

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



دانشگاه آزاد اسلامی

واحد تهران جنوب

دانشکده فنی مهندسی

پایان نامه کارشناسی

مهندسی کامپیوتر - نرم افزار

عنوان :

وب سایت برنامه نویسی

استناد راهنما:

دکتر محمد حمزه ئی

نام و نام خانوادگی دانشجو :

محمد رضا سلیمیان

شماره دانشجویی :

۹۴۱۱۲۱۲۲۲۴

خرداد ماه ۱۳۹۸

فهرست مطالب

شماره صفحه

عنوان مطلب

۱	چکیده
۲	فصل اول
۲	مقدمه شامل توضیح دامنه سیستم
۴	فصل دوم
۴	تحلیل نیازمندی ها
۵	تعریف محدوده سیستم (System Scope)
۵	نیازمندی های کیفیت :
۶	بررسی Use Case Diagram
۲۷	تحلیل ها Sequence Diagram
۲۸	مربوط به کامپایل سورس کد : Sequence Diagram
۲۹	مربوط به اجرای برنامه : Sequence Diagram
۳۹	مربوط به برقراری ارتباط با پایگاه داده : Sequence Diagram
۳۹	مربوط به رمزگذاری و Encryption Sequence Diagram
۴۱	مربوط به دانشجو : Activity Diagram
۴۱	شرح Activity Diagram
۴۳	مربوط به Actor استاد : Activity Diagram
۴۳	شرح Activity Diagram
۴۵	فصل سوم
۴۵	طراحی شامل معماری سیستم مولفه ها
۴۵	و ارتباطات و کلاس دیاگرام
۵۱	مربوط به کلاس Student Class Diagram
۵۱	مربوط به کلاس Professor Class Diagram
۵۲	مربوط به کلاس practice.java Class Diagram
۵۳	مربوط به DAO Class Diagram
۵۴	فصل چهارم
۵۴	پیاده سازی شامل تکنولوژی های مورد استفاده
۵۴	نکات مربوط به پیاده سازی ، تصاویر سیستم
۵۶	طراحی Data Model

چکیده

هدف اصلی وب سایت برنامه نویسی آنلاین ، بالا بردن سطح دانش برنامه نویسی دانشجویان است و برای این مهم ، یک ویرایشگر کد را فراهم کرده است که زبان های برنامه نویسی سی و جاوا و پایتون را پشتیابی می کند و به کاربران این امکان را می دهد که سورس کد های خود را کامپایل و اجرا کنند .

فصل اول

مقدمه شامل توضیح دامنه سیستم

برای ارتباط موثر مهندس نرم افزار با ذینفعان و درک بهتر نیازمندی های مشتریان نیاز داریم دانش خوب و عمیقی از دامنه‌ی سیستمی که قرار است آن را تحلیل، طراحی و پیاده‌سازی کنیم داشته باشیم.

اگر از تحلیلی که روی دامنه انجام می‌دهیم یک خلاصه تهیه کنیم این خلاصه به مهندسین نرم افزاری که بعداً به تیم ما ملحق می‌شوند در درک دامنه‌ی سیستم کمک می‌کند.

دامنه‌ی سیستم شامل مقدمه یا همان نام دامنه، واژگان دامنه، دانش عمومی درباره‌ی دامنه، مشتریان و کاربران می‌باشد

مقدمه : دامنه‌ی وب سایت برنامه نویسی

مهندسين نرم افزاري که در تیم توسعه وظیفه‌ی تحلیل و طراحی و پیازی سازی این سیستم را دارند به خوبی با زبان های برنامه نویسی کامپایلری و مفسری آشنا هستند و درک خوبی از فازها و مراحلی که برای کامپایل و اجرای سورس کد ها طی می‌شود دارند و اصطلاحات تخصصی را به خوبی می‌شناسند و این یعنی به خوبی با دامنه‌ی سیستمی که قرار است آن را پیاده‌سازی کنند آشنا هستند و ترمینولوژی آن را می‌شناسند.

میدانیم که در زبان‌های کامپایلری سی و جاوا فرانیید ترجمه‌ی سورس کد به صورت یکپارچه صورت می‌گیرد و زمانی که پروسه‌ی کامپایل کردن سورس کد اجرا می‌شود فاز‌های تحلیلگر لغوی، تحلیلگر نحوی و تحلیلگر معنایی و تولید کننده کد میانی و بهینه ساز کد و تولید کننده کد، سورس کد را از لحاظ لغوی تجزیه و تحلیل می‌کند و توکن‌های موجود در متن را شناسایی می‌کند و آنها را از لحاظ آرایش و ترتیب قرار گیری با گرامر زبان برنامه نویسی تطبیق می‌دهد و از لحاظ معنایی و منطقی وارسی می‌کند و در نهایت کد میانی که در زبان جاوا بایت کد است تولید می‌شود و رانر ماشین مجازی جاوا آن را اجرا می‌کند

در کامپایلر زبان سی بخش تولید کد میانی نداریم و کد حاصل ماشین کدی متناسب با معماری کامپیوتر ماشین سرور است و پردازنده دستورالعمل‌های باینری این زبان را فراخوانی، رمزگشایی و اجرا می‌کند.

اطلاعات و دانش کافی در این حیطه را دارد. مشتریان و کاربرانی که قرار است از این سیستم استفاده کنند دانشجویان و اساتید رشته مهندسی کامپیوتر هستند.

فصل دوم

تحلیل نیازمندی ها

تعريف محدوده سیستم (System Scope) :

برای تعریف محدوده سیستم ، مسئله را دقیق تر تعریف می کنیم ، برای این منظور تمام کار هایی که سیستم باید انجام دهد را لیست می نمائیم

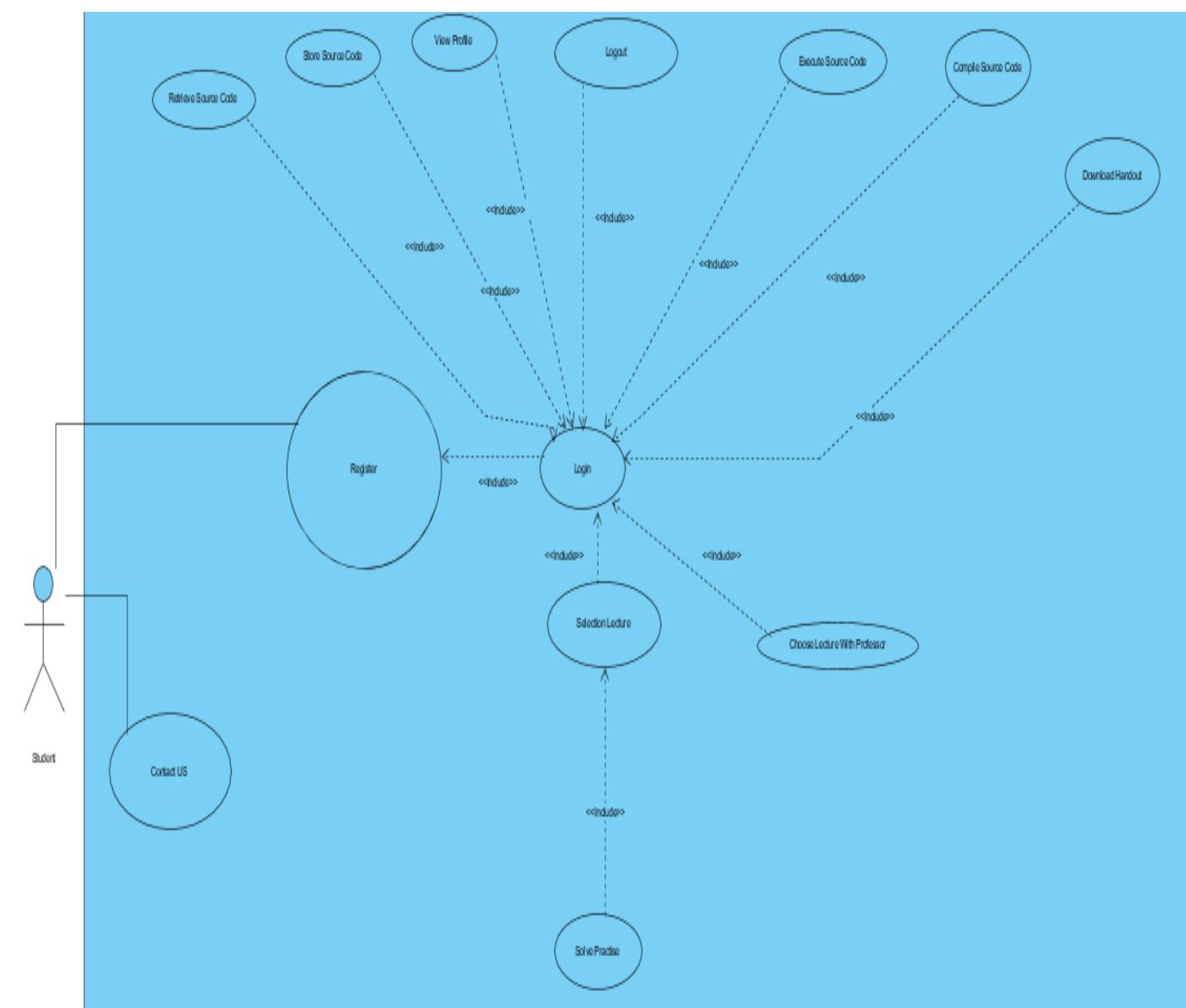
۱. ساختن حساب کاربری برای دانشجویان و اساتید
۲. احراز هویت کردن کاربران
۳. برداشتن و ثبت درس برای دانشجویان
۴. نمایش لیست دروس اخذ شده توسط دانشجویان
۵. ثبت دروس انتخاب شده توسط اساتید برای ارائه کردن
۶. نمایش لیست دروس اخذ شده توسط اساتید
۷. ذخیره و بازیابی فایل های دانشجویان و اساتید
۸. فراهم کردن ویرایشگر کد و محیط توسعه همراه با کامپایلر و مفسر برای تست کردن و اجرای کد های دانشجویان و نمایش خروجی به آنها
۹. ذخیره و بازیابی سورس کد های نوشته شده توسط دانشجویان از پایگاه داده
۱۰. فراهم کردن امکان طرح سوال و تمرین توسط اساتید برای دانشجویان و حل کردن مسائل و تمرینات توسط دانشجویان
۱۱. فراهم کردن امکان ارسال نظرات و پیشنهادات توسط دانشجویان و اساتید برای مدیر وب سایت برنامه نویسی
۱۲. فراهم کردن صفحه کاربری برای نمایش اطلاعات و مشخصات و تصویر کاربر
۱۳. منقضی کردن نشست کاربر پس از مدت زمانی معین (به صورت پیش فرض سی دقیقه)

موارد بالا به عنوان نیازمندی های عملکردی یا Functional Requirements شناخته می شود . در واقع موارد بالا تمام خصیصه ها و Feature ها و Function های وب سایت برنامه نویسی است و برای اینکه کاربران بتوانند Task هایشان را انجام دهند باید توسط توسعه دهنده پیاده سازی شوند .

نیازمندی های کیفیت :

پاسخ دهی به کاربران در کمترین زمان ممکن ، اجرای سریع پروسس های کامپایل و اجرا و مصرف کمترین منابع ممکن و قابلیت اطمینان و در دسترس بودن ، قابلیت استفاده مجدد

بررسی Use Case Diagram دانشجو



۲,۱

توصیف جزئیات	Use Case	بخش
یا دانشجو Student	Actor اصلی	
ثبت نام ، گام هایی را که Actor دانشجو برای ثبت نام در سیستم بایستی بردارد را مشخص می کند.	خلاصه	
کاربر باید به قسمت Navigation Register از Register بار برود	پیش شرط	
		جريان اصلی (سناریوی موفق)
۱. دانشجو روی Register Hyper Link از Navigation Bar کلیک می کند .		
۲. دانشجو اطلاعات خود را در فرم ثبت نام وارد می کند		
۳. سیستم جزئیات ثبت نام را بررسی می کند .		
۴. سیستم برای دانشجو یک حساب کاربری می سازد		
توصیف	عنوان	جريان های جایگزین
اگر در گام شماره ۱ دو از Main Flow جزئیات ثبت نام نامعتبر با Invalid باشد سیستم دانشجو را درباره ۱ جزئیات نامعتبر مطلع می کند و جريان مجدداً از ابتدای گام شماره ۱ دوادمه می یابد .	جريان جایگزین ۱	
دانشجو با کلیک کردن روی یکی از Hyper Link های موجود در Navigation Bar از صفحه ۱ ثبت نام خارج می شود و به این ترتیب use case Register خاتمه می یابد و End می شود .	جريان استثنای ۱	Exception Flow
توصیف	عنوان	پس شرط ها یا Post Conditions
دانشجو پروسه ۱ ثبت نام را تکمیل می کند و سیستم با موفقیت یک حساب کاربری برای او می سازد .	جريان اصلی	
پروسه ۱ ثبت نام دانشجو لغو می شود و use case End ثبت نام می شود	جريان استثنای ۱	

توصیف جزئیات	Use Case	بخش Case
یا دانشجو Student		اصلی Actor
تماس با ما یا Contact Us ، گام هایی را که Actor برای ارسال نظرات یا پیشنهادات باید بردارد را مشخص می کند .		خلاصه
User باید بر روی Hyper Link تماس با ما یا Navigation bar از Contact Us کلیک کند.		پیش شرط
جريان اصلی یا سناریوی موفق		

۱. دانشجو روی Navigation Bar از Hyper Link Contact Us کلیک می کند.
۲. دانشجو جزئیات شامل نام ، پست الکترونیکی و پیام یا همان نظر خود را در فرم Html نظرات و پیشنهادات وارد می کند
۳. سیستم اطلاعات و جزئیاتی که در فرم ثبت نام وارد شده است را بررسی و Verify می کند
۴. سیستم جزئیاتی را که دانشجو در فرم نظرات و پیشنهادات وارد کرده است معتبر تشخیص می دهد .

توصیف	عنوان	جريان های جایگزین
اگر در گام شماره ی دو از Main Flow جزئیات یا اطلاعات وارد شده در فرم نظرات / پیشنهادات نامعتبر یا Invalid باشد مثلاً پست الکترونیکی وارد شده توسط دانشجو نامعتبر و نادرست باشد سیستم دانشجو را درباره ی جزئیات نامعتبر مطلع می کند و جريان از ابتدای گام شماره ی دو از سر گرفته می شود و ادامه می یابد .	جريان جایگزین شماره یک	
دانشجو با کلیک روی یکی از Hyper Link های موجود در Navigation Bar از صفحه ی تماس با ما خارج می شود و به این ترتیب Use Case Contact Us خاتمه می یابد و End می شود .	جريان استثنایا Exception Flow	
دانشجو پروسه ی ثبت نظر و پیشنهاد را تکمیل میکند و سیستم نظر دانشجو را همراه با نام و پست الکترونیکی او در قالب یک رکورد یا Tuple در جدول پایگاه داده Insert می کند .	جريان اصلی	پس شرط ها یا Post Condition
پروسه ی ثبت نظر و پیشنهاد لغو می شود و Use Case Contact Us خاتمه می یابد و END می شود .	جريان استثنایا	

توصیف جزئیات	بخش Use Case
دانشجو Student	اصلی Actor
Use Case ورود به سیستم گام هایی را که دانشجو برای ورود به سیستم باید بردارد را مشخص می کند.	خلاصه
دانشجو می باشد قبل از سیستم ثبت نام کرده باشد ، زمانی که وب سایت بارگذاری می شود ، وب سرور ابتدا کاربر را به صفحه Navigate ، Login می کند.	پیش شرط

جريان اصلی یا سناریوی موفق

۱. دانشجو نام کاربری و پست الکترونیکی و رمز عبوری را که زمان ثبت نام حساب کاربری وارد کرده بود در فرم ورود به سیستم وارد می کند .
۲. سیستم جزئیات و اطلاعات وارد توسط کاربر را بررسی می کند .
۳. سیستم جزئیات و اطلاعات وارد شده توسط کاربر را معتبر تشخیص می دهد .

توصیف	عنوان	جريان های جایگزین
اگر در گام شماره ۱ دو از جريان اصلی اطلاعات وارد شده توسط دانشجو در فرم Login معتبر و صحیح نباشد مثلاً نام کاربری یا پست الکترونیکی یا رمز عبور وارد شده توسط User صحیح نباشد یا اینکه دانشجو به اشتباه تیک مربوط به Checkmark Professor سیستم به دانشجو می گوید که Credential وارد شده نامعتبر است و از کاربر می خواهد مجدداً Login کند و جريان از ابتدای گام شماره ۱ یک از سر گرفته می شود و ادامه می یابد و یا سیستم از کاربر میخواهد درصورتی که ثبت نام نکرده است درسیستم ثبت نام کند یعنی شروع Use Case Register	جريان جایگزین شماره ۱	
دانشجو با کلیک روی یکی از Hyper Link های موجود در Navigation Bar از صفحه Login خارج می شود و به این ترتیب خاتمه می یابد و Use Case Login END می شود.	جريان استثنای شماره ۱	جريان های استثنای
توضیف	عنوان	پس شرط ها یا Post Condition
دانشجو پروسه ۱ ورود به سیستم را تکمیل می کند یعنی فیلد های نام کاربری ، پست الکترونیکی و رمز عبور خود را به درستی وارد می کند و سیستم دانشجو را به صفحه User Page Navigate می کند.	جريان اصلی	
پروسه ۱ Login یا ورود به سیستم لغو می شود و Use Case Login خاتمه می یابد و END می شود .	جريان استثنای	

توصیف جزئیات	Use Case بخش
دانشجو یا Student	Actor اصلی
Use Case کامپایل سورس کد ، تمام آن گام هایی را که برای کامپایل سورس کد نیاز است که توسط Actor برداشته شود مشخص می کند.	خلاصه
دانشجو می بایست ابتدا در سیستم Login کند تا بتواند از امکانات فراهم شده در وب سایت برنامه نویسی آنلاین استفاده کند .	پیش شرط

جريان اصلی یا سناریوی موفق

۱. دانشجو بعد از اینکه با موفقیت به سیستم Login کرد وارد صفحه‌ی User Page می‌شود ، در این صفحه Actor روی آیکون Code Editor کلیک می‌کند .
۲. سیستم کاربر را به صفحه‌ی java server page ، ویرایشگر کد Navigate می‌کند .
۳. کاربر روی Button یا دکمه‌ی new کلیک می‌کند .
۴. سیستم از دانشجو می‌خواهد که زبان برنامه نویسی که می‌خواهد پروژه اش را با آن بنویسد وارد کند .
۵. دانشجو Programming Language مورد نظر خود را وارد می‌کند .
۶. سیستم از دانشجو می‌خواهد نام پروژه‌ی خود را وارد کند .
۷. دانشجو نام پروژه‌ی خود را در پنجره‌ی popup باز شده وارد می‌کند و محیط برنامه نویسی متناسب با زبان برنامه نویسی انتخاب شده فراهم می‌شود .
۸. دانشجو پروژه‌ی خود را می‌نویسد و بعد از اتمام کدنویسی ، روی Button کامپایل کلیک می‌کند .
۹. سیستم سورس کد نوشته شده توسط دانشجو را در سمت سرور کامپایل می‌کند .

