایی توسط کاربر را قرار می‌دهیم و سپس با الحاق تاریخ انتخاب شده توسط کاربر با این دو متغیر آن‌ها را در **wd.StartTime** و **wd.EndTime** ذخیره می‌کنیم تا بتوانیم موجودیت **wd** را در قالب یک رکورد در پایگاه داده ذخیره کنیم.

VIII. اکشن **ModifyWorkingDate**:

در این اکشن کلیه‌ی زمان‌های کاری را برای صاحب شغل (یا همکار) لیست می‌کنیم و به **View** می‌فرستیم. در **View** جلوی هر رکورد از زمان‌های کاری یک دکمه حذف قرار داده ایم که کاربر با فشردن آن، رکورد مورد نظر را پاک می‌کند. بدین ترتیب با **View**های **AddWorkingDate** و **ModifyWorkingDate** نیز کاربر می‌تواند استثناها را نیز مدیریت کند.

IX. اکشن **AppointmentList**:

در این اکشن لیست تمامی قرار ملاقات‌ها را برای صاحب شغل یا همکار از پایگاه داده می‌گیریم و سپس به **View** ارسال می‌کنیم.

X. اکشن **ExpiredAppointmentList**:

مانند **View**ی بالاست با این تفاوت که قرار ملاقات‌های قبل از تاریخ فعلی را لیست می‌کند.

XI. اکشن **FutureAppointmentList**:

مانند **View**ی بالاست با این تفاوت که قرار ملاقات‌های بعد از تاریخ فعلی را لیست می‌کند یعنی قرار ملاقات‌های آینده را به کاربر نشان می‌دهد.

XII. اکشن AppointmentDetails:

جزئیات هر قرار ملاقات اعم از نام طرف ملاقات کننده، شماره تلفن، زمان ملاقات و ... در اینجا به View فرستاده می شود و همچنین در View نیز دکمه کنسل ملاقات کاربر را به View بعدی منتقل می کند.

XIII. اکشن AppointmentCancellation:

در کنسل قرار ملاقات ها آن رکورد را از پایگاه داده حذف نمی کنیم، بلکه بیت isReserved را برابر صفر می گذاریم تا در آینده به کاربر و صاحب شغل آن را طور دیگری نشان دهیم. همچنین در View بعد از تایید کنسل کردن قرار ملاقات به اکشن زیر می رویم تا عملیات تغییر در پایگاه داده را در آن انجام دهیم.

XIV. اکشن AppointmentCancellationConfirmed:

همان طور که توضیح داده شد در این اکشن بیت isReserved را در رکورد مربوطه اش صفر می کند و در پایگاه داده ذخیره می کند. سپس کاربر را به صفحه لیست قرارهای کاری اش می فرستد.

۴.۳.۶: Main Controller

این کنترلر مسئول کنترل نیازهای کاربر عادی است. نیازهایی مانند رزرو وقت، جستجوی مشاغل و مشاهده لیست Reservation برای کاربر از جمله نیازهایی هستند که در این کنترلر پیاده سازی شده اند.

اکشن های این کنترلر عبارتند از:

I. اکشن Index:

در این اکشن صفحه ی جستجوی مشاغل را به کاربر نشان می دهیم.

II. اکشن Jobs

این اکشن هم صفحه‌ی جستجوی مشاغل را به کاربر نشان می‌دهد با این تفاوت که می‌توان با لینک دادن هم جستجو را انجام داد. برای مثال در صفحه **Index** از کنترلر **Home** لینک نوع شغل‌های مختلف گذاشته شده و با فشردن آن لینک، کاربر مستقیماً به صفحه **Jobs** منتقل شده و تنها نوع شغل مورد نظر خود را خواهد دید. **Query** یا همان فیلتر مورد استفاده در کنترلر این‌گونه پیاده‌سازی شده است:

```
Query = db.Jobs.Where(m=>
(string.IsNullOrEmpty(title) ? true : m.Title.Contains(title)) &&
(string.IsNullOrEmpty(city) ? true : m.City.Name == city) &&
(string.IsNullOrEmpty(jobtype) ? true : m.JobType.Title == jobtype));
```

III. اکشن **JobDetails**

کاربر می‌تواند جزئیات مربوط به هر کدام از مشاغلی که انتخاب می‌کند را مشاهده نماید.

