دانشگاه آزاد اسلامی

واحد تهران جنوب

دانشکده فنی مهندسی

پایان نامه کارشناسی

مهندسی فناوری اطلاعات

عنوان:

طراحی و پیاده سازی سیستم اشتراک فایل تحت اندروید و ویندوز

استاد راهنما:

دکتر صدیقه بختیاری

نام و نام خانوادگی دانشجو:

شایان آریان شعار

شماره دانشجویی:

۸۷۱۱۷۱۱۵۴۵

تابستان ۱۳۹۲

سپاسگذاری

با تشکر از استاد دلسوز و مهربانم، خانم دکتر صدیقه بختیاری که هم در دوران کارشناسی، و هم در نوشتن این پایان نامه از هیچ کمک و راهنمایی دریغ نکردند.

و همچنین خانواده ام، که با فراهم کردن محیطی مناسب، مرا در تمام دوران تحصیلم یاری دادند.

فهرست مطالب

فهرست شکلها:

فهرست جداول:

چکیده

در ابتدا به بررسی فواید برنامه نویسی تحت شبکه و فواید جاوا و اندروید می پردازم. سپس وارد فرایند طراحی سیستم اشتراک فایل با رویکرد RUP و با استفاده از UML خواهم شد. توضیحات مربوط به هر یک از دیسیپلین های RUP و نمودارهای UML در بخش مربوط به خودش داده خواهد شد. زبان انتخابی من برای نوشتن این برنامه جاوا است، دلیل این انتخاب را در ادامه خواهم گفت. بعد از طراحی نوبت به پیاده-سازی و نوشتن کد می رسد (بخش مورد علاقه ی من !). در آخر، خروجی این پروژه، نمودار های UML و دو برنامه ی کاربردی، یکی برای ویندوز و یکی برای اندروید، خواهد بود.

پیش گفتار

در دهه ی گذشته، برنامه نویسی تحت شبکه دیگر در قلمرو تعداد محدودی از متخصصان نبوده و تبدیل به بخش اصلی جعبه ابزار هر برنامه نویس شده است. امروزه تعداد نرم افزار های تحـــت شبکه، از تعـداد برنامه های دیگر بیشتر است! گذشته از برنامه های کلاسیک مثل مرورگرها و ایمیل ها، بیشتر برنامه های کاربردی، سطحی از شبکه گرایی را در خود دارند. برای مثال، ضد ویروس ها برای دریافت اطلاعات ویروس های جدید به سایت خود متصل می شوند، پخش کننده های موزیک، بخشی از فایل صوتی را آپلود می کنند و از پایگاه داده ی خود، اطلاعات مربوط به آن آهنگ را دانلود می کنند و ...

اکنون، ظهور وب سرویس ها بیش از پیش شبکه را در آغوش تمامی انواع نرم افزار ها قرار داده. وقوع همه ی این رویدادها در اینترنت است و تمامی آن می تواند در جاوا نوشته شود! جاوا اولین زبان برنامه نویسی بود که با تمرکز ویژه بر شبکه، طراحی شد. در ابتدا برای شبکه های تلویزیونی اختصاصی طراحی شده بود، اما همیشه در تفکر شبکه به عنوان نخستین امر بود! جاوا همچنین برای مسائل حیاتی نرم افزار های اینترنت، مثل وابستگی به پلتفرم و امنیت، راه حل هایی فراهم می کند. یکی از بزرگترین رازها در مورد جاوا این است که برنامه نویسی تحت شبکه را آسان می کند. در کد برنامه های جاوا مشهود اســـت که قطعه ی ساده تر آن به شبکه اختصاص یافته. حتی در نرم افزار های متمرکز برشبکه، مثل وب سرورها، تقریبا تمامی حجم کد متعلق به واسط کاربر و یا کار با داده می باشد.

شبکه به یک برنامه ی ساده، قدرتی دو چندان می دهد. با شبکه، یک برنامه می تواند اطلاعات را از میلیون ها کامپیوتر که در جای جای کره ی خاکی پراکنده اند، دریافت کند. یک برنامه می تواند با ده ها میلیون انسان ارتباط برقرار کند. یک برنامه می تواند قدرت تعدادی زیادی کامپیوتر را برای حل یک مسئله به کار گیرد. و این تازه شروع است! ...

۱. فصل اول :

بررسی منابع علمی

۱–۱ بررسی فواید برنامه نویسی تحت شبکه با جاوا:

عموما چند شکل از برنامه های تحت شبکه وجود دارد. رایج ترین آنها، سرویس دهنده[1] ها و مشتری[2] ها هستند. در ساده ترین حالت، برنامه ی مشتری، اطلاعات را از یک سرویس دهنده دریافت می کند و آن را نمایش می دهد. برنامه های پیچیده تر، داده را فیلتر و شناسایی می کنند، مرتبا داده های متغییر را دریافت می کنند، اطلاعات را به کامپیوتر ها و افراد دیگر ارسال می کنند، به طور بلادرنگ با چت یا بازی های چند نفره در تعامل اند. سرویس دهنده ها هم به درخواست ها پاسخ می دهند. سرویس دهنده های ساده فایلی را جستجو کرده و آن را در اختیار مشتری قرار می دهند، اما سرویس دهنده های پیچیده تر اغلب قبل از پاسخ به درخواست مشتری، پردازش های زیادی بر روی داده انجام می دهند. برنامه های نظیر به نظیر[3] مثل Gnutella کامپیوتر های زیادی را به هم متصل می کنند، که هر کدام هم مانند سرویس دهنده و هم مثل مشتری عمل می کنند.

در ادامه، نگاهی دقیق تر به قابلیت هایی که شبکه به برنامه ها می افزاید، می اندازیم:

دریافت داده

در ساده ترین حالت، یک مشتری داده را از سرور دریافت می کند. می تواند آن را برای نمایش به کاربر به فرمتی خاص تبدیل کند، آن را در پایگاه داده ی محلی ذخیره کند، با داده های دیگر ترکیب کند، آن را

[1] Server
[2] Client
[3] Peer-to-Peer

تحلیل کند، یا همه ی این موارد. مشتری ها در جاوا می توانند با پروتکل های استاندارد مثل HTTP[۴] ، FTP[۵] یا SMTP[۶] صحبت کنند تا با سرورهای نوشته شده در زبان های مختلف ارتباط برقرار کنند.

ارسال داده

مرورگر های وب برای دریافت داده به صورت بهینه عمل میکنند. یعنی فقط مقدار کمی داده را، اغلب به صورت یک فرم، به سرور ارسال می کنند. برنامه های نوشته شده در جاوا چنین محدودیتی ندارند. وقتی یک ارتباط برقرار شود، برنامه می تواند به همان راحتی که داده را دریافت می کند، ارسال داده را از طریق همان ارتباط انجام دهد، و این فرصت های جدیدی را ایجاد می کند:

- **انبار فایل:**

Applet[۷] ها معمولا نیاز به ذخیره ی اطلاعات در هر بار اجرای خود دارند (مثل ذخیره ی امتیاز بدست آمده در یک بازی). Applet های غیر مطمئن اجازه ی ذخیره ی اطلاعات در هارد دیسک را ندارند، اما میتوانند این اطلاعات را در سرور ذخیره کنند. به این صورت که Applet فقط یک اتصال به سرور را از طریق شبکه برقرار کرده و داده را به آن ارسال می کند. سرور ممکن است داده را از طریق HTTP POST ، FTP، SOAP[۸] یا Servlet[۹] بپذیرد.

- **پردازش موازی حجیم:**

---

[4] Hypertext Transfer Protocol
[5] File Transfer Protocol
[6] Simple Mail Transfer Protocol
[7] برنامه ای کوچک که در مرورگر وب اجرا میشود
[8] Simple Object Access Protocol
[9] برنامه ای در طرف سرور، که به قابلیت های سرور می افزاید

برای حل یک مسئله توسط یک کامپیوتر در زمانی قابل قبول، همیشه مشکلاتی وجود داشته است. گاهی اوقات تصمیم به خریدن کامپیوتری سریع تر (و البته پر هزینه تر) گرفته میشود، اما باید توجه داشت که هر چه به سمت تکنولوژی بالاتر می رویم، سرعت افزایش هزینه از سرعت کامپیوتر ها بیشتر میشود. به طوری که برنامه نویس ها ترجیح می دهند از قدرت چندین کامپیوتر ارزانتر استفاده کنند تا یک ابر کامپیوتر!

وقتی صحبت از پردازش های شبکه ای میشود، جاوا بهترین گزینه برای گروه های کلان از کامپیوتر های کوچک است. از آنجا که جاوا وابسته به پلتفرم[10] نیست، نرم افزار های آن میتوانند در تمام ماشین ها اجرا شوند، نه فقط در تمام ویندوز ها، تمام لینوکس ها یا تمام مکینتاش ها.

تعامل نظیر به نظیر[11] :

تمام موارد بالا، مثال هایی از معماری سرویس دهنده / مشتری هستند. اما در جاوا، مشتری ها در سرتاسر اینترنت میتوانند با یکدیگر نیز ارتباط برقرار کنند که این خود، فرصت های بسیاری را برای گروه-های نرم افزارها ایجاد میکند. اگرچه به دلایل امنیتی، این ارتباط به واسطه ی یک نرم افزار پروکسی در سروری که برنامه ی مشتری از آن دانلود شده، انجام میگیرد.

مشتری ها در جاوا به چند دسته تقسیم میشوند:

- **بازی:**

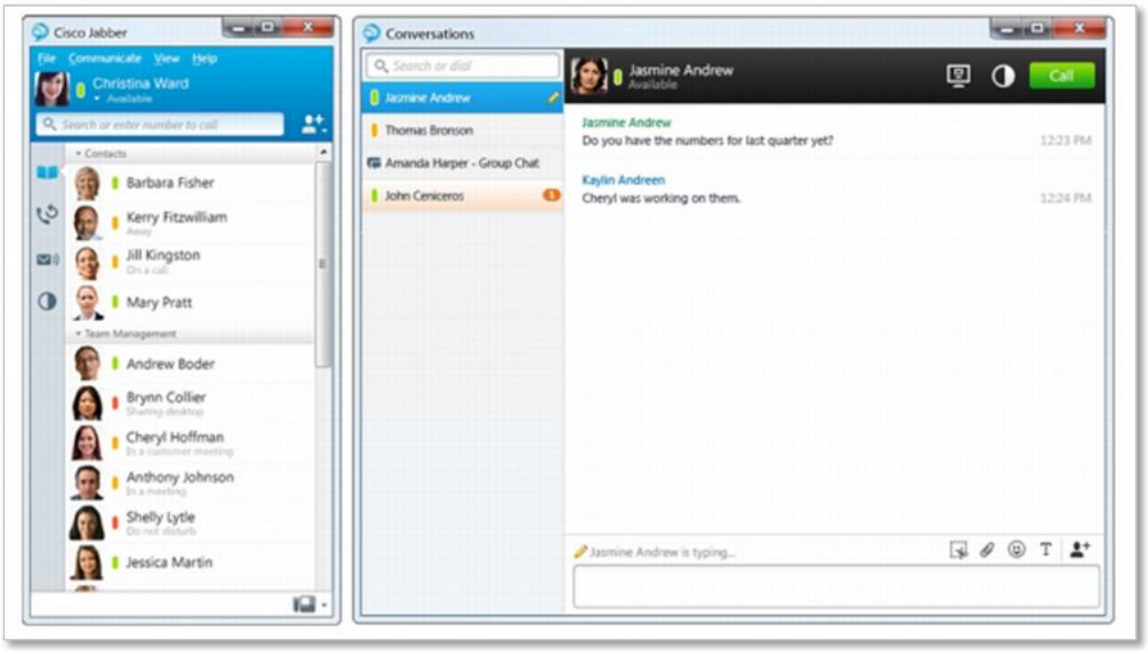
قابلیت شبکه را با گرافیک قدرتمند جاوا در برنامه هایتان توأم کنید، این دستورالعمل ساخت بسیاری از بازی های جذاب چند نفره است.

---

[10] Platform
[11] Peer to Peer

- **چت:**

IRC[۱۲] پروتکل اصلی چت در اینترنت، و دلیل بسیاری از شب بیداری ها در دهه ی ۷۰ است! برنامه هایی نظیر Yahoo! Messenger ، AOL Instant Message و Cisco Jabber همگی در جاوا نوشته شده- اند.



شکل ۱-۱ نرم افزار چت Cisco Jabber

- **اشتراک فایل[۱۳]:**

اشتراک فایل یکی از سه نرم افزار پرکاربرد و اولیـــه ی شبکه است (دو نرم افزار دیگر همان ایمیل و ورود از راه دور[۱۴] می باشند). در گذشته، انتقال فایل نیازمند یک سرور ثابت و در دسترس، با یک آدرس ثابت

[12] Internet Relay Chat
[13] File Sharing
[14] Remote login

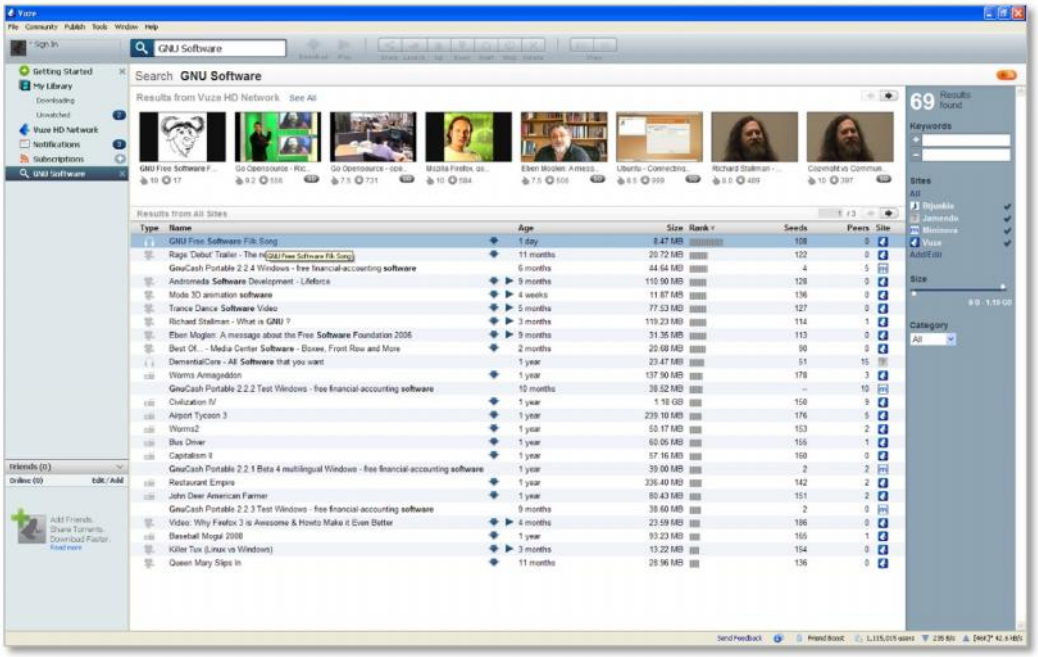
بود. پروتکل های اولیه ی اینترنت مثل FTP، با فرض اینکه سایت ها ۲۴ ساعت در روز و در ۷ روز هفته و در آدرس ثابت در دسترس می باشند، طراحی گردیدند. اما این فرض زمانی امکان پذیر بود که اینترنت اغلب از یونیکس باکس های[۱۵] چند کاربره و سرورهای بزرگ دیگر تشکیل شده بود، تا اینکه کاربران شروع به استفاده از کامپیوتر های شخصی رومیزی برای اتصال به شبکه کردند. این سیستم ها عموماً زمانی در دسترس بودند که یک کاربر در مقابل آنها نشسته باشد. به علاوه، اغلب آنها از اتصال های کند Dial-up استفاده میکردند که همیشه به شبکه متصل نبود، و آدرس های[۱۶]IP داشتند که با هر بار اتصال مجدد آنها به شبکه و یا راه-اندازی مجدد کامپیوتر، تغییر می کرد. گاهی نیز پشت دیوار های آتش[۱۷] یا پروکسی هایی پنهان بودند که اجازه ی اتصال به دنیای خارج را به آنها نمی داد.

وقتی کاربران بخواهند از هر کجا به شبکه متصل شوند و به هر کجا فایل ارسال کنند، ارتباط بین آنها آسان نخواهد بود. اینترنت ماهیاتاً به دو کلاس از کاربران تقسیم می شد: سایت های سرور با پهنای باند بالا، ثابت و همیشه متصل، و مشتری ها با پهنای باند پایین و اتصال های گاه و بی گاه به اینترنت. مشتری ها فقط از طریق یک سرور واسط می توانند با یکدیگر ارتباط داشته باشند. در سال های اخیر، این کلاس بندی شروع به از هم پاشیدن کرد. اتصال های پهنا باند-بالا توسط خطوط DSL[۱۸] رواج یافت. امروزه یک مرکز خرید کوچک، پهنای باندی دارد که شاید ۲۰ سال پیش، دانشگاهی بزرگ به آن حسادت می ورزید. مهم تر از آن، ابتدا Napster و بعد Gnutella، Kazaa، Freenet وBitTorrent پروتکل های انتقال فایل را به گونه-ای توسعه دادند که نیاز به اتصال ثابت به اینترنت ندارند. این پروتکل ها به مشتری های با آدرس متغییر و اتصال های نامنظم و حتی پنهان پشت دیوار های آتش، اجازه ی انتقال بی واسطه ی فایل به یکدیگر را میدهند. به این صورت که دانلود کننده ها هم زمان با دریافت یک فایل، آن را به اشتراک میگذارند یعنی فقط

---

15 Unix Box: ابر کامپیوتر هایی با سیستم عامل یونیکس
16 Internet Protocol
17 Firewall
18 Digital Subscriber Line

چند کاربر اول آن را از وب سایت اصلی دانلود می کنند. خیلی از این نرم افزار ها در جاوا نوشته شده اند. برای مثال LimeWire Gnutella، یک نرم افزار جاوای محض است که از کلاس های استاندارد شبکه در جاوا استفاده می کند. همچنین Azureus Vuze، یکی از بهترین نرم افزار های BitTorrent، کاملاً در جاوا نوشته شده.



شکل ۱-۱ . نرم افزار Azureus Vuze

اشتراک رایگان اطلاعات بین افراد بدون سرور های واسط قابل کنترل، گروه هایی را که تمایل به کنترل اطلاعات دارند، چه به دلایل منفعتی و چه سیاسی، مثل ناشران و اهالی هنر و دولت های مختلف، به ترس واداشت. خیلی از آنها با روش های قانونی یا با استفاده از تکنولوژی، سعی به مسدود کردن شبکه های نظیر به نظیر کردند. در مقابل، افراد خبره با رمز نگاری اطلاعات، طراحی پروتکل های ضد سانسور یا روش های

دیگر، پاسخگوی آنها شدند. یکی از جدی ترین این پروژه ها فری نت[۱۹] ابتکار "یان کلارک[۲۰]" بود. در این پروتکل فایل های کدگذاری شده به چندین فایل تقسیم و به کامپیوتر های مختلفی فرستاده می شدند که حتی نمی دانستند در حال به اشتراک گذاشتن کدام فایل هستند. به علاوه، انتقال فایل از طریق چندین میزبان واسط انجام می پذیرد. این اقدامات احتیاطی، کار پلیس سایبری را برای یافتن این که چه کسی چه فایلی را به اشتراک می گذارد، بسیار مشکل می کند. بار دیگر، پیاده سازی فری نت و بیشتر تحقیق و توسعه-ی آن در جاوا، به عنوان زبان انتخابی، انجام شده است.

سرورها:

برنامه های جاوا میتوانند در انتظار برقراری اتصال از سوی کاربران بمانند و به آنها پاسخ دهند، پس پیاده سازی سرور ها در جاوا امکان پذیر است. علاوه بر سرور های کلاسیک، جاوا روش های سطح بالاتری نیز برای ارتباط بین سرویس دهنده و سرویس گیرنده فراهم میکند. برای مثال، با استفاده از RMI[۲۱] میتوان متودهای اشیائی که در سرور هستند را فرا خواند، یا با Servlet میتوان قابلیت های یک سرور را افزایش داد (افزودن محتوی پویا) و ...

جستجو در وب:

[19] Freenet
[20] Ian Clarke
[21] Remote Method Invokation

برنامه های جاوا میتوانند در جستجوی اطلاعات، وبگردی کنند. این برنامه ها که بر روی سیستم مشتری اجرا میشوند، عنکبوت[۲۲] نام دارند. کار عنکبوت به این صورت است که یک صفحه وب را از یک URL[۲۳] مشخص دانلود میکند، در آن صفحه به شکار اطلاعات میپردازد، سپس URL های موجود در آن صفحه را استخراج میکند و صفحه های مرتبط با آن URL ها را نیز دانلود میکند و این فرآیند همینطور ادامه پیدا میکند. این روش، کم و بیش همان روشی است که در سرویس دهنده هایی مانند گوگل استفاده میشود.

امنیت:

برنامه هایی مانند Applet ها، که کد های فرستاده شده از راه دور را اجرا میکنند، به سطح بالایی از امنیت احتیاج دارند. نگرانی های زیادی راجع به این که کد های فرستاده شده از راه دور، چه میتوانند انجام دهند وجود دارد. برای رفع این نگرانی ها بد نیست بدانید، این کدها:

- نمیتوانند به یک آدرس دلخواه در حافظه دسترسی پیدا کنند.
- نمیتوانند بدون اجازه به فایل های محلی در سیستم دسترسی پیدا کنند.
- نمیتوانند اسناد را چاپ کنند.
- نمیتوانند در clipboard[۲۴] چیزی را ذخیره کنند یا از آن چیزی را بخوانند.
- نمیتوانند برنامه های دیگر را اجرا کنند. به عبارت دیگر نمیتوانند متودهای System.exec() یا Runtime.exec() را فرا بخوانند.

---

[22] Spider
[23] Uniform Resource Locator
[24] قسمتی از حافظه که آخرین اطلاعات کپی شده یا بریده شده، در آن ذخیره میگردند

۲-۱ تاریخچه ی جاوا :

جیمز گُسلینگ، مایک شریدان و پاتریک ناتون، صاحبان شرکت Sun Microsystems، در سال ۱۹۹۱ پروژه ی زبان جاوا را آغاز کردند (توجه کنید که این زبان با زبان جاوا اسکریپت[۲۵] کاملا متفاوت است). در اصل، جاوا برای شبکه های تلویزیونی تعاملی طراحی شده بود، اما در آن زمان، تکنولوژی جاوا پیشرفته تر از آن بود که بخواهد منحصر به صنعت تلویزیون بماند. در ابتدا این زبان را به خاطر درخت بلوطی که روبروی دفتر کار گُسلینک قرار داشت، "بلوط" نامیدند، سپس نام جاوا را برای آن انتخاب کردند (به خاطر قهوه-هایی به همین نام، که توسط سازندگان جاوا بسیار مصرف میشد).

جیمز گسلینگ، بنیانگذار جاوا

سینتکس[۲۶] جاوا شباهت بسیاری به زبان C و C++ دارد، دلیل آن هم، ساده شدن یادگیری جاوا برای توسعه دهندگان (برنامه نویسان) این دو زبان پرکاربرد است. شرکت سان، اولین نسخه ی جاوا را درسال ۱۹۹۵، با شعار "یک بار بنویس، همه جا اجرا کن" منتشر کرد. مرورگرهای بزرگ وب به زودی قابلیت اجرای applet های جاوا را در خود اضافه کردند و جاوا به سرعت عمومی شد. در سال ۱۹۹۸ جاوا ۲ (J2SE[۲۷] 1.2) روانه بازار شد بعد از آن نسخه های مختلفی برای پلتفرم های مختلف عرضه شد. J2EE[۲۸] برای طراحی نرم- افزار شرکت های بزرگ (مثل وب سرور ها) در سال ۱۹۹۹، و J2ME[۲۹] برای ساخت برنامه های کاربردی موبایل در سال ۲۰۰۶، عرضه شدند. این سه نسخه بعدها به Java SE ، Java EE و Java ME تغییر نام دادند. از سال ۱۹۹۷ تا ۲۰۱۱، نسخه های ۱,۱ تا ۱,۷ Java SE عرضه شدند، و قرار بر این است که امسال

[25] Java Script
[26] Syntax
[27] Java 2 Standard Edition
[28] Java 2 Enterprise Edition
[29] Java 2 Mobile Edition

(سال ۲۰۱۳) نسخه ی ۱,۸ آن نیز عرضه شود (البته این نسخه به صورت آزمایشی در اختیار توسعه دهندگان قرار داده شده، و این پروژه نیز با همین نسخه پیاده سازی شده است).

در سال ۲۰۰۷، شرکت سان، سورس کد[۳۰] جاوا را به صورت منبع آزاد[۳۱] در اختیار همه قرار داد. جاوا هر ساله ۹۳۰ میلیون بار توسط کاربران دانلود میشود و در حال حاضر بر روی ۳ میلیارد دستگاه (کامپیوتر های رومیزی، لپ تاپ ها، گوشی ها، مراکز بزرگ داده، کنسول های بازی و ابر کامپیوتر ها) در حال اجراست.

در سال ۲۰۱۰، شرکت سان توسط شرکت "اوراکل" خریداری شد و کمی بعد، جیمز گُسلینگ از اوراکل استفاء داد.

۳–۱ درباره ی اندروید:

اندروید، سیستم عاملی است مبنی بر لینوکس و طراحی شده برای گوشی های هوشمند مجهز به صفحه لمسی و تبلت[۳۲] ها. پروژه ی اندروید در سال ۲۰۰۳، توسط شرکتی به همین نام، با حمایت مالی شرکت گوگل کلید زده شد و دو سال بعد، گوگل این شرکت را خریداری کرد. در سال ۲۰۰۷ اولین نسخه ی آن با نام Cupcake عرضه شد و سال بعد اولین گوشی مجهز به اندروید[۳۳] روانه بازار گردید. تا امروز، ۸ نسخه ی مختلف از اندروید منتشر شده که آخرین نسخه ی آن، Jelly Bean با شماره نسخه ی ۴,۳ ، در جولای ۲۰۱۳ پرده- برداری شد.