توصیف	عنوان	جريان های جایگزین
اگر جزئیات وارد شده توسط دانشجو نامعتبر باشد مثلا دانشجو زبان برنامه نویسی ای را انتخاب کند که توسط سایت پشتیبانی نمی‌شود ، سیستم ورودی‌های نامعتبر را تشخیص می‌دهد و کاربر را مطلع می‌کند و جریان از ابتدای آن گامی شروع می‌شود که ورودی نامعتبر در آن اتفاق افتاده است .	جریان جایگزین شماره یک	
اگر دانشجو از صفحه‌ی جاری به صفحه‌ی قبلی بازگردد جریان از use case کامپایل خارج می‌شود و خاتمه می‌یابد و End می‌شود	جریان استثنا شماره یک	جریان استثنا
دانشجو پروسه‌ی کامپایل کردن سورس کد را به درستی تکمیل می‌کند و سیستم نتیجه‌ی کامپایل سورس کد دانشجو را در قالب یک رشته برای دانشجو نمایش می‌دهد .	جریان اصلی	پس شرط ها یا Post Conditions
پروسه‌ی کامپایل سورس کد دانشجو لغو می‌شود و Use Case کامپایل سورس کد خاتمه می‌یابد و End می‌شود .	جریان استثنا	

توصیف جزئیات	Use Case بخش	
دانشجو یا Student	Actor اصلی	
خلاصه Use Case اجرا ، گام هایی را که دانشجو برای اجرای پروژه اش باید بردارد مشخص می کند.		
دانشجو در وله‌ی اول می‌باشد در سیستم Login کرده تا بتواند از امکانات فراهم شده در وب سایت برنامه نویسی آنلاین استفاده کند و در صورتی که زبان برنامه نویسی انتخاب شده توسط دانشجو ، یک زبان Interpreter ای مثل Python نباشد و یک زبان Compiler ای باشد دانشجو می‌باشد قبل از اجرای کد ، سورس کدش را کامپایل کرده باشد .	پیش شرط	
جريان اصلی یا سناریوی موفق		
1. دانشجو روی دکمه‌ی Run ویرایشگر کد کلیک می‌کند . 2. سیستم اقدام به اجرای سورس کد می‌کند .		
توصیف	عنوان	جريان های جایگزین
اگر زبان برنامه نویسی انتخاب شده توسط دانشجو از نوع زبان‌های برنامه نویسی کامپایلری باشد و دانشجو قبل از کلیک کردن روی دکمه‌ی Run ، سورس کد خود را کامپایل نکرده باشد ، سیستم این خطا را متوجه شده و از دانشجو می‌خواهد که ابتدا سورس کد خود را کامپایل کند یعنی جريان از Use Case اجرا خارج می‌شود و کامپایل سورس کد شروع می‌شود.	جريان جایگزین شماره یک	
اگر دانشجو از صفحه‌ی جاری که در واقع صفحه‌ی Code Editor است خارج شود و به صفحه‌ی قبل برود . Use Case اجرای سورس کد خاتمه می‌یابد و End می‌شود .	جريان استثنای شماره یک	جريان استثنای Exception Flow
توصیف	عنوان	پس شرط‌ها یا Post Conditions
دانشجو پروسه‌ی اجرای سورس کد را تکمیل می‌کند و سیستم نتیجه‌ی اجرای سورس کد را در قالب یک رشته برای دانشجو به نمایش در می‌آورد.	جريان اصلی	
پروسه‌ی اجرای سورس کد دانشجو لغو می‌شود و Use Case اجرای سورس کد خاتمه می‌یابد و End می‌شود.	جريان استثنای	

توصیف جزئیات	Use Case بخش
Student یا دانشجو	Actor اصلی
Use Case ذخیره ای سورس کد ، گام هایی را که دانشجو برای ذخیره ای سورس کد پروژه اش در حساب کاربری خود باید بردارد مشخص می کند.	خلاصه
مثلاً باقی use case ها ، دانشجو برای استفاده از امکانات سیستم می بایست در سیستم Login کرده باشد.	پیش شرط ها
	جريان اصلی یا سناریوی موفق

۱. کاربر پس از نوشتن سورس کد خود روی **Store** ذخیره يا **Button** کلیک می کند.

۲. سیستم سورس کد دانشجو را ذخیره می کند.

توصیف	عنوان	جريان های جایگزین
در صورتی که دانشجو پروژه‌ی خود را New نکرده باشد یعنی سورس کد خود را ننوشته باشد Button ذخیره‌ی سورس کد disable خواهد بود سیستم اجازه‌ی ذخیره کردن سورس کد را به دانشجو نمی‌دهد	جريان جایگزین شماره یک	
اگر دانشجو روی هر Button ای غیر از Button ذخیره‌ی سورس کد کلیک کند ، جريان از Use Case ذخیره‌ی سورس کد خارج می‌شود و Use Case سورس کد خاتمه می‌یابد و End می‌شود	جريان استثنای شماره یک	جريان های استثنای Exception Flow
توصیف	عنوان	پس شرط‌ها یا Post Conditions
اگر دانشجو پروسه‌ی ذخیره‌ی سورس کد را تکمیل نماید سیستم سورس کد دانشجو را با موفقیت در حساب کاربری او و در قالب یک رکورد در جدول پایگاه داده Insert می‌کند.	جريان اصلی	
پروسه‌ی ذخیره‌سازی سورس کد در حساب کاربری دانشجو لغو می‌شود و Use Case ذخیره‌ی سورس کد خاتمه می‌یابد و End می‌شود.	جريان استثنای شماره یک	

توصیف جزئیات	Use Case	
دانشجو یا Student	اصلی Actor	
Use case بازیابی سورس کد ، گام‌هایی را که دانشجو برای بازیابی سورس کد مورد نظر خود باید بردارد را مشخص می‌کند .	خلاصه	
مانند سایر use case های موجود در سیستم برای استفاده از تمامی امکانات فراهم شده در وب سایت برنامه نویسی آنلاین دانشجو می‌باشد در سایت Login کند .	پیش شرط	
جريان اصلی یا سناریوی موفق		
۱. دانشجو بعد از Login در سیستم وارد User Page می‌شود و روی آیکون Code Editor کلیک می‌کند		
۲. سیستم دانشجو را به صفحه‌ی java server page ویرایشگر کد هدایت می‌کند .		
۳. دانشجو روی Button بازیابی سورس کد کلیک می‌کند .		
۴. سیستم از دانشجو می‌خواهد نام پروژه‌ی اسکریپت را که قبلاً در حساب کاربری خود ذخیره کرده است در popup window ای باز شده وارد کند .		
۵. دانشجو نام سورس کد یا پروژه‌ی مورد نظر خود را وارد می‌کند .		
۶. سیستم با توجه به نام پروژه‌ی وارد شده توسط دانشجو ، سورس کد دانشجو را بازیابی می‌کند .		
توصیف	عنوان	جريان های جایگزین

اگر در گام شماره پنج دانشجو نام پروژه‌ی مورد نظر خود را یک رشته تو خالی یعنی به طول صفر وارد کند سیستم این خطای دانشجو را تشخیص داده و به او اطلاع می‌دهد و از دانشجو می‌خواهد ورودی صحیح و معتبری را برای نام پروژه‌ی خود در پنجره‌ی Popup وارد نماید.	جريان جایگزین شماره یک	
اگر دانشجو به صفحه‌ی قبلی بازگردد یعنی صفحه‌ی User Page ، در این صورت جريان از Use Case بازیابی سورس کد خارج می‌شود و Use Case بازیابی سورس کد خاتمه می‌یابد و End می‌شود.	جريان استثنای شماره یک	جريان های استثنای
توصیف	عنوان	پس شرط‌ها یا Post Conditions
اگر دانشجو پروسه‌ی بازیابی سورس کد یا پروژه‌ی خود را به درستی تکمیل نماید سیستم ، سورس کد دانشجو را از جدول پایگاه داده بازیابی کرده و در قالب یک رشته در code editor برای کاربر برمی‌گرداند .	جريان اصلی	
در این صورت Use Case بازیابی سورس کد خاتمه می‌یابد و End می‌شود .	جريان استثنای	

توصیف جزئیات	Use Case بخش
دانشجو یا Student	Actor اصلی
Use Case مشاهده پروفایل ، مشخص می‌کند که کاربر برای مشاهده‌ی پروفایل کاربری خود باید چه گام‌هایی را بردارد.	خلاصه
دانشجو برای استفاده از code editor ، مشاهده پروفایل ، انتخاب درس ، انتخاب استاد ، دانلود کتاب و جزو و می‌بایست حتماً به سیستم Login کند .	پیش شرط
جريان اصلی یا سناریوی موفق	

۱. دانشجو بعد از **Login** کردن به سیستم وارد صفحه‌ی **User Page** ، **java server page** می‌شود و در این صفحه روی **icon** پروفایل کاربری کلیک می‌کند .
۲. سیستم دانشجو را به صفحه‌ی **Navigate** ، **user profile** می‌کند تا اطلاعات کاربری و عکس پروفایل خود را مشاهد کند .

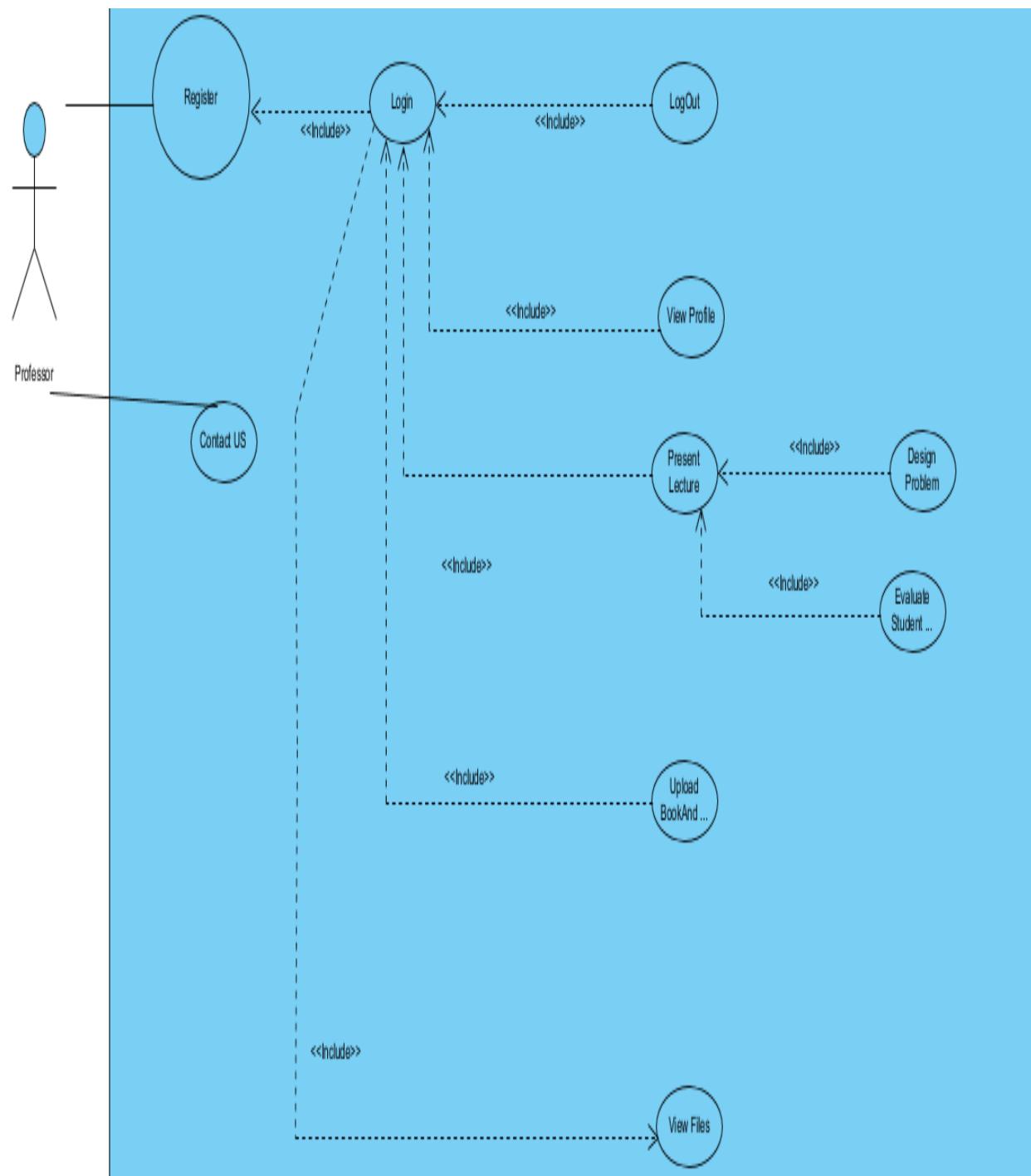
توصیف	عنوان	جريان های جایگزین
-----	جريان جایگزین شماره یک	
اگر دانشجو در صفحه‌ی User Page روی آیکون دیگری غیر از آیکون مشاهده پروفایل کاربری کلیک کند جريان از Use Case مشاهده پروفایل کاربر خارج می‌شود و Use Case مشاهده پروفایل کاربر خاتمه می‌یابد و End می‌شود .	جريان استثنای شماره یک	جريان های استثنای
توصیف	عنوان	پس شرط‌ها یا Post Conditions
اگر دانشجو پروسه‌ی مشاهده‌ی پروفایل کاربری خود را به درستی تکمیل نماید ، سیستم دانشجو را به صفحه‌ی profile.jsp هدایت می‌کند.	جريان اصلی	
Use Case مربوط به مشاهده پروفایل کاربر خاتمه می‌یابد و End می‌شود.	جريان استثنای	

توصیف جزئیات	Use Case بخش	
يا دانشجو Student	Actor اصلی	
Use Case خروج از سامانه ، گام هایی که User برای خروج از سیستم باید بردارد را مشخص می کند.	خلاصه	
دانشجو می باشد حتما در سیستم Login کرده باشد .	پیش شرط ها	
جريان اصلی یا سناریوی موفق		
1.دانشجو در صفحه‌ی User Page روی آیکون Logout کلیک میکند .		
2.سیستم کاربر را از سایت خارج می کند و Session مربوط به او را Expire می کند .		
توصیف	عنوان	جريان های جایگزین
-----	جريان جایگزین شماره یک	
اگر دانشجو در صفحه‌ی User Page روی هر آیکون موجود در صفحه غیر از آیکون Logout کلیک کند و با روی Hyper Link های موجود در Navigation Bar وب سایت برنامه نویسی آنلاین کلیک کند Flow از Use Case خارج می شود و End Use Case Logout	جريان استثنا شماره یک	جريان های استثنا
توصیف	عنوان	پس شرط ها یا Post Conditions
اگر دانشجو پرسه‌ی مربوط به Logout از سیستم را تکمیل کند ، سیستم او را از وب سایت خارج می کند و مربوط به دانشجو را منقضی می کند و می بنند و دانشجو را به صفحه‌ی Login یا ورود به سیستم Navigate می کند.	جريان اصلی	
Use Case خروج از سیستم خاتمه می یابد و End می شود .	جريان استثنا	

توصیف جزئیات	Use Case	بخش
دانشجو یا Student	Actor اصلی	
Use Case برداشتن درس ، گام هایی را که دانشجو برای برداشتن درس باید بردارد مشخص می کند.	خلاصه	
دانشجو حتما باید در سیستم Login کرده باشد تا بتواند درس مورد نظر را با استاد مورد نظر خود بردارد.	پیش شرط ها	
جريان اصلی یا سناریوی موفق		
۱. پس از Login کردن دانشجو در وب سایت برنامه نویسی آنلاین ، روی آیکون Lecture کلیک می کند . ۲. سیستم دانشجو را به صفحه Lecture هدایت می کند . ۳. دانشجو در صفحه Lecture روی آیکون برداشتن درس کلیک می کند ۴. سیستم دروسی را که تا کنون توسط دانشجو اخذ نشده است را برای دانشجو نمایش می دهد . ۵. دانشجو بر روی درس مورد نظر خود کلیک می کند . ۶. سیستم لیست اساتیدی که این درس را ارائه کرده اند ، نمایش می دهد . ۷. دانشجو روی Button استاد مورد نظر خود کلیک می کند . ۸. سیستم درس مورد نظر را با استاد انتخاب شده برای دانشجو ثبت می کند .		
توصیف	عنوان	جريان های جایگزین
-----	جريان جایگزین شماره يك	
اگر دانشجو به جای انتخاب Button برداشتن درس با استاد ، روی Button دروس انتخاب شده کلیک کند جريان يا Flow از Use Case انتخاب درس خارج می شود و Use Case انتخاب درس خاتمه می یابد و End می شود.	جريان استثنای شماره يك	جريان های استثنای
توصیف	عنوان	پس شرط ها یا Post Conditions
اگر دانشجو پرسه ی انتخاب درس با استاد مورد نظر را به درستی تکمیل کند ، سیستم ، درس و استاد انتخابی دانشجو را برای او ثبت می کند .	جريان اصلی	
Use Case خاتمه می یابد و End می شود .	جريان استثنای	

توصیف جزئیات	Use Case	بخش
دانشجو یا Student	Actor اصلی	
Use Case دانلود کتاب ، گام هایی را که دانشجو باید برای دانلود جزوی یا کتاب بردارد مشخص می کند.	خلاصه	
دانشجو قبل از انتخاب درس با استاد ، حتما باید در وب سایت برنامه نویسی آنلاین Login کرده باشد	پیش شرط ها	
جريان اصلی یا سناریوی موفق		

<p>۱. دانشجو در صفحه User Page روی آیکون Upload/Download کلیک می کند .</p> <p>۲. سیستم دانشجو را به صفحه Navigate می کند و لیست دروس اخذ شده توسط دانشجو را نشان می دهد و از دانشجو می خواهد بر روی Button درس مورد نظر خود کلیک کند .</p> <p>۳. دانشجو روی Button درس مورد نظر کلیک می کند</p> <p>۴. سیستم دانشجو را به صفحه ای هدایت می کند ، که شامل لیست جزوای و کتاب هایی است که برای این درس آپلود شده</p> <p>۵. دانشجو بر روی جزوی یا کتاب مورد نظر خود کلیک می کند و آن را دانلود میکند .</p>	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">توصیف</th><th style="text-align: left;">عنوان</th><th rowspan="2" style="text-align: left;">جريان های جایگزین</th></tr> </thead> <tbody> <tr> <td style="text-align: center;">-----</td><td style="text-align: center;">جريان جایگزین شماره یک</td></tr> </tbody> </table>	توصیف	عنوان	جريان های جایگزین	-----	جريان جایگزین شماره یک	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%;">اگر دانشجو بر روی یکی از Hyper Link های موجود در Navigation Bar کلیک کند ، جريان يا Flow از Use Case مربوط به دانلود کتاب خارج می شود و Use Case دانلود کتاب End میشود</td><td style="width: 50%;">جريان استثنای شماره یک</td><td style="width: 50%;">جريان های استثنای</td><td style="width: 50%;">جريان های جایگزین</td></tr> </table>	اگر دانشجو بر روی یکی از Hyper Link های موجود در Navigation Bar کلیک کند ، جريان يا Flow از Use Case مربوط به دانلود کتاب خارج می شود و Use Case دانلود کتاب End میشود	جريان استثنای شماره یک	جريان های استثنای	جريان های جایگزین
توصیف	عنوان	جريان های جایگزین									
-----	جريان جایگزین شماره یک										
اگر دانشجو بر روی یکی از Hyper Link های موجود در Navigation Bar کلیک کند ، جريان يا Flow از Use Case مربوط به دانلود کتاب خارج می شود و Use Case دانلود کتاب End میشود	جريان استثنای شماره یک	جريان های استثنای	جريان های جایگزین								
توصیف	عنوان										
اگر دانشجو پرسه ی مربوط به دانلود کتاب و جزوی را تکمیل کند سیستم از طریق سمت سرور فایل کتاب را برای دانشجو فراهم می کند ، و دانشجو فایل را از سرور سیستم دانلود میکند .	جريان اصلی	پس شرط ها يا Post Conditions									
Use Case دانلود کتاب يا جزوی خاتمه می یابد و End می شود.	جريان استثنای										