IV. اکشن **JobCorpsList**

کاربر پس از کلیک بر روی شغل مورد نظر که از قسمت جستجو یافته است به این صفحه هدایت می‌شود که لیست همکاران شغل مورد نظر در آن نمایش داده می‌شود. کاربر با کلیک بر روی هر کدام از همکاران به صفحه رزرو وقت از او وارد می‌شود. هم‌چنین در این صفحه لینکی به اکشن **JobDetails** وجود دارد.

V. اکشن **Reserve**

در متد **GET** این اکشن با توجه به **JobCorpId** همکار انتخابی لیست سرویس‌ها و زمان‌های کاری او را به **View** می‌فرستیم. در متد **POST** با توجه به تاریخ انتخابی کاربر باید تشخیص دهیم که زمان انتخاب شده توسط او در بازه‌ی کاری همکار هست یا خیر و هم‌چنین اگر کسی وقت دیگری در آن زمان گرفته باشد با پیغام خطا کاربر را مطلع سازیم. **Query** های زیر این کارها را انجام می‌دهند:

```
var isInWorkingTimes = db.WorkingTimes.Any(p => (p.StartTime <= ap.StartTime) &&
(p.EndTime >= ap.EndTime)&&(ap.StartTime >= DateTime.Now)&&(p.JobCorpId == s.JobCorpId));

var isInOtherAppointments = db.Appointments.Any(m => (ap.StartTime >= DateTime.Now) &&
(m.Service.JobCorpId==s.JobCorpId)&&(m.StartTime<ap.EndTime)&&(m.EndTime>ap.StartTime));
```

VI. اکشن **ChooseReserve**

برخی سختی‌ها در پیاده‌سازی اکشن بالا باعث شد این اکشن را برای رزرو کاربر استفاده کنیم. در اکشن بالا تاریخ شمسی را با استفاده از **JsDatePicker** از کاربر می‌گرفتیم. ولی با توجه به این که کاربر می‌خواهد

تنها روزهای کاری همکار مورد نظر را ببیند تصمیم گرفتیم به جای نشان دادن تقویم به کاربر تنها روزهایی که او کار می‌کند را به کاربر نشان دهیم. هم‌چنین پس از انتخاب تاریخ (با استفاده از DropDownList) به کاربر با استفاده از Ajax (اکشن JsonInfo) مشخصات شغل و هم‌چنین زمان دقیق کاری در آن روز توسط همکار را نشان می‌دهیم. هم‌چنین اگر کس دیگری در آن زمان وقت گرفته باشد به کاربر نشان می‌دهیم تا بداند که نمی‌تواند در آن زمان وقت رزرو کند. Query چک کردن islnWorkingTimes و islnOtherAppointments را هم که در اکشن بالا تعریف کردیم، دوباره اینجا مورد استفاده قرار داده‌ایم.

VII. اکشن JsonInfo

در View و در قسمت جاوا اسکریپت با استفاده از کد زیر این اکشن را فراخوانی می‌کنیم:

```
$(document).ready(function () {
    var url = '@Url.Action("JsonInfo")';
    $('#selDate').change(function () {
        $.getJSON(url, { wtid: $('#selDate').val() }, function (response) {
            if (response == false) {
                $(".ajaxRes").html("<span style='color: red;'>زمان کاری وجود ندارد</span>");
            }
            else{
                $(".ajaxRes").html(response.dow + "<span> از </span>" +
                    response.StartTime + "<span> تا </span>" +
                    response.EndTime + "<br/>" + response.JobTitle +
                    "<br/>" + response.JobCorp);

                if (response.ap.length > 0) {
                    $(".ajaxRes").append("<hr><span>دیگر قرارهای این روز:</span><br/>");
                }
                for (var j = 0; j < response.ap.length ; j++) {
                    $(".ajaxRes").append(response.ap[j].StartHour + " تا " +
                        response.ap[j].EndHour + "<br/>");
                }
            }
        });
    });
});
```