[30] Source code
[31] Open Source
[32] Tablet
[33] HTC Dream

اندروید سیستم عاملی با مبنع آزاد است، همین باعث میشود که تولید کنندگان بتوانند آن را متناسب با محصولات خود تغییر دهند، تا آنجا که علاوه بر تلفن های همراه، تلویزیون های هوشمند، دوربین های دیجیتال، ساعت های هوشمند، و حتی نوعی از عینک ها[۳۴] نیز مجهز به اندروید هستند. زبان برنامه نویسی در اندروید نیز **جاوا** است، و همین به محبوب شدن این سیستم عامل برای توسعه دهندگان برنامه های کاربردی می افزاید. طبق تحقیقی که در مِی ۲۰۱۳ درباره ی توسعه دهندگان انجام شد، اندروید در حال حاضر پرطرفدارترین پلتفرم برای توسعه دهندگان است و ۷۱ درصد توسعه دهندگان برنامه های موبایل، از آن استفاده میکنند. اکنون، تعداد دستگاههای مجهز به اندروید، به ۹۰۰ میلیون میرسد، که برنامه های کاربردی موجود در فروشگاه اینترنتی Google Play را ۴۸ میلیارد دفعه دانلود کرده اند!

۴-۱ کمی در باره ی RUP[۳۵] و UML[۳۶]:

RUP رویکردی نوین برای تولید نرم افزار، و گنجینه ای از راهکار های موفق پروژه های نرم افزاری است. در واقع RUP مشخص می کند که در یک فرآیند تولید نرم افزار، چه کسی، چه کاری را، کی و چگونه باید انجام دهد. عناصر کلیدی RUP ، یعنی فعالیت ها، نقش ها و دستاورد ها در قالب تعدادی دیسیپلین دسته بندی شده اند. از جمله ی این دیسیپلین ها می توان دیسیپلین مدلسازی کسب و کار (سازمان)، دیسیپلین نیازمندی ها، دیسیپلین تحلیل و طراحی، دیسیپلین پیاده سازی، دیسیپلین تست، دیسیپلین استقرار، دیسیپلین مدیریت تغییر و پیکر بندی، دیسیپلین مدیریت پروژه، و دیسیپلین محیط را نام برد. در فصل دوم بیشتر با دیسیپلین ها آشنا خواهید شد.

---

[34] Google Glass
[35] Rational Unified Process
[36] Unified Model Language

RUP تاکید زیادی بر مدلسازی دارد و زبان مدلسازی پیشنهادی آن UML است. پیش از این، روش ها و تکنیک های مختلفی برای مدلسازی بصری مورد استفاده قرار می گرفت. از جمله آنها میتوان DFD[۳۷]، ERD[۳۸]، STD[۳۹]، فلوچارت و پتری نت[۴۰] را نام برد. UML زبانی است که توسط مهندسین نرم افزار، به منظور یکپارچه سازی تکنیک های مدلسازی ارائه گردیده. امروزه UML دارای ۱۲ نوع نمودار مختلف است (از جمله نمودار مورد کاربرد، فعالیت، شیء، کلاس و ...) و همانطور که در فصل دوم خواهید دید، هر دیسیپلین RUP از یک یا چند نمودار UML بهره می برد.

[37] Data Flow Diagram
[38] Entity Relationship Diagram
[39] State Transition Diagram
[40] Petri Net

۲.فصل دوم:

روش انجام پروژه

در این فصل، فرآیند تولید سیستم نرم افزاری اشتراک فایل را برای شرکت تراشه (سهامی خاص)، با رویکرد RUP تشریح می کنم و همگام با دیسیپلین های RUP پیش میروم. البته چون RUP رویکردی مبتنی بر تکرار و تکامل تدریجی دارد، فعالیت های این دیسیپلین ها در واقع به ترتیب انجام نمیشوند بلکه همزمان انجام میشوند، اما دستاوردهای آنها را به ترتیب ذکر میکنم. به دلیل اینکه حجم این پروژه کوچک است، فقط ابعادی از سازمان را که مربوط به اشتـراک فایل است در نظر گرفتم و از تعدادی از دیسیپلین های RUP صرف نظر کردم.

۱–۲. دیسیپلین مدلسازی سازمان:

اهداف این دیسیپلین:

۱. درک ساختار و پویایی سازمان
۲. درک مسائل و مشکلات جاری سازمان
۳. استخراج و استنتاج نیازمندی های سازمان
۴. تعیین چشم انداز سازمان

۱-۱-۲. درک ساختار و پویایی سازمان:

در این سازمان بخصوص، در چرخه ی تولید محصول، لازم است که یک سری اطلاعات لازم برای تولید محصولات و مدیریت سازمان در قالب یک سری فایل، بین اشخاص مختلف سازمان از قبیل طراح، اپراتور دستگاه برش، سرپرست تولید، مسئول هماهنگی ها، کارمندان اداری، سرپرست IT[۴۱] و مدیریت سازمان، نقل و انتقال یابند. هر یک از اشخاص مذکور با یک کامپیوتر کار می کنند که می تواند یک کامپیوتر شخصی

[41] Information Technology

رومیزی، لپ تاپ و یا یک تبلت[۴۲] باشد، سیستم های عامل این کامپیوتر ها ویندوز[۴۳]، لینوکس و یا اندروید[۴۴] میباشد. یک فایل معمولا یک طرح برش، یک سفارش تولید، یک گزارش، و یا یک فاکتور میباشد. در حال حاضر شبکه ای در سازمان برقرار نیست و نقل و انتقال این فایل ها به صورت دستی و با فلش[۴۵]، یا کابل USB[۴۶] انجام می شود.

۲-۱-۲. درک مسائل و مشکلات جاری سازمان:

نقل و انتقال دستی فایل در سازمان مشکلات زیر را در پی دارد:

- لازمه ی هر انتقال فایل، رفت و آمد حضوری یک شخص می باشد که خود باعث اتلاف زمان و کاهش سرعت کار و بهره وری می باشد.
- مقصد بعضی انتقالات، تعداد زیادی از اشخاص هستند که این امر موجب انتقال فایل ها به تک تک اعضا می شود و هرچه تعداد اشخاص بیشتر باشد، انتقال نیز بیشتر زمان می برد.
- دستگاه هایی که درگیر انتقال هستند انواع مختلف و سیستم عامل های مختلفی دارند و ارتباط بین آنها آسان نیست. حتی وجود یک شبکه محلی و امکانات خود سیستم عامل ها برای اشتراک فایل کفایت نمی کند.

۳-۱-۲. استخراج و استنتاج نیازمندی های سازمان:

از بین بردن مشکلات مذکور، یک سری نیاز ها را خواستار می باشد. با پاسخ دادن به این نیازها در واقع مشکلات سازمان را از بین برده ایم. از جمله ی این نیاز ها:

---

[42] Tablet
[43] Microsoft Windows
[44] Android
[45] USB Flash
[46] Universal Serial Bus

- احتیاج به یک شبکه ی محلی (LAN[۴۷]) به عنوان بستر ارتباط
- نیاز به سیستمی به عنوان سرور مرکزی که در آن کاربران بتوانند فایل های خود را به اشتراک بگذارند.
- نیاز به اهراز هویت کاربران توسط سیستم، تا هر کسی با ورود به شبکه و داشتن نرم افزار سازمان، نتواند به فایل ها دسترسی پیدا کند یا اینکه آنها را ویرایش کند.
- نیاز به سیستم مدیریت پایگاه داده (DMBS[۴۸]) برای اطلاعات کاربران

۴-۱-۲. تعیین چشم انداز سازمان:

سیستمی که مورد نیاز سازمان است دارای ویژگی های زیر است:

در طرف کاربر :

۱. کاربر می تواند هر زمان که مایل باشد در فایل های به اشتراک گذاشته شده در سرور جستجو کند و یک کپی از فایل مورد نظر را از سرور دانلود کند.
۲. کاربر می تواند هر زمان که مایل باشد، فایلی دلخواه را به سرور ارسال کند.

در طرف سرور :

۳. مدیر سرور قادر است هر زمان که بخواهد، پوشه ای را به اشتراک بگذارد، یا از حالت اشتراک بردارد.
۴. مدیر سرور قادر است کاربری را در سرور تعریف یا از آن حذف کند.

همچنین:

۵. هر کاربر دارای شناسه و رمز عبور است که در سرور تعیین می شود. (**اهراز هویت**).

[47] Local Area Network
[48] Database Management System

۲-۲. دیسیپلین مدیریت نیازمندی ها:

دیسیپلین مدیریت نیازمندی ها مجموعه فعالیت ها، دستاورد ها، و نقش هایی است که با بهره گیری از روش های موفقی مانند مدلسازی نیازمندی ها به وسیله ی نمودار مورد کاربرد[۴۹]، ما را در جمع آوری، سازماندهی، و مدیریت نیازمندی های یک سیستم یاری می دهند.

مراحل این دیسیپلین:

۱- شناسایی اکتور ها[۵۰] و موارد کاربرد

۲- شرح موارد کاربرد

۳- استخراج مدل مورد کاربرد

۴- ایجاد نمونه ای از واسط کاربر.

۱-۲-۲. شناسایی اکتور ها و موارد کاربرد

اکتور یا بازیگر، معرف یک شخص یا یک سیستم است که به عنوان یک موجودیت خارجی با سیستم در ارتباط است. روابط، معمولاً راهی برای شناسایی بازیگران هستند.

اصولاً ۲ موجودیت خارجی با سیستم اشتراک فایل در ارتباط اند: کسی که استفاده کننده ی اصلی سیستم است و به اشتراک گذاشتن فایل می پردازد (کاربر)، و کسی دسترسی به سرور، و کنترل کامل بر حساب کاربران و فایل های اشتراکی دارد (مدیر سیستم[۵۱]).

[49] Use-Case
[50] Actor
[51] Administrator

مورد کاربرد هایی که می توان برای سیستم در نظر گرفت عبارتند از: اهراز هویت، دریافت فایل، ارسال فایل، نمایش لیست کاربران، حذف کاربر، ویرایش کاربر، ثبت کاربر جدید، نمایش لیست پوشه های به اشتراک گذاشته شده، اضافه کردن پوشه و حذف پوشه. البته تعداد این مورد کاربرد ها ممکن است در طول پروژه دستخوش تغییر شود، زیرا همانطور که در فصل ۱ گفته شد، RUP رویکردی مبتنی بر تکرار و تکامل تدریجی دارد.

۲-۲-۲. شرح مورد کاربرد

هر مورد کاربرد، نیاز به اطلاعات و توضیحات کتبی در مورد فرآیند، عملکرد و ویژگی های خود دارد که این اطلاعات در مجموعه ای به نام شرح مورد کاربرد، مستند می گردد. این اطلاعات به عنوان اطلاعات پایه در مراحل تحلیل، طراحی و کدنویسی مورد استفاده قرار می گیرد.

در ادامه، شرح موارد کاربرد طرف کاربر را میبینید:

جدول ۱-۲ شرح مورد کاربرد "اهراز هویت"

نام	اهراز هویت
شماره	۱
نویسنده	شایان آریان
تاریخ آخرین بروز رسانی	۱۳۹۲/۳/۲۰
فرضیات	اتصال به شبکه برقرار باشد سرویس دهنده برقرار و فعال باشد
پیش شرط ها	-
شروع	به محض اجرای نرم افزار طرف کاربر، این مورد کاربرد شروع می شود

گفتگو	از کاربر خواسته می شود شناسه کاربری و رمز عبور خود (و در صورت لزوم، آدرس سرور) را وارد کند. اگر ارتباط با سرویس دهنده برقرار نشود: نمایش پیغام خطای "سرویس دهنده پیدا نشد" در غیر اینصورت شناسه و رمز عبور کاربر در سرور اعتبار سنجی می شوند اگر شناسه و رمز عبور معتبر نبود نمایش پیغام خطای "شناسه یا رمز عبور معتبر نیست" درغیر اینصورت کاربر به سیستم وارد می شود
پایان	در حالت عادی: مورد کاربرد به طریق بیان شده در بخش گفتگو به پایان می رسد. در حالت غیر عادی: ممکن است کاربر انصراف دهد.
پس شرط ها	در صورت خروج عادی، شناسه و رمز عبور برای رمزدار کردن داده ها ذخیره می شوند.

جدول ۲-۲ شرح مورد کاربرد "دریافت فایل"

نام	دریافت فایل
شماره	۲
نویسنده	شایان آریان
تاریخ آخرین بروز رسانی	۱۳۹۲/۳/۲۱

فرضیات	کاربر معتبر باشد اتصال به سرویس دهنده بر قرار باشد
پیش شرط ها	-
شروع	اجرای این مورد کاربرد با انتخاب گزینه ی " دریافت فایل" توسط کاربر، شروع می شود
گفتگو	لیستی از فایل های به اشتراک گذاشته شده به کاربر نمایش داده می شود کاربر فایل مورد نظر را انتخاب می کند اگر فایل در حافظه ی دستگاه کاربر موجود نباشد، آن فایل دانلود می شود در غیر اینصورت: نمایش پیغام "فایل موجود است، آیا بازنویسی شود؟" در صورتی که جواب "بله" باشد، آن فایل دانلود و بازنویسی می شود در غیر اینصورت، مورد کاربرد متوقف می شود
پایان	در حالت عادی: کاربر گزینه ی "بازگشت" را انتخاب کند در حالت غیر عادی: شبکه قطع گردد
پس شرط ها	در صورتی که شبکه قطع گردد (در حین انتقال فایل)، به کاربر اطلاع داده میشود که "به دلیل مشکل در شبکه انتقال انجام نشد"

جدول ۳-۲ شرح مورد کاربرد "ارسال فایل"

نام	ارسال فایل
شماره	۳

نویسنده	شایان آریان
تاریخ آخرین بروز رسانی	۱۳۹۲/۳/۲۱
فرضیات	کاربر معتبر باشد اتصال به سرویس دهنده بر قرار باشد
پیش شرط ها	-
شروع	اجرای این مورد کاربرد با انتخاب گزینه ی " ارسال فایل " توسط کاربر، شروع می شود
گفتگو	یک مرورگر[1]، فایل ها و پوشه های موجود در دستگاه را به کاربر نمایش میدهد کاربر پس از جستجو، فایل مورد نظر را انتخاب می کند اگر فایلی با این نام در سرور موجود نباشد: آن فایل آپلود می شود در غیر اینصورت: نمایش پیغام "فایلی با این نام در سرور موجود است، تغییر نام میدهید؟" اگر جواب "بله" باشد، فایل تغییر نام داده شده و آپلود[2] میشود
پایان	در حالت عادی: فایل ارسال می شود در حالت غیر عادی: کاربر جواب "خیر" دهد یا شبکه قطع گردد
پس شرط ها	در صورتی که شبکه قطع گردد (در حین انتقال فایل)، به کاربر اطلاع داده میشود که "به دلیل مشکل در شبکه انتقال انجام نشد"

[1] Browser
[2] Upload

شرح موارد کاربرد سمت سرور:

جدول ۴-۲ شرح مورد کاربرد "نمایش لیست کاربران"

نام	نمایش لیست کاربران
شماره	۴
نویسنده	شایان آریان
تاریخ آخرین بروز رسانی	۱۳۹۲/۳/۲۲
فرضیات	مدیر سیستم در حال استفاده از این مورد کاربرد است (دسترسی به سرور فقط برای مدیر سیستم امکان پذیر می باشد)
پیش شرط ها	-
شروع	اجرای این مورد کاربرد با انتخاب گزینه ی "نمایش لیست کاربران" توسط مدیر سیستم، شروع می شود
گفتگو	لیستی از شناسه های کاربری نمایش داده می شود کاربر یکی از شناسه ها را انتخاب می کند اگر کاربر گزینه ی "ویرایش کاربر" را انتخاب کند: مورد کاربرد ویرایش کاربر شروع می شود اگر کاربر گزینه ی "حذف کاربر" را انتخاب کند: مورد کاربرد ویرایش کاربر شروع می شود اگر کاربر گزینه ی " ثبت کاربر جدید" را انتخاب کند: مورد کاربرد ثبت کاربر جدید شروع می شود
پایان	در حالت عادی: اگر کاربر هر کدام از ۳ گزینه ی بالا را انتخاب کند

	در حالت غیر عادی: ممکن است کاربر انصراف دهد
پس شرط ها	-

جدول ۵-۲ شرح مورد کاربرد "ویرایش کاربر"

نام	ویرایش کاربر
شماره	۵
نویسنده	شایان آریان
تاریخ آخرین بروز رسانی	۱۳۹۲/۳/۲۲
فرضیات	مدیر سیستم در حال استفاده از این مورد کاربرد است
پیش شرط ها	-
شروع	این مورد کاربرد از طریق مورد کاربرد "نمایش لیست کاربران" شروع می شود
گفتگو	سیستم از کاربر می خواهد شناسه و رمز عبور جدید را وارد کند (رمز عبور جدید ۲ مرتبه وارد شود) ۱-اگر شناسه دیگری با این نام وجود داشته باشد: نمایش پیغام خطای "کاربری با این نام در سیستم موجود است" ۲-اگر دو رمز عبور وارد شده همخوانی نداشته باشند: نمایش پیغام خطای "رمز عبور ها همخوانی ندارند" اگر شرط ۱ و ۲ برقرار نبود، اطلاعات کاربر مورد نظر در پایگاه داده ی سرور بروز رسانی می شود و پیغام "ویرایش انجام شد" نمایش داده می شود
پایان	در حالت عادی: ویرایش انجام میشود و مورد کاربرد پایان می یابد

	در حالت غیر عادی: ممکن است کاربر انصراف دهد
پس شرط ها	-

جدول ۶-۲ شرح مورد کاربرد " ثبت کاربر جدید"

نام	ثبت کاربر جدید
شماره	۶
نویسنده	شایان آریان
تاریخ آخرین بروز رسانی	۱۳۹۲/۳/۲۲
فرضیات	مدیر سیستم در حال استفاده از این مورد کاربرد است
پیش شرط ها	-
شروع	این مورد کاربرد از طریق مورد کاربرد "نمایش لیست کاربران" شروع می شود
گفتگو	سیستم از کاربر می خواهد شناسه و رمز عبور کاربر جدید را وارد کند (رمز عبور جدید ۲ مرتبه وارد شود) ۱-اگر شناسه دیگری با این نام وجود داشته باشد: نمایش پیغام خطای " کاربری با این نام در سیستم موجود است " ۲-اگر دو رمز عبور وارد شده همخوانی نداشته باشند: نمایش پیغام خطای "رمز عبور ها همخوانی ندارند" اگر شرط ۱ و ۲ برقرار نبود، اطلاعات کاربر مورد نظر در پایگاه داده ی سرور ثبت می شود و پیغام "کاربر ثبت شد" نمایش داده می شود
پایان	در حالت عادی: ثبت کاربر انجام میشود و مورد کاربرد پایان می یابد

	در حالت غیر عادی: ممکن است کاربر انصراف دهد
پس شرط ها	ثبت کاربر در پایگاه داده تأیید شود

جدول ۷-۲ شرح مورد کاربرد " حذف کاربر"

نام	حذف کاربر
شماره	۷
نویسنده	شایان آریان
تاریخ آخرین بروز رسانی	۱۳۹۲/۳/۲۲
فرضیات	مدیر سیستم در حال استفاده از این مورد کاربرد است
پیش شرط ها	-
شروع	این مورد کاربرد از طریق مورد کاربرد "نمایش لیست کاربران" شروع می شود
گفتگو	از کاربر پرسیده می شود که آیا از حذف کردن کاربر مورد نظر از سیستم مطمئن است؟ اگر جواب "بله" باشد، نام کاربر از پایگاه داده ی سیستم حذف شده و پیغام "کاربر حذف شد" نمایش داده می شود در غیر این صورت، مورد کاربرد متوقف می شود
پایان	در حالت عادی: حذف کاربر انجام میشود و مورد کاربرد پایان می یابد در حالت غیر عادی: ممکن است کاربر انصراف دهد
پس شرط ها	حذف کاربر از پایگاه داده تأیید شود

جدول ۸-۲ شرح مورد کاربرد "نمایش پوشه های اشتراکی"

نام	نمایش پوشه های اشتراکی
شماره	۸
نویسنده	شایان آریان
تاریخ آخرین بروز رسانی	۱۳۹۲/۴/۸
فرضیات	مدیر سیستم در حال استفاده از این مورد کاربرد است
پیش شرط ها	
شروع	اجرای این مورد کاربرد با انتخاب گزینه ی "نمایش پوشه های به اشتراک گذاشته شده" توسط مدیر سیستم، شروع می شود
گفتگو	لیستی از پوشه های به اشتراک گذاشته شده نمایش داده می شود با انتخاب گزینه ی "اضافه کردن پوشه" توسط کاربر، مورد کاربرد "اضافه کردن پوشه" اجرا می شود با انتخاب گزینه ی "حذف پوشه" توسط کاربر، مورد کاربرد "حذف پوشه" اجرا می شود
پایان	در حالت عادی: اگر کاربر هر کدام از ۲ گزینه ی بالا را انتخاب کند در حالت غیر عادی: ممکن است کاربر انصراف دهد

جدول ۹-۲ شرح مورد کاربرد " اضافه کردن پوشه"

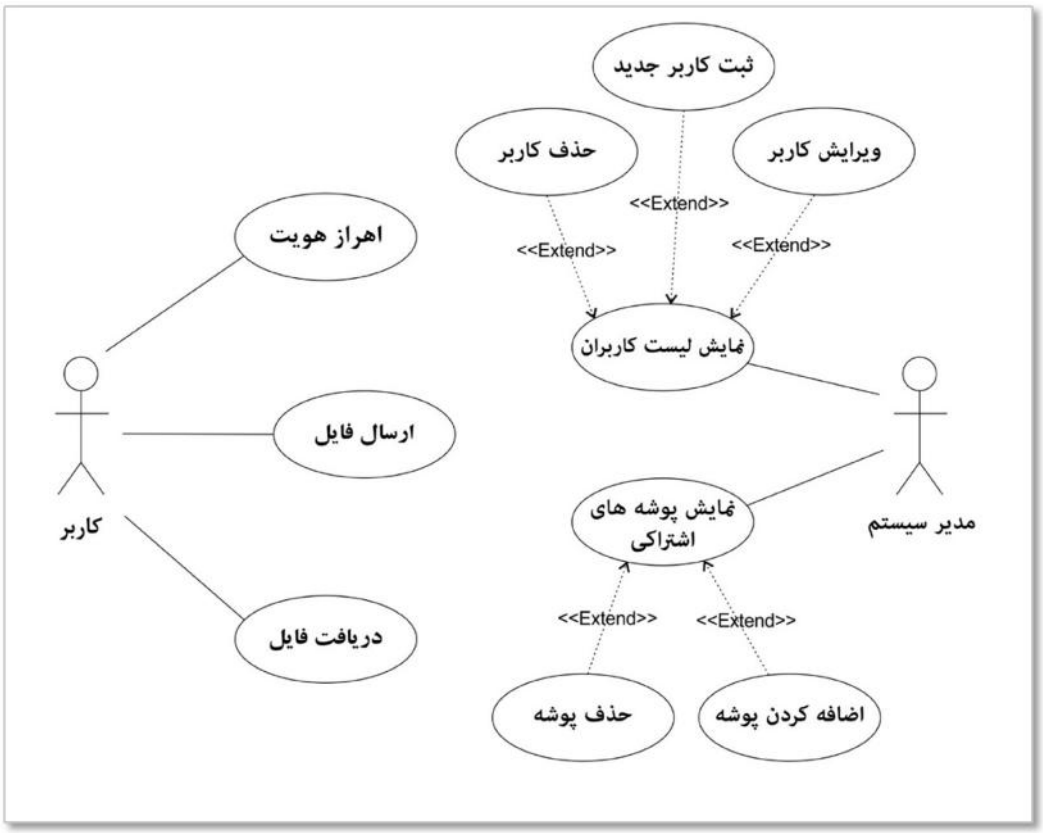
نام	اضافه کردن پوشه
شماره	۹
نویسنده	شایان آریان
تاریخ آخرین بروز رسانی	۱۳۹۲/۴/۸
فرضیات	مدیر سیستم در حال استفاده از این مورد کاربرد است
پیش شرط ها	
شروع	اجرای این مورد کاربرد با انتخاب گزینه ی " اضافه کردن پوشه" توسط مدیر سیستم، شروع می شود
گفتگو	یک مرورگر نمایش داده شده که در آن کاربر میتواند پوشه ی مورد نظر را جستجو کند اگر پوشه ای انتخاب شود، آن پوشه به لیست پوشه های اشتراکی اضافه میشود
پایان	در حالت عادی: پوشه ای به اشتراک گذاشته می شود در حالت غیر عادی: ممکن است کاربر انصراف دهد
پس شرط ها	

جدول ۱۰-۲ شرح مورد کاربرد "حذف پوشه"

نام	حذف پوشه
شماره	۱۰
نویسنده	شایان آریان
تاریخ آخرین بروز رسانی	۱۳۹۲/۴/۸
فرضیات	مدیر سیستم در حال استفاده از این مورد کاربرد است
پیش شرط ها	
شروع	اجرای این مورد کاربرد با انتخاب گزینه ی " حذف پوشه" توسط مدیر سیستم، شروع می شود
گفتگو	کاربر پوشه ای را انتخاب می کند از کاربر پرسیده می شود که آیا پوشه حذف شود؟ اگر جواب بله باشد، پوشه دیگر به اشتراک گذاشته نمی شود
پایان	در حالت عادی: پوشه ای از حالت اشتراک خارج می شود در حالت غیر عادی: ممکن است کاربر انصراف دهد
پس شرط ها	

۳-۲-۲. استخراج مدل مورد کاربرد

با اطلاعات داده شده در شرح موارد کاربرد، رسم نمودار مورد کاربرد سیستم کار دشواری نیست:

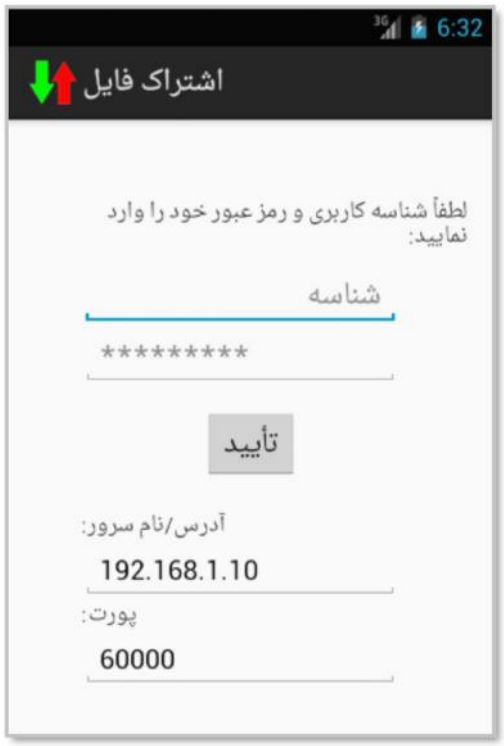


شکل ۱-۲-۲ نمودار مورد کاربرد سیستم اشتراک فایل

۴-۲-۲. ایجاد نمونه ای از واسط کاربر:

در این قسمت، یک نمونه ی ظاهری از واسطه های گرافیکی کاربران[3] می سازیم. این نمونه ای است برای توافق میان ذینفعان سیستم، و ممکن است واسطه ها در انتهای پروژه کمی دستخوش تغییر شوند.