توصیف جزئیات	Use Case بخش	
استاد یا Professor	Actor اصلی	
Use Case ثبت نام یا Register تمام گام هایی که استاد به عنوان Actor خارج این سیستم برای ثبت نام در سایت برنامه نویسی آنلاین باید بردارد را مشخص می کند .	خلاصه	
استاد باید روی Hyper Link ثبت نام از Navigation Bar کلیک نماید	پیش شرط ها	
جریان اصلی یا سناریوی موفق		
۱. استاد پس از بازگذاری صفحه ی وب سایت بر روی Register Hyper Link موجود در Navigation Bar کلیک می کند. ۲. سیستم استاد را به صفحه ی Register که شامل یک HTML Form است میکند . ۳. استاد اطلاعات خود را در فیلد های فرم ثبت نام وارد می نماید و تیک Check Mark که نشان دهنده ی این است که User استاد می باشد را می زند . ۴. با زدن تیک check mark تابع Java اسکریپت بخش خاصی را در صفحه ی html جاری show می کند که شامل فیلد هایی است مربوط به مدارک و رشته تحصیلی استاد و اینکه استاد قرار است چه Lecture هایی را برای دانشجویان ارائه نماید . ۵. استاد پس از تکمیل فرم ثبت نام بر روی دکمه ثبت نام کلیک می نماید ۶. سیستم ، کاربر با اطلاعات وارد شده را به عنوان استاد در سایت ثبت نام می کند.		
توصیف	عنوان	جريان های جایگزین
اگر در گام شماره ی سه از Main Flow اطلاعات وارد شده توسط استاد ، نامعتبر یا Invalid باشد ، سیستم کاربر را از اطلاعات نامعتبر وارد شده با خبر می کند واز کاربر می خواهد بخش مورد نظر را با ورودی صحیح و معتبر مجدداً پر نماید و یا در صورتی که کاربر بر روی دکمه Reset از فرم ثبت نام کلیک کند ، جریان از ابتدای گام شماره سه از سر گرفته می شود و ادامه می یابد .	جریان جایگزین شماره یک	
در صورتی که کاربر بر روی یکی از Hyper Link های موجود در Navigation Bar (به جز Register) کلیک نماید ، جریان از Use Case Register ثبت نام خارج می شود و Use Case خاتمه می یابد و END می شود و یا در صورتی که تیک check mark مربوط به استاد را نزند Use Case ثبت نام دانشجو آغاز می شود .	جریان استثنای شماره یک	جریان های استثنای
توصیف	عنوان	پس شرط ها یا Post Conditions
اگر کاربر پروسه ی ثبت نام استاد را به درستی تکمیل نماید ، سیستم در سمت سرور یک حساب کاربری برای استاد مورد نظر می سازد و اطلاعات استاد در سامانه ثبت می شود .	جریان اصلی	
Use Case ثبت نام خاتمه می یابد و END می شود .	جریان استثنای	

توصیف جزئیات	Use Case بخش	
استاد یا Professor	Actor اصلی	
Use Case ورود به سیستم گام هایی را که دانشجو برای ورود به سیستم باید بردارد را مشخص می کند .	خلاصه	
استاد می بایست قبلا در سیستم ثبت نام کرده باشد ، زمانی که وب سایت بارگذاری می شود ، وب سرور ابتدا کاربر را به صفحه Navigate ، Login می کند.	پیش شرط	
جريان اصلی یا سناریوی موفق		
1. استاد نام کاربری و پست الکترونیکی و رمز عبوری را که زمان ثبت نام حساب کاربری وارد کرده بود در فرم ورود به سیستم وارد می کند . 2. سیستم جزئیات و اطلاعات وارد شده توسط کاربر را بررسی می کند . 3. سیستم جزئیات و اطلاعات وارد شده توسط کاربر را معتبر تشخیص می دهد .		
توصیف	عنوان	جريان های جایگزین
اگر در گام شماره ی دو از جريان اصلی اطلاعات وارد شده توسط استاد در فرم Login معتبر و صحیح نباشد مثلا نام کاربری یا پست الکترونیکی یا رمز عبور وارد شده توسط User صحیح نباشد یا اینکه استاد به اشتباه تیک مربوط به Checkmark Professor را نزدیک باشد سیستم به استاد می گوید که وارد شده نامعتبر است و از کاربر می خواهد مجددا Login Credential کند و جريان از ابتدای گام شماره ی یک از سر گرفته می شود و ادامه می یابد و یا سیستم از کاربر میخواهد درصورتی که ثبت نام نکرده است درسیستم ثبت نام کند یعنی شروع Use Case Register	جریان جایگزین شماره ۱	جريان های جایگزین
استاد با کلیک روی یکی از Hyper Link های موجود در Navigation Bar از صفحه Login خارج می شود و به این ترتیب Use Case Login خاتمه می یابد و END می شود .	جریان استثنای شماره یک	جريان های استثنای Post Condition
توصیف	عنوان	پس شرط ها یا Post Condition
استاد پروسه ی ورود به سیستم را تکمیل می کند یعنی فیلد های نام کاربری ، پست الکترونیکی و رمز عبور خود را به درستی وارد می کند و سیستم دانشجو را به صفحه Navigate ، User Page می کند.	جريان اصلی	
پروسه ی Login یا ورود به سیستم لغو می شود و Use Case Login خاتمه می یابد و END می شود .	جريان استثنای	

توصیف جزئیات	Use Case	بخش
استاد یا استاد Professor	Actor اصلی	
Use Case خروج از سامانه ، گام هایی که User برای خروج از سیستم باید بردارد را مشخص می کند.	خلاصه	
استاد می بایست حتما در سیستم Login کرده باشد .	پیش شرط ها	
جريان اصلی یا سناریوی موفق		
1. استاد در صفحه‌ی User Page روی آیکون Logout کلیک میکند . 2. سیستم کاربر را از سایت خارج می کند و Session مربوط به او را Expire می کند .		
توصیف	عنوان	جريان های جایگزین
-----	جريان جایگزین شماره یک	
اگر استاد در صفحه‌ی User Page روی هر آیکون موجود در صفحه غیر از آیکون Logout کلیک کند و یا روی Hyper Link های موجود در Navigation Bar نویسی آنلاین کلیک کند Flow از Use Case Logout خارج می شود و خاتمه می یابد و End می شود .	جريان استثنا شماره یک	جريان های استثنا
توصیف	عنوان	پس شرط ها یا Post Conditions
اگر استاد پرسه‌ی مربوط به Logout از سیستم را تکمیل کند ، سیستم او را از وب سایت خارج می کند و Session مربوط به دانشجو را منقضی می کند و می بنند و استاد را به صفحه‌ی Navigate یا ورود به سیستم Login می کند .	جريان اصلی	
Use Case خروج از سیستم خاتمه می یابد و End می شود .	جريان استثنا	

توصیف جزئیات	Use Case	بخش
استاد یا Professor	Actor اصلی	
Use Case مشاهده پروفایل ، مشخص می کند که کاربر برای مشاهده‌ی پروفایل کاربری خود باید چه گام هایی را بردارد .	خلاصه	
استاد برای مشاهده پروفایل ، ارائه درس آپلود کردن کتاب و جزو و می بایست حتما به سیستم Login کند .	پیش شرط	
جريان اصلی یا سناریوی موفق		

۱. استاد بعد از Login کردن به سیستم وارد صفحه‌ی User Page، java server page می‌شود و در این صفحه روی icon پروفایل کاربری کلیک می‌کند.
۲. سیستم، استاد را به صفحه‌ی Navigate، user profile می‌کند تا اطلاعات کاربری و عکس پروفایل خود را مشاهد کند.

توصیف جزئیات	عنوان	جريان های جایگزین
----	جريان جایگزین شماره یک	
اگر استاد در صفحه‌ی User Page روی آیکون دیگری غیر از آیکون مشاهده پروفایل کاربری کلیک کند جريان از Use Case مشاهده پروفایل کاربر خارج می‌شود و مشاهده پروفایل کاربر خاتمه می‌یابد و End می‌شود.	جريان استثنا شماره یک	جريان های استثنا
توصیف	عنوان	پس شرط‌ها یا Post Conditions
اگر استاد پروسه‌ی مشاهده‌ی پروفایل کاربری خود را به درستی تکمیل نماید سیستم، استاد را به صفحه‌ی profile.jsp هدایت می‌کند.	جريان اصلی	
مربوط به مشاهده پروفایل کاربر خاتمه می‌یابد و End می‌شود.	جريان استثنا	

توصیف جزئیات	Use Case بخش
استاد یا Professor	اصلی Actor
استاد Use Case آپلود کتاب و جزوه، گام‌هایی را که استاد باید برای آپلود جزوه یا کتاب بردارد مشخص می‌کند.	خلاصه
استاد قبل از اینکه درسی را که ارائه کرده است انتخاب کند، حتماً باید در وب سایت برنامه نویسی آنلاین Login کرده باشد	پیش شرط‌ها
استاد در صفحه‌ی User Page روی آیکون Upload/Download که یک آیکون Cloud می‌باشد کلیک می‌کند.	جريان اصلی یا سناریوی موفق
سیستم، استاد را به صفحه‌ی آپلود Navigate می‌کند و لیست دروس ارائه شده توسط استاد را در قالب یک Drop Down لیست نشان می‌دهد و از استاد می‌خواهد از بین دروسی که ارائه کرده است درس مورد نظرش را انتخاب و سپس فایل را آپلود نماید	
استاد درس مورد نظرش را انتخاب و روی دکمه Browse کلیک و فایل کتاب یا جزوه مورد نظر را از فایل سیستم ماشین خود انتخاب می‌کند و روی دکمه Upload کلیک می‌کند.	
سیستم، پس از اینکه استاد فایل مربوط به درس مورد نظر را آپلود کرد، آن فایل را در Insert، Data Base می‌کند.	

توصیف	عنوان	جريان های جایگزین
-----	جريان جایگزین شماره یک	
اگر استاد بر روی یکی از Hyper Link های موجود در Navigation Bar کلیک کند ، جريان يا Use Case از Flow مربوط به آپلود کتاب خارج می شود و آپلود کتاب End میشود	جريان استثنا شماره یک	جريان های استثنا
توصیف	عنوان	پس شرط ها يا Post Conditions
اگر استاد پرسه هی مربوط به آپلود کتاب و جزو را تکمیل کند سیستم از طریق سمت سرور فایل کتاب را در Data Base درج می کند .	جريان اصلی	
Use Case آپلود کتاب یا جزو خاتمه می یابد و End می شود.	جريان استثنا	

توصیف جزئیات	Use Case بخش
استاد یا Professor	Actor اصلی
Use Case برداشتن درس ، گام هایی را که استاد برای ارائه کردن یک درس باید بردارد مشخص می کند.	خلاصه
استاد حتما باید در سیستم Login کرده باشد تا بتواند درس مورد نظر را برای دانشجویان ارائه کند .	پیش شرط ها
جريان اصلی یا سناریوی موفق	
۱. پس از Login کردن استاد در وب سایت برنامه نویسی آنلاین ، روی آیکون Lecture کلیک می کند .	
۲. سیستم استاد را به صفحه Lecture هدایت می کند .	
۳. استاد در صفحه Lecture روی آیکون برداشتن درس کلیک می کند	
۴. سیستم دروسی را که تا کنون توسط استاد اخذ ارائه نشده است را در قالب یک Drop Down لیست برای استاد نمایش می دهد	
۵. استاد درس مورد نظرش را از Present Lecture Select می کند و روی دکمه Present Lecture کلیک می نماید .	
۶. سیستم درس یا Lecture انتخاب شده توسط استاد را برای استاد برمی دارد .	
توصیف	عنوان
-----	جريان جایگزین شماره یک

اگر استاد به جای کلیک روی Button ارائه‌ی درس ، روی Button دروس ارائه شده کلیک کند جریان با Use Case Flow از Use Case انتخاب درس خارج می‌شود و End ارائه‌ی درس خاتمه می‌یابد و می‌شود.	جریان استثنا شماره یک	جریان های استثنا
توصیف	عنوان	پس شرط ها یا Post Conditions
اگر استاد پرسه‌ی ارائه‌ی درس را به درستی تکمیل کند ، سیستم درس مورد نظر را برای استاد بر می‌دارد و این Lecture جزو Lecture های ارائه شده توسط این استاد قرار می‌گیرد .	جریان اصلی	
Use Case خاتمه می‌یابد و End می‌شود .	جریان استثنا	

توضیف جزئیات	Use Case بخش
استاد یا Professor	اصلی Actor
Use Case طراحی مسئله و تمرین ، گام‌هایی را که استاد برای طراحی مسئله باید بردارد را مشخص می‌کند.	خلاصه
Login کردن استاد به وب سایت و انتخاب درس مورد نظر برای طرح کردن تمرین برای دانش آموزان آن درس	پیش شرط ها
حریان اصلی یا سناریویی موفق	
۱. پس از Login کردن استاد به وب سایت برنامه نویسی آنلاین ، وارد User page خود می‌شود و روی آیکون Lecture کلیک می‌کند.	
۲. سیستم استاد را به صفحه‌ی Navigate , professorLecture.jsp می‌کند	
۳. استاد بر روی دکمه Presented Lectures کلیک می‌کند	
۴. سیستم لیست دروس ارائه شده توسط استاد را در قالب یک Drop Down لیست نمایش می‌دهد .	
۵. استاد از بین دروسی که ارائه کرده است ، درسی را که قصد دارد برای دانشجویانش سوال و مسئله طرح کند را انتخاب می‌کند .	
۶. سیستم استاد را به صفحه‌ی studentListAndPractiseList.jsp ، هدایت می‌کند .	
۷. استاد روی دکمه طرح کردن تمرین کلیک می‌کند .	
۸. سیستم یک پنجره‌ی Popup برای استاد نمایش می‌دهد	

۹. استاد مسئله را برای درس موردنظر طرح می کند و زبان برنامه نویسی و عنوان تمرین را نیز مشخص می کند و روی دکمه ارسال کلیک می کند
۱۰. سیستم تمرین طرح شده توسط استاد را برای دانشجویانی که درس موردنظر را با این استاد دارند در نظر می گیرد و ثبت می کند

توصیف	عنوان	جريان های جایگزین
-----	جريان جایگزین شماره یک	
اگر استاد بر روی لینک های موجود در Navigation Bar کلیک کند ، جريان از Use Case طرح سوال خارج می شود و اين Use Case خاتمه می يابد	جريان استثنا شماره یک	جريان های استثنا
توصیف	عنوان	پس شرط ها یا Post Conditions
اگر استاد پرسوه ای طرح سوال را به درستی طی و تکمیل نماید ، سیستم سوال مطرح شده توسط استاد را برای دانشجویان آن استاد ثبت می کند و سوال در لیست تمارین درس دانشجویان قرار می گیرد .	جريان اصلی	
Use Case طرح سوال خاتمه می يابد و END می شود .	جريان استثنا	

در سیستم ما دو Actor داریم

(۱) دانشجو

(۲) استاد

در Student use case diagram دانشجو برای استفاده از امکانات سایت باید در وب سایت ثبت نام یا Register نماید و بعد به سیستم Login کند .

یعنی پیش شرط ها یا Pre Condition ها برای انتخاب استاد و درس توسط دانشجو و همچنین استفاده از ادیتور وب سایت ، Register کردن و Login کردن است .

بین دانشجو و Business Use Case Association ثبت نام یک رابطه بود که نشان می دهد دانشجو که در خارج سیستم قرار دارد از این Functionality ثبت نام که در داخل سیستم قرار دارد استفاده می کند .

بین Business Use Case و Register یا Login یک رابطه بود Include وجود دارد یعنی Business Use Case ثبت نام در داخل Business Use Case ورود قرار گرفته است به Business Use Case ثبت نام که در داخل Business Use Case می گوییم و به Included Business Use Case ورود می گوییم .

چون User برای Login کردن به سیستم باید از قبل در سیستم Register کرده باشد .

Use Case مشاهده پروفایل ، Login را Use Case ورود یا Include کرده است یعنی Use Case ابتدا باید به User از Use Case View Profile ، جزئی از Login مشاهده پروفایل کاربری ، Login سیستم ، Login کند .

به همین ترتیب Use Case های خروج از سیستم ، کامپایل کردن سورس کد ، اجرای ماشین کد ، ذخیره سیستم کد ، بازیابی سورس کد ، برداشتن درس با استاد ، دانلود کردن جزو و کتاب ، Use Case Login را Include کرده اند .

بين Actor دانشجو و Business Use Case تماش با ما یک رابطه ای وجود دارد و این Association Relationship های داخل سیستم هیچ Use Case و دیگر Case وجود ندارد .

در Use Case Diagram مربوط به استاد :

بين Actor Register و Contact US های Business Use Case استاد که خارج از سیستم قرار دارد و که داخل سیستم قرار دارد یک رابطه ای Association وجود دارد . یعنی استاد از Actor Functionality های Contact US و Register استفاده می کند .

مانند Business Use Case دانشجو در Use Case Diagram استاد نیز View Files ، Upload Books And Handouts ، Present Lecture و View Profile های Business Use Case ورود را Include کرده اند .

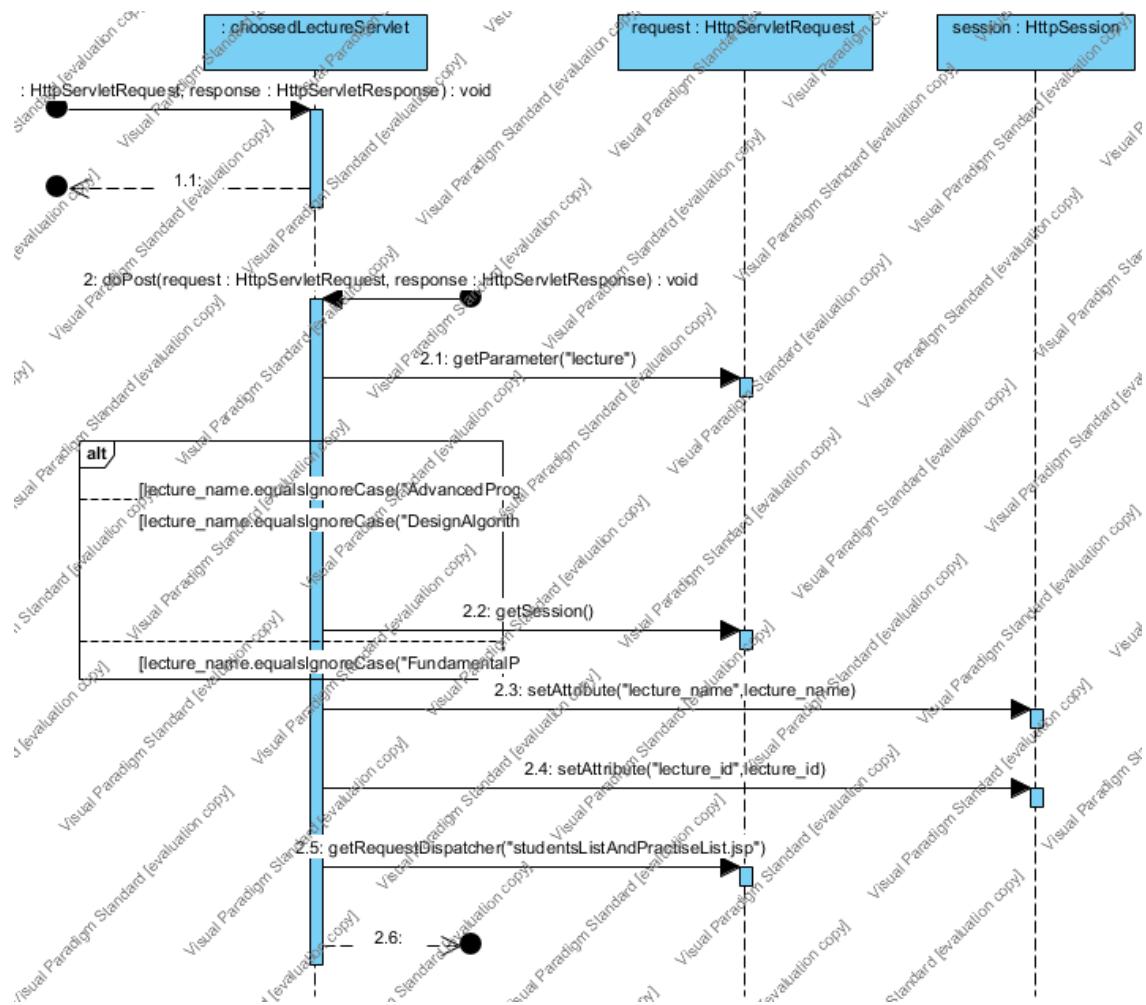
Evaluate Student Solution و Design Problem های Business Use Case نیز Select Lecture را Include کرده اند .

چون استاد ابتدا از بین درس ها یا Lecture هایی که ارائه کرده است درس مورد نظرش را انتخاب می کند و بعد برای دانشجویان آن Lecture تمرین طرح می کند .

پس Use Case انتخاب درس جزئی از Use Case طرح کردن تمرین است .