حال wtid که همان روز انتخابی کاربر (WorkingTime همکار مورد نظر) است را به JsonInfo می‌فرستیم و با استفاده از مدل LittleWorkingTime که قبلاً در قسمت مدل‌ها توضیح دادیم، اشیائی که می‌خواهیم با

استفاده از AJAX به View بفرستیم را مقداردهی می‌کنیم. لازم به ذکر است لیست ap را که دربرگیرنده قرار ملاقات‌های JobCorp در زمان کاری مورد نظر است را با استفاده از Query زیر به دست می‌آوریم:

```
lwt.ap = (from s in db.Services
         from a in db.Appointments
         from w in db.WorkingTimes
         where w.Id == wt.Id && w.StartTime <= a.StartTime && w.EndTime >= a.EndTime
            && w.JobCorpId == s.JobCorpId && s.Id == a.ServiceId
         select new Item{ StartTime = a.StartTime, EndTime = a.EndTime }).ToList();
```

حال باید اشیاء را با استفاده از AJAX به View بفرستیم. پس شیء lwt را که یک instance از مدل LittleWorkingTime است را به عنوان شیء ارسالی می‌فرستیم. دستور مورد نظر Json را به View ارسال می‌کند:

```
return Json(lwt, JsonRequestBehavior.AllowGet);
```

VIII. اکشن ReserveEdit:

در متد GET این اکشن لیست سرویس‌ها و زمان‌های کاری همکار مورد نظر را لود کرده و به View می‌فرستیم. در متد POST نیز اطلاعات ویرایش شده را می‌خوانیم و سپس در پایگاه داده آن‌ها را تغییر می‌دهیم.

IX. اکشن ReserveDetails:

جزئیات هر قرار ملاقات اعم از عنوان شغل، شماره تلفن شغل، مشخصات همکاری که از آن وقت گرفته شده است، زمان ملاقات و ... در اینجا به View فرستاده می‌شود و هم‌چنین در View نیز دکمه کنسل ملاقات کاربر را به View بعدی منتقل می‌کند.

X. اکشن ReserveCancellation:

در کنسل قرار ملاقات‌ها آن رکورد را از پایگاه داده حذف نمی‌کنیم، بلکه بیت isReserved را برابر صفر می‌گذاریم تا در آینده به کاربر و صاحب شغل آن را طور دیگری نشان دهیم. هم‌چنین در View بعد از تایید کنسل کردن قرار ملاقات به اکشن زیر می‌رویم تا عملیات تغییر در پایگاه داده را در آن انجام دهیم.

.XI اکشن AppointmentCancellationConfirmed:

همان‌طور که توضیح داده شد در این اکشن بیت `isReserved` را در رکورد مربوطه‌اش صفر می‌کند و در پایگاه داده ذخیره می‌کند. سپس کاربر را به صفحه لیست قرارهایش می‌فرستد.

.XII اکشن Reservations:

در این اکشن لیست تمامی قرارهای کاربر را از پایگاه داده می‌گیریم و سپس به `View` ارسال می‌کنیم.

.XIII اکشن ExpiredAppointmentList:

مانند `View`ی بالاست با این تفاوت که قرارهای قبل از تاریخ فعلی را لیست می‌کند.

.XIV اکشن FutureAppointmentList:

مانند `View`ی بالاست با این تفاوت که قرارهای بعد از تاریخ فعلی را لیست می‌کند یعنی قرارهای آینده را به کاربر نشان می‌دهد.