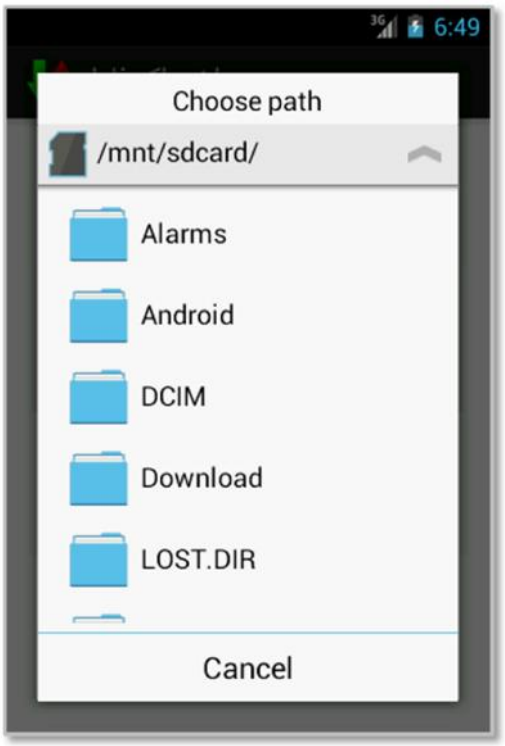
[3] Graphical User Interfaces



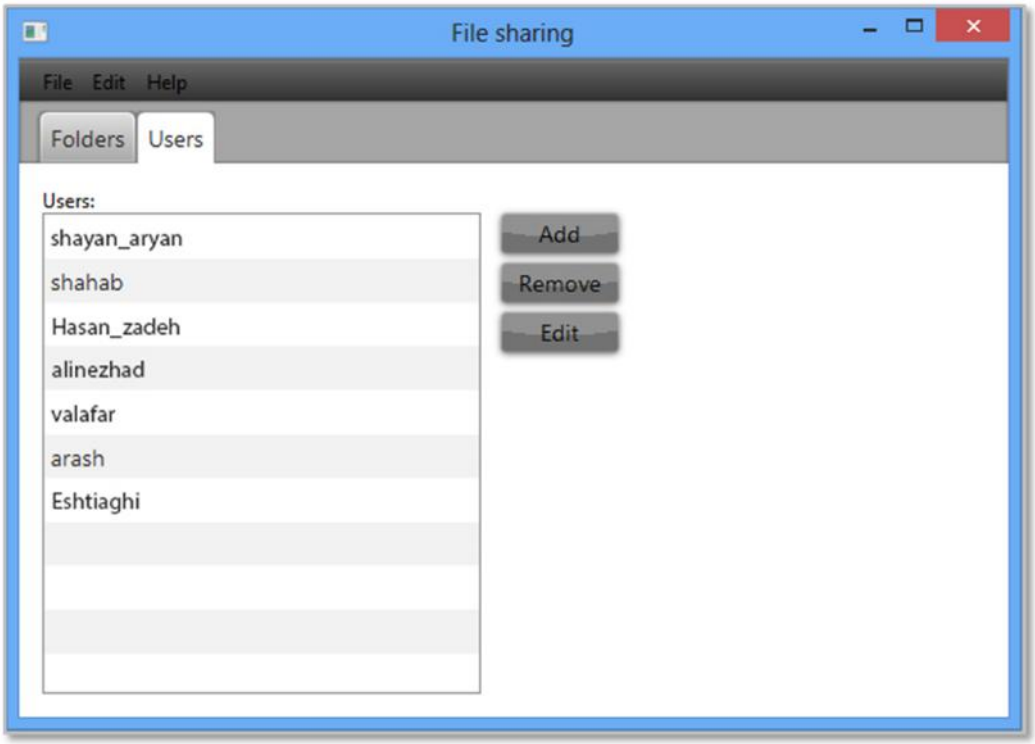
شکل ۲-۲-۲ صفحه ی نخست نرم افزار کاربر (متناظر با مورد کاربرد احراز هویت)

شکل ۳-۲-۲ صفحه ی انتخاب دریافت یا ارسال فایل

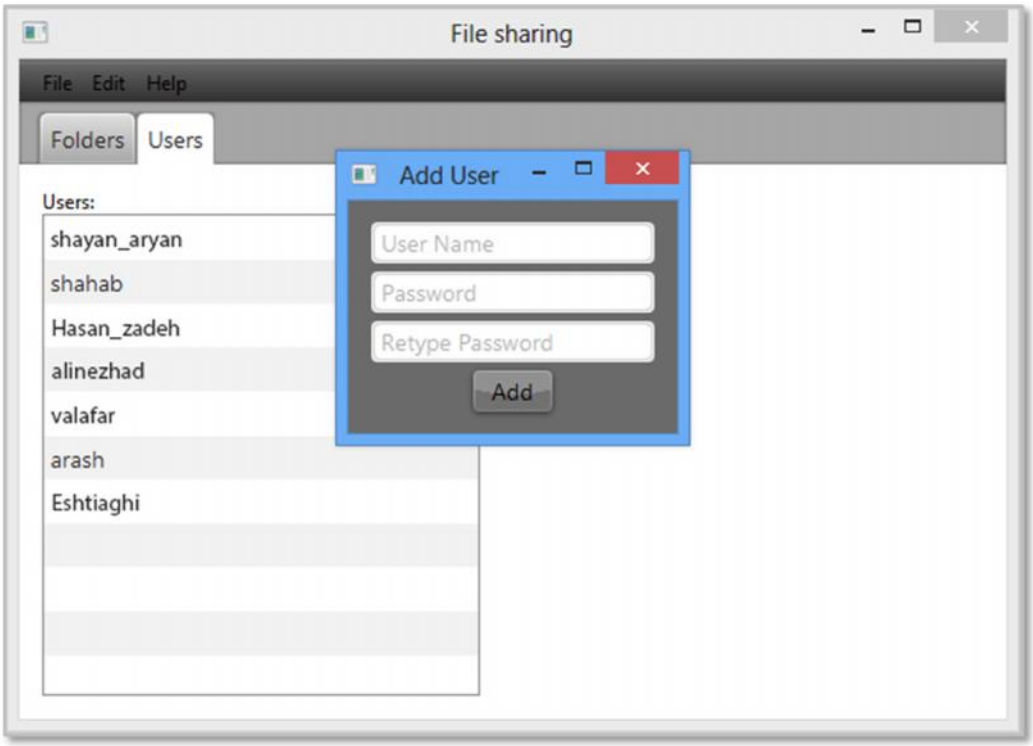
شکل ۴-۲-۲ صفحه ی انتخاب فایل برای دریافت (متناظر با مورد کاربرد "دریافت فایل")



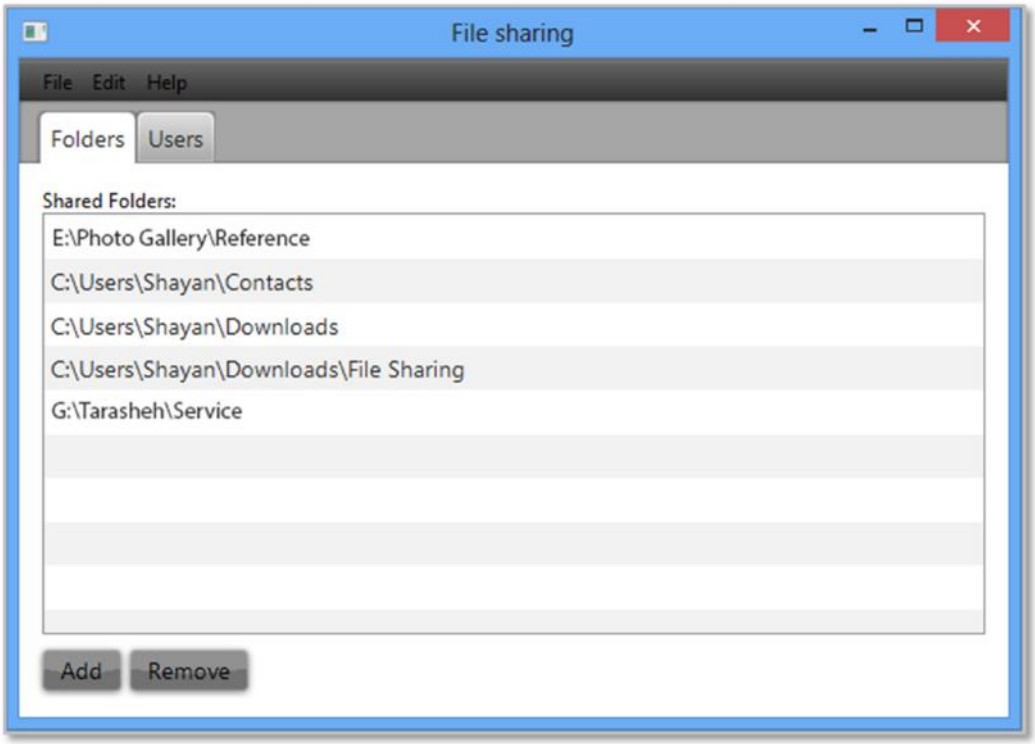
شکل ۵-۲-۲ صفحه ی انتخاب فایل برای ارسال (متناظر با مورد کاربرد "ارسال فایل")



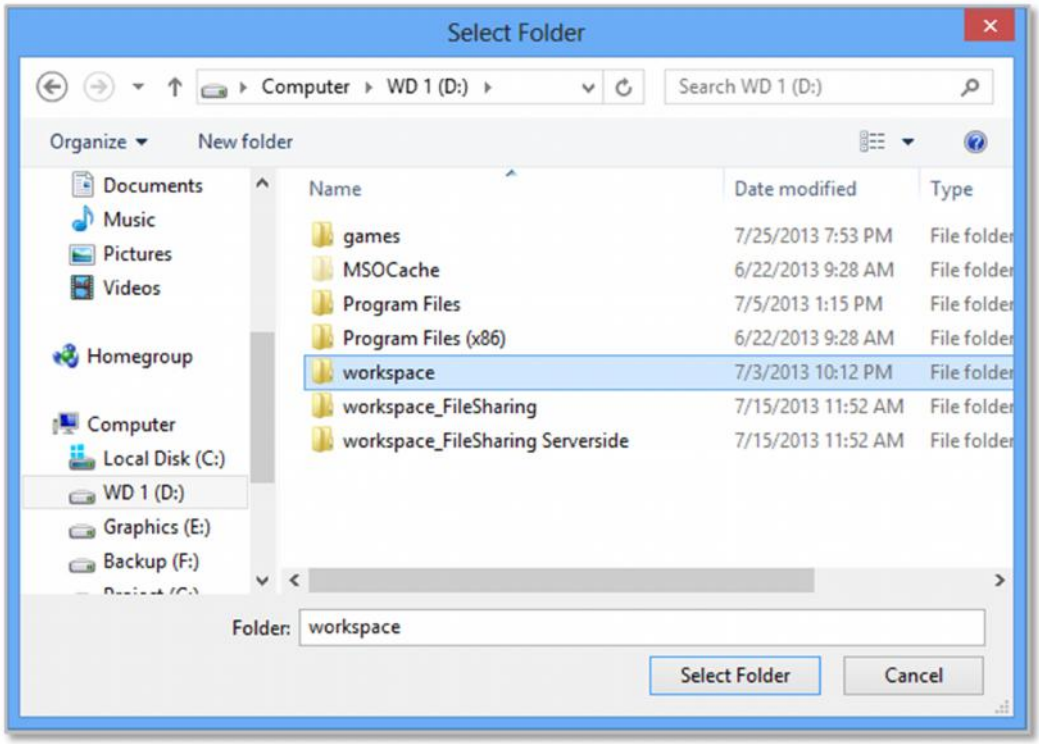
شکل ۶-۲-۲ نرم افزار طرف مدیر سیستم – صفحه متناظر با مورد کاربرد " نمایش لیست کاربران "



شکل ۷-۲-۲ پنجره ی ثبت کاربر جدید (متناظر با مورد کاربرد "ثبت کاربر")



شکل ۸-۲-۲ صفحه ی لیست پوشه های اشتراکی (متناظر با مورد کاربرد "نمایش پوشه های اشتراکی")



شکل ۹-۲-۲ پنجره ی انتخاب پوشه ی جدید برای اشتراک (متناظر با مورد کاربرد "اضافه کردن پوشه")

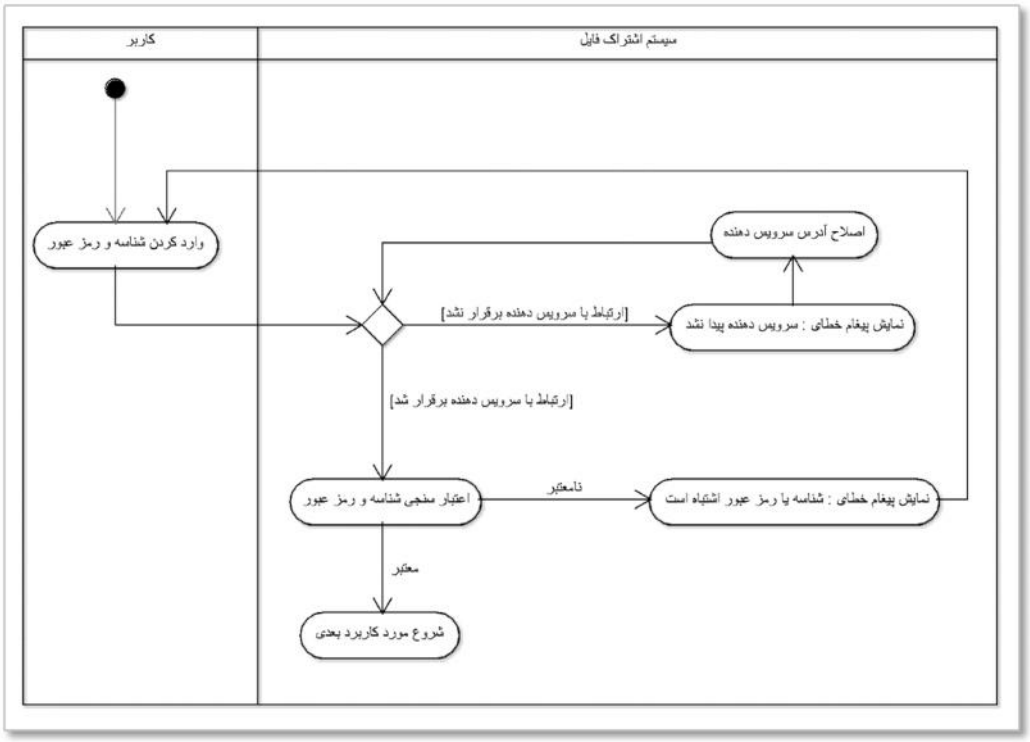
۳-۲ دیسیپلین تحلیل و طراحی:

هدف اصلی دیسیپلین تحلیل و طراحی، انتقال نیازمندی های سیستم از زبان و دنیای مشتری به زبان و دنیای تیم توسعه است. تحلیل، با معرفی سیستم به صورت یک جعبه ی خاکستری از اجزای درونی آن، بدون وارد شدن به جزئیات پیاده سازی، چگونگی تحقق موارد کاربرد سیستم را نشان می دهد. با رفتن از نیازمندی ها به تحلیل، طراحی و پیاده سازی، به تدریج به جزئیات مورد نیاز اضافه شده و میزان تجرید[۴] کمتر می شود.

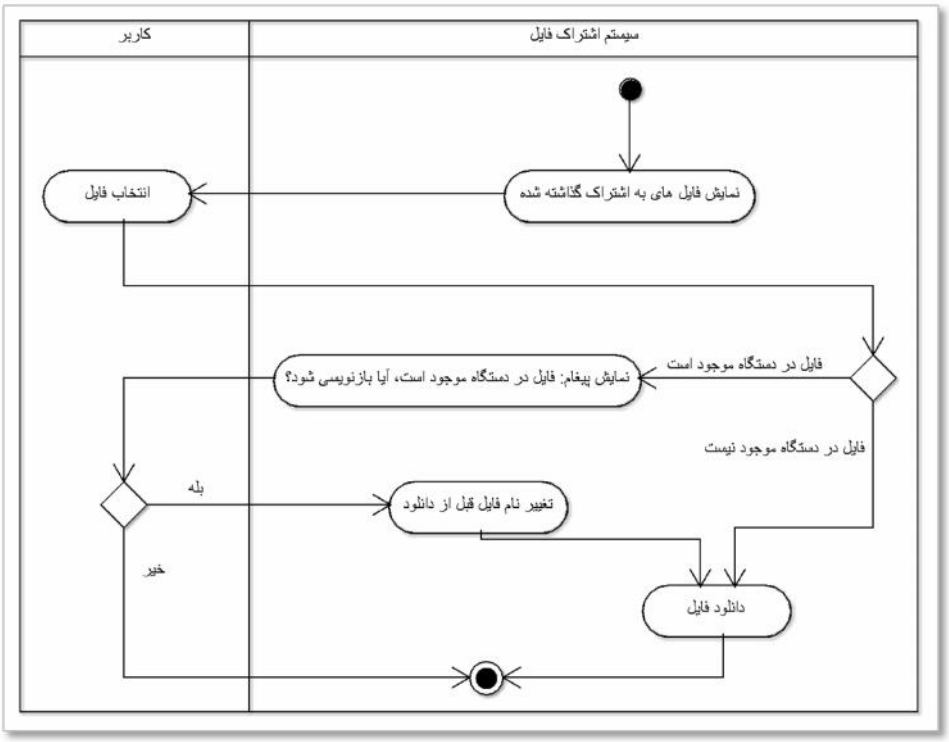
۱-۳-۲ نمودار فعالیت:

نمودار فعالیت، نسخه ی توسعه یافته ی فلوچارت است که در UML مورد استفاده قرار می گیرد. این نمودار در واقع بازنمایی بصری شرح موارد کاربرد است و روند منطقی اجرای عملیات و گردش های کار را نشان می دهد. در ادامه، نمودار های فعالیت سمت کاربر را میبینید:

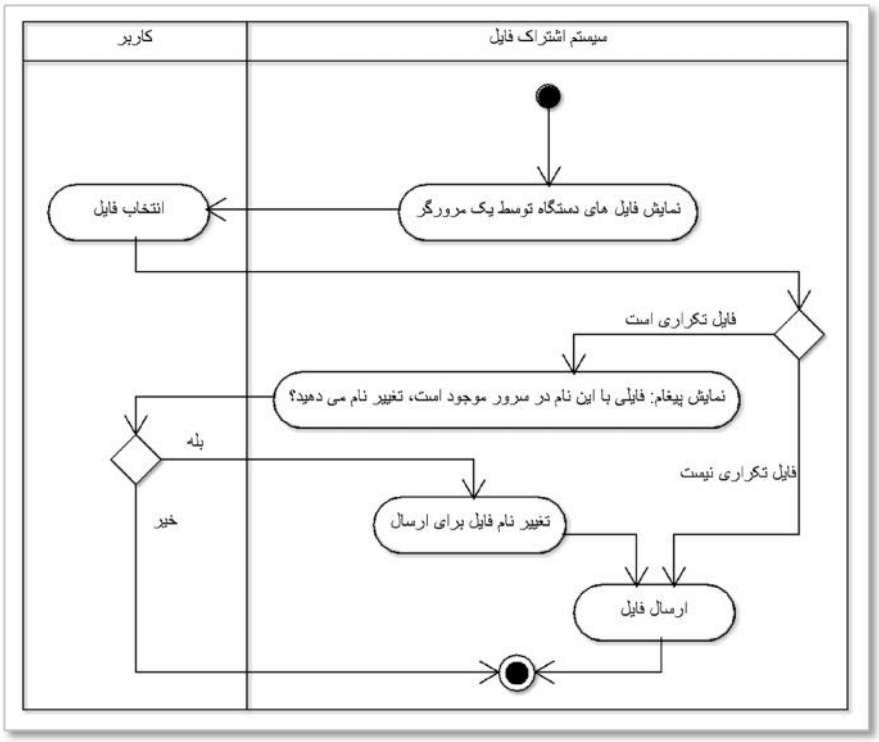
[4] Abstraction



شکل ۱-۳-۲ نمودار فعالیت متناظر با مورد کاربرد "اهراز هویت"



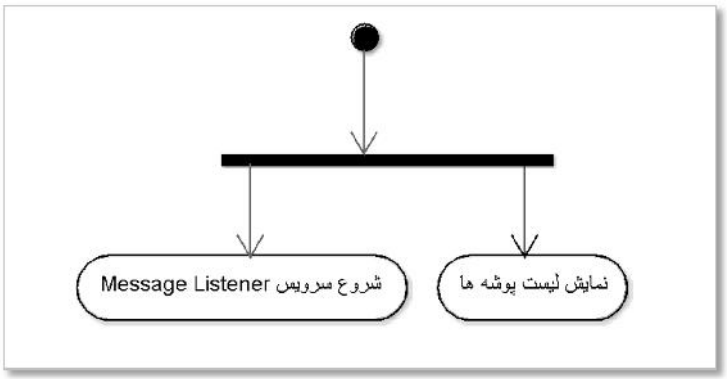
شکل ۲-۳-۲ نمودار فعالیت متناظر با مورد کاربرد "دریافت فایل"



شکل ۳-۳-۲ نمودار فعالیت متناظر با مورد کاربرد "ارسال فایل"

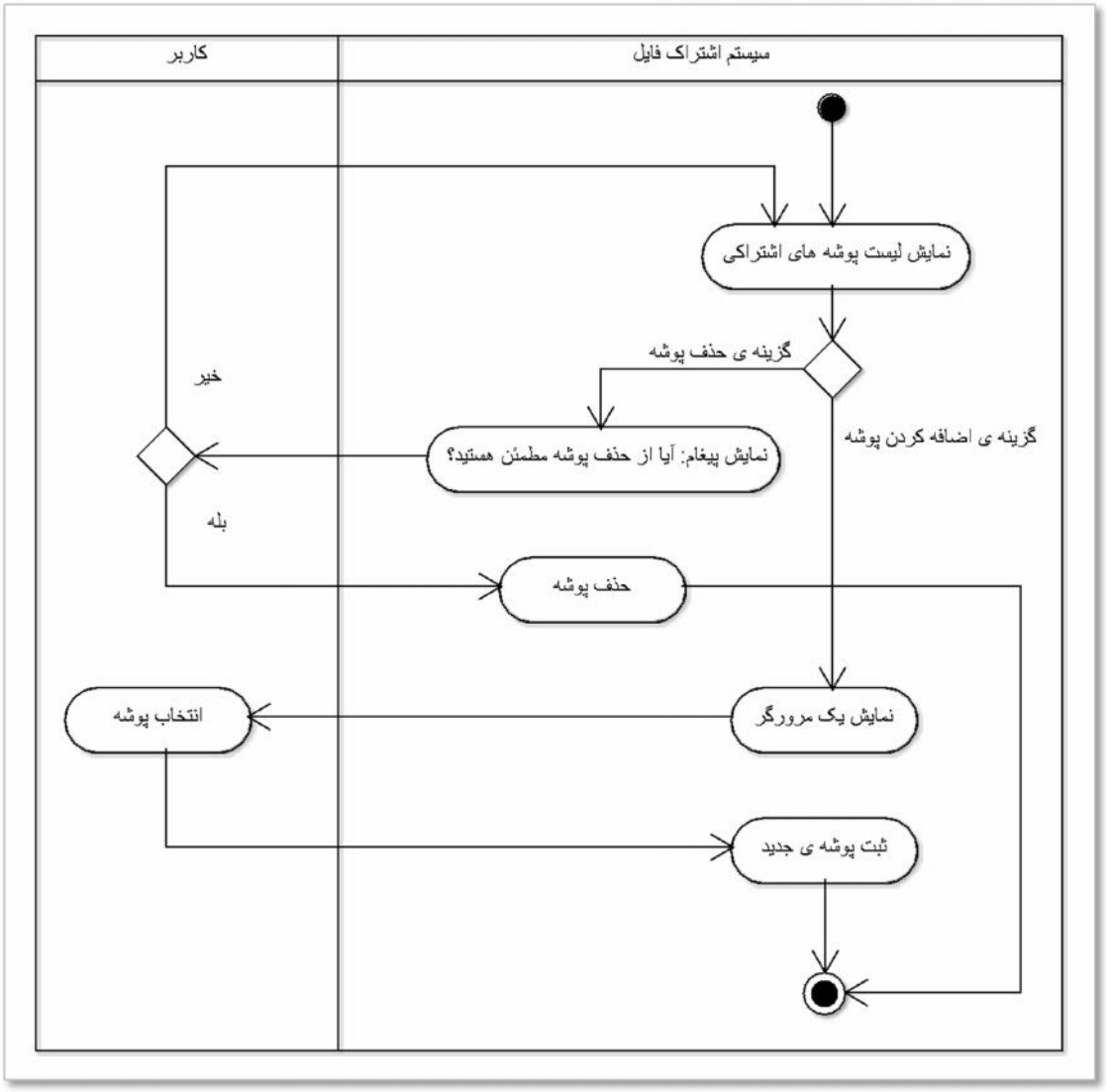
نمودار های فعالیت سمت سرور:

برای اینکه کمی به جزئیات نزدیکتر شویم، بد نیست بدانید هنگامیکه نرم افزار در طرف سرور اجرا میشود، همزمان با نمایش واسط گرافیکی به کاربر، یک سرویس شروع به کار میکند که وظیفه ی آن پاسخگویی به درخواست کاربران است. این سرویس تا زمانی که برنامه در حال اجراست، منتظر دریافت درخواست می ماند. نمودار فعالیت زیر، این موضوع را به خوبی نشان میدهد:

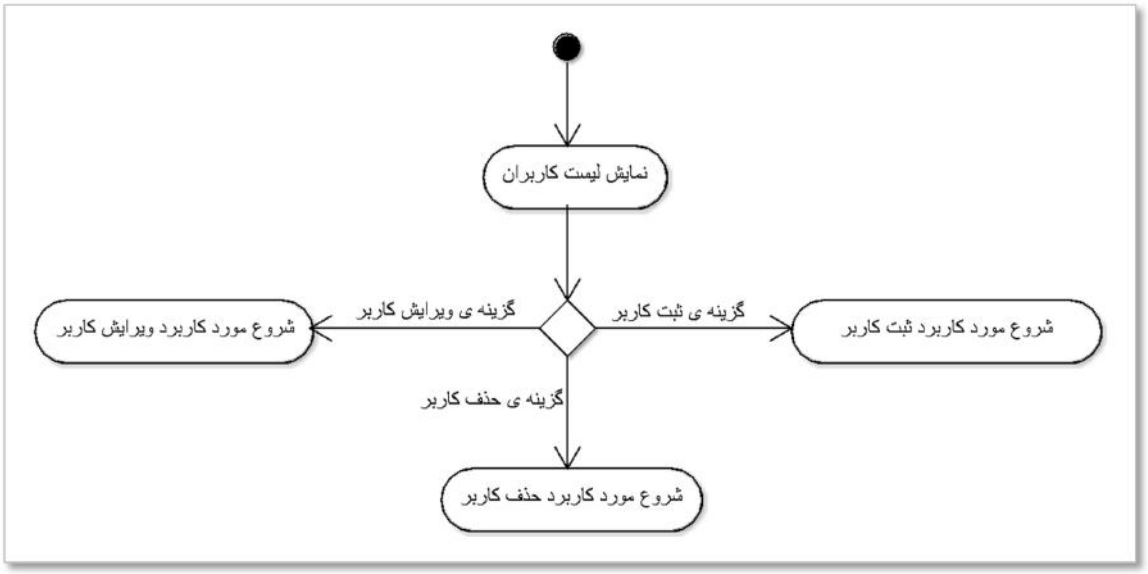


شکل ۴-۳-۲ نمودار فعالیت شروع (طرف سرور)

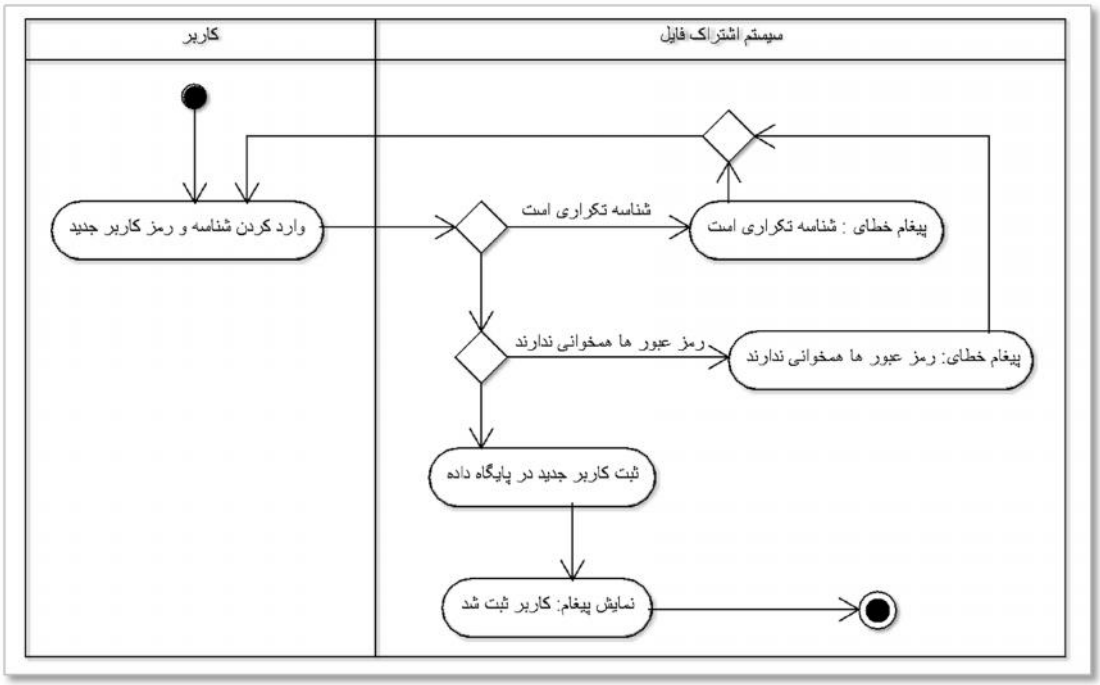
چند نمودار فعالیت دیگر:



شکل ۵-۳-۲ نمودار فعالیت متناظر با چند مورد کاربرد مربوط به اشتراک پوشه



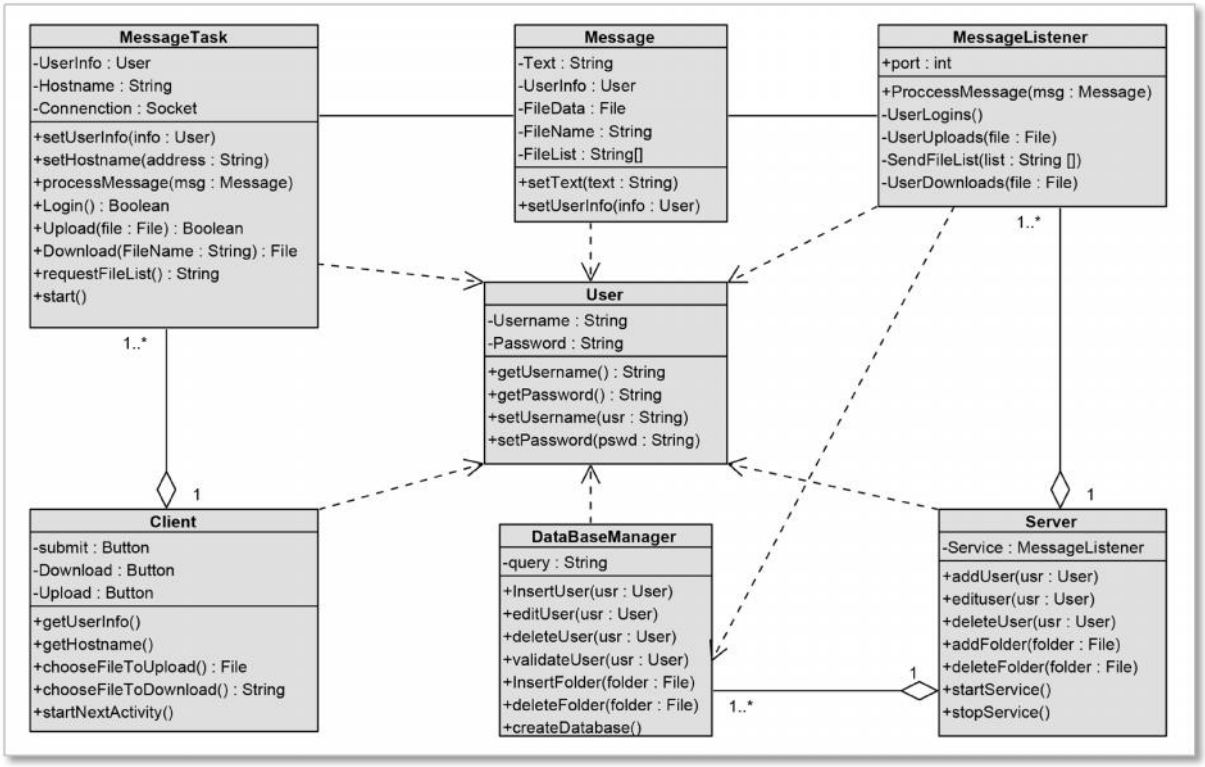
شکل ۶-۳-۲ نمودار فعالیت متناظر با مورد کاربرد "نمایش لیست کاربران"



شکل ۷-۳-۲ نمودار فعالیت متناظر با مورد کاربرد "ثبت کاربر"

۲-۳-۲ نمودار کلاس:

نمودار کلاس منبع اصلی تولید کد محسوب می شود. از نمودار های دیگر جهت تکمیل اطلاعات مربوط به نمودار کلاس و نیز تست آن استفاده می شود.



شکل ۸-۳-۲ نمودار کلاس سیستم اشتراک فایل

همانطور که میبینید، این سیستم از روش انتقال پیغام[5] برای ایجاد ارتباط بین کاربر و سرور استفاده میکند. به این صورت که هم در طرف سرور و هم در طرف کاربر، کلاسی برای مدیریت پیغام ها نوشته میشود (در طرف سرور، این کلاس یک سرویس است)، یک کلاس نیز برای خود پیغام در نظر گرفته میشود. یک

[5] Message Passing

شیء پیغام که از طرف کاربر به سرور فرستاده میشود، میتواند یکی از این ۴ حالت باشد:(۱) درخواست ورود[۶]، که حاوی شناسه و رمز عبور کاربر می باشد، (۲) درخواست ارسال فایل، که حاوی یک فایل و نام آن فایل میباشد، (۳) درخواست لیست فایل ها، که صرفا یک در خواست برای دریافت لیست نام فایل های اشتراکی است و (۴) درخواست دریافت فایل، که حاوی نام یک فایل اشتراکی است که کاربر درخواست دریافت آن را دارد. سرویس مدیریت پیغام در طرف سرور، یعنی MessageListener، با دریافت هر کدام از این پیغام ها، رفتار خاصی نشان میدهد و در جواب یک شیء پیغام به کاربر می فرستد.(۱) اگر درخواست ورود باشد، شناسه و رمز عبور ضمیمه شده به پیغام را در پایگاه داده ی سیستم اعتبار سنجی میکند (ارتباط با پایگاه داده از طریق کلاس DatabaseManager برقرار می شود) اگر معتبر بود، یک پیغام با متن "معتبر" به کاربر ارسال میکند، در غیر اینصورت، پیغام با متن "نا معتبر" ارسال می شود. (۲) اگر درخواست ارسال فایل باشد، سرور نام فایلی را که در محتوی پیغام وجود دارد با فایل های اشتراکی خود میسنجد، اگر فایل تکراری بود یک پیغام با متن "فایل تکراری است" به کاربر میفرستد، در غیر این صورت فایل را ذخیره میکند و یک پیغام با متن "انجام شد" به کاربر میفرستد. (۳) اگر درخواست لیست فایل ها باشد، سرور نام فایل های اشتراکی را در یک آرایه، ضمیمه ی پیغام میکند و میفرستد. (۴) اگر درخواست دریافت یک فایل باشد، سرور فایل مورد نظر را ضمیمه ی پیغام کرده و میفرستد.

پس یک شیء پیغام باید بتواند انواع مختلفی از متغییر ها را ضمیمه ی خود کند. همانطور نمودار کلاس میبینید، کلاس Message متغییر هایی برای ذخیره ی متن پیام، شناسه کاربر، رمز عبور کاربر، داده ی فایل، نام فایل و لیست نام فایل ها دارد.

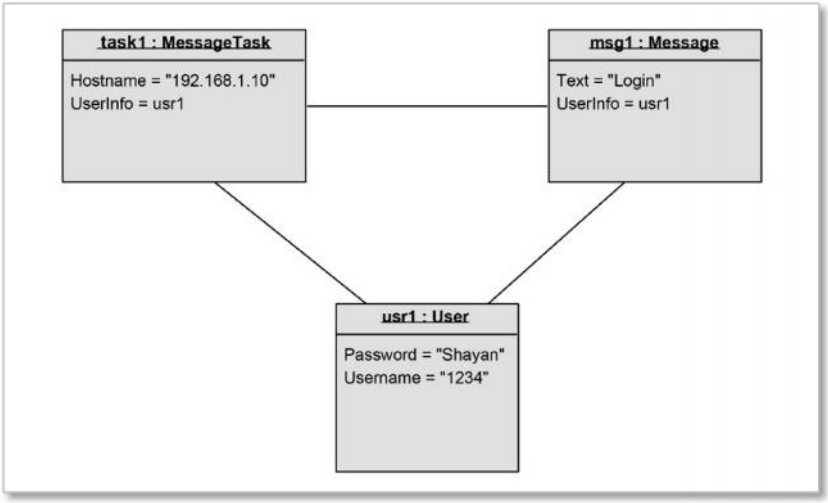
واسط های گرافیکی کاربر نیز در کلاس های Server و Client تعریف میشود که دهها متغییر و متود را به این کلاس ها می افزاید، اما برای حفظ سادگی از ذکر آنها خودداری کردم.

---

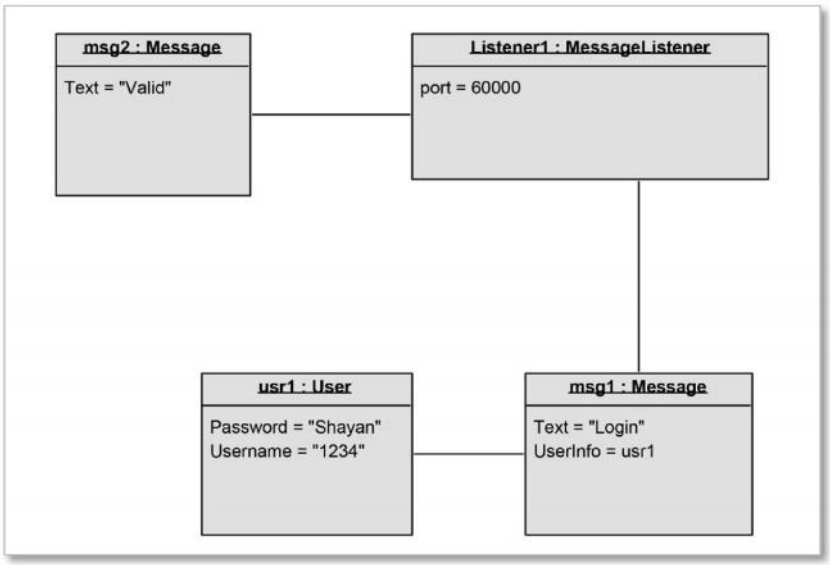
[6] Login

۳-۳-۲ نمودار شیء:

نمودار شیء کاربرد های زیادی در مدل سازی درست یک سیستم دارد و به عنوان مبنای ایجاد و تست نمودار کلاس به کار میرود. این نمودار در واقع مثال هایی از نمودار کلاس ارائه میکند:



شکل ۹-۳-۲ نمودار شیء ۱. در این نمودار، شیء task1 قصد ارسال پیام به مقصد ۱۹۲,۱۶۸,۱,۱۰ را دارد، اطلاعات کاربر که در شیء usr1 قرار دارد را در شیء msg1 میریزد و متن پیام را "Login" قرار میدهد.

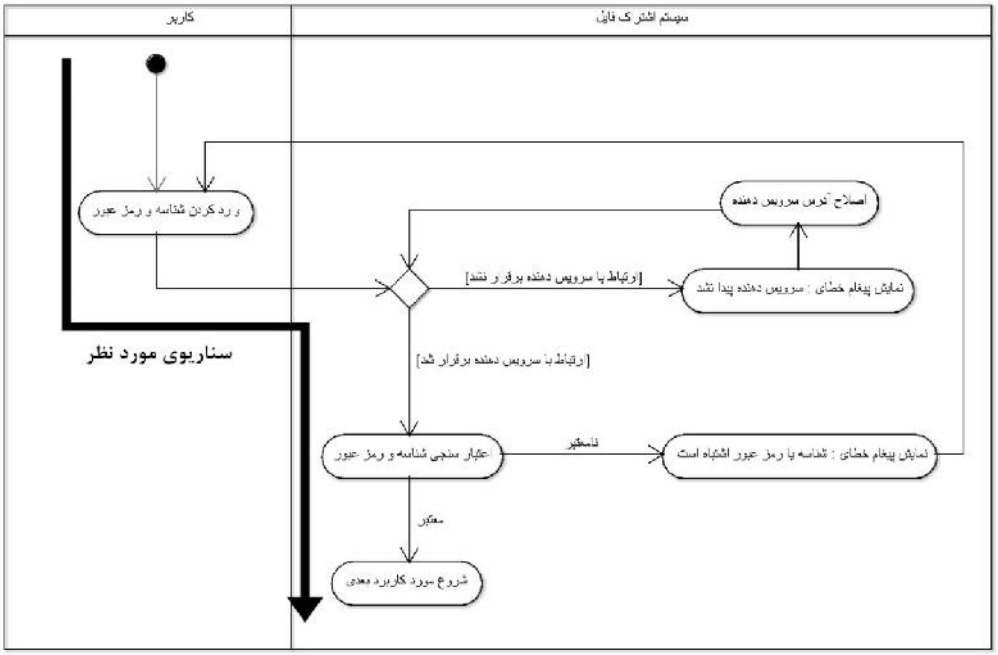


شکل ۱۰-۳-۲ نمودار شیء ۲. شیء Listener1 پیام را دریافت کرده و در جواب آن شیء msg2 را ایجاد میکند، متن آن را "Valid" قرار داده و می فرستد.

۴-۳-۲ نمودار توالی[۷]:

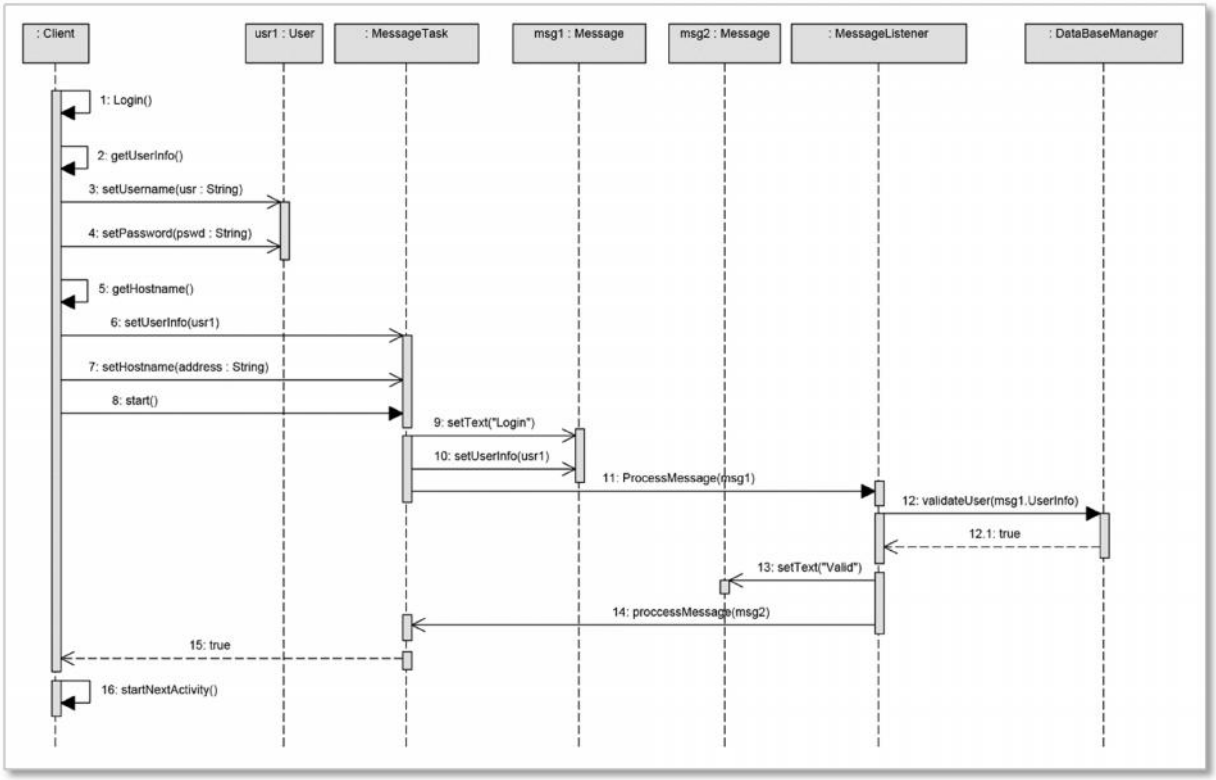
هر نمودار UML از یک دید خاص، قسمت هایی از رفتار یا ساختار سیستم را مدل میکند. نمودار توالی نشان میدهد که چگونه در یک سیستم، اشیاء با یکدیگر در تعامل هستند، چگونه سیستم به درخواست کاربر پاسخ میدهد، چگونه اطلاعات را در اختیار کاربر قرار میدهد و چگونه اشیاء، مدیریت و کنترل میشوند. در نمودار فعالیت ترتیب و تقدم اعمال انجام شده در سیستم را نشان دادیم، حال، نمودار توالی نشان میدهد که هر عمل مختص به کدام کلاس و شیء است و ترتیب انجام متود های کلاس ها چگونه است.

طراحی نمودار توالی از روی نمودار فعالیت انجام میشود. بدین صورت که، از روی نمودار فعالیت یک سناریوی خاص انتخاب میشود (یک مسیر که حالت های خاصی را از میان تمام حالات ممکن انتخاب میکند)، سپس اجزای نمودار توالی از روی اجزای نمودار فعالیت تعیین میشوند.



شکل ۱۱-۳-۲ سناریوی اهراز هویت

[7] Sequence Diagram

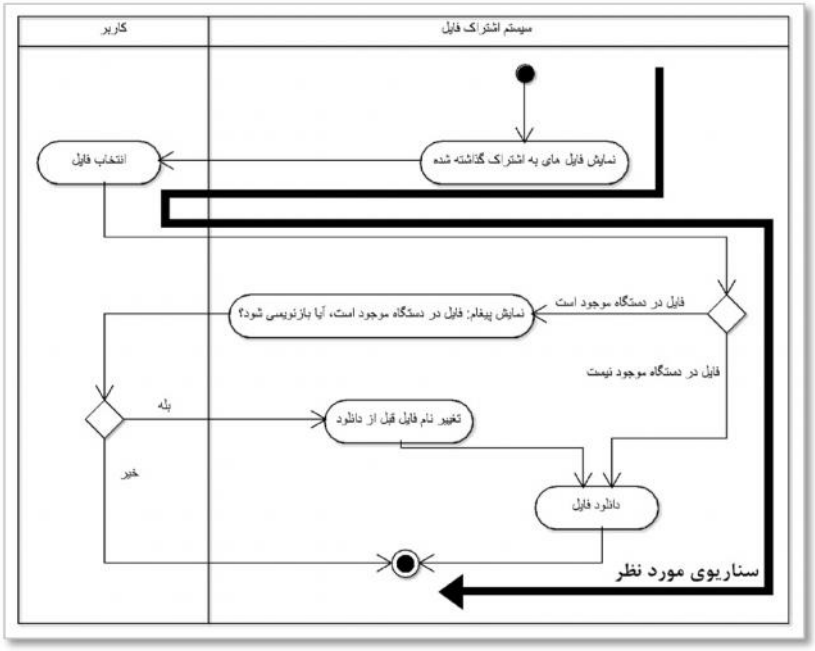


شکل ۱۲-۳-۲ نمودار توالی اهراز هویت. متناظر با سناریوی شکل قبل

توضیح: ابتدا اطلاعات کاربر، از او خواسته میشود و در مرحله ی بعد یک شیء از کلاس User ساخته، و اطلاعات در آن ذخیره میشود، سپس آدرس سرور از کاربر خواسته میشود (این آدرس به صورت پیش فرض 192.168.1.10:60000 مشخص شده، مگر آنکه کاربر آن را تغییر دهد) و این اطلاعات به MessageTask داده میشود تا آنها را در پیغامی با برچسب "Login" ذخیره کند. پیغام را MessageListener دریافت میکند و محتوی آن را خوانده و اطلاعات کاربر را در پایگاه داده سیستم اعتبارسنجی میکند (توسط کلاس DataBaseManager). با فرض اینکه اطلاعات معتبر است، پیغامی با متن "معتبر" برای MessageTask فرستاده میشود و این نتیجه به کلاس Client بازگردانده میشود تا کاربر وارد سیستم گردد.