تحليل Sequence Diagram ها :

مربط به درس انتخاب شده : Sequence Diagram

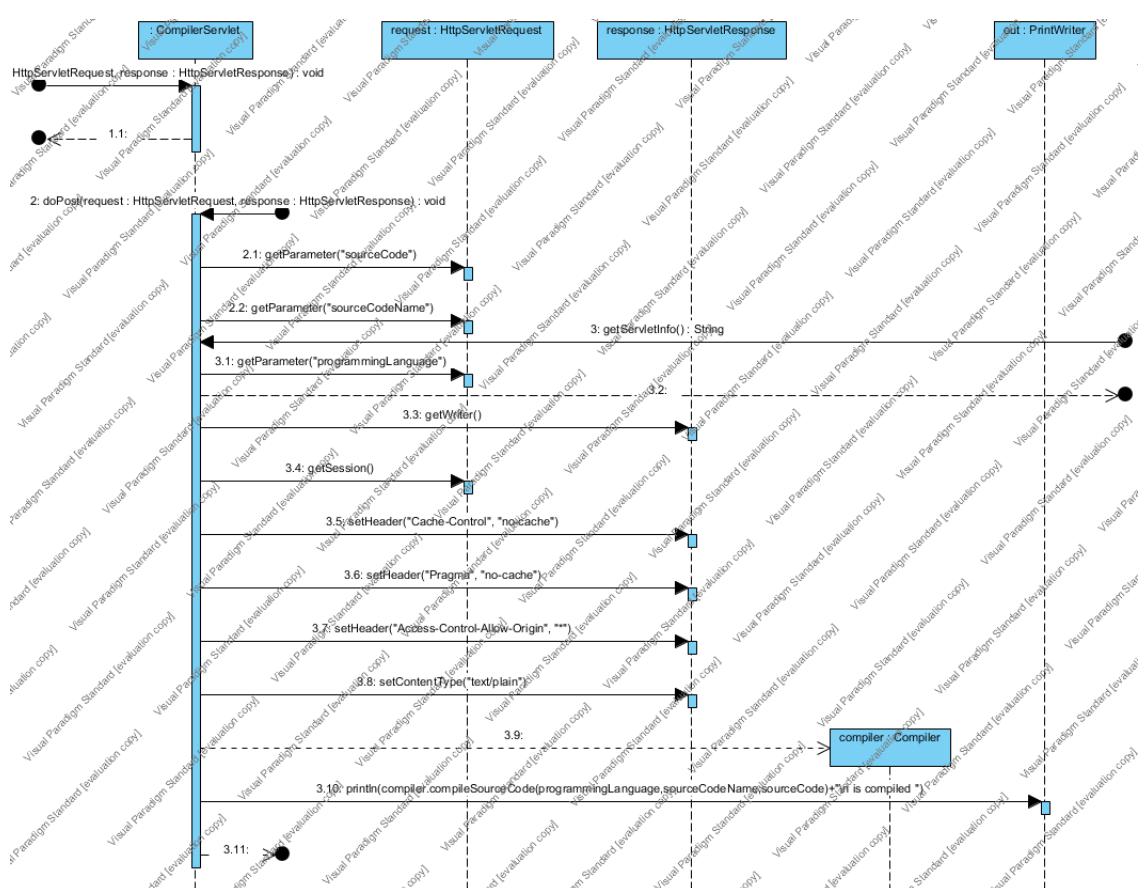


این نمودار توالی controller های کلاس سرولت choosedLectureServlet در پکیج Instruction را نشان می دهد .

این سرولت دو متد (doGet() و doPost()) را دارد .

زمانی که استاد در سیستم Log in کلیک کند سرور زمانی که استاد در صفحه ProfessorLecture.jsp میکند اگر بر روی آیکون Lecture کلیک کند استاد ارائه شده است برای او نشان داده می شود استاد از میان دروسی که ارائه کرده است روی درس مورد نظر خود کلیک میکند و کد جاوا اسکریپت در سمت کلاینت یک ajax request Post از نوع lecture میکند و پارامتر lecture_id را در بدن خود دارد به این کلاس سرولت ارسال می کند و متد (doPost()) این کلاس سرولت را فراخوانی می کند این متد مقدار پارامتر را از روی http servlet request بدست می آورد ، سپس مقدار lecture_name و lecture_id را که نشان دهنده نام و شناسه درس ارائه شده استاد توسعه دهنده درس ارائه شده توسط استاد روی session تنظیم می کند و http servlet response و http servlet request object را به java server page studentListAndPractiseList ارسال می کند .

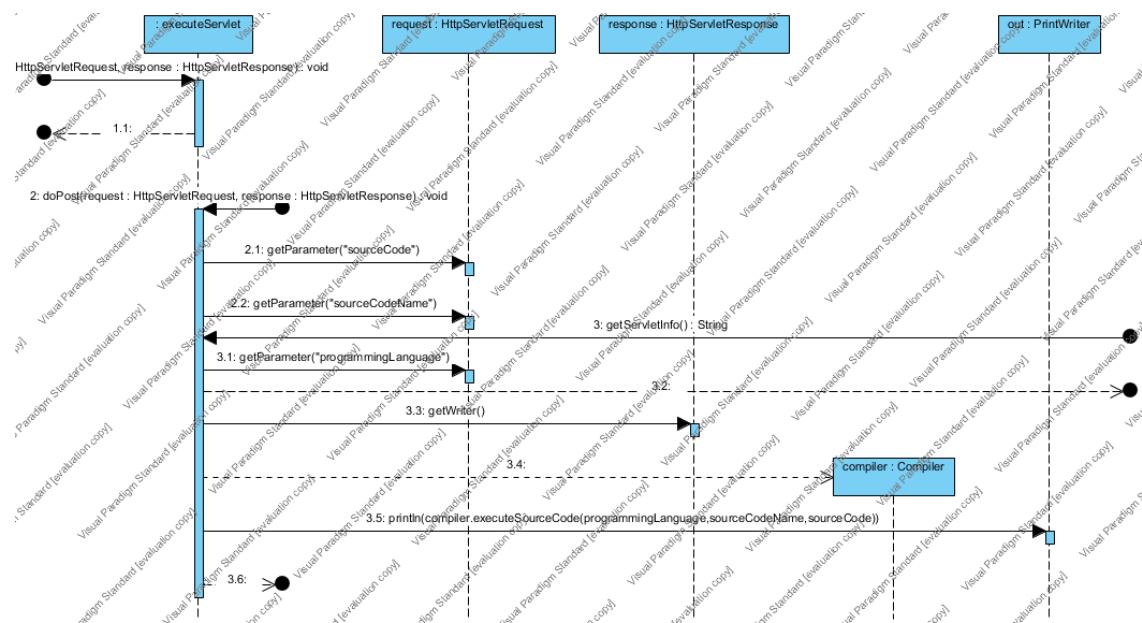
مربوط به کامپایل سورس کد : Sequence Diagram



این نمودار توالی **Instruction** های کلاس **CompilerServlet** را نشان می دهد. تمامی کلاس های سرولت در پکیج **controller** قرار دارد . وقتی کاربر وارد صفحه **java server page** ویرایشگر کد می شود ، با کلیک بر روی دکمه **new** ، سیستم از کاربر می خواهد که زبان برنامه نویسی خود را انتخاب کند و با انتخاب زبان برنامه نویسی توسط کاربر ، سیستم از او میخواهد برای پروژه خود یک نام برگزیند و با وارد کردن نام پروژه توسط کاربر و نوشتن برنامه کامپیوتری خود ، بر روی دکمه کامپایل کلیک می کند و یکتابع جاوا اسکریپت در سمت کلاینت فراخوانی می شود این تابع ، یک **ajax request** از نوع متده **Post** مربوط به پروتکل **http** می سازد و مقدار مربوط به **html document object model** ، ویرایشگر کد را کد گذاری می کند (بخاطر وجود **special character** در پارامتر و مشکلات پارس کردن درخواست در وب سرور) و همچنین زبان برنامه

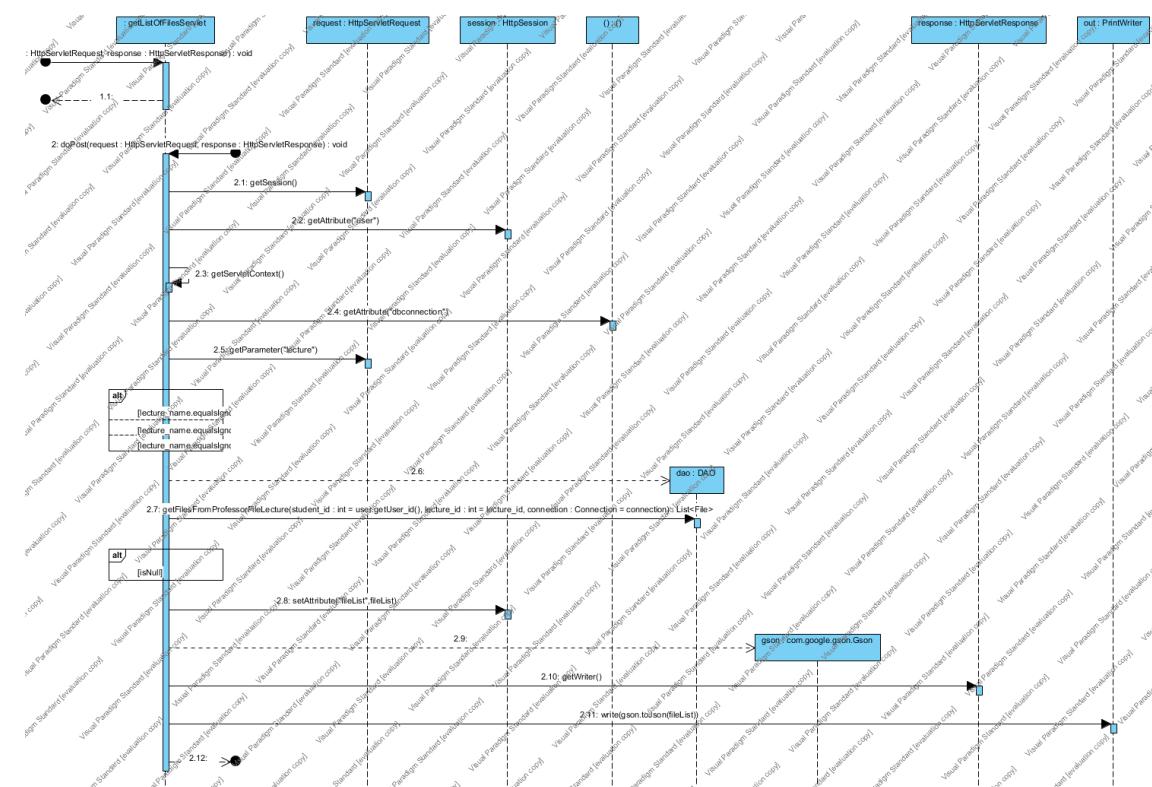
نویسی انتخاب شده توسط کاربر و نام پروژه‌ی کاربر را به عنوان پارامتر روی این `request` تنظیم می‌کند و به سمت سرور ارسال می‌کند و متدهای `doPost()` این کلاس سرولت را فراخوانی می‌کند، این متدهای پارامترهای سورس کد و نام پروژه و زبان برنامه نویسی را از روی `http servlet request` بر می‌دارد و یک `instance` از کلاس کامپایلر ایجاد می‌کند و متدهای `compileSourceCode()` آن را فراخوانی می‌کند و پارامترهای سورس کد و نام `PrintWriter` و زبان برنامه نویسی را به آن پاس می‌دهد و نتیجه‌ی ایجاد کامپایل را روی آبجکت `out` از کلاس `PrintWriter` می‌نویسد و به عنوان `response` به سمت کلاینت ارسال می‌کند.

Sequence Diagram مربوط به اجرای برنامه:



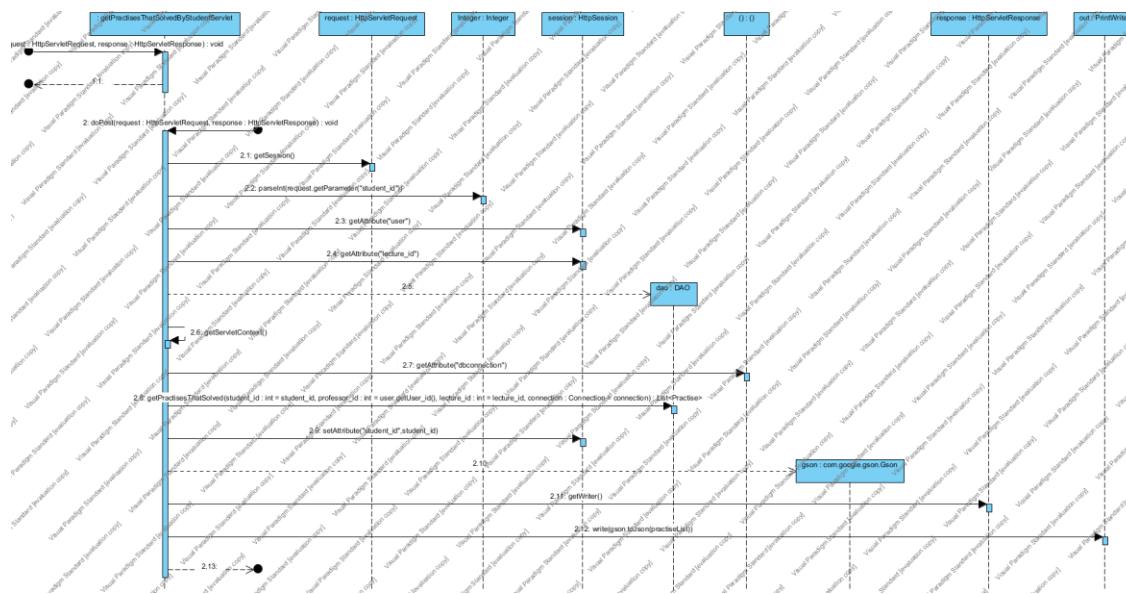
کاربر روی دکمه `run` کلیک می‌کند و تابع `javaScript` در سمت کلاینت یک `ajax request` از نوع متدهای `Post` پروتکل `http` ایجاد می‌کند و پارامتر سورس کد کد گذاری شده و پارامترهای زبان برنامه نویسی و نام پروژه را روی `executeService` قرار می‌دهد و به سمت سرور ارسال می‌کند در سمت سرور کلاس سرولت `doPost` کلاس سرولت `call` می‌شود و پارامتر فراخوانی می‌شود و همانطور که در نمودار توالی مشخص است متدهای `instance` از کلاس کامپایلر های سورس کد و نام پروژه و زبان برنامه نویسی را از `request` بر میدارد و یک `instance` از کلاس کامپایلر می‌سازد و متدهای `executeSourceCode` آن را فراخوانی می‌کند و پارامترهای سورس کد و نام پروژه و زبان برنامه نویسی را به آن متدهای پاس می‌دهد و نتیجه‌ی اجرای برنامه را روی آبجکت `out` از کلاس `PrintWriter` نوشته می‌شود و به عنوان `response` به سمت سرور ارسال می‌شود و تابع `javaScript` در سمت کلاینت می‌شود و به عنوان `text area` برای کلاینت در یک `String` نمایش می‌دهد.

Sequence Diagram مربوط به مشاهده لیست کتاب ها و جزو ات توسط دانشجو که این کتاب ها و جزو ات قبل از درس مورد نظر آپلود شده است :



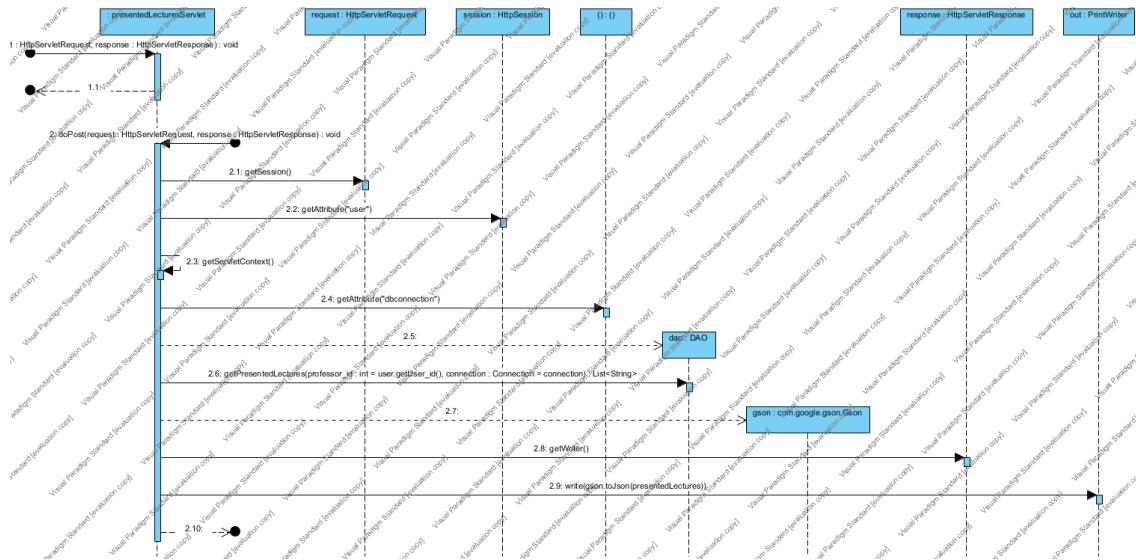
کاربر بعد از ورود به وب سایت برنامه نویسی وارد صفحه‌ی کاربری خود می‌شود و روی آیکون **upload / download** کلیک می‌کند، سرور کاربر را به صفحه‌ی **html** دانلود هدایت می‌کند در این صفحه، یک لیست از دروسی که توسط دانشجو برداشته شده است، برای کاربر نمایش داده می‌شود و کاربر از میان **Lecture** هایی که انتخاب کرده است، بر روی درس مورد نظر خود کلیک می‌کند با کلیک کاربر بر روی دکمه‌ی درس مورد نظر، یک تابع جاوا اسکریپت فراخوانی می‌شود و این تابع یک **ajax request** از نوع متدهای **post** پروتکل **http** می‌سازد و پارامتر **lecture** را در **request** **lecture** قرار می‌دهد و به سمت سرور ارسال می‌کند و درست سرور متدهای **http servlet** **getServletList** کلاس سرولت **getListOfServlet**، فراخوانی می‌شود و پارامتر نام درس از روی آبجکت **request** برداشته می‌شود و آبجکت **connection** که برای برقراری ارتباط با پایگاه داده است و روی **context** **attribute** به صورت **context** **servlet** تنظیم شده است را از روی **context** **servlet** بازیابی می‌کنیم و متدهای **getFilesFromProfessor()** از کلاس **DAO** فراخوانی می‌شود و آی دی دی دانشجو و آی دی درس و آبجکت **connection** به عنوان پارامتر به این متدهای پاس داده می‌شود، این متدهای تمام جزو ات و فایل هایی که توسط استاد این درس برای دانشجویان آپلود شده است را برمی‌گردانند و نتیجه در یک آبجکت **json** ریخته می‌شود و به عنوان **response** برای سمت کلاینت ارسال می‌شود.

Sequence Diagram مربوط به لیست تمرینات طرح شده برای یک درس توسط استاد :



دانشجو بعد از اینکه در وب سایت برنامه نویسی Log in کرد ، وارد صفحه‌ی کاربری خود می‌شود و با کلیک بر روی آیکون Lecture ، سیستم ، دانشجو را به صفحه‌ی java server page ، studentLecture می‌هدایت می‌کند . در این صفحه لیست دروسی که توسط دانشجو انتخاب شده است نمایش داده می‌شود . دانشجو با کلیک بر روی درس مورد نظر وارد صفحه‌ی java serve page می‌شود . زمانی که این صفحه بارگذاری می‌شود یکتابع listOfPractisesThatWillSolveByStudent می‌شود . این تابع یک ajax request از نوع متدهای http پروتکل می‌سازد و به جاوا اسکریپت فراخوانی می‌شود و این تابع یک url می‌ارسال می‌کند . در سمت سرور متدهای url /practisesOfLectureServlet و به سمت سرور می‌رسد که آن برای url pattern آن باشد فراخوانی می‌شود . در متدهای doPost() و doGet() سرولت کاربر بازیابی می‌شود و شناسه دانشجو را از فیلد user id آبجکت user بدست می‌آوریم و همچنین شناسه‌ی استاد و شناسه‌ی درس را که روی کاربر (دانشجو) می‌شود است بازیابی می‌کنیم و متد getPractises() از کلاس PractiseDAO را فراخوانی می‌کنیم و شناسه‌ی دانشجو و شناسه‌ی استاد و شناسه‌ی درس را به عنوان پارامتر به این متد پاس می‌دهیم ، این متد لیست تمرينات مربوط به درسی که دانشجو با استاد با اين شناسه برداشته است بر می‌گرداند . لیست تمرينات را در یک آبجکت Gson ذخیره می‌کنیم و به عنوان response به سمت کاربر بر می‌گردانیم .