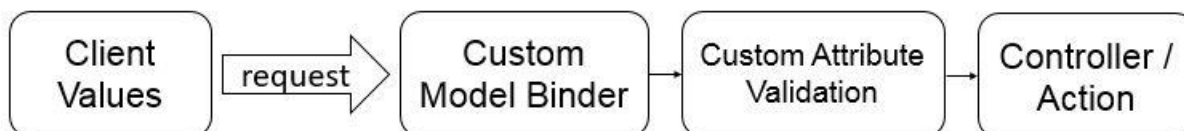
۴.۴: برخی پیاده‌سازی‌های کاربردی سیستم

۴.۴.۱: نحوه تبدیل تاریخ شمسی به میلادی در سیستم

-اطلاعات از سمت کاربر چگونه به سرور می‌رسد؟

زمانی که یک کاربر مقداری را به سمت سرور می‌فرستد با استفاده از یک `request` آن را به سرور می‌فرستد (با استفاده از متدهایی مانند `POST`، `GET` و `JSON`) مقدار وارد شده به `Custom Model Binder` می‌رود. وظیفه `Custom Model Binder` تبدیل مقدار داده شده به اشیائی است که در `.NET` تعریف شده اند می‌باشد. پس در همین مرحله باید تبدیل به تاریخ میلادی را انجام دهیم. پس از این مرحله نوبت به

Custom Attribute Validation است که کار اعتبارسنجی اطلاعات را انجام می‌دهد و در صورت درست بودن اطلاعات (true شدن مقدار ModelState.IsValid) اطلاعات را به سمت کنترلر و اکشن مربوطه هدایت می‌کند. لازم به ذکر است Custom Attribute Validation هر دو Validation در سمت Client و Server را انجام می‌دهد. مثلاً استفاده از اعتبارسنجی Regular Expression ها در سمت Client و اعتبارسنجی مقادیری که به سرور فرستاده می‌شوند توسط خود Server انجام می‌شوند.



برای توضیح دادن این تبدیل مثال کلی زیر را در نظر بگیرید:

در یکی از مدل‌های خود مثلاً WorkingTime یک متغیر DateTime به صورت زیر تعریف می‌کنیم:

```
public DateTime myDate { get; set; }
```

برای میلادی کردن تاریخ شمسی باید از Custom Model Binder استفاده کنیم. برای نوشتن آن یک پوشه جدید در پروژه به نام Utilities ایجاد کرده و سپس در آن پوشه‌ای به نام ModelBinders ایجاد می‌کنیم. سپس کلاس PersianDateTimeModelBinder.cs را در آن ایجاد می‌کنیم. این کلاس حتماً باید از اینترفیس IModelBinder ارث بری کند. در این کلاس مقادیری که کاربر به عنوان تاریخ شمسی وارد کرده را گرفته و سپس چک می‌کنیم اگر مقدار داده شده توسط کاربر اشتباه باشد ModelState.Error را اضافه می‌کنیم تا سرور خطا دهد. در غیر این صورت ModelState.IsValid را برابر با true می‌کنیم. اگر تاریخ وارد شده توسط کاربر مشکلی نداشت آن را به تاریخ میلادی تبدیل می‌کنیم و سپس برمی‌گردانیم. پس از این در پارامتر ورودی اکشن خود تاریخ میلادی نخواهیم داشت و تاریخ به میلادی تبدیل شده است.

کد این کلاس بدین صورت پیاده سازی شده است. لازم به ذکر است در برخی جاها از کلاس Utilities معرفی خواهیم کرد، استفاده کرده‌ایم:

```
public class PersianDateTimeModelBinder : IModelBinder
{
    public object BindModel(ControllerContext controllerContext, ModelBindingContext bc){
        var valueResult = bc.ValueProvider.GetValue(bc.ModelName);
        var modelState = new ModelState { Value = valueResult };
        object actualValue = null;
        try{
```

```

        if (valueResult.AttemptedValue.IsPersianDateTime() == false){
            var metadata = bindingContext.ModelMetadata;
            var displayName = metadata.DisplayName ?? metadata.PropertyName ??
            bc.ModelName.Split('.').Last();
            modelState.Errors.Add(string.Format("{0} را به درستی وارد کنید", displayName));
        }
        else{
            var datetime = Convert.ToDateTime(valueResult.AttemptedValue);
            var miladi = datetime.ToMiladiDateTime();
            actualValue = miladi;
        }
    }
    catch (FormatException e) {
        modelState.Errors.Add(e);
    }
    bindingContext.ModelState.Add(bindingContext.ModelName, modelState);
    return actualValue;
}
}