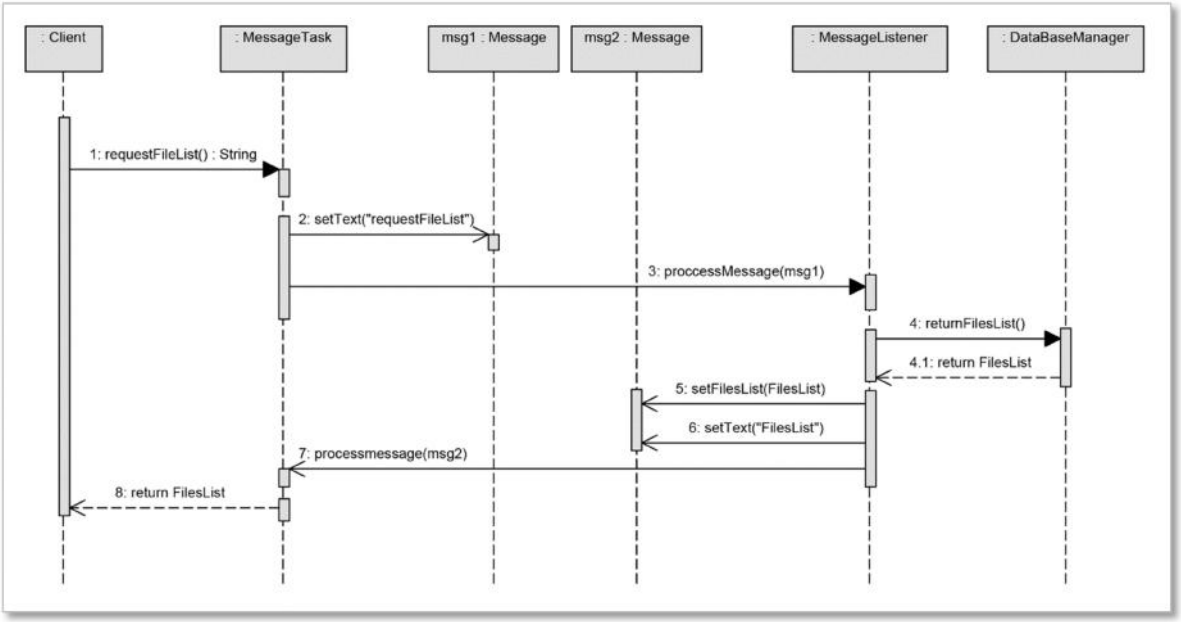
نکته: نقش کلاس MessageTask در این سیستم ممکن است به نظر پیچیده برسد. دلیل طراحی این کلاس، جدا کردن پردازش های شبکه از واسط گرافیکی است. این کار باید به صورت همروند[8] انجام شود تا واسط گرافیکی همیشه قادر به تعامل با کاربر باشد. برای دور شدن از جزئیات میتوان کلاس MessageTask و Client را یکی در نظر گرفت. اما نقش کلاس MessageListener متفاوت است! این کلاس در طرف سرور، به صورت یک سرویس همروند، همیشه در حال اجراست. در حالی که مدیر سیستم در حال استفاده از واسط گرافیکی برنامه (کلاس Server) است، این سرویس در حال پاسخگویی به درخواست -های کاربران است. پس با کلاس Server هیچ اشتراکی ندارد و هیچ یک از وظیفه های یکدیگر را انجام نمی دهند!

طراحی نمودار توالی دریافت فایل:



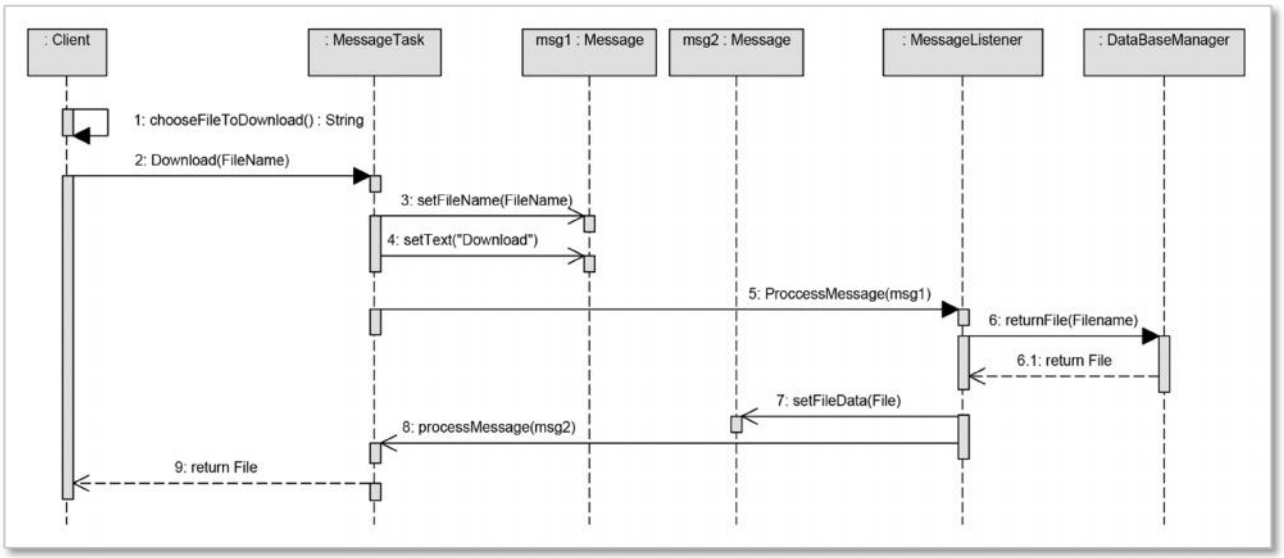
شکل۱۳-۳-۲ سناریوی دریافت فایل

[8] Multi-Threaded



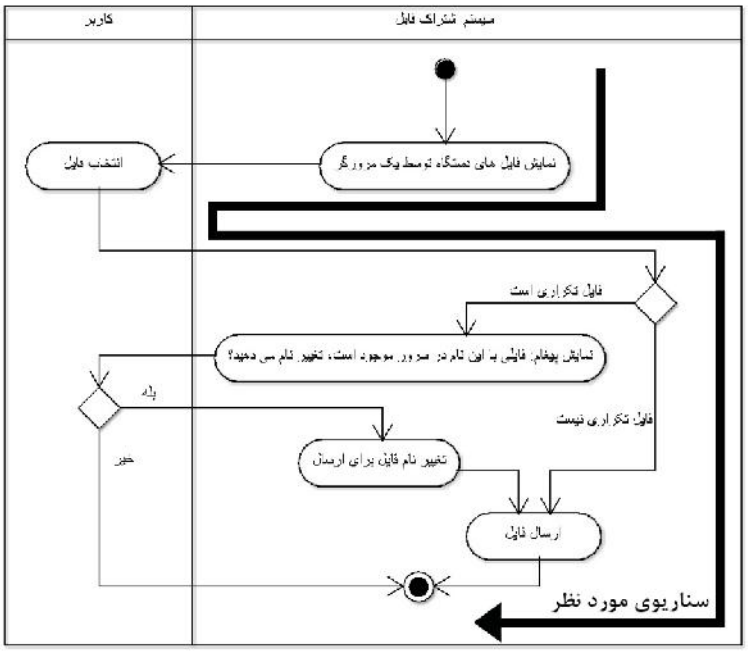
شکل ۱۴-۳-۲ نمودار توالی "نمایش فایل های به اشتراک گذاشته شده"

توضیح: این نمودار، عملیاتی را نشان میدهد که فقط در بخشی از سناریوی شکل قبل اتفاق می افتد. به این صورت که، ابتدا یک پیغام با متن "درخواست لیست فایل ها" از کاربر به سرور (کلاس MessageListener) فرستاده میشود، سرور محتوی پیغام را خوانده و لیست اسامی فایل ها را از پایگاه داده استخراج میکند، در یک آرایه قرار میدهد، این آرایه را در یک پیغام با متن"لیست فایلها" ذخیره میکند و میفرستد. نرم افزار طرف کاربر، این آرایه را در یک لیست به کاربر نمایش میدهد. در ادامه ممکن است کاربر فایلی را انتخاب نکند و انصراف دهد، اما اگر فایلی را انتخاب کند عملیات لازم برای دریافت آن فایل شروع میشود (شکل ۱۵-۳-۲):

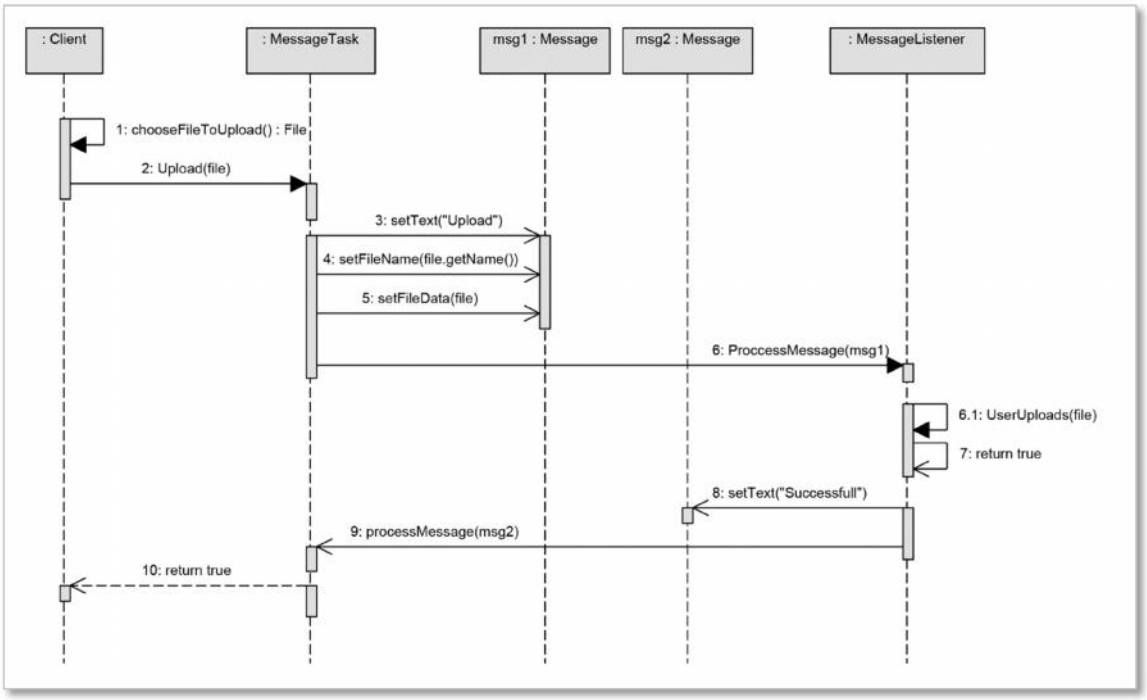


توضیح: ابتدا کاربر فایلی را انتخاب میکند، سپس نام این فایل در پیغامی با متن "دریافت فایل" ذخیره، و فرستاده میشود. سرور پیغام را بررسی کرده و نام فایل را در پایگاه داده جستجو میکند، محتوی آن فایل را در یک پیغام و در یک متغییر از نوع []byte ذخیره کرده، و میفرستد. سپس نرم افزار طرف کاربر محتوی پیغام را در یک فایل، در محلی مشخص در دستگاه ذخیره میکند.

طراحی نمودار توالی ارسال فایل:



شکل ۱۶-۳-۲ سناریوی ارسال فایل



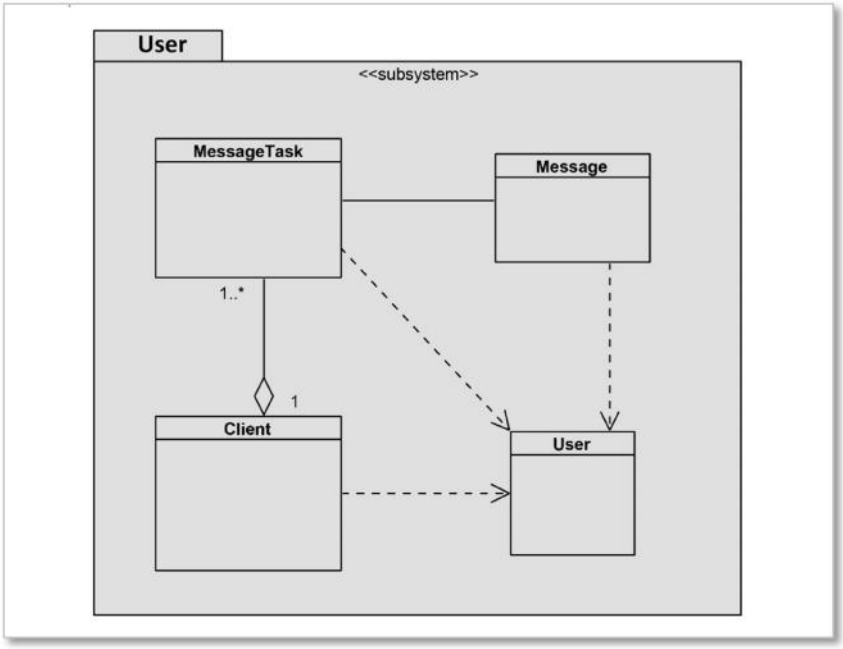
شکل ۱۷-۳-۲ نمودار توالی ارسال فایل

توضیح: ابتدا کاربر، فایل مورد نظر برای ارسال را به کمک یک مرورگر انتخاب میکند. سپس نام فایل و محتوی آن در یک پیغام ذخیره، و فرستاده میشود (یعنی فایل با همین پیغام ارسال میشود). سرور پیغام را بررسی کرده و نام فایل را با فایل های به اشتراک گذاشته شده مقایسه میکند، فایل تکراری نیست، پس در سرور به اشتراک گذاشته میشود و پیغامی با متن "موفقیت آمیز" برای کاربر فرستاده میشود تا از ارسال موفق فایل مطلع شود (در یک سناریوی دیگر، اگر فایل تکراری بود، آن فایل ذخیره نمیشد و پیغامی با متن "تکراری" برای کاربر فرستاده میشد تا اقدامات لازم صورت گیرد).

۵-۳-۲ نمودار بسته:

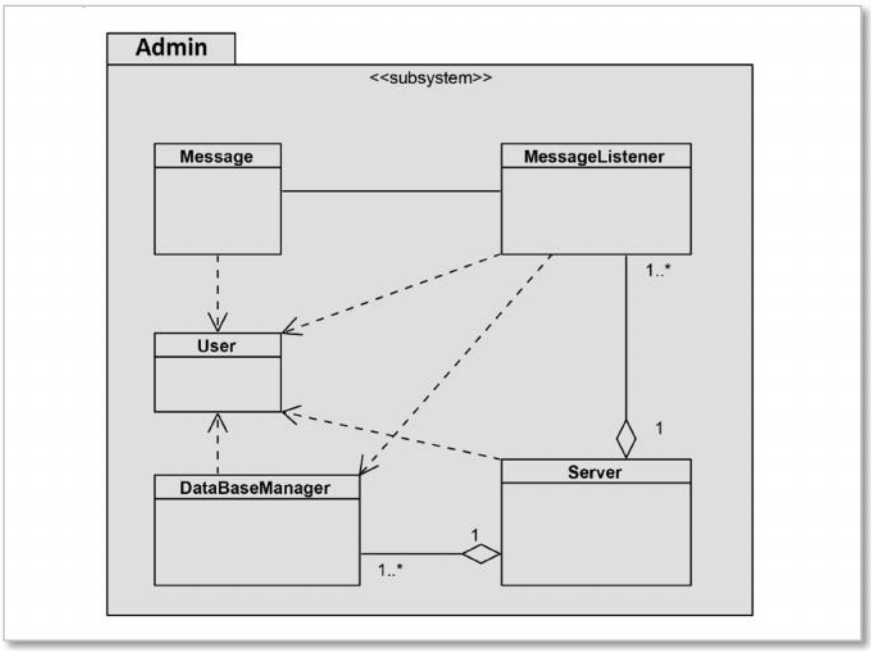
با پیچیده تر شدن سیستم و افزایش تعداد نمودار ها، نیاز به کنترل و مدیریت آنها بیشتر میشود. نمودار بسته، این ابزار را در اختیار ما قرار میدهد. هر بسته مانند یک پوشه مدلسازی میشود، و عموماً در سه نقش زیرسیستم، مدل و یا پوشه ظاهر میشود.

بسته ی کاربر: در ارتباط با کاربر است و کلاس های مربوط به کاربر را در یک زیرسیستم قرار داده است.



۲-۳-۱۸ نمودار بسته ی کاربر

بسته ی مدیر سیستم: در ارتباط با مدیر سیستم اشتراک فایل است و کلاس های مربوط به مدیریت را در یک زیرسیستم قرار داده است.



۲-۳-۱۹ نمودار بسته ی مدیر سیستم

۲-۴ دیسیپلین پیاده سازی:

بخش عمده ی فعالیت های دیسیپلین پیاده سازی به برنامه نویسی مرتبط می شود. یک قطعه کد قابل اجرا، یک دستاورد مهـــم دیسیپلین پیاده سازی محسوب می شود.

اهداف اصلی این دیسیپلین عبارتند از:

- تعریف ساختار کد های برنامه در قالب زیر سیستم های هر لایه از معماری سیستم
- پیاده سازی کلاس ها و اشیاء در قالب مؤلفه ها
- تست مؤلفه های پیاده سازی شده به صورت واحد
- یکپارچه سازی و مجتمع سازی مؤلفه ها در قالب یک سیستم قابل اجرا

۲-۴-۱ تعریف ساختار کد های برنامه:

از هرچه بگذریم، سخن "کد" خوش تر است! از آنجا که RUP رویکرد تکرار و تکامل تدریجی دارد، تا به اینجا، کد نویسی نرم افزار اشتراک فایل تقریباً انجام شده است، بیشتر آن با زبان جاوا و بخشی هم با زبان XML[9] (برای تعریف واسط های گرافیکی). حجم کد ها تقریباً به ۱۵۰۰ خط میرسد که توضیح همه ی آن در این پروژه نمی گنجد. پس قسمت های مهم تر آن را که بیشتر مربوط به شبکه یا انتقال داده است، همراه با شرح آن (خطوط سبز رنگ) می نویسم ، و از قطعه کد های مربوط به واسط گرافیکی که تا اندازه ای پیچیده است، صرف نظر میکنم.

[9] Extensible Markup Language

کلاس **پیغام**، که از آن به مراتب در کلاس های دیگر استفاده میشود:

```
package Model;

import java.io.Serializable;

public class Message implements Serializable {

	private static final long serialVersionUID = 1L;
	//متن پیغام
	private String Text=null;
	//شناسه و رمز عبور کاربر که در پیغام ذخیره میشود
	private String Username=null;
	private String Password=null;
	//محتوی یک فایل
	private byte[] FileData;
	//نام یک فایل
	private String FileName;
	//لیست نام فایل ها
	private String[] FilesList;

	//متودهای setter و getter که به وسیله ی آنها میتوان مقدار یک متغییر
	//را تغییر داد، یا از مقدار یک متغییر استفاده کرد
	public void setText(String text){
		this.Text=text;
	}
	public void setPassword(String Password){
		this.Password=Password;
	}
	public void setUsername(String Username){
		this.Username=Username;
	}
	public void setFileData(byte[] Data){
		FileData=Data;
	}
	public void setFileName(String Name){
		FileName=Name;
	}
	public void setFilesList(String[] Files){
		FilesList=Files;
	}
	public String getText(){
		return Text;
	}
	public String getUsername(){
		return Username;
	}
	public String getPassword(){
		return Password;
	}
	public byte[] getFileData(){
		return FileData;
	}
	public String[] getFilesList(){
		return FilesList;
	}
	public String getFileName(){
		return FileName;
	}
```

```
}
```

کلاس **اطلاعات کاربر**:

```
package Model;

public class User {
	//اطلاعات کاربری
	private String Username;
	private String Password;
	//getter و setter متودهای
	public void Set(String Username,String Password){
		this.Username=Username;
		this.Password=Password;
	}

	public String getUsername(){
		return Username;
	}
	public String getPassword(){
		return Password;
	}
}
```

کلاس MesageTask :

```
package com.aryan.filesharing;

import Model.Message;
import Model.User;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.net.UnknownHostException;
import android.content.Context;
import android.content.Intent;
import android.util.Log;
import android.widget.Toast;
import android.os.Handler;
import android.os.Looper;

public class MessageTask extends Thread {
	//آدرس سرور
	public static String HostName;
	//پورت سرور
	public static int Port;
	/* کانال ارتباط با سرور که یک سوکت می باشد و از
	 *متغییر های آدرس و پورت استفاده میکند
	 *سوکت در جاوا، پروتکل تی سی پی[10] را پیاده سازی میکند
```

[10] TCP/IP

```
*/و دیتا را به صورت رمزگذاری شده انتقال می دهد
private Socket connection;
//دستوری که کلاس کلاینت به این کلاس میدهد
private String order;
//اطلاعات کاربر که در پیغام ذخیره میشود
private User UserInfo;
//یک درگاه برای خروج جریان اطلاعات
private ObjectOutputStream Out=null;
//یک درگاه برای ورود جریان اطلاعات
private ObjectInputStream in=null;
//فایلی که ارسال میشود
private File fileToUpload;
//نام فایلی که دانلود میشود
private String fileToDownload=null;
public static String DOWNLOAD_PATH="/mnt/sdcard/Download";
private String TAG = "File Sharing App";


//این متود، قبل از همه ی متودها در این کلاس اجرا میشود
public void start(){
	try {
		//تلاش برای برقراری ارتباط با سرور
		connection = new Socket(HostName,Port);
		//ایجاد یک درگاه برای خروج جریان اطلاعات از کاربر به سرور
		Out=new ObjectOutputStream(connection.getOutputStream());
		//اینجا تصمیم گرفته میشود که کدامیک از اعمال ٤گانه،
		// یعنی ورود، ارسال فایل، دریافت لیست فایل ها، یا
		//دریافت فایل باید صورت گیرد
		ProcessOrder();
		Connection.close();
		Out.close();
	} catch (UnknownHostException uhe) {
		//اگر آدرس سرور اشتباه وارد شده باشد، به اطلاع کاربر میرسد
		Log.d("TAG","UnknownHostException");
	}
	catch (IOException ioe){
		//اگر تلاش برای برقراری ارتباط با شکست
		//مواجه شود، به اطلاع کاربر میرسد
		Log.d("TAG","IOException");
	}
}

//متود تصمیم گیرنده ی عمل بعدی
private void ProcessOrder(){
	if(order.equals("Login"))
		Login();
	}else if(order.equals("Upload"))
		Upload();
	else if(order.equals("File List Request"))
		fileListRequest();
	else if(order.equals("Download"))
		Download();
	else Log.d("TAG","Unknown order");
}

//متود مرتبط با ورود کاربر به سیستم
private void Login(){
	//یک شیء پیغام با متن "ورود" ساخته میشود و اطلاعات
	//کاربر در آن ذخیره میشود
	Message OBJ=new Message();
	OBJ.setText(“Login”);
```

```
                OBJ.setUserInfo(UserInfo);
                try {
                                //تلاس برای ارسال شیء پیغام به سرور
                                Out.writeObject(OBJ);
                                Out.flush();
                        //ایجاد درگاه برای ورود جریان اطلاعات از سرور
                        in=new ObjectInputStream(connection.getInputStream());
                                //انتظار برای دریافت پیام از سرور
                                OBJ = (Message) in.readObject();
                                //اگر پیغام سرور با متن "معتبر" باشد،
                                //کاربر وارد سیستم میشود
                                if(OBJ.getText().equals("Valid")){
                                        startChooseActivity();
                                //اگر پیغام سرور با متن "نامعتبر" باشد،
                                //این موضوع به اطلاع کاربر میرسد
                                }else if(OBJ.getMessage().equals("Invalid"))
                handler.sendMessage(Message.obtain(handler,INVALID_USER_ID));
                //اگر مشکلی در انتقال پیغام پیش بیاید به اطلاع کاربر میرسد
                } catch (IOException e) {
                        Log.d("TAG","IOException");
                handler.sendMessage(Message.obtain(handler,IOEXCEPTION_ID));
                } catch (ClassNotFoundException e) {
                        Log.d("TAG","ClassNotFoundException");}
        }
        //متود مرتبط با ارسال فایل
        private void Upload(){
                //پیغامی با متن "ارسال" ساخته میشود
                Message OBJ=new Message();
                OBJ.setText("Upload");
                try {
                        FileInputStream fis=new FileInputStream(fileToUpload);
                        //فایلی که توسط کلاس کلاینت انتخاب شده
                        //در متغییری ذخیره میشود
                        byte[] b=new byte[(int)fileToUpload.length()];
                        fis.read(b);
                        //اطلاعات آن متغییر در پیغام ذخیره میشود
                        OBJ.setFileData(b);
                        OBJ.setFileName(fileToUpload.getName());
                        //پیام که شامل فایل است برای سرور فرستاده میشود
                        Out.writeObject(OBJ);
                        Out.flush();
                        //ایجاد درگاه برای ورود جریان اطلاعات از سرور
                        in=new ObjectInputStream(connection.getInputStream());
                        //انتظار برای دریافت پیام از سرور
                        OBJ = (Message) in.readObject();
                        //اگر پیغام سرور با متن "موفقیت آمیز" باشد،
                        //این موضوع به اطلاع کاربر میرسد
                        if(OBJ.getMessage().equals("OK"))
        handler.sendMessage(Message.obtain(handler,SENT_SECCESSFULL_ID));
                        //اگر پیغام سرور با متن "فایل تکراری" باشد،
                        //این موضوع به اطلاع کاربر میرسد
                        else if(OBJ.getMessage().equals("File Exists"))
        handler.sendMessage(Message.obtain(handler,FILE_EXISTS _ID));
                        //اگر به هر دلیل ارسال ناموفق باشد،
                        //این موضوع به اطلاع کاربر میرسد
                        else if(OBJ.getMessage().equals("Failed"))
        handler.sendMessage(Message.obtain(handler,SENDING_FAILED_ID));

                } catch (FileNotFoundException e) {
        handler.sendMessage(Message.obtain(handler,FILE_NOT_FOUND_EXCEPTION_I
D));
```

```
                }
            catch (IOException ioe) {
            handler.sendMessage(Message.obtain(handler,IOEXCEPTION_ID));
                }
            catch (ClassNotFoundException e) {
            handler.sendMessage(Message.obtain(handler,EXCEPTION_ID));
                }
        }

        //متود مربوط به درخواست اسامی فایل های به اشتراک گذاشته شده
        private void fileListRequest(){
            Message OBJ=new Message ();
            OBJ.setText("File List Request");
            try {
                Out.writeObject(OBJ);
                Out.flush();
                in=new ObjectInputStream(connection.getInputStream());
                OBJ = (Message) in.readObject();
                SharedFilesListActivity.LIST=OBJ.getFilesList();
                //نمایش لیست فایل ها
                startListActivity();
                } catch (IOException e) {
                Log.d(TAG,e.toString());
            handler.sendMessage(Message.obtain(handler,IOEXCEPTION_ID));
                } catch (ClassNotFoundException e) {
                Log.d(TAG,e.toString());
            handler.sendMessage(Message.obtain(handler,EXCEPTION_ID));
            }
        //متود مربوط به دریافت فایل انتخاب شده
        private void Download(){
            try {
            File file=new File(DOWNLOAD_PATH+"/"+fileToDownload);
            //بررسی میشود که فایل تکراری است یا خیر
            if(!file.exists()){
                Message OBJ = new Message ();
                OBJ.setFileName(fileToDownload);
                OBJ.setMessage(mMessage);
                Out.writeObject(OBJ);
                Out.flush();
                in=new ObjectInputStream(connection.getInputStream());
                OBJ = (Message) in.readObject();
                if(OBJ.getMessage().equals("File Found")){
                    //فایل به کمک یک بافر دریافت میشود
                    long FileLength = in.readLong();
                    File DownloadFolder = new File(DOWNLOAD_PATH);
                    if(!DownloadFolder.exists())
                        DownloadFolder.mkdir();
                    BufferedOutputStream bos=new
BufferedOutputStream(new FileOutputStream(file));
                    byte[] buffer=new byte[100000];
                    int count=0;
                    long current=0;
                    while(current<FileLength &&
(count=in.read(buffer,0,(int)Math.min(FileLength-
current,buffer.length)))!=-1){
                        bos.write(buffer,0,count);
                        bos.flush();
                        current+=count;
                    }
                    bos.close();
```

```
		handler.sendMessage(Message.obtain(handler,FILE_DOWNLOADED_ID));
                }
                else //فایل در سرور یافت نشد

		handler.sendMessage(Message.obtain(handler,FILE_NOT_FOUND_EXCEPTION_I
D));
		}
		else //فایل تکراری است

		handler.sendMessage(Message.obtain(handler,FILE_EXISTS_ID));

		} catch (IOException e) {
		handler.sendMessage(Message.obtain(handler,IOEXCEPTION_ID));

		} catch (ClassNotFoundException e) {
		handler.sendMessage(Message.obtain(handler,EXCEPTION_ID));
		}
	}
	//متود مربوط به نمایش صفحه ی بعدی
	private void startChooseActivity(){
		Intent i= new Intent(context,ChooseActivity.class);
		context.startActivity(i);
	}
	//متود مربوط به نمایش لسیت فایل ها
	private void startListActivity(){
		Intent i= new Intent(context,SharedFilesListActivity.class);
		context.startActivity(i);
	}
}
```