مربوط به دروس ارائه شده توسط استاد : Sequence Diagram



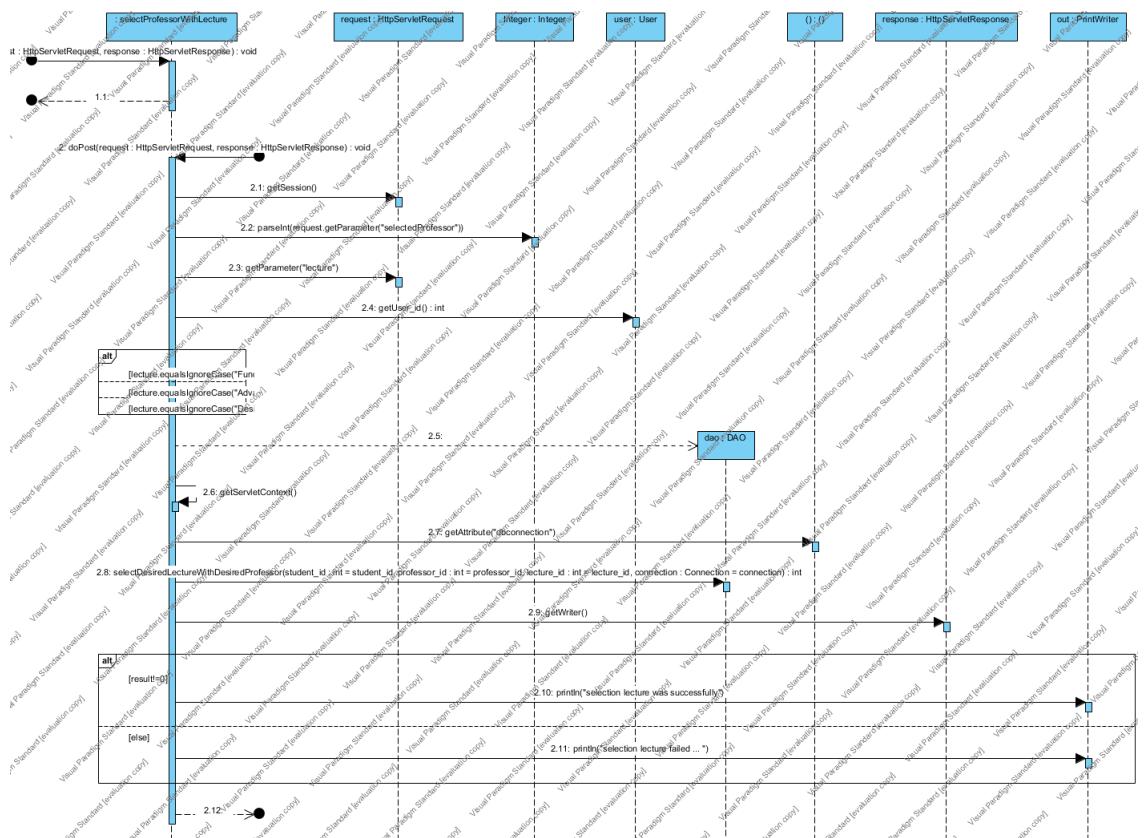
این نمودار توالی Instruction های مربوط به کلاس سرولت presentedLecturesServlet را نمایش می دهد که داخل پکیج کنترلر قرار گرفته است .

زمانی که استاد در وب سایت برنامه نویسی Login می کند ، اینکه کاربر استاد است را به عنوان یک user مربوط به session تنظیم می کنیم . بعد از Login کردن ، استاد وارد صفحه ای کاربری خود می شود و با کلیک بر روی آیکون Lecture ، یک Post Request از نوع متده است با url ، و با ارسال اطلاعات studentProfessorServlet از طریق این سرولت کار کنترل کاربر را انجام می دهد و با استفاده از Attribute session کاربر تنظیم شده است تصمیم می گیرد که کاربر را به کدام صفحه هدایت کند . در اینجا چون user به عنوان استاد به وب سایت برنامه نویسی Login کرده است این کنترلر استاد را به صفحه ای که روی session مربوط شده است تضمین می کند . در این صفحه استاد روی دکمه ای دروس ارائه شده کلیک می کند ، با کلیک کردن استاد یک ajax request با استفاده از XMLHttpRequest ایجاد می کنیم و نوع متده را Post می گذاریم و به url /presentedLecturesServlet ارسال می کنیم .

شرح Sequence Diagram از اینجا شروع می شود : در سمت سرور ، سرولت کانتینر url درخواست آزادی را با url pattern سرولت ها تطبیق می دهد و متده `doPost()` سرولتی را فراخوانی می کند که آن مطابق url ای که در بالا آمده است باشد . در متده `doPost()` ، آبجکت user را از session مربوط به استاد بازیابی می کنیم و با فراخوانی متده `getUserId` ، شناسه ای استاد را از روی این آبجکت user بدست می آوریم آبجکت connection پایگاه داده که به عنوان servlet context attribute روی connection تنظیم شده است را بازیابی می کنیم . متده `getPresentedLectures()` از کلاس DAO را فراخوانی می کنیم و شناسه ای کاربری استاد را به عنوان پارامتر به این متده می دهیم . این متده لیست دروسی را که توسط استاد با این شناسه کاربری ارائه شده است بر می گرداند این لیست را در یک آبجکت Gson ذخیره می کنیم و روی آبجکت out از کلاس

پرینٹر می نویسیم

Sequence Diagram مربوط به انتخاب و برداشت درس توسط دانشجو با استاد مورد نظر:



زمانی که دانشجو در وب سایت برنامه نویسی آنلاین Login می کند ، وارد صفحه‌ی کاربری خود می شود و با کلیک بر روی آیکون Lecture وارد صفحه‌ی jsp ، studentLecture می شود در این صفحه می تواند به صفحه‌ی دروسی که قبلا برداشته است برود یا ایتكه درس جدیدی را با یک استاد بردارد . دانشجو از لیست Drop Down درس مورد نظر خود را انتخاب می کند و بر روی دکمه جستجو کلیک می کند و سیستم لیست اساتیدی که این درس را ارائه کرده اند به دانشجو نشان می دهد و دانشجو با کلیک بر روی نام استاد ، درس مورد نظر خود با آن استاد بر می دارد . با کلیک دانشجو بر روی نام استاد یک تابع جاوا اسکریپت فراخوانی می شود ، این تابع ، یک ajax با استفاده از request http xml ایجاد می کند و شناسه‌ی استاد و نام درس انتخاب شده توسط دانشجو را به عنوان پارامتر روی request قرار می دهد و به url ، /selectProfessorWithLecture ، ارسال می کند .

شرح Sequence Diagram مربوط به انتخاب درس از اینجا شروع می شود :

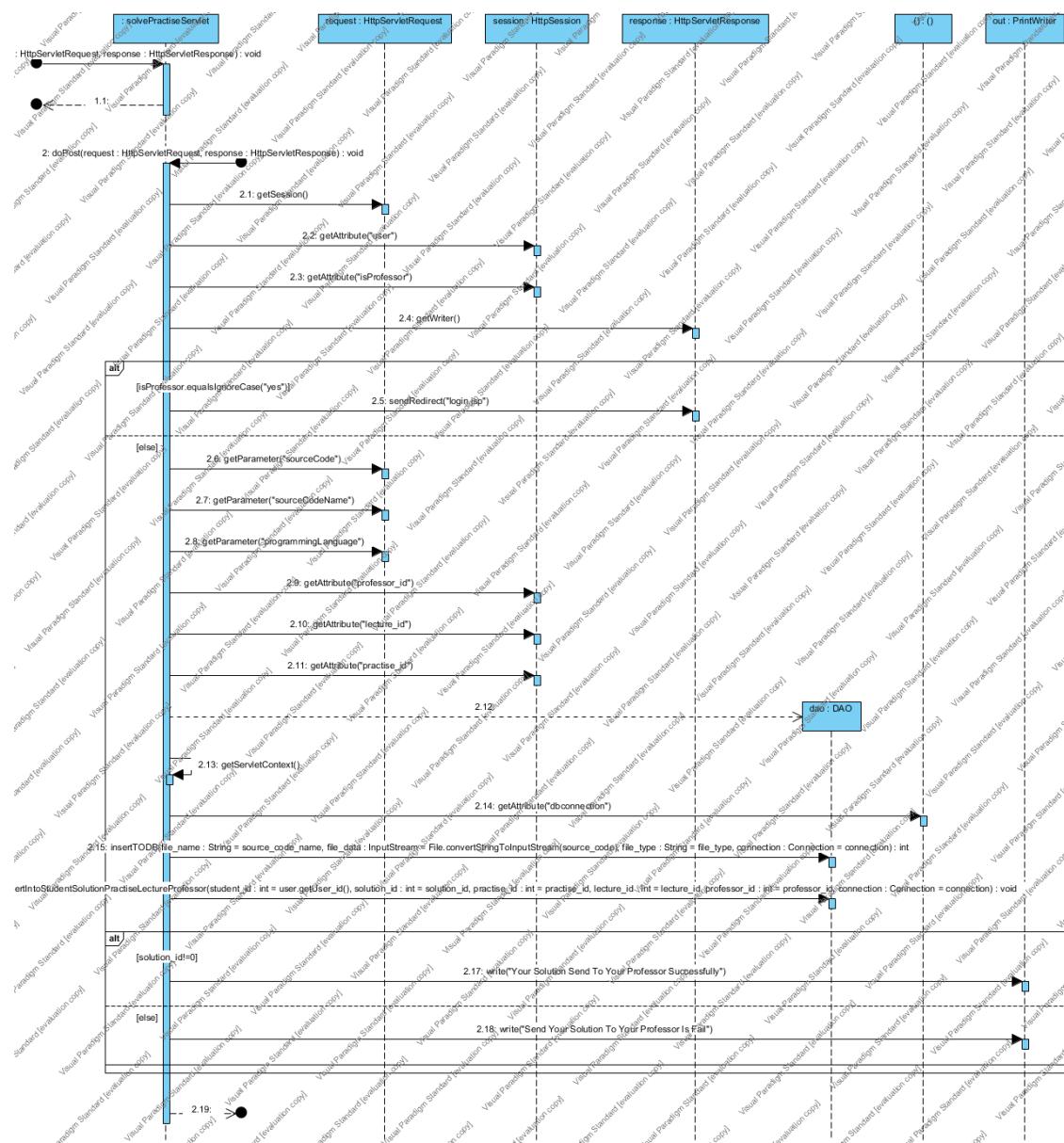
در سمت سرور متد `doPost()` سرولتی که `url pattern` آن با `url` بالا تطبیق داشته باشد فراخوانی می شود و در این متد پارامتر های شناسه‌ی کاربری استاد و نام درس از روی آبجکت `http servlet request` بازیابی می شود و آبجکت `connection` با پایگاه داده از روی `servlet context` بازیابی می شود و یک `instance` از کلاس

DAO، می سازیم و متدا
و

`selectDesiredLectureWithDesiredProfessor()` را فراخوانی می کنیم و پارامتر های شناسه‌ی

کاربری دانشجو و شناسه‌ی کاربری استاد و شناسه‌ی درس را همراه با آبجکت **connection** به این متدهای پاس می‌دهیم. این متدهای یک رکورد در جدول **student_professor_lecture** ایجاد می‌کنند. شناسه‌ی دانشجو را از روی **session** مربوط به کاربر (دانشجو) بازیابی کردیم.

مربوط به حل تمرین توسط Sequence Diagram



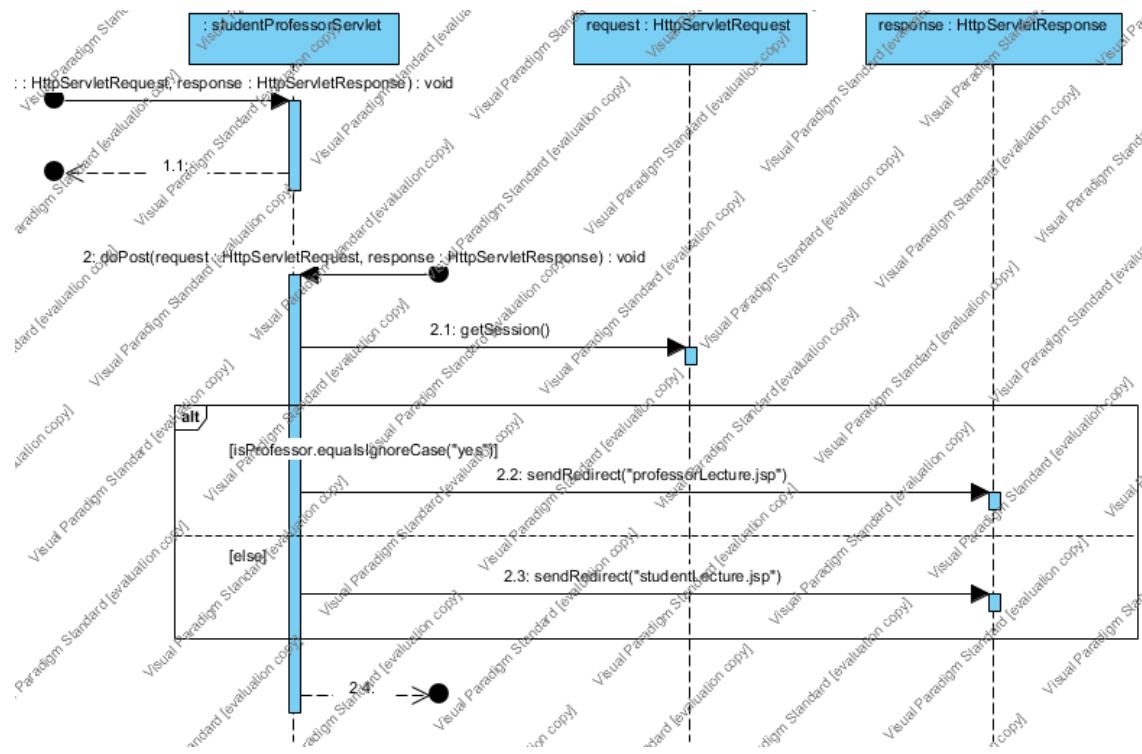
زمانی که دانشجو بر روی تمرین مورد نظر کلیک می‌کند یک صفحه‌ی **popup** مربوط به آن تمرین باز می‌شود، دانشجو بعد از مشاهده‌ی صورت سوال با کلیک بر روی حل تمرین، به صفحه‌ی ویرایشگر کد می‌رود و با کلیک بر

روی دکمه `new` یک پروژه‌ی جدید ایجاد می‌کند و زبان برنامه نویسی و نام پروژه‌ی خود را انتخاب می‌کند و شروع به نوشتن کد برنامه‌ی خود می‌کند. دانشجو با کلیک بر روی دکمه‌ی حل تمرین در پایین ویرایشگر کد یک `xml http request` را با استفاده از `ajax request` می‌کند و یک `call` را با استفاده از `xml http request` می‌کند و این `call` را با استفاده از `ajax request` می‌کند و یک `request` را به عنوان پارامتر می‌سازد و پارامترهای سورس کد، کد گذاری شده و زبان برنامه نویسی و نام پروژه را به عنوان پارامتر روی `request` قرار می‌دهد و این `request` را به `/solvePractiseServlet` ارسال می‌کند.

شرح و توضیح Sequence Diagram از اینجا شروع می‌شود:

متد `doPost()` کلاس سرولتی فراخوانی می‌شود که `url` آن با `url pattern` آمده در بالا مطابقت داشته باشد، در متد `doPost`، پارامترهای سورس کد تمرین حل شده، زبان برنامه نویسی، نام پروژه از روی `request` بدست می‌آوریم و مقادیر شناسه‌ی کاربری استادی که تمرین را طرح کرده است و شناسه‌ی درسی که تمرین برای آن طرح شده است و شناسه‌ی تمرین را همراه با آبجکت `user` از روی `session` کاربر بدست می‌آوریم، با فراخوانی متد `get user id` روی آبجکت `user` شناسه‌ی کاربری دانشجو را بدست می‌آوریم و همچنین آبجکت `connection` پایگاه داده را از روی `servlet context` بدست می‌آوریم و یک `Instance` از روی کلاس `DAO` می‌سازیم و متد `insertIntoFile()` را روی این آبجکتی که از `DAO` ساختیم فراخوانی می‌کنیم و پارامترهای نام پروژه، سورس کد، نوع فایل را به عنوان پارامتر به این متد پاس می‌دهیم نوع فایل یک نتغیر از جنس کلاس `String` است که در واقع مشخص کننده‌ی نوع فایل می‌باشد که در اینجا زبان برنامه نویسی با رشته همان شناسه‌ی فایلی است که به `Data Base` اضافه شده است را بر می‌گرداند `insertIntoStudentSolutionPractiseLectureProfessor()` فراخوانی می‌کنیم و پارامترهای شناسه‌ی کاربری دانشجو، شناسه‌ی فایلی که در بالا به پایگاه داده اضافه شد و شناسه‌ی تمرینی که توسط دانشجو پاسخ داده شده است و شناسه‌ی درسی که دانشجو به تمرین آن پاسخ داده است و شناسه‌ی کاربری استادی که درس مورد نظر را برای این دانشجو ارائه کرده است را به متد پاس می‌دهیم.

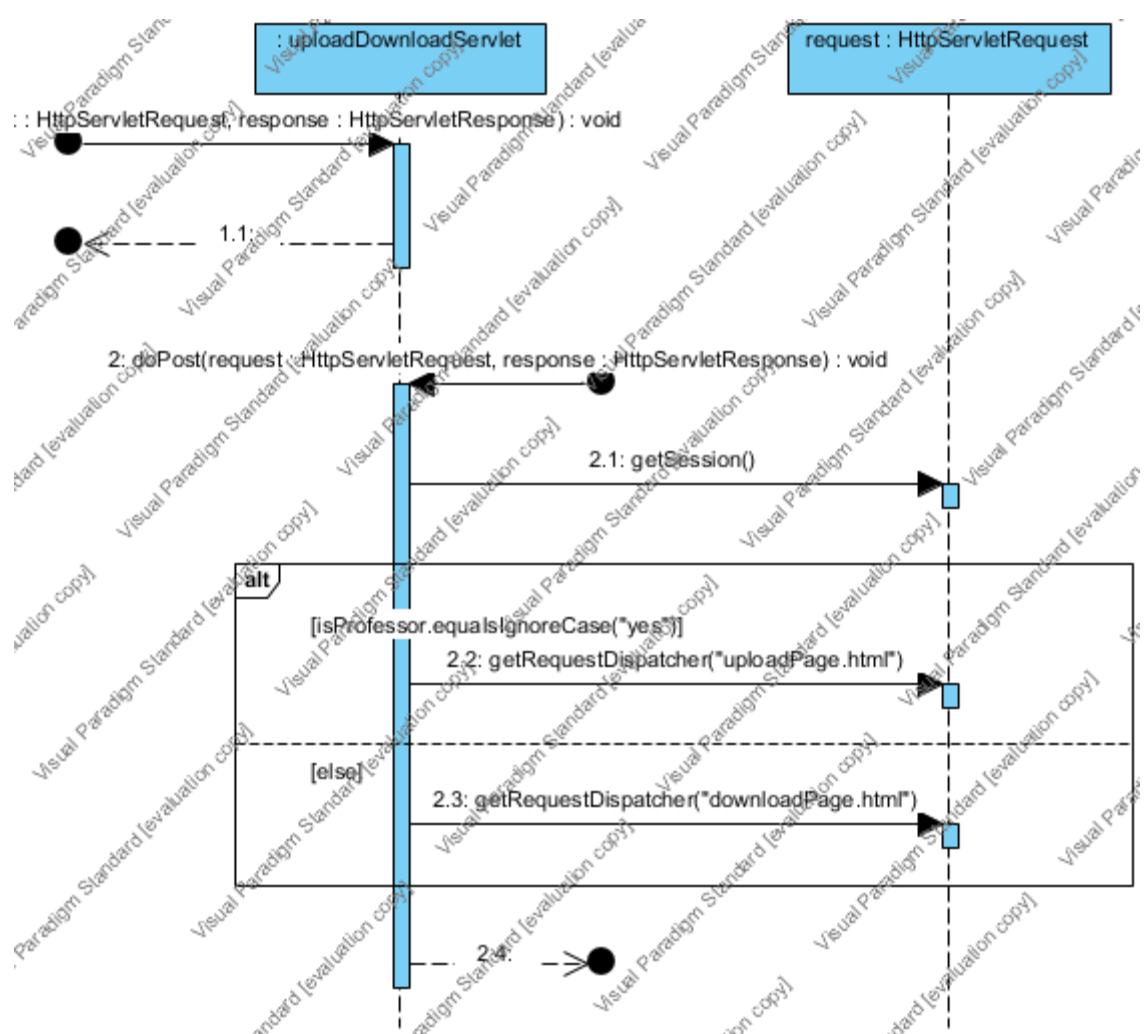
: Lecture مربوط به هدایت دانشجو و کاربر به صفحه‌ی Sequence Diagram



زمانی که کاربر به وب سایت برنامه نویسی Login می کند یک key Attribute می کند که نشان می دهد کاربر ، دانشجو است یا استاد ؟ زمانی که کاربر بر روی آیکون Lecture کلیک می کند یک Post از نوع متده است url ، ارسال می شود در سمت سرور متده Post از کلاس سرولت /studentProfessorServlet فراخوانی می شود این متده session Attribute را که روی studentProfessorServlet می کند و تشخیص می دهد که آیا کاربر دانشجو است یا استاد ، در صورتی که کاربر دانشجو باشد به صفحه studentLecture.jsp هدایت می شود . در صورتی که کاربر استاد باشد به صفحه professorLecture.jsp .