```

یک کلاس دیگر در پوشه Utilities به نام Utility.cs می‌سازیم. در این کلاس سه متد کاربردی را پیاده سازی کرده‌ایم که تمامی آن‌ها Extension Method هستند (به علت static بودن توابع و استفاده از this در متغیرهای ورودی آن‌ها). یعنی می‌توانیم این متدها را بعد از هر property فراخوانی کنیم و نیازی به new کردن این کلاس نخواهیم داشت. متد اول با استفاده از متد PersianCalender که از توابع .NET است استفاده کرده و تاریخ میلادی را به شمسی تغییر می‌دهد. متد دوم دقیقاً برعکس متد اول است. و متد سوم نیز با استفاده از عبارات منظم تعیین می‌کند که رشته ارسالی تاریخ شمسی معتبری هست یا خیر.

پیاده سازی این توابع به صورت زیر است:

```

public static DateTime ToPersianDateTime(this DateTime datetime) {
    var pc = new PersianCalendar();
    return new DateTime(pc.GetYear(datetime), pc.GetMonth(datetime),
        pc.GetDayOfMonth(datetime), 0, 0, 0);
}

public static DateTime ToMiladiDateTime(this DateTime datetime){
    var pc = new PersianCalendar();
    return pc.ToDateTime(datetime.Year, datetime.Month, datetime.Day, 0, 0, 0);
}

public static bool IsPersianDateTime(this string datetime) {
    return Regex.IsMatch(datetime, @"^(13\d{2})|([1-9]\d)/([1012]|0?[1-9])/([12]\d|3[01]|0?[1-9])$");
}

```

حال که CustomModelBinder خود را نوشته‌ایم باید آن را به پروژه معرفی کنیم. این معرفی را در کلاس Global.asax انجام می‌دهیم. لذا خط زیر را در این کلاس اضافه می‌کنیم تا این ModelBinder به پروژه اضافه گردد:

```
ModelBinders.Binders.Add(typeof(DateTime), new PersianDateTimeModelBinder());
```

با این معرفی هر جا که به سمت سرور نوع DateTime ارسال شود کار اعتبارسنجی آن متغیر را کلاس PersianDateTimeModelBinder انجام می‌دهد.

حال تمام اعتبارسنجی سمت سرور برای تاریخ شمسی را ساختیم. پس از انجام این اعتبارسنجی، در کنترلر و View خود اعمال همیشگی را انجام می‌دهیم. مثلاً می‌توانیم در قسمت JavaScript صفحه خودمان هم یک عبارت منظم برای input تاریخ شمسی در نظر بگیریم تا سمت client هم اعتبارسنجی کنیم. برای زیباسازی بیشتر پروژه یک jquery component تاریخ شمسی را استفاده می‌کنیم تا کاربر بتواند به راحتی تاریخ خود را انتخاب کند. این ابزار را در قسمت ۱.۵.۲ فصل اول معرفی کرده بودیم.

۴.۴.۲: نحوه ست کردن متغیرهای Session توسط SessionPersister

برای دسترسی به برخی متغیرها در صفحات html نیاز به چند Session داریم. مانند نام و نام خانوادگی کاربر که در بالای همه صفحات به کاربر نشان دهیم. همینطور id کاربر و نقش کاربر را هم در متغیرهای Session جداگانه نگهداری می‌کنیم تا در اکشن‌های کنترلرها از آن‌ها استفاده نماییم.