کلاس MessageListener:

```
package com.Aryan.FileSharing.Server;

import javafx.concurrent.Service;
import javafx.concurrent.Task;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.net.ServerSocket;
import Model.Message;


public class MessageListener extends Service<String> {
	public static int port=60000;//پورتی که سرور روی آن منتظر ارتباط است
	private Socket connection=null;
	private ObjectInputStream in=null;
	private ObjectOutputStream out=null;
	private Message OBJ;
	private File fileToUpload=null;
	private boolean fileNotFound=false;
```

```
	public static void setPort(int PortNumber){
		port=PortNumber;
		}
//این متود قبل از دیگر متود ها اجرا میشود//
protected Task<String> createTask(){
		return new Task<String>(){
			protected String call(){
				System.out.println("Listening for connections on
port: " + port);
				String result=new String();
				try (ServerSocket SS=new ServerSocket(port);
					Socket connection = SS.accept();)
					{
					System.out.println("connection accepted");
					in = new ObjectInputStream(new
BufferedInputStream(connection.getInputStream()));
					out = new ObjectOutputStream(new
BufferedOutputStream(connection.getOutputStream()));
					//پیغام در اینجا دریافت میشود
					OBJ=(Message) in.readObject();
					result = OBJ.getMessage();
					//محتوی پیغام بررسی مشود
					ProccessMessage(OBJ.getMessage());
					in.close();
					out.close();
				} catch (IOException e) {
					e.printStackTrace();}
				catch (ClassNotFoundException e) {
					e.printStackTrace();}

				return result;
			}
		};
	}
	//این متود محتوی پیغام را بررسی میکند//
	private void ProccessMessage(String Message){
		if(Message.equals("Login")){
			System.out.println("Message is: Login");
			ValidateUser();
		}else if(Message.equals("Upload")){
			System.out.println("Message is: Upload, and the file name
is: "+OBJ.getFileName());
			UserUploads();
		}
		else if(Message.equals("File List Request")){
			System.out.println("Message is: File List Request");
			sendFileList();
		}
		else if(Message.equals("Download")){
			System.out.println("Message is: Download, and the file
name is: "+OBJ.getFileName());
			UserDownloads();
		}
	}
	//متود مربوط به اعتبار سنجی کاربری که قصد ورود به سیستم را دارد//
	private void ValidateUser(){
		try {
			DatabaseManager DBM=new DatabaseManager();
			if(DBM.UserIsValid(OBJ.getUsername(),OBJ.getPassword()))
				OBJ.setMessage("Valid");
```

```
                        else
                              OBJ.setMessage("Invalid");
                        out.writeObject(OBJ);
                        out.flush();

                  } catch (IOException e) {
                        e.printStackTrace();}
            }
            //متود مربوط به زمانی که کاربری قصد ارسال فایلی دارد
            private void UserUploads(){
                  String Message;
                  char SC=File.separatorChar;
                  File file=new
File(System.getProperty("user.home")+SC+"Downloads"+SC+"File
Sharing"+SC+OBJ.getFileName());
                  if(file.exists()){ //فایل تکراری است
                        Message="File Exists";
                        System.out.println("File: "+file.getName()+" supposed to
download, but it already exists in system");
                        try {
                              //پیامی با متن "تکراری" برای کاربر ارسال میشود
                              OBJ=new Message();
                              OBJ.setMessage(Message);
                              out.writeObject(OBJ);
                              out.flush();
                        } catch (IOException e) {e.printStackTrace();
                        }
                  }
                  else{ //فایل تکراری نیست
                        try {
                              OBJ=new Message();
                              OBJ.setMessage("OKSend");
                              out.writeObject(OBJ);
                              out.flush();
                        } catch (IOException e) {e.printStackTrace();
                        }
                        try { //فایل به صورت بافر شده دریافت میشود
                              BufferedOutputStream bos=new
BufferedOutputStream(new FileOutputStream(file));
                              long FileLength = in.readLong();
                              byte[] buffer=new byte[100000];
                              int count=0;
                              long current=0;
                              while(current<FileLength &&
(count=in.read(buffer,0,(int)Math.min(FileLength-
current,buffer.length)))!=-1){
                                    bos.write(buffer,0,count);
                                    bos.flush();
                                    current+=count;
                              }
                              bos.close();
                              System.out.println("File downloaded:
"+file.getAbsolutePath());
                              Message="OK";
                        } catch (FileNotFoundException e) {
                              Message="Failed";
                              e.printStackTrace();
                        } catch (IOException e) {
                              Message="Failed";
                              e.printStackTrace();
                              }
```

```
                        try {
                              //نتیجه ی دریافت فایل به کاربر اطلاع داده میشود
                              OBJ=new Message();
                              OBJ.setMessage(Message);
                              out.writeObject(OBJ);
                              out.flush();
                        } catch (IOException e) {e.printStackTrace();}
                  }//end of else
            }
            //متود مربوط به زمانی که کاربر درخواست لیست فایل ها را دارد//
            private void sendFileList(){
                  String[] Folders=new DatabaseManager().returnFolders();
                  StringBuffer sb=new StringBuffer();
                  for(int i=0;i<Folders.length;i++){
                        AppendFiles(Folders[i], sb);
                  }
                  String[] items=sb.toString().split(":");
                  OBJ.setFilesList(items);
                  try {
                        out.writeObject(OBJ);
                        out.flush();
                  } catch (IOException e) {
                        e.printStackTrace();
                  }
            }
            //این متود فقط در متود قبلی استفاده میشود
            private void AppendFiles(String Directory,StringBuffer sb){
                  File Folder=new File(Directory);
                  File[] Files=Folder.listFiles();
                  for(int j=0;j<Files.length;j++){
                        if(!Files[j].isDirectory())
                              sb.append(Files[j].getName()+":");
                        else AppendFiles(Files[j].getAbsolutePath(), sb);
                  }
            }
            //متود مربوط به زمانی که کاربر قصد دریافت فایلی را دارد
            private void UserDownloads(){
                  try {
                        //فایل در سیستم جستجو میشود
                        SearchFile(OBJ.getFileName());
//ممکن است فایل یافت نشود، زیرا احتمال دارد زمانی که کاربر در حال//
//مشاهده ی لیست فایل ها میباشد، فایل مورد نظر از سیستم حذف گردد//
                        if(!fileNotFound){//The file found
                              OBJ.setMessage("File Found");
                              out.writeObject(OBJ);
                              out.flush();
                              //فایل به صورت بافر شده ارسال میشود
                              out.writeLong(fileToUpload.length());
                              out.flush();
                              BufferedInputStream bis=new BufferedInputStream(new
FileInputStream(fileToUpload));
                              byte[] buffer = new byte[100000];
                              int count=0;
                              long current=0;
                              while(current<fileToUpload.length() &&
(count=bis.read(buffer,0,(int)Math.min(buffer.length,
fileToUpload.length()-current)))!=-1){
                                    out.write(buffer,0,count);
                                    out.flush();
                                    current+=count;
                              }
```

```
                        bis.close();
                        System.out.println("File sent");
                  }
                  else{ //اگر فایل پیدا نشود، به اطلاع کاربر میرسد
                        OBJ.setMessage("File Not Found");
                        out.writeObject(OBJ);
                        out.flush();
                        System.out.println("The file: "+OBJ.getFileName()+"
not found");
                  }
            } catch (IOException e) {
                  e.printStackTrace();
            }

      }
      //این متود فقط در متود قبلی استفاده میشود
      private void SearchFile(final String FileName){
            fileNotFound=true;
            String[] Foldernames=new DatabaseManager().returnFolders();
            for(int i=0;i<Foldernames.length;i++){
                  if(!fileNotFound)
                        break;
                  File Folder=new File (Foldernames[i]);
                  SearchFileInFolder(FileName,Folder);
            }
      }
      //این متود فقط در متود قبلی استفاده میشود
      private void SearchFileInFolder(final String FileName,File dir){
                  File[] files=dir.listFiles();
                  for(File file : files){
                        if(!fileNotFound)
                              break;
                        if(file.isDirectory())
                              SearchFileInFolder(FileName,file);
                        else if(file.getName().equals(FileName)){
                                    fileToUpload=file;
                                    fileNotFound=false;
                              }
                  }
      }
}
```

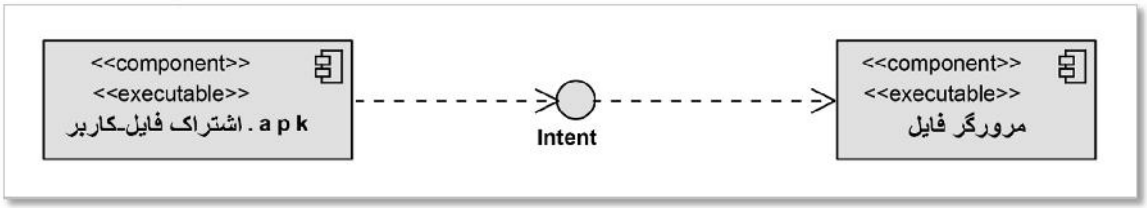
۲-۴-۲ نمودار مؤلفه[11]:

در هنگام پیاده سازی نرم افزار، لازم است که کد ها در بخش های کوچکتری دسته بندی شده تا فقط در صورت لزوم به حافظه فراخوانی شوند. در UML، مؤلفه به مجموعه ای از کد ها یا داده های نرم افزار گفته

[11] Component Diagram

میشود که در یک محل از حافظه قرار میگیرند. از آنجایی که در یک مؤلفه فقط بخشی از کد ها یا داده های نرم افزار قرار میگیرند، هر مؤلفه ممکن است نیازمند ارتباط با مؤلفه ی دیگر باشد. گاهی برای انجام این ارتباط، برای مؤلفه یک واسط در نظر گرفته میشود تا از طریق واسط، مؤلفه های دیگر بتوانند با آن ارتباط برقرار کنند.

از آنجا که در این پروژه، نرم افزار سمت کاربر برای سیستم عامل اندروید نوشته میشود، می توان از امکانات این سیستم عامل استفاده کرد. از جمله این امکانات، همان مرورگر فایلی است که کاربر برای ارسال فایل از آن استفاده میکند. این یکی از مزیت های اندروید است که به برنامه های کاربردی اجازه میدهد تا از امکانات یکدیگر استفاده کنند. برای ایجاد ارتباط بین برنامه های کاربردی، از یک واسط به نام Intent استفاده میشود که مسئول انتقال اطلاعات بین برنامه ها، و پیدا کردن برنامه ی مورد نیاز برای انجام عمل مورد نظر است. نمودار مؤلفه ی زیر، ارتبـــاط بین مؤلفه ی اشتراک فایل-کاربر (فایل اجرایی اصلی نرم افزار در سمت کاربر) و مرورگر فایل (که یک برنامه ی کاربردی جداگانه است و در همه ی دستگاههای مجهز به اندروید وجود دارد) را نشان میدهد.



شکل ۱-۴-۲ نمودار مؤلفه سمت کاربر

شکل ۲-۴-۲ نمودار مؤلفه سمت سرور

در سیستم عامل ویندوز نیز امکان استفاده از مرورگر فایل وجود دارد، اما یک برنامه ی جداگانه نیست، بلکه جزئی از ویندوز است. شکل ۲-۴-۲، نمودار مؤلفه سمت سرور را نشان میدهد. همانطور که میبینید، فایل اجرایی اصلی نرم افزار در سمت سرور، از یک پایگاه داده استفاده میکند، که خود نرم افزار اشتراک فایل آن را قبلا ایجاد کرده و شامل اطلاعات کاربران و پوشه های به اشتراک گذاشته شده است.

۵-۲ دیسیپلین استقرار:

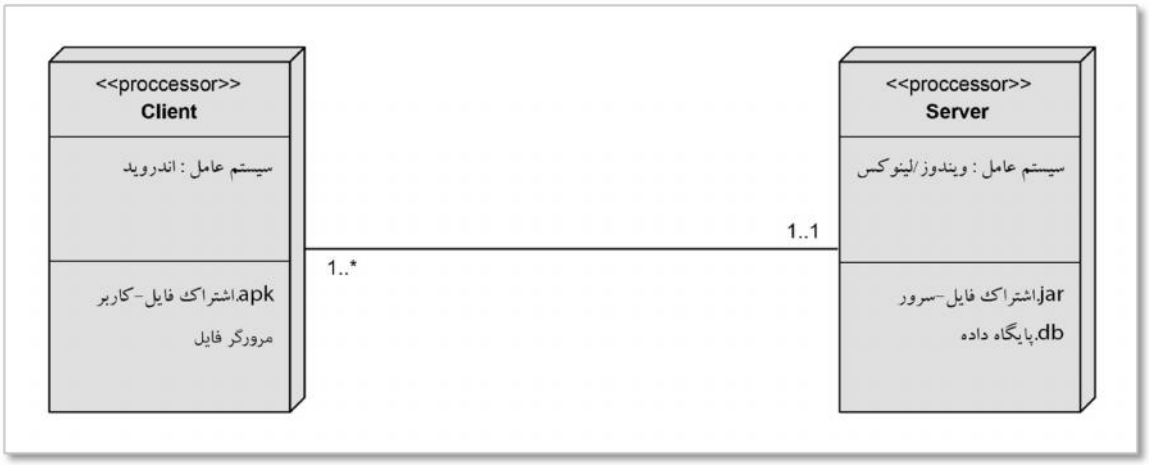
فعالیت های این دیسیپلین:

- تست نرم افزار در محیطی با شرایط عملیاتی مورد نظر
- بسته بندی نرم افزار برای تحویل
- توزیع نرم افزار
- نصب و راه اندازی نرم افزار
- آموزش کاربران نهایی و نیروهای فروش

البته فعالیت های ذکر شده اکثراً عملیاتی هستند و آنچه میتوان مستندسازی کرد، نمودار استقرار است:

۱-۵-۲ نمودار استقرار:

در این نمودار، نُد ها[12] اجزای معماری ســخت افزار را مدل میکنند. نُد در UML مبین یک منبــع سخت افزاری است که عملیات پردازش را انجام میدهد. برای مثال، هر کامپیوتر یک نُد محسوب میشود.



شکل ۱-۵-۲ نمودار استقرار سیستم

هر نُد مانند هر کلاس، دارای ویژگی ها و رفتارهای خاص خود است. در قسمت بالایی، نام و نوع نُد، در قسمت میانی، ویژگی های نُد، و در قسمت پایینی، نام مؤلفه هایی که در این نُد اجرا میشوند، مشخص شده است. همانطور که در شکل میبینید، سرور میتواند با تعداد نامحدودی کاربر در ارتباط باشد.

لازم به ذکر است، طبق آنچه در دیسیپلین مدلسازی سازمان مطرح شد، پیاده سازی نرم افزار طرف کاربر برای سه سیستم عامل اندروید، ویندوز و لینوکس لازم است (تحلیل و طراحی آنها یکسان میباشد). اما طبق عنوان این پایان نامه، پیاده سازی نرم افزار سمت کاربر، فقط برای سیستم عامل اندروید انجام گرفت.

[12] Nodes

نتیجه گیری:

با پیشروی در دیسیپلین های RUP، ابتدا یک سازمان را مدلسازی کردیم و با تحلیل این مدل، مشکلات جاری سازمان را مشخص کردیم و نیاز هایی را که رفع آنها، مشکلات سازمان را رفع میکرد، استخراج کردیم. سپس با طراحی و پیاده سازی سیستم اشتراک فایل، وعده های داده شده در دیسیپلین نیازمندی ها را عملی کردیم.

فرآیند ساخت نرم افزار اشتراک فایل، نمونه ی بسیار کوچکی از تولید یک برنامه به روش RUP و با استفاده از UML و مفاهیم شیء گرایی بود. شکی نیست که اگر این پایان نامه، با توجه به حجم آن، به روش RUP و با رویکردی مبتنی بر تکرار و تکامل تدریجی انجام نمی شد، هیچ گاه به پایان نمیرسید و در مرحله پیاده سازی با شکست مواجه میشد. همچنین استفاده از جاوا، به عنوان زبانی قدرتمند در برنامه نویسی تحت شبکه، ریسک های پروژه را بسیار کمرنگ تر کرد. البته فرآیند تولید یک نرم افزار، به هیچ وجه کاری یک نفره نبوده، و اصولاً هیچ پروژه مهندسی موفقی در قالبی غیر از یک گروه منسجم انجام نمیگیرد.

امید است که این پایان نامه برای مشتاقان برنامه نویسی و جاوا مفید واقع شود...

منابع:

1. P. Kruchten, The Rational Unified Process: An Introduction, 3rd Edition, Addison Wesley, 2003.
2. Elliotte Rusty Harold, Java Network Programming, O'Reilly, 2004.
3. Jeff Friesen, Beginning Java 7, Apress, 2011.
4. Monica Pawlan, Essentials of the Java Programming Language: A Hands-On Guide, Sun Microsystems Inc, 2003.
5. James L. Weaver, Weiqi Gao, Ph.D., Stephen Chin, Dean Iverson, Johan Vos, Ph.D., Pro JavaFX 2, Apress, 2011.
6. Donn Felker, Joshua Dobbs, Android Application Development For Dummies, Wiley, 2011.
7. M.Fowler, K. Scott, UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd Edition, Addison Wesley, 2005.
8. http://docs.oracle.com

۹. عباس رسول زادگان، مدلسازی نرم افزار به کمک UML، علوم رایانه، ۱۳۹۲.

پیوست : سورس کد

در دیسیپلین پیاده سازی ، برای ساده تر شدن کد ها، فقط قطعه کدهای مربوط به شبکه و انتقال داده آورده شدند و عملاً قابل استفاده نیستند. به همین خاطر، سورس کد کامل نرم افزار طرف کاربر و طرف سرور را ضمیمه کردم.

لازم به ذکر است، کلاس های Client، MessageTask، Message و Server به ترتیب به نام های Authentication، ConnectionThread، MyObject و Main تغییر نام داده شده اند.

سورس کد طرف کاربر:

Authentication.java

```
package com.shayan.filesharing;

import android.os.Bundle;
import android.app.Activity;
import android.content.Context;
import android.view.Menu;
import android.view.View;
import android.widget.*;
import android.util.Log;

public class Authentication extends Activity {
	public static Context CONTEXT;

	public Authentication(){
		CONTEXT=this;
	}

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.authentication);
		//
		setSubmitClickListener();
	}

	protected void setSubmitClickListener(){
		Button submitButton = (Button)findViewById(R.id.button_submit);
		submitButton.setOnClickListener(new View.OnClickListener(){
```

```
                  @Override
                  public void onClick(View V) {
                        Log.d("TAG","Button clicked");
                        ConnectionThread thread=new
ConnectionThread(CONTEXT,"Login");
                        ConnectionThread.Port=getPort();
                        ConnectionThread.Address=getAddress();
                        thread.setUserPass(getUsername(), getPassword());
                        thread.start();
                  }
            });
      }
      private String getAddress(){
            EditText editText= (EditText)
findViewById(R.id.editText_ServerAddress);
            return editText.getText().toString();
      }
      private int getPort(){
            EditText editText = (EditText)
findViewById(R.id.editText_port);
            return Integer.parseInt(editText.getText().toString());
      }
      private String getUsername(){
            EditText editText=(EditText)
findViewById(R.id.editText_username);
            return editText.getText().toString();
      }
      private String getPassword(){
            EditText editText=(EditText)
findViewById(R.id.editText_password);
            return editText.getText().toString();
      }

      @Override
      public boolean onCreateOptionsMenu(Menu menu) {
            // Inflate the menu; this adds items to the action bar if it is
present.
            getMenuInflater().inflate(R.menu.authentication, menu);
            return true;
      }

}
```