مربوط به هدایت کاربر به صفحه دانلود یا آپلود (بسته به اینکه کاربر

استاد است یا دانشجو ؟)

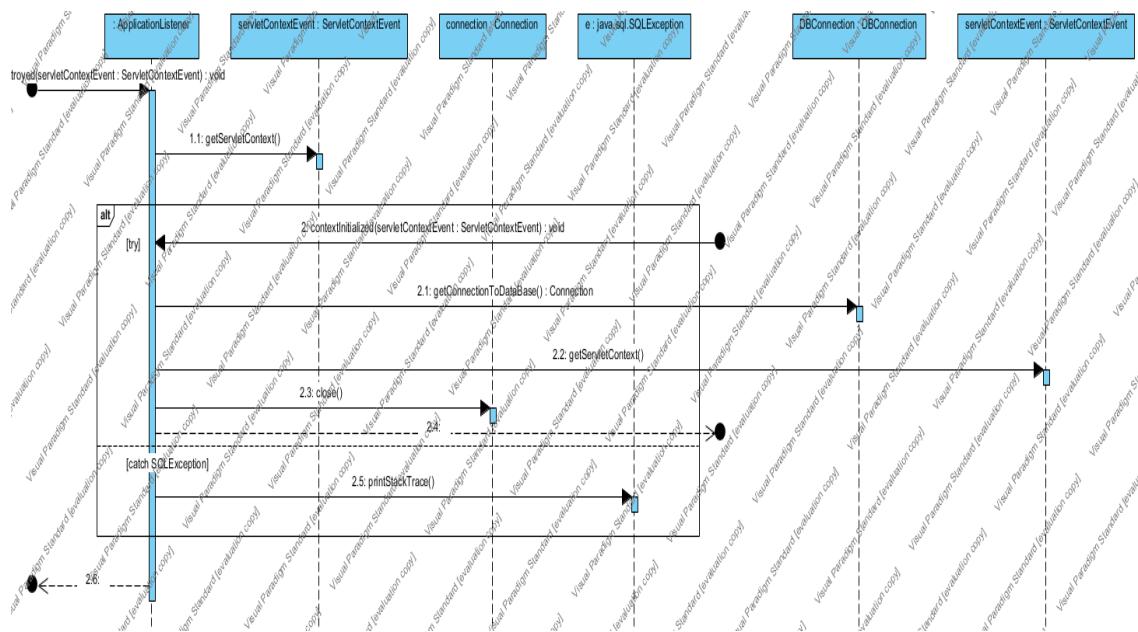


همانطور که در شرح `isProfessor` Attribute قبلی گفته‌یم یک **Sequence Diagram** روی `isProfessor` مربوط به کاربر `set` می‌کنیم برای راحت `session` زمانی که دانشجو به وب سایت برنامه نویسی `Login` کرد، یا کلیک بر روی آیکون `Cloud` مربوطبه آپلود / دانلود یک `Http Request` از نوع متده `Post` به `url` `/uploadDownloadServlet` ارسال می شود.

شرح **Sequence Diagram** از اینجا شروع می شود :

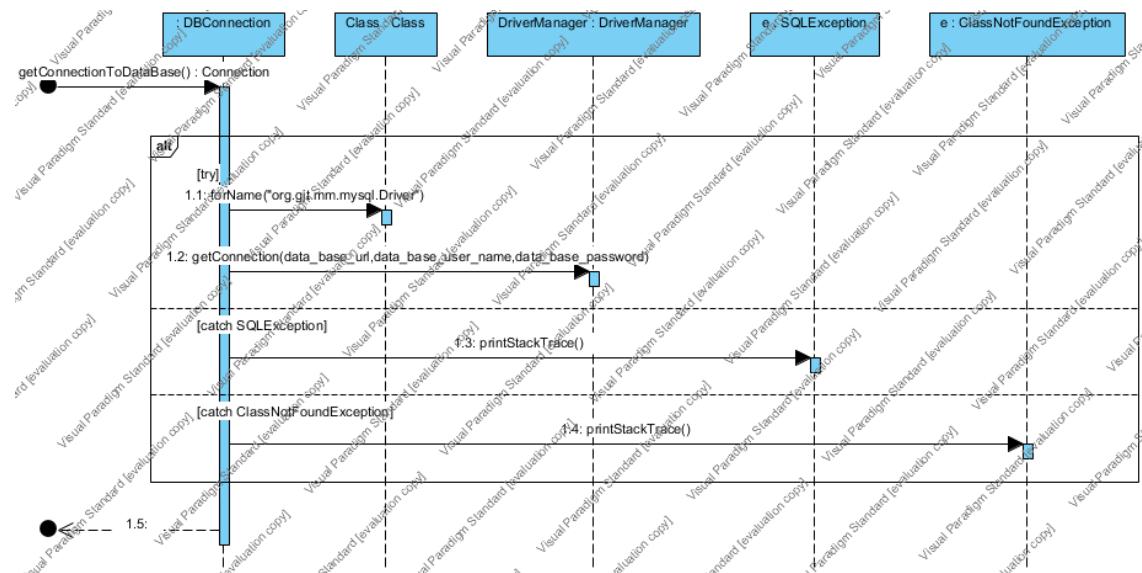
در سمت سرور ، متده `doPost()` از کلاس سرولت `uploadDownloadServlet` فراخوانی می شود در این متده `user` با کلید `isProfessor` از روی `session` مربوط به `user` ای که به سیستم کرده است `get` می‌شود و از روی متغیر از `Type String` کلاس `String` کنترلر تشخیص می دهد که کاربر را به چه صفحه ای `forward` کند اگر کاربر دانشجو باشد آنرا به صفحه ای دانلود یعنی `download.jsp` و اگر کاربر استاد باشد آنرا به صفحه ای آپلود یعنی صفحه ای `upload.jsp` می فرستد .

: Application Listener مربوط به Sequence Diagram



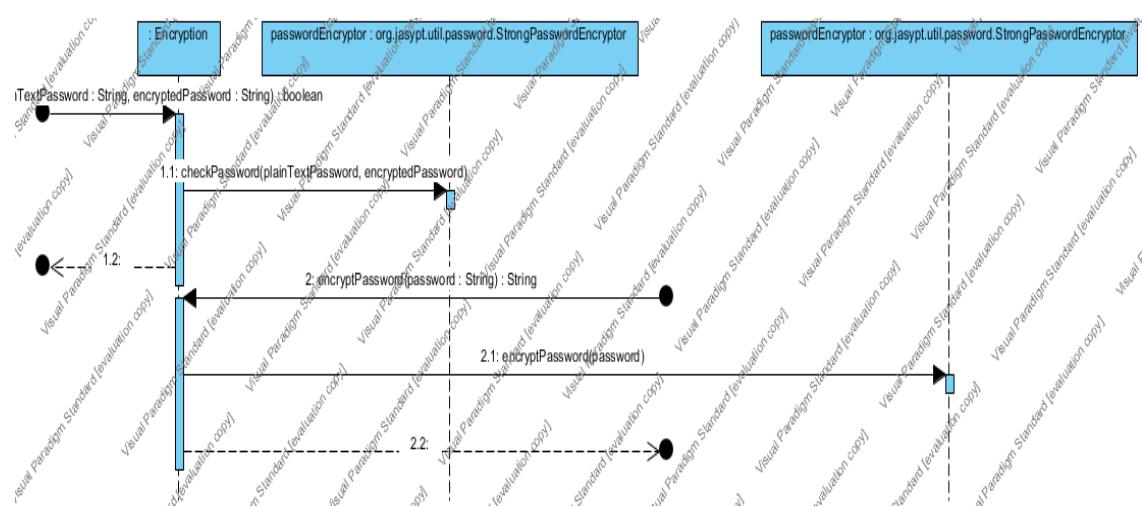
کلاس **Servlet Context Listener** اینترفیس **Application Listener** را همراه با Method های `contextDestroyed()` و `contextInitialized()` پیاده سازی می کند. در متدهای `contextDestroyed()` و `contextInitialized()` زمانی که وب سرور up می شود، این متدهای `contextDestroyed()` و `contextInitialized()` را OPEN Data Base connection با `connection` میکنند و در متدهای `contextDestroyed()` و `contextInitialized()` زمانی که وب سرور Down می شود این متدهای `contextDestroyed()` و `contextInitialized()` با Close Data Base Connection با `connection` را Destroyed کنند. این رویکرد موجب افزایش کارایی و Performance کلی Web Application می شود و به این ترتیب نیازی نیست به ازای هر `method` ای که از لایه Model فراخوانی می شود یک `connection` برای ارتباط با data base باز شود.

Sequence Diagram مربوط به برقراری ارتباط با پایگاه داده :



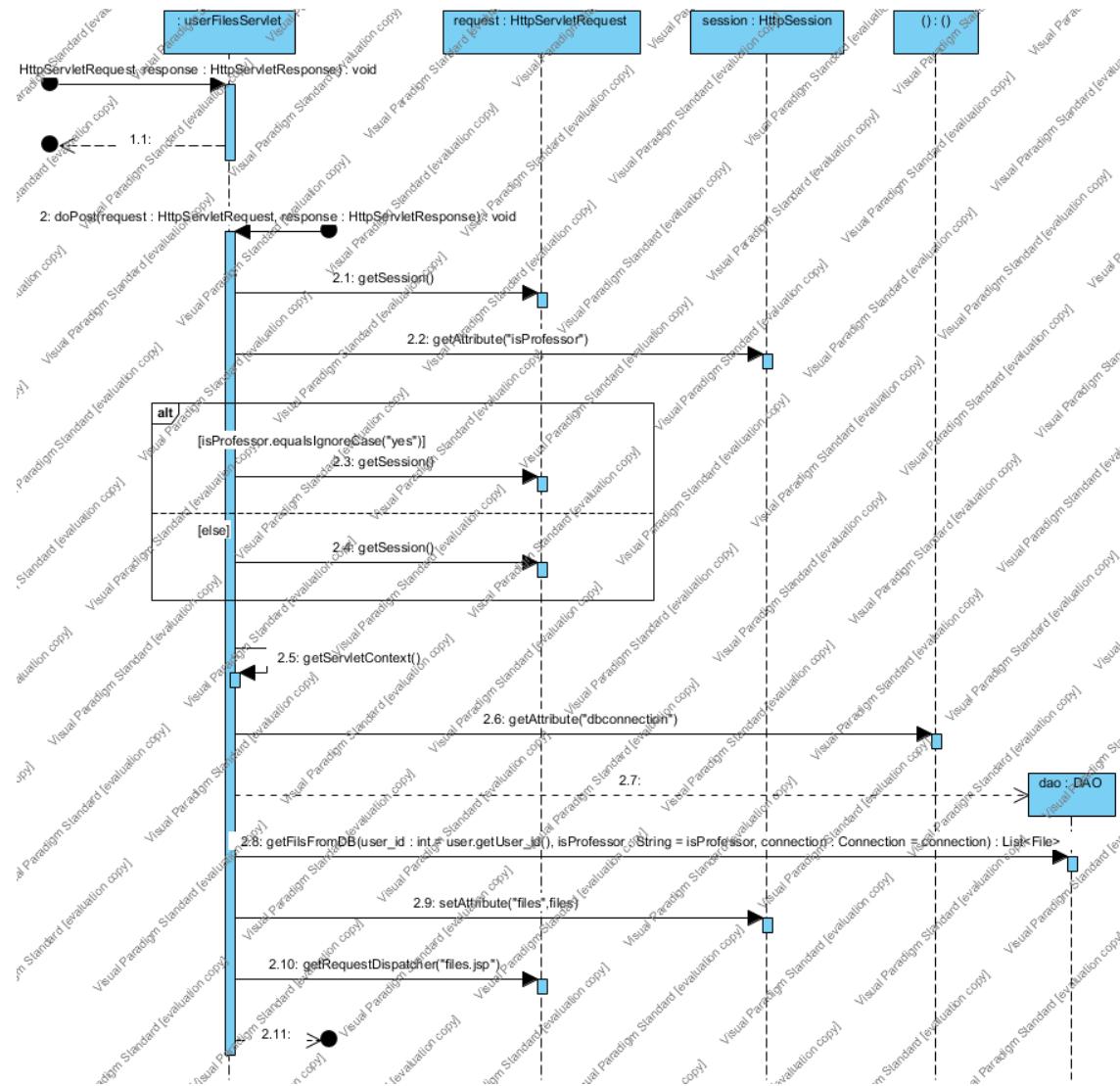
در کلاس Connection یک آبجکت DBConnection تعریف می کنیم و در متدهای register mysql را با url و user name کنیم . یک Connection با پایگاه داده برقرار می کنیم .

Sequence Diagram مربوط به رمزگذاری و Encryption :



در کلاس String ، password Encryptor Object . Encryption و به صورت Encrypt Plain Text است checkPasword می کند و متدهای Encrypt و password وارد شده توسط کاربر را با پسورد رمزگذاری شده تطبیق می دهد برای رمزگذاری پسورد ها از کتابخانه jasypt استفاده می کنیم .

Sequence Diagram مربوط به نمایش لیست فایل های کاربران (دانشجویان و اساتید) :

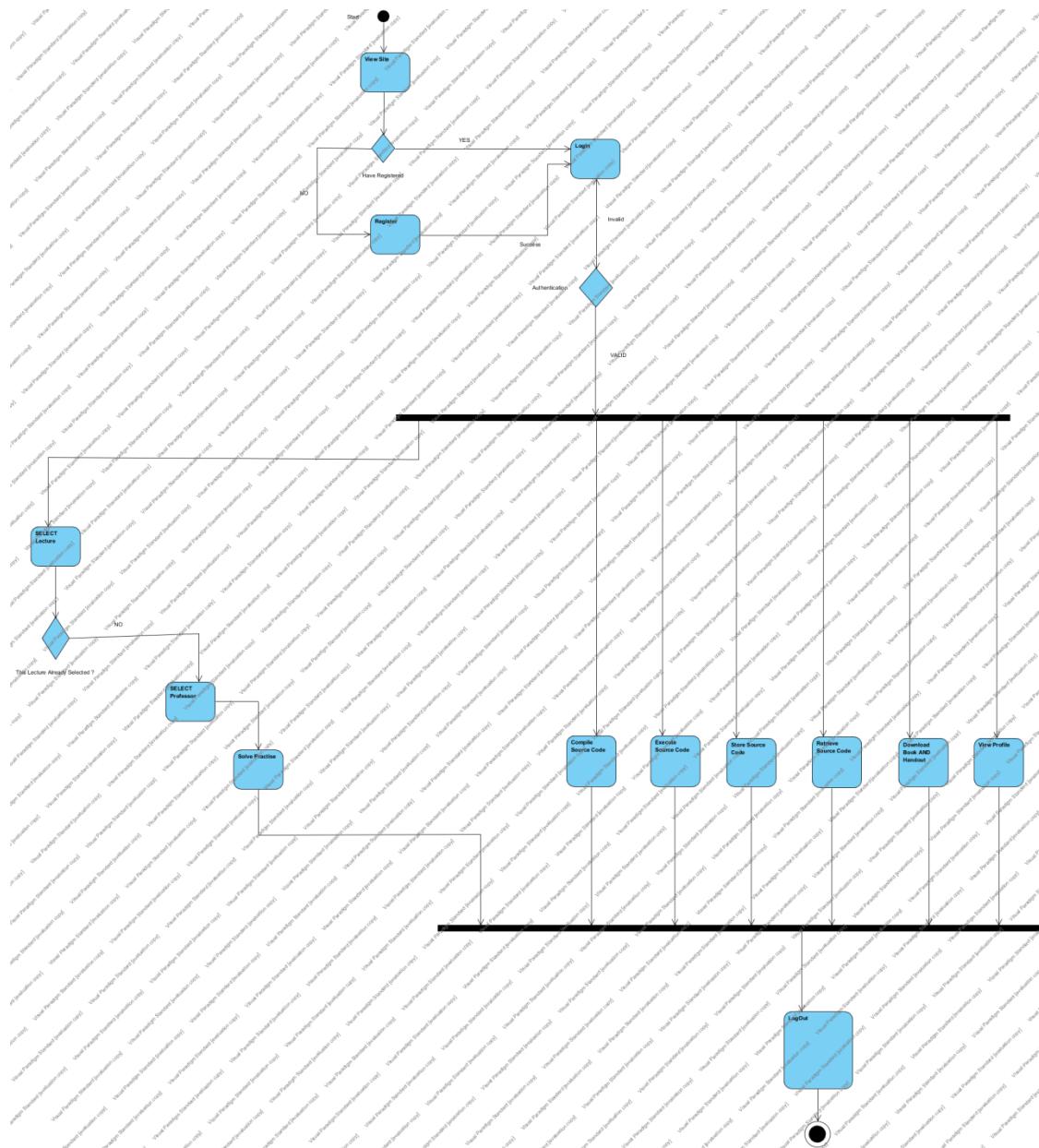


زمانی که کاربر در وب سایت برنامه نویسی Login می کند ، وارد صفحه ی کاربری خود می شود با کلیک بر روی آیکون فایل کلیک می کند و یک Post Request از نوع متده است url /userFilesServlet به url ارسال می شود .

شرح Sequence Diagram از اینجا شروع می شود : در سمت سرور متده `doPost()` کلاس سرولت شرخ اینجا شروع می شود : در سمت سرور متده `doPost()` کلاس سرولت **userFilesServlet** که url pattern این سرولت با url بالا مطابقت دارد و در این متده ابتدا مقدار Attribute `isProfessor` را از `HttpServletRequest` با کلید `getAttribute("isProfessor")` می کنیم و آبجکت `dao:DAO` را روى این آبجکت `getFilesFromDB()` می سازیم و متده `getFilesFromDB()` را روى این آبجکت `get` فراخوانی می کنیم و پارامتر های شناسه ی کاربر و مقدار `isProfessor` را به آن پاس می دهیم ، متده بالا یک لیست از فایل های `User` بر می گرداند و این لیست را در یک آبجکت `Gson` ذخیره می کنیم و به عنوان `response` به سمت کاربر ارسال می کنیم .