کلاس SessionPersister را در پوشه Security به صورت استاتیک تعریف می‌کنیم. یکی از توابع این کلاس به صورت زیر پیاده‌سازی می‌شود:

```
public static string UserRole
{
    get
    {
        if (HttpContext.Current == null)
            return string.Empty;
        var sessionvar = HttpContext.Current.Session["userRole"];
        if (sessionvar != null)
            return sessionvar as string;
        return null;
    }
    set
```

```

    {
        HttpContext.Current.Session["userRole"] = value;
    }
}

```

که مشخصاً مقدار `SessionPersister.UserRole` را برابر `Session["userRole"]` می‌کند. این تابع را بعدها در کنترلرها استفاده کرده‌ایم. لازم به ذکر است مثلاً در اکشن `SignIn` کاربر اینگونه این `Session` را مقداردهی می‌کنیم:

```
SessionPersister.UserRole = user.UserRole.Name;
```

بعداً به طور مثال در کنترلر `JCDashboard` و در ابتدای متد `GET` اکشن `AddWorkingTime` باید چک کنیم که اگر کاربر، صاحب شغل یا همکار نباشد او را به جای دیگری هدایت کنیم تا به آن صفحه دسترسی نداشته باشد. مانند کد زیر:

```

if (SessionPersister.UserRole.ToString() == "User"){
    return RedirectToAction("Index", "Home");
}

```

کد بالا در صورتی که کاربر عادی بخواهد وارد اکشن `AddWorkingTime` شود او را به صفحه `Index` از کنترلر `Home` هدایت می‌کند. لذا صفحه `AddWorkingTime` هیچ‌گاه برای او لود نخواهد شد.

۴.۵: خلاصه

در این فصل پیاده‌سازی مدل‌ها و کنترلرها را خیلی مبسوط شرح دادیم و همچنین برخی پیاده‌سازی‌های دیگر در پروژه را نیز توضیح دادیم. در این فصل به `View`ها پرداخته نشد چون باید اصل ایجاز را رعایت می‌کردیم. در قسمت ۲.۳ فصل دو که روند کار^{۲۸} را به همراه تصویر صفحات شرح دادیم به `View`ها پرداخته بودیم.

[1] Craig Larman, "Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development", United States of America, Prentice Hall PTR, 2004

[2] Roger S. Pressman, "Software Engineering: A Practitioner's Approach", United States of America, McGraw-Hill Education. 2010.

<https://docs.asp.net>

<https://www.visualstudio.com/en-us/docs/vs/overview>

https://en.wikipedia.org/wiki/Web_server

https://en.wikipedia.org/wiki/Application_server

https://en.wikipedia.org/wiki/Internet_Information_Services

<http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html>

https://en.wikipedia.org/wiki/ASP.NET_MVC

https://en.wikipedia.org/wiki/Entity_Framework

https://en.wikipedia.org/wiki/Integrated_development_environment

https://en.wikipedia.org/wiki/Microsoft_Visual_Studio

<https://www.en.wikipedia.org/wiki/JavaScript>

<https://en.wikipedia.org/wiki/JQuery>

https://en.wikipedia.org/wiki/Font_Awesome

[https://en.wikipedia.org/wiki/Bootstrap_\(front-end_framework\)](https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework))

[https://msdn.microsoft.com/en-us/library/mt590198\(v=sql.1\).aspx](https://msdn.microsoft.com/en-us/library/mt590198(v=sql.1).aspx)

https://en.wikipedia.org/wiki/Microsoft_SQL_Server

https://en.wikipedia.org/wiki/Agile_software_development

https://en.wikipedia.org/wiki/User_story

<https://en.wikipedia.org/wiki/Requirement>

http://www.tutorialspoint.com/uml/uml_use_case_diagram.htm

https://en.wikipedia.org/wiki/Sequence_diagram

<http://agilemodeling.com/artifacts/sequenceDiagram.htm>

<https://www.birij.com/asp-net-mvc/persian-date-time-and-picker-in-asp-mvc-part1>



Zanjan University

Faculty of Engineering

Department of Computer

Business Appointing System

In Partial Fulfillment of the Requirement for the degree of Bachelor of
Science in

Computer Software Engineering

Advisor

Mr. Davoud Mohammadpour

By

Sina Ebrahimi

Winter 2016