ConnectionThread.java

```
package com.shayan.filesharing;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
```

```
import java.net.UnknownHostException;

import Model.MyObject;
import android.content.Context;
import android.content.Intent;
import android.util.Log;
import android.widget.Toast;
import android.net.Uri;
import android.os.Handler;
import android.os.Looper;
import android.os.Message;

public class ConnectionThread extends Thread {
	private Context context;
	public static String Address=null;
	public static int Port=0;
	private Socket connection=null;
	private String mMessage=null;
	private ObjectOutputStream Out=null;
	private ObjectInputStream in=null;
	private String Username=null;
	private String Password=null;
	private File fileToUpload=null;
	private String fileToDownload=null;
	public static String DOWNLOAD_PATH="/mnt/sdcard/Download";
	private String TAG = "File Sharing app";

	public static final int UNKNOWN_HOST_ID=0;
	public static final int IOEXCEPTION_ID=1;
	public static final int FILE_NOT_FOUND_EXCEPTION_ID=5;
	public static final int EXCEPTION_ID=7;
	public static final int INVALID_USER_ID=6;
	public static final int SENT_SECCESSFULL_ID=2;
	public static final int SENDING_FAILED_ID=3;
	public static final int FILE_EXISTS_ID=4;
	public static final int FILE_DOWNLOADED_ID=8;

	Handler handler = new Handler(){
		public void handleMessage(Message msg){
			switch(msg.what){
			case UNKNOWN_HOST_ID: Toast.makeText(context, "Unknown
host!", Toast.LENGTH_SHORT).show();
				break;
			case IOEXCEPTION_ID: Toast.makeText(context, "I/O problem
occured!", Toast.LENGTH_SHORT).show();
				break;
			case FILE_NOT_FOUND_EXCEPTION_ID: Toast.makeText(context,
"File not found!", Toast.LENGTH_SHORT).show();
				break;
			case EXCEPTION_ID: Toast.makeText(context, "some
exception occured!", Toast.LENGTH_SHORT).show();
				break;
			case INVALID_USER_ID: Toast.makeText(context, "Invalid
username or password!", Toast.LENGTH_SHORT).show();
				break;
			case SENT_SECCESSFULL_ID: Toast.makeText(context, "File
successfully sent", Toast.LENGTH_SHORT).show();
				break;
			case SENDING_FAILED_ID: Toast.makeText(context, "Sending
failed for some reason!", Toast.LENGTH_LONG).show();
				break;
```

```
                  case FILE_EXISTS_ID: Toast.makeText(context, "File
already exists!", Toast.LENGTH_LONG).show();
                        break;
                  case FILE_DOWNLOADED_ID: Toast.makeText(context,
"Downloaded to: "+DOWNLOAD_PATH, Toast.LENGTH_SHORT).show();
                        break;
                  default: break;
                  }
                  super.handleMessage(msg);
            }
      };

      public ConnectionThread(Context c,String Message){
            context=c;
            mMessage=Message;
      }
      public void run(){
            Looper.prepare();
            try {
                  connection = new Socket(Address,Port);
                  Out=new ObjectOutputStream(new
BufferedOutputStream(connection.getOutputStream()));
                  ProcessMessage(mMessage);
            } catch (UnknownHostException uhe) {
                  Log.d(TAG,"UnknownHostException");

      handler.sendMessage(Message.obtain(handler,UNKNOWN_HOST_ID));
            }
            catch (IOException ioe){
                  Log.d(TAG,"IOException");

      handler.sendMessage(Message.obtain(handler,IOEXCEPTION_ID));
                  }
            finally{
                  if(connection!=null)
                        try {
                              connection.close();
                        } catch (IOException e) {Log.d(TAG,"IOException");}
                  if(Out!=null)
                        try {
                              Out.close();
                        } catch (IOException e) {Log.d(TAG,"IOException");}
                  if(in!=null)
                        try {
                              in.close();
                        } catch (IOException e) {Log.d(TAG,"IOException");}
            }
            Looper.loop();
      }
      public void setUserPass(String Username,String Password){
            this.Password=Password;
            this.Username=Username;
      }
      public void setFileToDownload(String file){
            fileToDownload=file;
      }
      public void setFileToUpload(File file){
            fileToUpload=file;
      }
      private void ProcessMessage(String Message){
            if(Message.equals("Login")){
```

```
                        Log.d(TAG,"Loging in...");
                        Login();
                  }else if(Message.equals("Upload")){
                        Log.d(TAG,"Uploading file: "+fileToUpload+" ...");
                        Upload();
                  }
                  else if(Message.equals("File List Request")){
                        Log.d(TAG,"Requesting files list...");
                        fileListRequest();
                  }
                  else if(Message.equals("Download")){
                        Log.d(TAG,"Downloading file: "+fileToDownload+" ...");
                        Download();
                  }
                  else Log.d(TAG,"Unknown Message");
            }
            private void Login(){
                  MyObject OBJ=new MyObject();
                  OBJ.setMessage(mMessage);
                  OBJ.setPassword(Password);
                  OBJ.setUsername(Username);
                  try {
                              Out.writeObject(OBJ);
                              Out.flush();
                              Log.d(TAG,"Object sent");
                              in=new
      ObjectInputStream(connection.getInputStream());
                              OBJ = (MyObject) in.readObject();
                              if(OBJ.getMessage().equals("Valid")){
                                    Log.d(TAG,"Valid");
                                    startChooseActivity();
                              }else if(OBJ.getMessage().equals("Invalid")){
                                    Log.d(TAG,"Invalid");

            handler.sendMessage(Message.obtain(handler,INVALID_USER_ID));
                              }
                              else
                                    Log.d(TAG,"Unknown Message");
                  } catch (IOException e) {
                        Log.d(TAG,"IOException");

            handler.sendMessage(Message.obtain(handler,IOEXCEPTION_ID));
                  } catch (ClassNotFoundException e) {
                        Log.d(TAG,"ClassNotFoundException");}
            }
            private void Upload(){
                  MyObject OBJ=new MyObject();
                  OBJ.setMessage(mMessage);
                  try {
                        OBJ.setFileName(fileToUpload.getName());
                        Out.writeObject(OBJ);
                        Out.flush();
                        in=new ObjectInputStream(connection.getInputStream());
                        OBJ = (MyObject) in.readObject();
                        if(OBJ.getMessage().equals("File Exists")){
                              //File Exists!
                              Log.d(TAG,"File already exists");

            handler.sendMessage(Message.obtain(handler,FILE_EXISTS_ID));

                        }else if(OBJ.getMessage().equals("OKSend")){
```

```
                        //File Not Exists
                        Out.writeLong(fileToUpload.length());
                        BufferedInputStream bis=new BufferedInputStream(new
FileInputStream(fileToUpload));
                        byte[] buffer = new byte[100000];
                        int count=0;
                        long current=0;
                        while(current<fileToUpload.length() &&
(count=bis.read(buffer,0,(int)Math.min(buffer.length,
fileToUpload.length()-current)))!=-1){
                              Out.write(buffer,0,count);
                              Out.flush();
                              current+=count;
                        }
                        bis.close();
                        Log.d(TAG,"File sent");
                        OBJ = (MyObject) in.readObject();
                        if(OBJ.getMessage().equals("OK")){
                              Log.d(TAG,"File delivery recieved");

      handler.sendMessage(Message.obtain(handler,SENT_SECCESSFULL_ID));
                        }
                        else if(OBJ.getMessage().equals("Failed")){
                              Log.d(TAG,"File sending failed");

      handler.sendMessage(Message.obtain(handler,SENDING_FAILED_ID));
                        }
                  }//end of first else
            } catch (FileNotFoundException e) {
                  Log.d(TAG,e.toString());

      handler.sendMessage(Message.obtain(handler,FILE_NOT_FOUND_EXCEPTION_I
D));
                  }
            catch (IOException ioe) {
                  Log.d(TAG,ioe.toString());

      handler.sendMessage(Message.obtain(handler,IOEXCEPTION_ID));
                  }
            catch (ClassNotFoundException e) {
                  Log.d(TAG,e.toString());

      handler.sendMessage(Message.obtain(handler,EXCEPTION_ID));
                  }
      }
      private void fileListRequest(){
            MyObject OBJ=new MyObject();
            OBJ.setMessage(mMessage);
            try {
                  Out.writeObject(OBJ);
                  Out.flush();
                  in=new ObjectInputStream(connection.getInputStream());
                  OBJ = (MyObject) in.readObject();
                  SharedFilesListActivity.LIST=OBJ.getFilesList();
                  startListActivity();
            } catch (IOException e) {
                  Log.d(TAG,e.toString());

      handler.sendMessage(Message.obtain(handler,IOEXCEPTION_ID));
            } catch (ClassNotFoundException e) {
                  Log.d(TAG,e.toString());
```

```
	handler.sendMessage(Message.obtain(handler,EXCEPTION_ID));
		}
	}
	private void Download(){
		try {
		File file=new File(DOWNLOAD_PATH+"/"+fileToDownload);
		if(!file.exists()){
			MyObject OBJ = new MyObject();
			OBJ.setFileName(fileToDownload);
			OBJ.setMessage(mMessage);
			Out.writeObject(OBJ);
			Out.flush();
			in=new ObjectInputStream(connection.getInputStream());
			OBJ = (MyObject) in.readObject();
			if(OBJ.getMessage().equals("File Found")){
				Log.d(TAG,"File found");
				long FileLength = in.readLong();
				File DownloadFolder = new File(DOWNLOAD_PATH);
				if(!DownloadFolder.exists())
					DownloadFolder.mkdir();
				BufferedOutputStream bos=new
BufferedOutputStream(new FileOutputStream(file));
				byte[] buffer=new byte[100000];
				int count=0;
				long current=0;
				while(current<FileLength &&
(count=in.read(buffer,0,(int)Math.min(FileLength-
current,buffer.length)))!=-1){
					bos.write(buffer,0,count);
					bos.flush();
					current+=count;
				}
				bos.close();

	handler.sendMessage(Message.obtain(handler,FILE_DOWNLOADED_ID));
			}
			else //File not found in server

	handler.sendMessage(Message.obtain(handler,FILE_NOT_FOUND_EXCEPTION_I
D));
		}
		else //File already exists in device

	handler.sendMessage(Message.obtain(handler,FILE_EXISTS_ID));

		} catch (IOException e) {
			Log.d(TAG,e.toString());

	handler.sendMessage(Message.obtain(handler,IOEXCEPTION_ID));
		} catch (ClassNotFoundException e) {
			Log.d(TAG,e.toString());

	handler.sendMessage(Message.obtain(handler,EXCEPTION_ID));
		}
	}
	private void startChooseActivity(){
		Intent i= new Intent(context,ChooseActivity.class);
		context.startActivity(i);
	}
	private void startListActivity(){
```

```
            Intent i= new Intent(context,SharedFilesListActivity.class);
            context.startActivity(i);
      }
}
```

ChooseActivity.java

```
package com.shayan.filesharing;

import java.io.File;
import java.io.IOException;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.provider.MediaStore;
import android.util.Log;
import android.view.View;
import android.widget.Button;

public class ChooseActivity extends Activity {
      public static Context CONTEXT;
      static final int SEND_FILE_REQUESTCODE=1;
      private Button UploadButton;
      private Button DownloadButton;


      public ChooseActivity(){
            CONTEXT=this;
      }
      @Override
      protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.choose);

            setUploadButton();
            setDownloadButton();
      }
      private void setUploadButton(){
            UploadButton = (Button) findViewById(R.id.Button_Upload);
            UploadButton.setOnClickListener(new View.OnClickListener() {
                  @Override
                  public void onClick(View arg0) {
                        Intent intent = new
Intent(Intent.ACTION_GET_CONTENT);
                        Uri
uri=Uri.parse(Environment.getExternalStorageDirectory().getPath());
                        intent.setDataAndType(uri, "text/csv");
                        startActivityForResult(Intent.createChooser(intent,
"Open folder"), SEND_FILE_REQUESTCODE);

                  }
            });
      }
      @Override
```

```
	protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
		if(requestCode==SEND_FILE_REQUESTCODE)
			if(resultCode==RESULT_OK){
				String path=data.getData().getPath();
				if(path.contains("/external/images/media"))
					path=convertImageUriToPath(data.getData());
				Log.d("TAG","File to upload is: "+path);
				File file=new File(path);
				ConnectionThread thread=new
ConnectionThread(CONTEXT,"Upload");
				thread.setFileToUpload(file);
				thread.start();
			}
	}
	protected String convertImageUriToPath(Uri uri){
		String[] projection={MediaStore.Images.Media.DATA};
		Cursor cursor=getContentResolver().query(uri, projection, null,
null, null);
		int
ColumnIndex=cursor.getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
		cursor.moveToFirst();
		String path=cursor.getString(ColumnIndex);
		cursor.close();
		return path;
	}
	private void setDownloadButton(){
		DownloadButton=(Button) findViewById(R.id.Button_download);
		DownloadButton.setOnClickListener(new View.OnClickListener() {
			@Override
			public void onClick(View arg0) {
				new ConnectionThread(CONTEXT,"File List
Request").start();
			}});
	}
}
```

SharedFilesListActivity.java

```
package com.shayan.filesharing;

import java.io.File;

import android.app.ListActivity;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class SharedFilesListActivity extends ListActivity {

	public static Context CONTEXT;
	public static String[] LIST;

	public SharedFilesListActivity(){
		CONTEXT=this;
	}
```

```
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.shared_files_list);

		ArrayAdapter<String> adapter=new
ArrayAdapter<String>(this,R.layout.shared_files_row,R.id.text1,LIST);
		setListAdapter(adapter);
	}
	@Override
	protected void onListItemClick(ListView l,View v,int position,long
id){
		super.onListItemClick(l, v, position, id);
		String
FileName=getListView().getItemAtPosition(position).toString();
		ConnectionThread thread= new
ConnectionThread(CONTEXT,"Download");
		thread.setFileToDownload(FileName);
		thread.start();

	}
	protected void getItemAtPosition(int position){

	}
}
```

MyObject.java

```
package Model;

import java.io.Serializable;

public class MyObject implements Serializable {

	private static final long serialVersionUID = 1L;
	private String Message=null;
	private String Username=null;
	private String Password=null;
	private byte[] FileData;
	private String FileName;
	private String[] FilesList;

	public void setMessage(String Message){
		this.Message=Message;
	}
	public void setPassword(String Password){
		this.Password=Password;
	}
	public void setUsername(String Username){
		this.Username=Username;
	}
	public void setFileLength(int Length){
		FileData=new byte[Length];
	}
	public void setFileData(byte[] Data){
		FileData=Data;
	}
	public byte[] getFileData(){
		return FileData;
	}
```

```
	public void setFileName(String Name){
		FileName=Name;
	}
	public void setFilesList(String[] Files){
		FilesList=Files;
	}
	public String[] getFilesList(){
		return FilesList;
	}
	public String getFileName(){
		return FileName;
	}
	public String getMessage(){
		return Message;
	}
	public String getPassword(){
		return Password;
	}
	public String getUsername(){
		return Username;
	}
}
```

---

authentication.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".Authentication" >

    <TextView
        android:id="@+id/textView_login"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_marginLeft="32dp"
        android:layout_marginTop="35dp"
        android:text="@string/loginText" />

    <EditText
        android:id="@+id/editText_username"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/textView_login"
        android:layout_below="@+id/textView_login"
        android:layout_marginTop="10dp"
        android:ems="10"
        android:inputType="textPersonName"
        android:hint="@string/username" />


    <EditText
        android:id="@+id/editText_password"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/editText_username"
        android:layout_below="@+id/editText_username"
        android:ems="10"
        android:inputType="textPassword"
        android:hint="@string/password" />

    <Button
        android:id="@+id/button_submit"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/editText_password"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="14dp"
        android:text="@string/submit" />

    <TextView
        android:id="@+id/textView_Hostname"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/editText_port"
        android:layout_above="@+id/editText_ServerAddress"
        android:text="@string/server_name_address" />

    <EditText
        android:id="@+id/editText_ServerAddress"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/editText_port"
        android:layout_above="@+id/textView_port"
        android:ems="10"
        android:inputType="text"
        android:text="@string/_192_168_1_10"
        android:textSize="18sp" />

    <TextView
        android:id="@+id/textView_port"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/editText_port"
        android:layout_above="@+id/editText_port"
        android:text="@string/port" />

    <EditText
        android:id="@+id/editText_port"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/textView_login"
        android:layout_alignParentBottom="true"
        android:layout_marginBottom="15dp"
        android:ems="10"
        android:text="@string/_5706"
        android:inputType="number" />


</RelativeLayout>
```

Choose.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <Button
        android:id="@+id/Button_download"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="150dp"
        android:text="@string/Download" />

    <Button
        android:id="@+id/Button_Upload"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/Button_download"
        android:layout_alignRight="@+id/Button_download"
        android:layout_below="@+id/Button_download"
        android:layout_marginTop="10dp"
        android:text="@string/Upload" />

</RelativeLayout>
```

Shared_files_list.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
      <ListView android:id="@+id/android:list"
             android:layout_width="match_parent"
             android:layout_height="match_parent" />
      <TextView android:id="@+id/android:empty"
         android:layout_width="wrap_content"
         android:layout_height= "wrap_content"
         android:text="nothing!" />

</LinearLayout>
```

Shared_files_row.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/text1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="10dip" />
```

---

strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">File Sharing</string><string
name="action_settings">Settings</string>
    <string name="loginText">Please enter your username and
password:</string>
    <string name="username">Username</string>
    <string name="password">********</string>
    <string name="server_name_address">Server Name/Address:</string>
    <string name="_192_168_1_10">192.168.1.10</string>
    <string name="port">Port:</string>
    <string name="submit">Submit</string>
    <string name="_5706">60000</string>
    <string name="invalid_UserPass">نام یا رمز عبور اشتباه است!</string>

</resources>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.shayan.filesharing"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />
    <uses-permission
        android:name="android.permission.INTERNET"/>
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme"
        android:debuggable="true"
        android:permission="android.permission.INTERNET">
        <activity
            android:name=".Authentication"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
/>
```

```
            </intent-filter>
        </activity>
        <activity
            android:name=".SharedFilesListActivity"
            android:label="@string/app_name" />
        <activity
            android:name=".ChooseActivity"
            android:label="@string/app_name" />
    </application>

</manifest>
```

سورس کد طرف سرور:

Main.java

```
package com.Shayan.FileSharing.Server;

import java.io.IOException;

import javafx.application.Application;
import javafx.concurrent.Task;
import javafx.concurrent.WorkerStateEvent;
import javafx.event.EventHandler;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Stage;

public class Main extends Application {

      private MessageListener messageListener;

      public static void main(String[] args) {
            launch(Main.class, (String[])null);
      }

      @Override
      public void start(Stage primaryStage) {
            initializeDatabase();
            AnchorPane rootPane=null;
            try {

      rootPane=(AnchorPane)FXMLLoader.load(getClass().getResource("Main.fxm
l"));
            } catch (IOException e) {System.err.println(e);}
                  Scene scene = new Scene(rootPane);
            primaryStage.setScene(scene);
            primaryStage.setTitle("File sharing");
            primaryStage.show();


            messageListener = new MessageListener();
            messageListener.setOnSucceeded(new
EventHandler<WorkerStateEvent>() {
```

```
                        @Override
                        public void handle(WorkerStateEvent event) {
                              messageListener.restart();
                        }
                  });
            messageListener.setOnFailed(new
EventHandler<WorkerStateEvent>() {
                        @Override
                        public void handle(WorkerStateEvent event) {
                              System.out.println("MessageListener failed");
                              messageListener.reset();
                        }
                  });
            messageListener.start();

      }
      public void initializeDatabase(){
            Task<Void> task = new Task<Void>(){
                  @Override
                  protected Void call() throws Exception {
                        DatabaseManager DBM=new DatabaseManager();
                        DBM.initalizeDatabase();
                        return null;
                  }
            };
            new Thread(task).start();
      }


}
```

MessageListener.java

```
package com.Shayan.FileSharing.Server;

import javafx.concurrent.Service;
import javafx.concurrent.Task;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FilenameFilter;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.net.ServerSocket;
import java.nio.file.SimpleFileVisitor;

import org.apache.derby.iapi.services.io.FileUtil;

import Model.MyObject;


public class MessageListener extends Service<String> {
```

```
	public static int port=60000;
	private Socket connection=null;
	private ObjectInputStream in=null;
	private ObjectOutputStream out=null;
	private MyObject OBJ;
	private File fileToUpload=null;
	private boolean fileNotFound=false;

	public static void setPort(int PortNumber){
		port=PortNumber;
		}

	public Socket getConnection(){
		return connection;
	}

	protected Task<String> createTask(){
		return new Task<String>(){
			protected String call(){
				System.out.println("Listening for connections on
port: " + port);
				String result=new String();
				try (ServerSocket SS=new ServerSocket(port);
					Socket connection = SS.accept();)
					{
					System.out.println("connection accepted");
					in = new ObjectInputStream(new
BufferedInputStream(connection.getInputStream()));
					out = new ObjectOutputStream(new
BufferedOutputStream(connection.getOutputStream()));
					OBJ=(MyObject) in.readObject();
					result = OBJ.getMessage();
					ProccessMessage(OBJ.getMessage());
					in.close();
					out.close();
				} catch (IOException e) {
					e.printStackTrace();}
				catch (ClassNotFoundException e) {
					e.printStackTrace();}

				return result;
			}
		};
	}
	private void ProccessMessage(String Message){
		if(Message.equals("Login")){
			System.out.println("Message is: Login");
			ValidateUser();
		}else if(Message.equals("Upload")){
			System.out.println("Message is: Upload, and the file name
is: "+OBJ.getFileName());
			UserUploads();
		}
		else if(Message.equals("File List Request")){
			System.out.println("Message is: File List Request");
			sendFileList();
		}
		else if(Message.equals("Download")){
			System.out.println("Message is: Download, and the file
name is: "+OBJ.getFileName());
			UserDownloads();
```

```
            }
      }
      private void ValidateUser(){
            try {
                  System.out.println("UserName: " + OBJ.getUsername());
                  System.out.println("Password: " + OBJ.getPassword());
                  DatabaseManager DBM=new DatabaseManager();
                  if(DBM.UserIsValid(OBJ.getUsername(),OBJ.getPassword()))
                        OBJ.setMessage("Valid");
                  else
                        OBJ.setMessage("Invalid");
                  out.writeObject(OBJ);
                  out.flush();

            } catch (IOException e) {
                  e.printStackTrace();}
      }

      private void UserUploads(){
            String Message;
            char SC=File.separatorChar;
            File file=new
File(System.getProperty("user.home")+SC+"Downloads"+SC+"File
Sharing"+SC+OBJ.getFileName());
            if(file.exists()){
                  Message="File Exists";
                  System.out.println("File: "+file.getName()+" supposed to
download, but it already exists in system");
                  try {
                        OBJ=new MyObject();
                        OBJ.setMessage(Message);
                        out.writeObject(OBJ);
                        out.flush();
                  } catch (IOException e) {e.printStackTrace();
                  }
            }
            else{
                  try {
                        OBJ=new MyObject();
                        OBJ.setMessage("OKSend");
                        out.writeObject(OBJ);
                        out.flush();
                  } catch (IOException e) {e.printStackTrace();
                  }
                  try {
                        BufferedOutputStream bos=new
BufferedOutputStream(new FileOutputStream(file));
                        long FileLength = in.readLong();
                        byte[] buffer=new byte[100000];
                        int count=0;
                        long current=0;
                        while(current<FileLength &&
(count=in.read(buffer,0,(int)Math.min(FileLength-
current,buffer.length)))!=-1){
                              bos.write(buffer,0,count);
                              bos.flush();
                              current+=count;
                        }
                        bos.close();
                        System.out.println("File downloaded:
"+file.getAbsolutePath());
```

```
                        Message="OK";
                  } catch (FileNotFoundException e) {
                        Message="Failed";
                        e.printStackTrace();
                  } catch (IOException e) {
                        Message="Failed";
                        e.printStackTrace();
                        }
                  try {
                        OBJ=new MyObject();
                        OBJ.setMessage(Message);
                        out.writeObject(OBJ);
                        out.flush();
                  } catch (IOException e) {e.printStackTrace();}
            }//end of else
      }
      private void sendFileList(){
            String[] Folders=new DatabaseManager().returnFolders();
            StringBuffer sb=new StringBuffer();
            for(int i=0;i<Folders.length;i++){
                  AppendFiles(Folders[i], sb);
            }
            String[] items=sb.toString().split(":");
            OBJ.setFilesList(items);
            try {
                  out.writeObject(OBJ);
                  out.flush();
            } catch (IOException e) {
                  e.printStackTrace();
            }
      }
      private void AppendFiles(String Directory,StringBuffer sb){
            File Folder=new File(Directory);
            File[] Files=Folder.listFiles();
            for(int j=0;j<Files.length;j++){
                  if(!Files[j].isDirectory())
                        sb.append(Files[j].getName()+":");
                  else AppendFiles(Files[j].getAbsolutePath(), sb);
            }
      }
      private void UserDownloads(){
            try {
                  SearchFile(OBJ.getFileName());
                  if(!fileNotFound){//The file found
                        System.out.println("The file:
"+fileToUpload.getName()+" found");
                        OBJ.setMessage("File Found");
                        out.writeObject(OBJ);
                        out.flush();
                        out.writeLong(fileToUpload.length());
                        out.flush();
                        BufferedInputStream bis=new BufferedInputStream(new
FileInputStream(fileToUpload));
                        byte[] buffer = new byte[100000];
                        int count=0;
                        long current=0;
                        while(current<fileToUpload.length() &&
(count=bis.read(buffer,0,(int)Math.min(buffer.length,
fileToUpload.length()-current)))!=-1){
                              out.write(buffer,0,count);
                              out.flush();
```

```
                        current+=count;
                    }
                    bis.close();
                    System.out.println("File sent");
                }
                else{//The file not found
                    OBJ.setMessage("File Not Found");
                    out.writeObject(OBJ);
                    out.flush();
                    System.out.println("The file: "+OBJ.getFileName()+"
not found");
                }
            } catch (IOException e) {
                e.printStackTrace();
            }

        }
        private void SearchFile(final String FileName){
            fileNotFound=true;
            String[] Foldernames=new DatabaseManager().returnFolders();
            for(int i=0;i<Foldernames.length;i++){
                if(!fileNotFound)
                    break;
                File Folder=new File (Foldernames[i]);
                SearchFileInFolder(FileName,Folder);
            }
        }
        private void SearchFileInFolder(final String FileName,File dir){
                File[] files=dir.listFiles();
                for(File file : files){
                    if(!fileNotFound)
                        break;
                    if(file.isDirectory())
                        SearchFileInFolder(FileName,file);
                    else if(file.getName().equals(FileName)){
                            fileToUpload=file;
                            fileNotFound=false;
                        }
                }
        }
}
```

DatabaseManager.java

```
package com.Shayan.FileSharing.Server;

import java.io.File;
import java.io.IOException;
import java.sql.Connection;
import java.sql.Driver;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Properties;
```

```
import org.apache.derby.jdbc.EmbeddedDriver;

import Model.User;


public class DatabaseManager {
	private String sql=null;
	private Properties prop;
	private Driver driver=null;
	private String url="jdbc:derby:FileSharingDatabase";


	public DatabaseManager(){
		prop=new Properties();
		prop.put("create", "false");
		driver=new EmbeddedDriver();
	}
	public boolean InsertIntoUsers(User user){
			try(Connection con= driver.connect(url, prop);
					Statement stmt=con.createStatement();)
{
			sql="INSERT INTO Users
VALUES('"+user.getUsername()+"','"+user.getPassword()+"')";
			stmt.executeUpdate(sql);
			System.out.println("row inserted");
			return true;
			} catch (SQLException e) {
				//e.printStackTrace();
				return false;}
	}
	public boolean DeleteUser(User user){
		try(Connection con= driver.connect(url, prop);
				Statement stmt=con.createStatement();) {
			sql="DELETE FROM Users WHERE
Username='"+user.getUsername()+"'";
			stmt.executeUpdate(sql);
			System.out.println("User: "+user.getUsername()+"
deleted");
			return true;
		} catch (SQLException e) {
			e.printStackTrace();
			return false;}
	}
	public boolean UserIsValid (String Username,String Password){
		sql="SELECT Username,Password FROM Users WHERE Username=?";
		try(Connection con=driver.connect(url, prop);){
			PreparedStatement pstmt=con.prepareStatement(sql);
			pstmt.setString(1, Username);
			ResultSet rs=pstmt.executeQuery();
			if(!rs.next()){
				System.out.println("user not exists");
				return false;}
			if(!rs.getString(2).equals(Password)){
				System.out.println("password is invalid, it was:
"+rs.getString(2));
				return false;}
		}catch(SQLException sqlex){
			sqlex.printStackTrace();
			return false;
			}
		System.out.println("passwords match, user logged in");
```

```
            return true;
      }
      public User[] returnUsers() throws SQLException{
            Connection con=driver.connect(url, prop);
                  Statement stmt=con.createStatement();
                  sql="SELECT count(*) FROM Users";
                  ResultSet rs = stmt.executeQuery(sql);
                  rs.next();
                  System.out.println("\nUsers count = "+rs.getInt(1));
                  User[] users=new User[rs.getInt(1)];
                  sql="SELECT Username,Password FROM Users";
                  rs = stmt.executeQuery(sql);
                  int i=0;
                  while(rs.next()){
                        users[i]=new User(rs.getString(1),rs.getString(2));
                        i++;}
                  con.close();
                  stmt.close();
                  rs.close();
                  return users;
            }
      public String[] returnFolders(){
            try (Connection con=driver.connect(url, prop);
                        Statement stmt=con.createStatement();){
                  sql="SELECT count(*) FROM Folders";
                  ResultSet rs=stmt.executeQuery(sql);
                  rs.next();
                  String[] Folders=new String[rs.getInt(1)];
                  System.out.println("Shared folders count:
"+rs.getInt(1));
                  sql="SELECT * FROM Folders";
                  rs=stmt.executeQuery(sql);
                  for(int i=0;rs.next();i++)
                        Folders[i]=rs.getString(1);
                  return Folders;
            } catch (SQLException e) {
                  e.printStackTrace();
                  return null;
            }
      }
      public boolean InsertIntoFolders(File Directory){
            try (Connection con=driver.connect(url, prop);
                        Statement stmt=con.createStatement();){
                  sql="INSERT INTO Folders
VALUES('"+Directory.getCanonicalPath()+"')";
                  stmt.executeUpdate(sql);
                  System.out.println("Folder shared:
"+Directory.getAbsolutePath());
                  return true;
            } catch (SQLException e) {
                  e.printStackTrace();
                  return false;
            } catch (IOException e) {
                  e.printStackTrace();
                  return false;
            }
      }
      public boolean DeleteFolder(File Directory){
            try (Connection con = driver.connect(url, prop);
                        Statement stmt=con.createStatement();){
```