شرح Activity Diagram ها :

Activity Diagram مربوط به دانشجو :



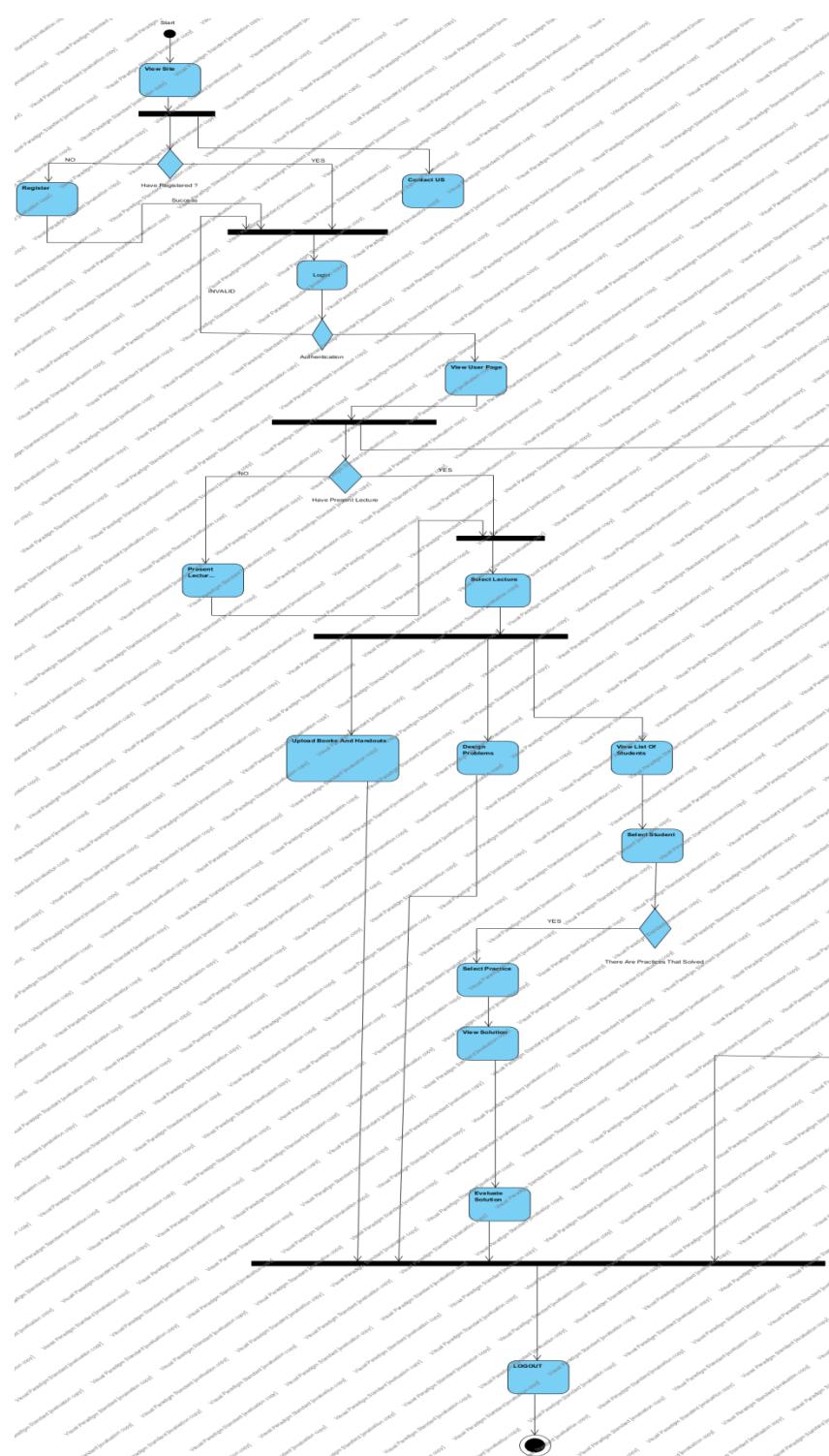
شرح Activity Diagram :

از Initial Node در Activity Diagram شروع می کنیم . زمانی که مرورگر ، وب سایت را بارگذاری می کند ، دانشجو ابتدا وب سایت را مشاهده می نماید و سپس برای انجام Functionality هایی که در سیستم فراهم شده است و استفاده از امکانات وب سایت باید در سیستم Login کند در صورتی که کاربر قبلا در سیستم ثبت نام نکرده باشد حتما باید ثبت نام کند و سپس Login کند . اما در صورتی که قبلا در سیستم ثبت نام کرده باشد ، تنها با وارد کردن نام کاربری و پس کلمه عبور می تواند در سیستم Login کند .

دانشجو بعد از ورود به حساب کاربری خود می تواند با کلیک بر روی آیکون ویرایشگر کد می تواند از محیط برنامه نویسی ای که سایت برای دانشجویان فراهم کرده است ، استفاده کند ، کد و برنامه های کامپیوتری خود را به زبان

های برنامه نویسی سی ، جاوا و پایتون بنویسد و اجرا کند یا در حساب کاربری خود ذخیره کند و یا برنامه هایی که قبلانوشته و در اکانت خود ذخیره کرده است بازیابی نموده و اجرا کند . دانشجو در صفحه‌ی کاربری خود می‌تواند با کلیک بر روی آیکون **Lecture** دروس مورد نظر خود را با اساتیدی که می‌خواهد بردارد و تمرینات طرح شده توسط اساتید را حل کند و همچنین کتاب‌ها و جزوات آموزشی تکمیلی که توسط اساتید آپلود شده است را از صفحه‌ی مربوط به درس خود دانلود کند و در انتهای آنها بعد از اتمام کارش از حساب کاربری خود خارج شود . استفاده از تمامی امکانات وب سایت منوط به ثبت نام کاربر می‌باشد .

استاد : Actor مربوط به Activity Diagram



شرح Activity Diagram : استاد

زمانی که مرورگر کاربر ، وب سایت برنامه نویسی آنلاین را بارگذاری می کند ، اولین صفحه ای که برای استاد نمایش داده می شود صفحه **Login** می باشد و سیستم از کاربر می خواهد ابتدا در وب سایت **Login** کند و سپس از امکانات فراهم شده استفاده کند . درصورتی که استاد در وب سایت برنامه نویسی آنلاین ثبت نام کرده باشد ، با وارد کردن نام کاربری ، پست الکترونیکی و رمز عبور به سایت برنامه نویسی **Login** میکند و وارد حساب کاربری خود می شود ، اما درصورتی که استاد تاکنون در وب سایت برنامه نویسی ثبت نام نکرده باشد با انتخاب گزینه **Register** در **Navigation** بار وارد صفحه ای ثبت نام می شود و با وارد کردن مشخصات خود در وب سایت برنامه نویسی ثبت

نام می کند ، پس از ساختن حساب کاربری می تواند وارد حساب کاربری خود شود . استاد بعد از ورود به حساب کاربری خود با کلیک بر روی آیکون **Lecture** می تواند دروس مد نظرش را ارائه کند و سپس برای دانشجویان هر کدام از درس هایی که ارائه کرده است تمرین برنامه نویسی طرح کند . برای راهنمایی دانشجویان خود و اضافه کردن مطالب و موضوعات تکمیلی برای دانشجویان جزو و کتاب آموزشی آپلود کند و در انتهای انتها بعد از اتمام کار هایش با کلیک بر روی آیکون خروج ، از حساب کاربری خود خارج شود .

فصل سوم

طراحی شامل معماری سیستم مولفه ها و ارتباطات و کلاس دیاگرام

در طراحی وب سایت برنامه نویسی آنلاین از معماری نرم افزار لایه ای استفاده کرده ایم .

MVC Structural Pattern یک رویکرد توسعه‌ی نرم افزار و یک معماری نرم افزار لایه ای است . این معماری نرم افزار از یک ایده‌ی ابتدایی پیروی می‌کند ، ما در هر اپلیکیشن باید مسئولیت‌ها را از هم جدا کنیم .

MVC Architectural Pattern ، اساساً یک معماری سه لایه است . مشخصه‌های اپلیکیشن را از هم مجزا می‌کند

اولین لایه‌ی **MVC** ، به **User Input Logic** مربوط است

دومین لایه‌ی **MVC** ، به **Business Logic** مربوط است .

سومین لایه‌ی **MVC** برای پیاده‌سازی **User Interface Logic** استفاده شده است .

میان این لایه‌یک **Coupling** خیلی سست و شل فراهم می‌کند .

برای مشخص کردن مکان هر **MVC** در **Application Logic** استفاده شده است . الگوی‌های **MVC** سهولت توسعه‌ی موازی را فراهم می‌آورد ، این معنی را می‌دهد که هر لایه‌ی **Application** مستقل از دیگری است ، به عبارت دیگر سه **Developer** می‌توانند به صورت همزمان روی یک **Application** کار کنند . یک **User Developer** روی **Controller Logic** ، **User Input Logic** یا **View Interface Logic** میان این لایه‌یک **Coupling** خیلی سست و شل فراهم می‌کند .

سوم روی **Model** یا **Business Logic** به طور همزمان و موازی کار خواهد کرد . جدا بودن مولفه‌ها و لایه‌ها در **Web Application** تست کردن را برای ما آسان خواهد کرد .

معماری **MVC** به ما کمک می‌کند ، با تقسیم کردن **Application** به سه مولفه ، پیچیدگی را کنترل کنیم . رویکرد توسعه‌ی مبتنی بر تست توسط **MVC** پشتیبانی می‌شود

(۱) **Model View Controller** ، **MVC Pattern** می‌کند : (۲) **View Model** ، **Application** ، **Controller**

برای ارتباط با پایگاه داده استفاده شده است **Model**

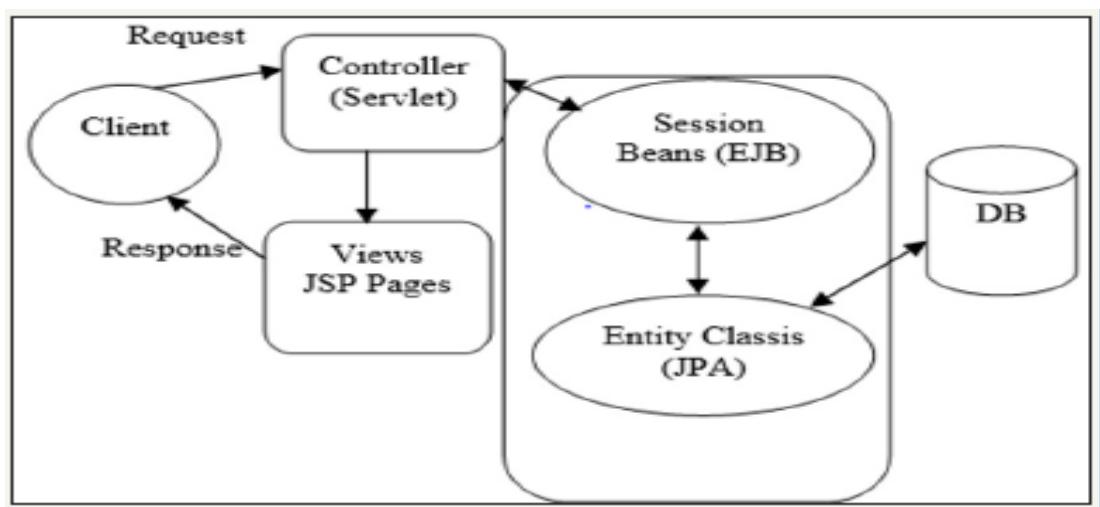
برای گرفتن **Data** از **Model** و نمایش آن به **User** استفاده شده است و در آخر **Controller** برای **Application** استفاده شده است تعریف رفتار

با ورودی‌های کاربران تعامل دارد ، به طور کلی در **Application** های مبتنی بر وب ، ورودی‌های کاربران **Http Post** و **Http Get** هستند .

در این **Java Server Page** ، **Technology** ما با **Web Application** کار می‌کنیم صفحات **Controller** به **Servlet** کامپایل می‌شوند در این معماری کلاس‌های سرولت ما در لایه‌ی **Server Page** قرار دارند که کار دریافت ورودی‌های کاربران ، اعتبار سنجی و تعامل با لایه **Model** را انجام می‌دهند . کاربران

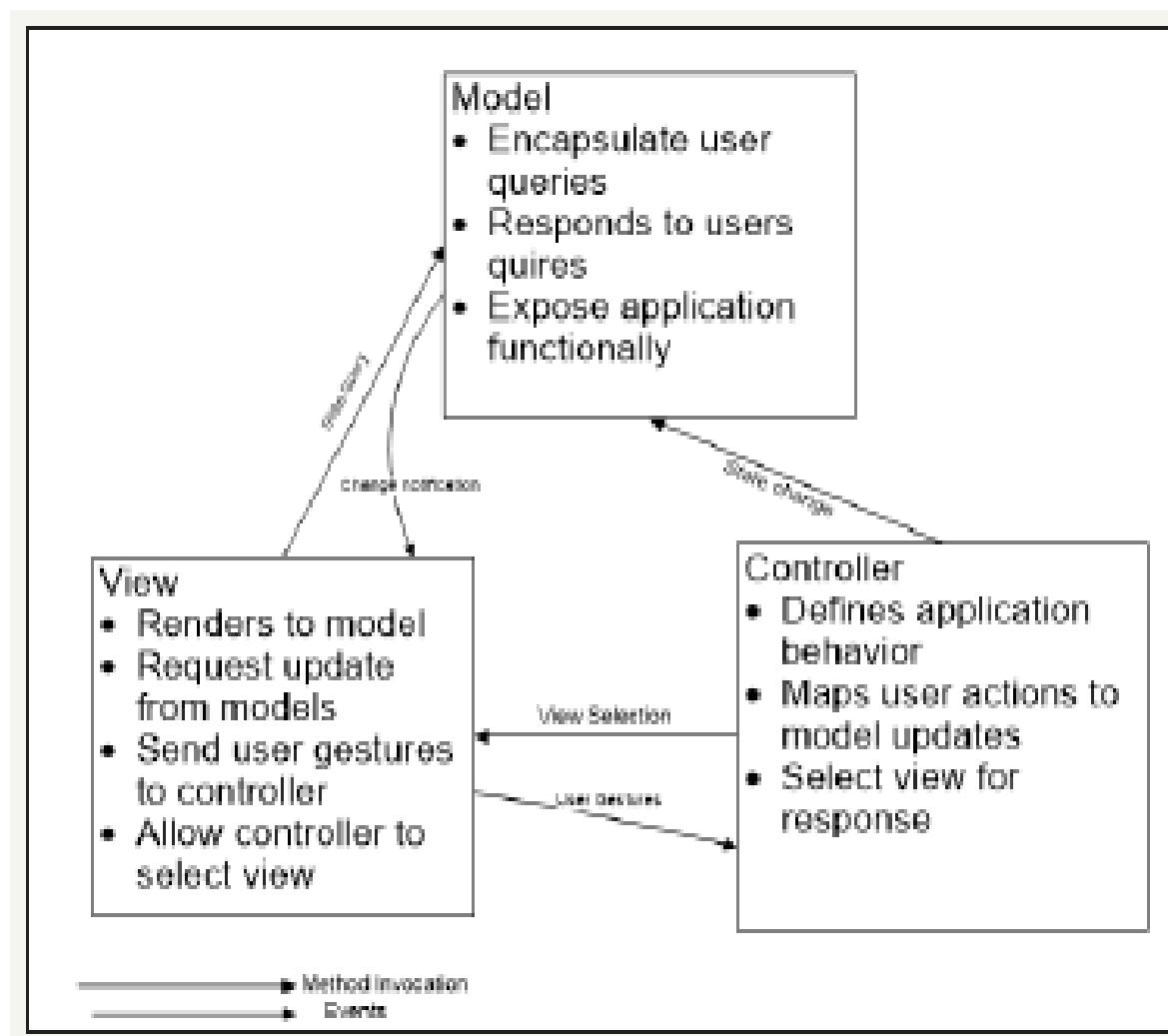
های خود را به لایه **Request** مدهند و **Controller** های خود را از این لایه دریافت می کنند. کد های **User Interface Logic** در لایه **View** یا **CSS** ، **HTML** ، **java script** قرار می گیرند و **وظیفه** **نمايش** **Application** به کاربران را بر عهده دارند. کلاس های **DAO** یا **Data Access Object** و **Model** در لایه **Data Base Connection** قرار دارند و برای درج اطلاعات در جداول پایگاه داده ، بازیابی اطلاعات از جداول پایگاه داده و بروزرسانی اطلاعات در جداول پایگاه داده به کار می روند.

شکل زیر تعامل لایه های معماری نرم افزار **MVC** را نمایش می دهد .



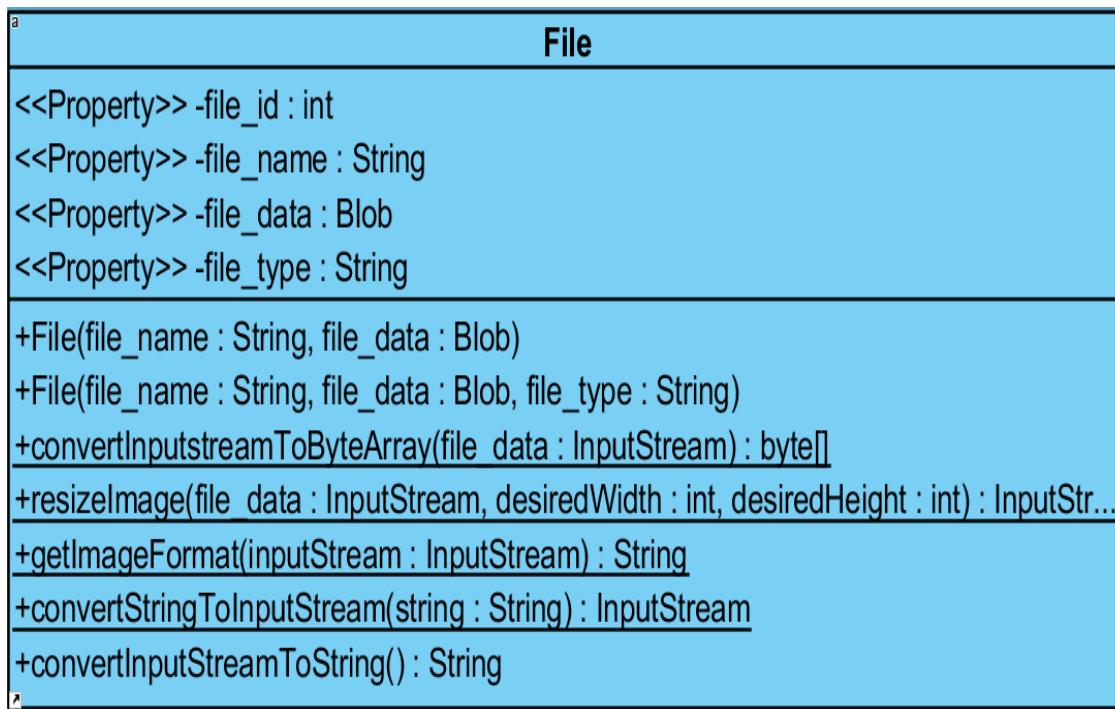
در پیاده سازی وب سایت برنامه نویسی ، کد ها و کلاس های مربوط به هر لایه در پکیج مربوط به آن لایه قرار داده شده است .

هر لایه از معماری نرم افزار **MVC** در شکل زیر نشان داده شده است :



UML Class Diagram مربوط به وب سایت برنامه نویسی آنلاین :

File.java مربوط به کلاس Class diagram



در کلاس فایل ، ما متغیر های نام فایل ، نوع فایل که آنها Data Type است و متغیر دیتا فایل که آن Blob است و متغیر شناسه هی فایل که آن Integer Data Type است را به عنوان آنها Access Modifier که Property ها یا صفات کلاس داریم که

Object Constructor می باشد . در کلاس فایل ، Method های File را داریم که Private فایل هستند که تفاوت آنها در پارامتر های ورودی است . دو متد Overload شده اند . متد convertInputStreamToByteArray

Access می گیرد و دیتای Input Stream را به Byte Array تبدیل می کند و آن Public Modifier است این متد یک متد Static است و فراخوانی آن از طریق ذکر نام کلاس صورت می گیرد و مستقل از Object است . متد استاتیک get Image Format یک پارامتر از نوع Input Stream می گیرد و فرمت آن را به عنوان خروجی بر می گرداند .

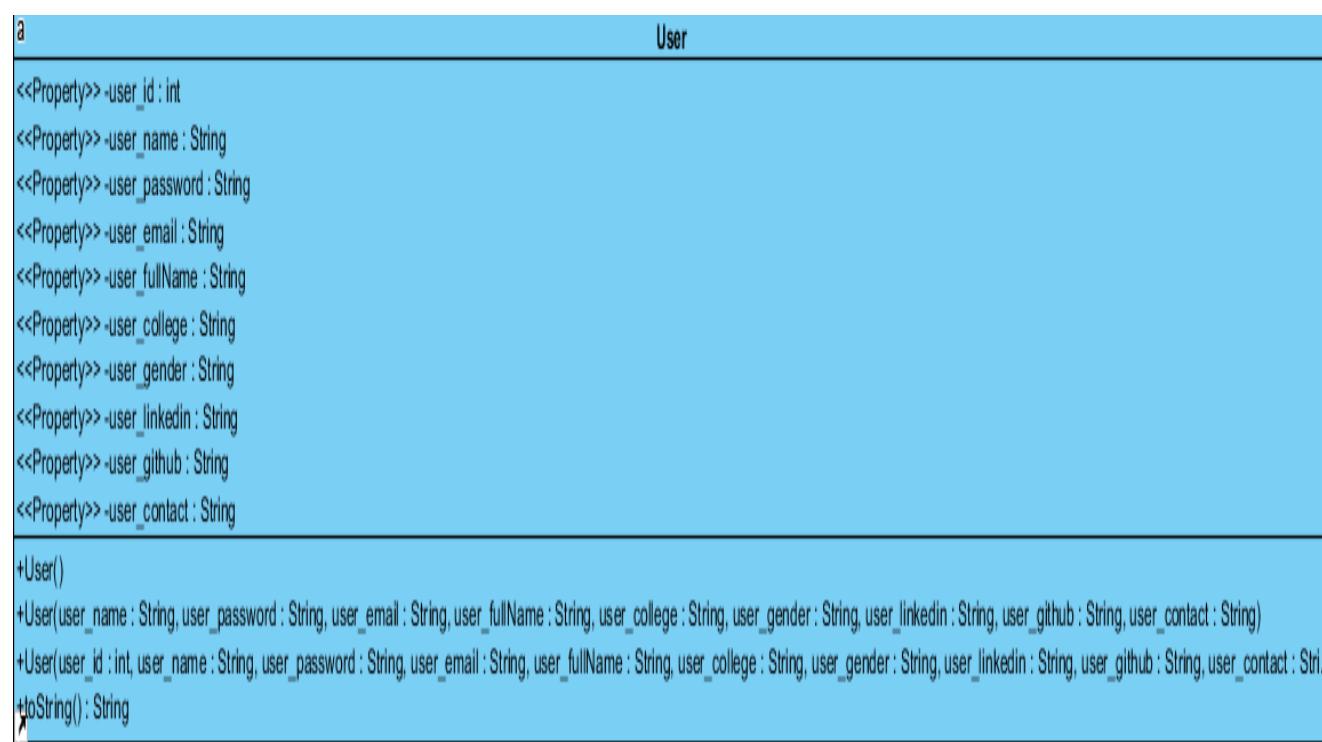
متد استاتیک convert String To Input Stream یک پارامتر String می گیرد و آن را به Stream تبدیل می کند .

متد غیر استاتیک Return Type آن call روی آبجکت convert Input Stream To String می شود و String است .

متده استاتیک **resize Image** یک پارامتر ورودی به عنوان عرض ، یک پارامتر ورودی دیگر به عنوان طول و یک پارامتر ورودی به عنوان دیتای عکس از نوع کلاس **Input Stream** می گیرد .

برای هر کدام از **Property** های کلاس متدهای **Getter** و **Setter** تعریف شده است .

کلاس دیاگرام مربوط به کلاس **User.java** :



متغیر شناسه‌ی کاربر از نوع **Integer** و نام کاربری ، پسورد کاربری ، پست الکترونیکی ، نام و نام خانوادگی کاربر و دانشگاه و جنسیت کاربر ، اکانت لینکدین کاربر ، اکانت گیت‌هاب کاربر و شماره‌ی تماس کاربر از نوع **String** ، فیلد های کلاس یا **Property** ها یا صفات کلاس هستند که **Access Modifier** آنها **Private** است .

این کلاس سه **Method Constructor** دارد که از متده **OverLoad** دارد و یک متده **OverRide** کلاس پدر کلاس‌های جاوا **toString** شده است .

برای هر کدام از **Property** های کلاس متدهای **Getter** و **Setter** تعریف شده است .

این کلاس **Professor** و **Student** برای کلاس‌های **Super Class** یا **Parent Class** است و توسط این کلاس‌ها **Extends** (ارث بری) شده است .

: Student مربوط به کلاس Class Diagram

a	Student
	+Student(user_name : String, user_password : String, user_email : String, user_fullName : String, user_college : String, user_gender : String, user_linkedin : String, user_github : String, user_contact : String)
	+Student(user_id : int, user_name : String, user_password : String, user_email : String, user_fullName : String, user_college : String, user_gender : String, user_linkedin : String, user_github : String, user_contact : String)
x	+Student()

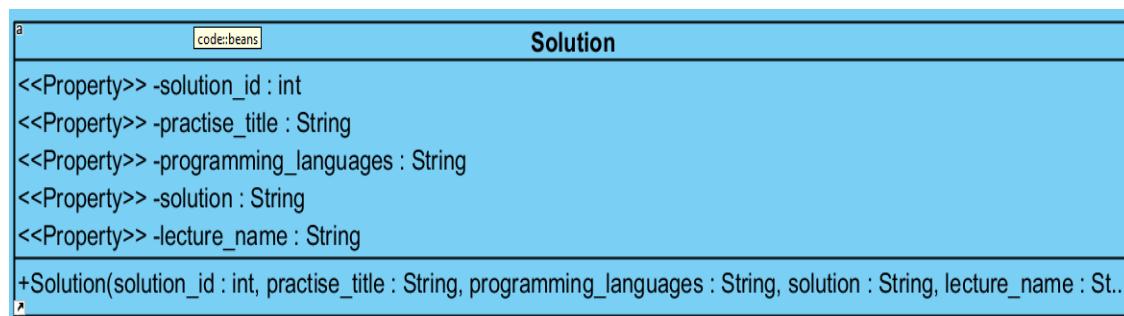
کلاس Student کلاس User را Property کرده است و تمامی ها و متدهای کلاس پدر User را به ارث برده است. کلاس Student می باشد و کلاس User Child Class یا Sub Class . Student Parent Class یا Super Class .

: Professor مربوط به کلاس Class Diagram

a	Professor
	<<Property>> -lectures : List<String>
	-field_study : List<String>
	<<Property>> -practise : List<String>
	+Professor()
	+Professor(user_name : String, user_password : String, user_email : String, user_fullName : String, user_college : String, user_gender : String, user_linkedin : String, user_github : String, user_contact : String)
	+Professor(user_id : int, user_name : String, user_password : String, user_email : String, user_fullName : String, user_college : String, user_gender : String, user_linkedin : String, user_github : String, user_contact : String)
	+setDegree(field_study : List<String>) : void
x	+getDegree() : List<String>

کلاس Professor کلاس User را Field های آن را به ارث برده است . کلاس Professor سه String Practice و Lectures . ArrayList field_study از نوع دارد که آنها private است و برای هر کدام از سه Property با Access Modifier Getter و Setter Overload تعریف شده است . همچنین سه متدهای Constructor دارد که از هم شده اند .

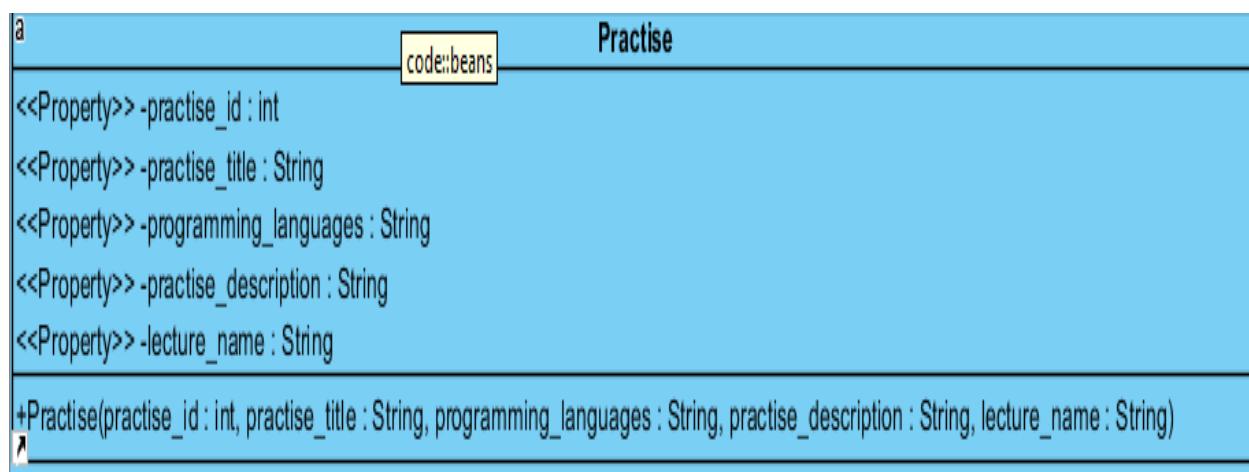
: Solution مربوط به Class Diagram



متغیر های شناسه‌ی راه حل از نوع **Integer** و عنوان تمرین و زبان‌های برنامه نویسی و راه حل و نام درس از نوع **String** ، فیلد‌ها یا **Property**‌های کلاس هستند که آنها **Access Modifier** است .

متد سازنده‌ی کلاس **Solution** است . برای هر کدام از **Property**‌های کلاس **Solution** متد **Getter** و **Setter** تعریف شده است .

practice.java مربوط به کلاس Class Diagram



متغیر های شناسه‌ی تمرین از نوع **Integer** و عنوان تمرین و زبان‌های برنامه نویسی و صورت تمرین و نام درس از نوع **String** ، **Practice**‌های کلاس **Practise** هستند و سطح دسترسی این فیلد‌ها **Private** است و از طریق متد های **Getter** قابل دسترسی هستند و مقدار این **Property**‌ها را می‌توان از طریق متد های **Setter** تغییر داد .

Class Diagram مربوط به DAO

```
a          DAO
+RegisterUser(user_name : String, user_password : String, user_email : String, user_fullName : String, user_college : String, user_gender : String, user_linkedin : String, user_github : String, user_contact : String, isProfessor : String, connection : Connection) : int
+validateUser(user_name : String, user_email : String, user_password : String, isProfessor : String, connection : Connection) : User
+insertTODB(file_name : String, file_data : InputStream, file_type : String, connection : Connection) : int
+pfOrsf(user_id : int, file_id : int, isProfessor : String, connection : Connection) : void
+insertIntoProfessorLecture(professor_id : int, lecture_id : int, connection : Connection) : void
+getFilesFromDB(user_id : int, isProfessor : String, connection : Connection) : List<File>
+getLecturesOfProfessor(professor_id : int, connection : Connection) : List<String>
+retrieveFileFromDataBase(user_id : int, isProfessor : String, file_name : String, connection : Connection) : File
+retrieveUserProfileImage(user_id : int, file_type : String, isProfessor : String, connection : Connection) : File
+insertToContactUSTable(name : String, email : String, message : String, connection : Connection) : int
+selDegreeFieldStudyOfProfessor(professor_id : int, degree_id : int, field_study_id : int, connection : Connection) : void
+getDegreeFieldStudyOfProfessor(professor_id : int, connection : Connection) : List<String>
+getProfessorsThatPresentThisLecture(lecture_name : String, connection : Connection) : List<Professor>
+selectDesiredLectureWithDesiredProfessor(student_id : int, professor_id : int, lecture_id : int, connection : Connection) : int
+selectedLectures(student_id : int, connection : Connection) : List<Professor>
+getPractices(student_id : int, professor_id : int, lecture_id : int, connection : Connection) : List<Practise>
+getPresentedLectures(professor_id : int, connection : Connection) : List<String>
+professorPresentDesiredLecture(professor_id : int, lecture_id : int, connection : Connection) : String
+alreadyPresentLecture(professor_id : int, lecture_id : int, connection : Connection) : boolean
+getListOfStudentOfThisLecture(professor_id : int, lecture_id : int, connection : Connection) : List<Student>
+getPracticesThatSolved(student_id : int, professor_id : int, lecture_id : int, connection : Connection) : List<Practise>
+insertIntoPractiseTable(file_id : int, practise_title : String, programming_languages : String, connection : Connection) : int
+insertIntoProfessorPractiseLectureTable(professor_id : int, practise_id : int, lecture_id : int, connection : Connection) : void
+insertIntoStudentSolutionPractiseLectureProfessor(student_id : int, solution_id : int, practise_id : int, lecture_id : int, professor_id : int, connection : Connection) : void
+insertIntoProfessorFileLecture(professor_id : int, file_id : int, lecture_id : int, connection : Connection) : void
+getFilesFromProfessorFileLecture(student_id : int, lecture_id : int, connection : Connection) : List<File>
+getSolutionOfPractise(student_id : int, practise_id : int, lecture_id : int, professor_id : int, connection : Connection) : Solution
```

این کلاس در پکیج **Model** قرار دارد و تمامی متدهای مربوط به **Data Base Logic** در این کلاس تعریف شده است.

فایل مربوط به **UML Class Diagram** در کنار فایل **Project Document** قرار گرفته است. برای مشاهده جزئیات دقیق می توانید به این فایل مراجعه کنید.

فصل چهارم

پیاده سازی شامل تکنولوژی های مورد استفاده
نکات مربوط به پیاده سازی ، تصاویر سیستم

تکنولوژی استفاده شده در لایه های Controller User Input Logic و لایه های Business Logic یا View User Interface Logic و همچنین لایه های Model

زبان برنامه نویسی جاوا است . زبان جاوا از تکنیک شی گرایی و کپسوله سازی که شامل Object های کپسوله شده است که با رد و بدل کردن و پاس دادن اطلاعات به یکدیگر با هم ارتباط برقرار می کنند . جاوا شامل بررسی های زمان کاپایل و زمان اجرا برای شناسایی خطاهای است . جاوا همچنین دارای خصیصه های امنیتی شبکه است ، قابل حمل است Application بدون کوچکترین اصلاحی ، به راحتی می تواند از یک Platform به یک Platform دیگر منتقل شود

مفسر جاوا مستقل از محیط run time است ، بنابراین ما می توانیم به سرعت برنامه ها را اجرا کنیم . جاوا همچنین یک خودکار را برای آزاد کردن Memory بلاستفاده فراهم کرده است . Object ها یا Reference Variable هایی که ساخته می شود در بخش heap حافظه قرار می گیرند و یک Instance این Object ها ارجاع یا اشاره دارد زمانی که این Reference Variable ها به آدرس دیگری از حافظه اشاره کنند این Garbage Instance ها می شوند و نمی توان از آن استفاده کرد زبان برنامه نویسی جاوا کار برنامه نویس را راحت کرده است و با استفاده از Garbage Collector زبان خود ، آزاد کردن این فضاهای اشغال شده را خود بر عهده گرفته است . جاوا فقط یک زبان برنامه نویسی نیست یک Platform کامل است که شامل دو مولفه است . یک) ماشین مجازی جاوا : یک موتور که دستورالعمل های ترجمه شده توسط کامپایلر جاوا را اجرا می کند . آن به عنوان یک Java Runtime Environment از Instance سیستم عامل ، مرورگر ، وب سرور و نظایر آن گذاشته شده است .

مولفه ی دوم پلتفرم جاوا : Java Application Program Interface است که کدهای از پیش نوشته شده که داخل پکیج هایی از پیش گذاشته شده اند . این پکیج ها شامل کلاس هایی برای ساختن User Interface ها نظیر فونت ، دکمه ، منوها و نظایر آن .

از میان نسخه های زبان برنامه نویسی جاوا یعنی

JAVA Standard Edition

JAVA Enterprise Edition

JAVA Micro Edition

ما از نسخه ی تجاری زبان برنامه نویسی جاوا استفاده می کنیم .

نسخه ی تجاری جاوا تکنولوژی هایی را برای توسعه و مدیریت قوی ، اصولی و مقیاس پذیر و امن Application های سمت سرور فراهم می کند .

تکنولوژی های نسخه ی تجاری زبان برنامه نویسی جاوا به دو بخش تقسیم شده است . یک) تکنولوژی های Web Enterprise Application دو) تکنولوژی های Application

ما در وب سایت برنامه نویسی آنلاین از تکنولوژی های Web Application جاوا یعنی Servlet و JSP استفاده می کنیم .

طراحی Data Model

برای طراحی Data Base و ساخت مدل مفهومی یا Conceptual Model از سیستم ذخیره سازی موجودیت های موجود در سیستم و روابط بین آنها را تعریف می کنیم .

Data ما حاوی پارامتر های طراحی فیزیکی و منطقی است نیاز داریم که با استفاده از Data Model Script یا DDL ، یک ای بنویسیم که در ساخت پایگاه داده و جداول و Constraint های آنها مفید باشد .

از Data Model برای طراحی Data Base Server MySQL استفاده می کنیم .

تمامی دستورات DDL در Sub Directory ،

از /src/ DATA_BASE_SQL_QUERY/Query.sql از پوشه ی پروژه وب سایت برنامه نویسی قرار دارد .

در توسعه ی وب سایت برنامه نویسی ابتدا از Front وب سایت شروع می کنیم و فایل های CSS و HTML و Css و Java server page را می سازیم و در پوشه ی web قرار می دهیم .

از تکنولوژی های View و Html و CSS و همچنین زبان برنامه نویسی جاوا اسکریپت و JSON و AJAX در Front وب سایت استفاده کردیم

سپس کلاس های لایه کنترلر یا User Input Logic یا HttpServlet را برای Handle کردن Request های Http می نویسیم . کلاس های HttpServlet را Extends HttpServlet می کنند

همچنین یک فایل xml به نام WEB-INF/web.xml در مسیر می سازیم . این فایل Application Descriptor حاوی همه ی اطلاعات پیکربندی مربوط به Deployment است .

در لایه ی Data Model یا Business Logic Pattern از DAO استفاده می کنیم . همچنین کلاس Connection را برای ارتباط با Data Base ایجاد می کنیم . برای ارتباط و برقراری Connection با پایگاه داده مای اس کیو ال از Driver MySQL Connector ورژن ۲,۵ استفاده می کنیم ، شماره Port آن را برابر ۳۳۰۶ قرار می دهیم .

تمامی متدهای مربوط به DAO را در کلاس Select ، Update ، Retrieve می نویسیم هر دستور DML را در فرمت String به PreparedStatement پاس می دهیم برای ساختن Statement هایی که توسط JDBC اجرا می شوند .

برای برقراری ارتباط امن میان کاربر و سرور Apache Tomcat ، پیکربندی می کنیم . برای این منظور ابتدا یک کلید keystore می سازیم و یک Certificate برای آن ایجاد می کنیم سپس فایل server.xml موجود در پوشه `conf` سرور آپاچی تامکت را ویرایش می کنیم و port وеб اپلیکیشن را برابر ۴۴۳ قرار می دهیم و همچنین پسورد keystore و باقی اطلاعات را در server.xml می نویسیم .

رمز عبور کاربران که به صورت Plain Text به دست سرور می رسد را با استفاده از کتابخانه `jasypt` می کنیم و در پایگاه داده ذخیره می کنیم . Encrypt

برای کامپایل و اجرای سورس کد های کاربران از Process در جاوا استفاده می کنیم و سورس کد کاربران را در Machine Server Command Line اجرا می کنیم .

با توجه به Session Track برای Http Stateless بودن پروتکل کاربران از URL Hijacking به خاطر مسئله `URI` نمی توانیم از مکانیزم Rewriting استفاده کنیم .

از کامپایلر python و genu c compiler برای کامپایل و تفسیر برنامه های C و python استفاده می کنیم . این جاوا وеб اپلیکیشن روی MohammadRS ، CompilerOnline ، Repository ، اکانت clone Local Repository قرار دارد و با دانلود و نصب git می توانید این پروژه را روی github.com خود کنید .

مربوط به پایگاه داده وب سایت برنامه نویسی : Data Base Diagram