```
                  sql="DELETE FROM Folders WHERE
Directory='"+Directory.getCanonicalPath()+"'";
                  stmt.executeUpdate(sql);
                  System.out.println("This folder is not shared anymore:
"+Directory.getCanonicalPath());
                  return true;
            } catch (SQLException e) {
                  e.printStackTrace();
                  return false;
            } catch (IOException e) {
                  e.printStackTrace();
                  return false;
            }
      }
      //Create Database and its tables if they not exist
      public void initalizeDatabase(){
            String[] TableNames = new String[]{"Users","Folders"};
            String CreateUsersTableStatement="CREATE TABLE Users(Username
VARCHAR(20) PRIMARY KEY,Password VARCHAR(20))";
            String CreateFoldersTableStatement="CREATE TABLE
Folders(Directory VARCHAR(200) PRIMARY KEY)";
            String[] CreateTableStatements = new
String[]{CreateUsersTableStatement,CreateFoldersTableStatement};
            prop.put("create", "true");
            for(int i=0;i<TableNames.length;i++){
                  try(Connection con=driver.connect(url, prop);){
                        PreparedStatement pstmt =
con.prepareStatement("SELECT t.tablename FROM sys.systables t WHERE
t.tablename=?");
                        pstmt.setString(1, TableNames[i]);
                        ResultSet rs=pstmt.executeQuery();
                        if(!rs.next()){
                              Statement stmt=con.createStatement();
                              stmt.executeUpdate(CreateTableStatements[i]);
                              stmt.close();
                        }
                        pstmt.close();
                        rs.close();
                  }catch(SQLException sqlex){System.out.println("Table
'"+TableNames[i]+"' already exsists");}
            }
            //Create new folder to download user files and add it (the
folder) to Folders table
            prop.put("create", "false");
            char SC=File.separatorChar;
            File downloadDir=new
File(System.getProperty("user.home")+SC+"Downloads"+SC+"File Sharing");
            if (!downloadDir.exists())
                  downloadDir.mkdir();
            try(Connection con=driver.connect(url, prop);){
                  Statement stmt=con.createStatement();
                  stmt.executeUpdate("INSERT INTO Folders
Values('"+downloadDir.getAbsolutePath()+"')");
                  stmt.close();
            }catch(SQLException sqlex){System.out.println("Download folder
already exists: "+downloadDir.getAbsolutePath());}
      }

}
```

MyObject.java

```
package Model;

import java.io.Serializable;

public class MyObject implements Serializable {

	private static final long serialVersionUID = 1L;
	private String Message=null;
	private String Username=null;
	private String Password=null;
	private byte[] FileData;
	private String FileName;
	private String[] FilesList;

	public void setMessage(String Message){
		this.Message=Message;
	}
	public void setPassword(String Password){
		this.Password=Password;
	}
	public void setUsername(String Username){
		this.Username=Username;
	}
	public void setFileLength(int Length){
		FileData=new byte[Length];
	}
	public void setFileData(byte[] Data){
		FileData=Data;
	}
	public byte[] getFileData(){
		return FileData;
	}
	public void setFileName(String Name){
		FileName=Name;
	}
	public void setFilesList(String[] Files){
		FilesList=Files;
	}
	public String[] getFilesList(){
		return FilesList;
	}
	public String getFileName(){
		return FileName;
	}
	public String getMessage(){
		return Message;
	}
	public String getPassword(){
		return Password;
	}
	public String getUsername(){
		return Username;
	}
}
```

User.java

```
package Model;

public class User {
	private String Username;
	private String Password;

	public String getUsername(){
		return Username;
	}
	public String getPassword(){
		return Password;
	}
	public User(String Username,String Password){
		this.Username=Username;
		this.Password=Password;
	}
	public void Set(String Username,String Password){
		this.Username=Username;
		this.Password=Password;
	}
	public String toString(){
		return Username;
	}
}
```

FileSharingController.java

```
package com.Shayan.FileSharing.Server;

import java.io.File;
import java.io.IOException;
import java.net.URL;
import java.sql.SQLException;
import java.util.ResourceBundle;

import org.controlsfx.control.action.Action;
import org.controlsfx.dialog.Dialog;
import org.controlsfx.dialog.Dialogs;

import Model.User;

import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.concurrent.Task;
import javafx.concurrent.WorkerStateEvent;
import javafx.event.Event;
import javafx.event.EventHandler;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.ListView;
import javafx.scene.control.TextField;
import javafx.scene.layout.AnchorPane;
import javafx.stage.DirectoryChooser;
import javafx.stage.FileChooser;
import javafx.stage.Stage;
```

```
public class FileSharingController implements Initializable {
	MessageListener messageListener;
	private ObservableList<User> UsersListData;
	private ObservableList<String> FoldersListData;
	@FXML
	private ListView<User> UsersList;
	@FXML
	private ListView<String> FoldersList;
	@FXML
	private TextField PortTextField;
	@FXML
	private Button OkPortSetedBtn;
	@FXML
	private Button OkDirectoryAdded;


	public FileSharingController(){
		UsersListData=FXCollections.observableArrayList();
		FoldersListData=FXCollections.observableArrayList();
	}

	@Override
	public void initialize(URL location, ResourceBundle resources) {
		assert UsersList!=null : "fx:id=\"UsersList\" was not injected:
check your FXML file 'Main.fxml'.";
		assert FoldersList!=null : "fx:id=\"FoldersList\" was not
injected: check your FXML file 'Main.fxml'.";
		System.out.println("initializing...");

		//Update Folders ListView
		final Task<ObservableList<String>> FoldersListTask=new
Task<ObservableList<String>>() {
			@Override
			protected ObservableList<String> call() throws Exception
{
				FoldersListData.clear();
				FoldersListData.addAll(new
DatabaseManager().returnFolders());
				return FoldersListData;
			}
		};
		FoldersListTask.setOnSucceeded(new
EventHandler<WorkerStateEvent>() {
			@Override
			public void handle(WorkerStateEvent arg0) {

	FoldersList.setItems(FXCollections.observableArrayList(FoldersListTas
k.getValue()));
			}
		});
		FoldersListTask.setOnCancelled(new
EventHandler<WorkerStateEvent>() {

			@Override
			public void handle(WorkerStateEvent event) {
				System.out.println(FoldersListTask.getException());

			}
		});
		new Thread(FoldersListTask).start();
```

```
            //Update Users ListView
            final Task<ObservableList<User>> UsersListTask=new
Task<ObservableList<User>>(){
                        protected ObservableList<User> call() throws
SQLException{
                              UsersListData.clear();
                              UsersListData.addAll(new
DatabaseManager().returnUsers());
                              return UsersListData;}};
            UsersListTask.setOnSucceeded(new
EventHandler<WorkerStateEvent>() {
                        @Override
                        public void handle(WorkerStateEvent arg0) {

      UsersList.setItems(FXCollections.observableArrayList(UsersListTask.ge
tValue()));
                        }
            });
            UsersListTask.setOnFailed(new EventHandler<WorkerStateEvent>()
{
                        @Override
                        public void handle(WorkerStateEvent event) {

      System.out.println(UsersListTask.getException());
                        }
            });
            new Thread(UsersListTask).start();

      }
      @FXML
      public void showAddUserLayout(){
            try {Stage stage=new Stage();
            AnchorPane
rootPane=FXMLLoader.load(getClass().getResource("AddUser.fxml"));
            Scene scene = new Scene(rootPane);
            stage.setScene(scene);
            stage.setTitle("Add User");
            stage.show();
      } catch (IOException e) {e.printStackTrace();}
      }
      @FXML
      public void chooseFolder(){
            Stage stage=(Stage) FoldersList.getScene().getWindow();
            File Directory=new DirectoryChooser().showDialog(stage);
            if(new DatabaseManager().InsertIntoFolders(Directory))
                  showDirectoryAddedDialog();
      }
      @FXML
      private void showDirectoryAddedDialog(){
            try {Stage stage=new Stage();
            AnchorPane
rootPane=FXMLLoader.load(getClass().getResource("DirectoryAdded.fxml"));
            Scene scene = new Scene(rootPane);
            stage.setScene(scene);
            stage.show();
            } catch (IOException e) {e.printStackTrace();}
      }
      @FXML
      private void closeDirectoryAddedDialog(){
            Stage stage=(Stage) OkDirectoryAdded.getScene().getWindow();
            stage.close();
```

```
	}
	@FXML
	private void deleteFolder(){
		File Directory= new
File(FoldersList.getSelectionModel().getSelectedItem());
		if(new DatabaseManager().DeleteFolder(Directory))
			showFolderDeletedDialog(Directory);
	}
	private void showFolderDeletedDialog(File Directory){
		try {
			AnchorPane
pane=FXMLLoader.load(getClass().getResource("FolderDeletedDialog.fxml"));
			Scene scene=new Scene(pane);
			Stage stage=new Stage();
			stage.setScene(scene);
			stage.setTitle("Action completed");
			stage.show();
		} catch (IOException e) {
			e.printStackTrace();
		}
	}
	@FXML
	public void showSetPortLayout(){
		try {Stage stage=new Stage();
			AnchorPane
rootPane=FXMLLoader.load(getClass().getResource("SetPort.fxml"));
			Scene scene = new Scene(rootPane);
			stage.setScene(scene);
			stage.setTitle("Set port");
			stage.show();
		} catch (IOException e) {e.printStackTrace();}
	}
	@FXML
	public void setPort(){

	MessageListener.setPort(Integer.parseInt(PortTextField.getText()));
		System.out.println("Port changed to: "+MessageListener.port);
		Stage stage = (Stage)OkPortSetedBtn.getScene().getWindow();
		stage.close();

	}
	@FXML
	public void startService(){
		messageListener= new MessageListener();
	  messageListener.start();
	}
	@FXML
	public void stopService(){
		messageListener.cancel();
	}

	@FXML
	private void deleteUser(){
		User user=UsersList.getSelectionModel().getSelectedItem();
		DatabaseManager DB=new DatabaseManager();
		if(DB.DeleteUser(user)){
			showUserDeletedLayout();
		}
	}
	private void showUserDeletedLayout(){
		try {Stage stage=new Stage();
```

```
            AnchorPane
rootPane=FXMLLoader.load(getClass().getResource("UserDeleted.fxml"));
            Scene scene = new Scene(rootPane);
            stage.setScene(scene);
            stage.show();
      } catch (IOException e) {e.printStackTrace();}
      }
      @FXML
      private void refresh(){
            initialize(null, null);

      }
}
```

AddUserController.java

```
package com.Shayan.FileSharing.Server;

import java.io.IOException;

import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Stage;
import Model.User;

public class AddUserController {
      @FXML
      private TextField AddUsernameTextfield;
      @FXML
      private TextField AddPasswordTextfield;
      @FXML
      private TextField AddPassword2Textfield;
      @FXML
      private Button OkUserAddedBtn;
      @FXML
      private Button AddUserBtn;
      @FXML
      private Button OkPasswordsDontmatchBtn;
      @FXML
      private Button OkUserExistsBtn;



      private void showPasswordsDontMatchLayout(){
            try {Stage stage=new Stage();
            AnchorPane
rootPane=FXMLLoader.load(getClass().getResource("PasswordsDontMatch.fxml"))
;
            Scene scene = new Scene(rootPane);
            stage.setScene(scene);
            stage.setTitle("Error");
            stage.show();
      } catch (IOException e) {e.printStackTrace();}
```

```
	}
	@FXML
	private void closePasswordsDontMatchLayout(){
		Stage stage=
(Stage)OkPasswordsDontmatchBtn.getScene().getWindow();
		stage.close();
	}
	private void showUserExistsLayout(){
		try {Stage stage=new Stage();
		AnchorPane
rootPane=FXMLLoader.load(getClass().getResource("UserExists.fxml"));
		Scene scene = new Scene(rootPane);
		stage.setScene(scene);
		stage.setTitle("Error");
		stage.show();
	} catch (IOException e) {e.printStackTrace();}
	}
	@FXML
	private void closeUserExistsLayout(){
		Stage stage=(Stage) OkUserExistsBtn.getScene().getWindow();
		stage.close();
	}
	private void showUserAddedLayout(){
		try {Stage stage=new Stage();
		AnchorPane
rootPane=FXMLLoader.load(getClass().getResource("UserAddedMessage.fxml"));
		Scene scene = new Scene(rootPane);
		stage.setScene(scene);
		stage.show();
	} catch (IOException e) {e.printStackTrace();}
	}
	@FXML
	public void CloseUserAddedDialog(){
		Stage stage= (Stage)OkUserAddedBtn.getScene().getWindow();
		stage.close();
	}
	@FXML
	public void addUser(){
		String Username=AddUsernameTextfield.getText();
		String Password1=AddPasswordTextfield.getText();
		String Password2=AddPassword2Textfield.getText();
		System.out.println("information:
"+Username+Password1+Password2);

		if(!Password1.equals(Password2))
			showPasswordsDontMatchLayout();
		else if(Username==null || Password1==null || Password2==null)
			System.out.println("please fill the user informations");
		else {
			User user=new User(Username,Password1);
			DatabaseManager DM=new DatabaseManager();
			if(DM.InsertIntoUsers(user)){
				showUserAddedLayout();
				Stage stage=
(Stage)AddUserBtn.getScene().getWindow();
				stage.close();
			}else showUserExistsLayout();
		}
}
}
```

FolderDeletedController.java

```
package com.Shayan.FileSharing.Server;

import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.stage.Stage;

public class FolderDeletedController {
	@FXML
	private Button Ok;
	@FXML
	private void CloseDialog(){
		Stage stage= (Stage)Ok.getScene().getWindow();
		stage.close();
	}


}
```

UserDeletedController.java

```
package com.Shayan.FileSharing.Server;

import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.stage.Stage;

public class UserDeletedController {
	@FXML
	private Button OkUserRemovedBtn;
	@FXML
	private void CloseUserDeletedDialog(){
		Stage stage= (Stage)OkUserRemovedBtn.getScene().getWindow();
		stage.close();
	}


}
```

Main.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>
<?import java.net.*?>
<?import java.util.*?>
```

```
<?import javafx.scene.control.*?>
<?import javafx.scene.control.TabPane?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.layout.AnchorPane?>

<AnchorPane onContextMenuRequested="#refresh" prefHeight="400.0"
prefWidth="600.0" styleClass="theme" xmlns:fx="http://javafx.com/fxml/1"
xmlns="http://javafx.com/javafx/2.2"
fx:controller="com.Shayan.FileSharing.Server.FileSharingController">
  <!-- TODO Add Nodes -->
  <children>
    <TabPane prefHeight="374.0" prefWidth="600.0"
tabClosingPolicy="UNAVAILABLE" AnchorPane.bottomAnchor="0.0"
AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0"
AnchorPane.topAnchor="26.0">
      <tabs>
        <Tab text="Folders">
          <content>
            <AnchorPane id="Content" minHeight="0.0" minWidth="0.0"
prefHeight="180.0" prefWidth="200.0">
              <children>
                <Label layoutX="14.0" layoutY="14.0" text="Shared Folders:"
/>
                <ListView fx:id="FoldersList" prefHeight="282.0"
prefWidth="572.0" AnchorPane.bottomAnchor="50.0"
AnchorPane.leftAnchor="14.0" AnchorPane.rightAnchor="14.0"
AnchorPane.topAnchor="30.0" />
                <Button layoutX="14.0" mnemonicParsing="false"
onMouseClicked="#chooseFolder" text="Add" AnchorPane.bottomAnchor="16.0" />
                <Button layoutX="67.0" mnemonicParsing="false"
onMouseClicked="#deleteFolder" text="Remove" AnchorPane.bottomAnchor="16.0"
/>
              </children>
            </AnchorPane>
          </content>
        </Tab>
        <Tab text="Users">
          <content>
            <AnchorPane id="Content" minHeight="0.0" minWidth="0.0"
prefHeight="180.0" prefWidth="200.0">
              <children>
                <Label layoutX="14.0" layoutY="14.0" text="Users:" />
                <VBox id="VBox" alignment="CENTER" layoutX="291.0"
layoutY="30.0" prefHeight="81.0" prefWidth="74.0" spacing="5.0">
                  <children>
                    <Button mnemonicParsing="false"
onMouseClicked="#showAddUserLayout" prefWidth="72.0" text="Add" />
                    <Button mnemonicParsing="false"
onMouseClicked="#deleteUser" text="Remove" />
                    <Button mnemonicParsing="false" prefWidth="72.0"
text="Edit" />
                  </children>
                </VBox>
                <ListView fx:id="UsersList" layoutX="14.0"
prefHeight="292.0" prefWidth="266.0" AnchorPane.bottomAnchor="14.0"
AnchorPane.topAnchor="30.0" />
              </children>
            </AnchorPane>
          </content>
        </Tab>
      </tabs>
```

```
    </TabPane>
    <MenuBar prefWidth="600.0" AnchorPane.leftAnchor="-1.0"
AnchorPane.rightAnchor="1.0" AnchorPane.topAnchor="2.0">
      <menus>
        <Menu mnemonicParsing="false" text="File">
          <items>
            <MenuItem mnemonicParsing="false" text="Exit" />
          </items>
        </Menu>
        <Menu mnemonicParsing="false" text="Edit">
          <items>
            <MenuItem mnemonicParsing="false" onAction="#showSetPortLayout"
text="Set port" />
            <MenuItem mnemonicParsing="false" onAction="#startService"
text="Start Service" />
            <MenuItem mnemonicParsing="false" onAction="#stopService"
text="Stop Service" />
            <MenuItem mnemonicParsing="false" onAction="#refresh"
text="Refresh" />
          </items>
        </Menu>
        <Menu mnemonicParsing="false" text="Help">
          <items>
            <MenuItem mnemonicParsing="false" text="About" />
          </items>
        </Menu>
      </menus>
    </MenuBar>
  </children>
  <stylesheets>
    <URL value="@../../../../IssueTrackingLite.css" />
  </stylesheets>
</AnchorPane>
```

AddUser.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>
<?import java.net.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.layout.AnchorPane?>

<AnchorPane prefHeight="141.0" prefWidth="200.0" styleClass="theme"
xmlns:fx="http://javafx.com/fxml"
fx:controller="com.Shayan.FileSharing.Server.AddUserController">
  <!-- TODO Add Nodes -->
  <children>
    <VBox alignment="CENTER" prefHeight="141.0" spacing="5.0"
AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="14.0"
AnchorPane.rightAnchor="14.0" AnchorPane.topAnchor="0.0">
      <children>
        <TextField fx:id="AddUsernameTextfield" focusTraversable="false"
prefWidth="172.0" promptText="User Name" />
```

```
        <TextField fx:id="AddPasswordTextfield" focusTraversable="false"
prefWidth="172.0" promptText="Password" />
        <TextField fx:id="AddPassword2Textfield" focusTraversable="false"
prefWidth="172.0" promptText="Retype Password" />
        <Button fx:id="AddUserBtn" mnemonicParsing="false"
onMouseClicked="#addUser" text="Add" />
      </children>
    </VBox>
  </children>
  <stylesheets>
    <URL value="@../../../../IssueTrackingLite.css" />
  </stylesheets>
</AnchorPane>
```

DirectoryAdded.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>
<?import java.net.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.layout.AnchorPane?>

<AnchorPane prefHeight="70.0" prefWidth="150.0" styleClass="theme"
xmlns:fx="http://javafx.com/fxml"
fx:controller="com.Shayan.FileSharing.Server.FileSharingController">
  <!-- TODO Add Nodes -->
  <children>
    <Label layoutX="39.0" text="Folder shared" AnchorPane.topAnchor="14.0"
/>
    <Button fx:id="OkDirectoryAdded" layoutX="55.0" mnemonicParsing="false"
onMouseClicked="#closeDirectoryAddedDialog" text="OK"
AnchorPane.topAnchor="34.0" />
  </children>
  <stylesheets>
    <URL value="@../../../../IssueTrackingLite.css" />
  </stylesheets>
</AnchorPane>
```

FolderDeletedDialog.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>
<?import java.net.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.layout.AnchorPane?>
```

```
<AnchorPane prefHeight="70.0" prefWidth="200.0" styleClass="theme"
xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/2.2"
fx:controller="com.Shayan.FileSharing.Server.FolderDeletedController">
  <!-- TODO Add Nodes -->
  <children>
    <Label layoutX="23.0" text="Folder is not shared anymore"
AnchorPane.topAnchor="13.0" />
    <Button id="OkUserRemovedBtn" fx:id="Ok" layoutX="80.0"
mnemonicParsing="false" onMouseClicked="#CloseDialog" text="OK"
AnchorPane.topAnchor="34.0" />
  </children>
  <stylesheets>
    <URL value="@../../../../IssueTrackingLite.css" />
  </stylesheets>
</AnchorPane>
```

PasswordsDontMatch.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>
<?import java.net.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.layout.AnchorPane?>

<AnchorPane prefHeight="70.0" prefWidth="200.0" styleClass="theme"
xmlns:fx="http://javafx.com/fxml"
fx:controller="com.Shayan.FileSharing.Server.AddUserController">
  <!-- TODO Add Nodes -->
  <children>
    <Label layoutX="21.0" text="Error: Passwords Don't Match!"
AnchorPane.topAnchor="14.0" />
    <Button fx:id="OkPasswordsDontmatchBtn" layoutX="80.0"
mnemonicParsing="false" onMouseClicked="#closePasswordsDontMatchLayout"
text="OK" AnchorPane.topAnchor="34.0" />
  </children>
  <stylesheets>
    <URL value="@../../../../IssueTrackingLite.css" />
  </stylesheets>
</AnchorPane>
```

SetPort.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>
<?import java.net.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.layout.AnchorPane?>
```

```
<AnchorPane prefHeight="50.0" prefWidth="220.0" styleClass="theme"
xmlns:fx="http://javafx.com/fxml"
fx:controller="com.Shayan.FileSharing.Server.FileSharingController">
  <!-- TODO Add Nodes -->
  <children>
    <TextField fx:id="PortTextField" focusTraversable="false"
layoutX="14.0" layoutY="14.0" prefWidth="140.0"
promptText="1024&lt;Port&lt;65536" />
    <Button fx:id="OkPortSetedBtn" layoutX="165.0" layoutY="14.0"
mnemonicParsing="false" onMouseClicked="#setPort" text="OK" />
  </children>
  <stylesheets>
    <URL value="@../../../../IssueTrackingLite.css" />
  </stylesheets>
</AnchorPane>
```

UserAddedMessage.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>
<?import java.net.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.layout.AnchorPane?>

<AnchorPane prefHeight="70.0" prefWidth="200.0" styleClass="theme"
xmlns:fx="http://javafx.com/fxml"
fx:controller="com.Shayan.FileSharing.Server.AddUserController">
  <!-- TODO Add Nodes -->
  <children>
    <Label layoutX="37.0" text="User successfully added"
AnchorPane.topAnchor="14.0" />
    <Button fx:id="OkUserAddedBtn" layoutX="80.0" mnemonicParsing="false"
onMouseClicked="#CloseUserAddedDialog" text="OK"
AnchorPane.topAnchor="34.0" />
  </children>
  <stylesheets>
    <URL value="@../../../../IssueTrackingLite.css" />
  </stylesheets>
</AnchorPane>
```

UserDeleted.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>
<?import java.net.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.layout.AnchorPane?>
```

```
<AnchorPane prefHeight="70.0" prefWidth="200.0" styleClass="theme"
xmlns:fx="http://javafx.com/fxml"
fx:controller="com.Shayan.FileSharing.Server.UserDeletedController">
  <!-- TODO Add Nodes -->
  <children>
    <Label layoutX="70.0" text="User deleted." AnchorPane.topAnchor="13.0"
/>
    <Button fx:id="OkUserRemovedBtn" layoutX="80.0" mnemonicParsing="false"
onMouseClicked="#CloseUserDeletedDialog" text="OK"
AnchorPane.topAnchor="34.0" />
  </children>
  <stylesheets>
    <URL value="@../../../../IssueTrackingLite.css" />
  </stylesheets>
</AnchorPane>
```

UserExists.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>
<?import java.net.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.layout.AnchorPane?>

<AnchorPane prefHeight="70.0" prefWidth="200.0" styleClass="theme"
xmlns:fx="http://javafx.com/fxml"
fx:controller="com.Shayan.FileSharing.Server.AddUserController">
  <!-- TODO Add Nodes -->
  <children>
    <Label layoutX="70.0" text="User Exists!" AnchorPane.topAnchor="13.0"
/>
    <Button fx:id="OkUserExistsBtn" layoutX="80.0" mnemonicParsing="false"
onMouseClicked="#closeUserExistsLayout" text="OK"
AnchorPane.topAnchor="34.0" />
  </children>
  <stylesheets>
    <URL value="@../../../../IssueTrackingLite.css" />
  </stylesheets>
</AnchorPane>
```

IssueTrackingLite.css

```
.theme {
    master-color: black;
    -fx-background-color: derive(master-color, 70%);
    -fx-font-size: 14px;
}

.split-pane {
    -fx-padding: 0;
```

```
    -fx-border-width: 0;
    -fx-background-color: derive(master-color, 100%);
}

.button {
    -fx-background-color:
        linear-gradient(
            derive(master-color, 120%),
            derive(master-color, 90%)
        ),
        radial-gradient(
            center 50% -40%,
            radius 180%,
            derive(master-color, 95%) 55%,
            derive(master-color, 75%) 55%
        );
    -fx-background-radius: 4, 3;
    -fx-background-insets: 0, 1;
    -fx-effect: dropshadow( three-pass-box , rgba(0,0,0,0.55) , 5, 0.0 , 0
, 1 );
    -fx-text-fill: derive(master-color, -60%);
    -fx-padding: 3 11 3 11;
}

.darkList {
    -fx-background-color: derive(master-color, -60%);
}

.darkList .list-cell {
    -fx-padding: 6 0 0 13;
    -fx-background-color: derive(master-color, -60%);
    -fx-text-fill: lightgrey;
}

.darkList .list-cell:focused {
    -fx-text-fill: white;
}

.table-view {
    -fx-border-width: 0;
    -fx-border-color: red;
}

.text-area, .text-field {
    -fx-background-radius: 4;
    -fx-border-radius: 4;
}

.text-area {
    -fx-padding: 2;
}

.text-field {
    -fx-padding: 3 6 3 6;
}

.label {
    -fx-text-fill: derive(master-color, -20%);
    -fx-font-size: 12px;
}
```