

دانشگاه فني و حرفه‌اي

دانشكده فني دکتر شريعتي

اپلیکیشن بهبود سبک زندگی و برنامه­­­‌ریزی با استفاده از بازی‌انگاری برای سیستم عامل اندروید

پايان‌نامه براي دريافت درجه کارشناسي

در رشته مهندسي کامپيوتر گرايش نرم‌افزار

نرگس فرازان، زهرا نجمی

استاد راهنما:

دکتر ایمان شریفی

نيم‌سال اول 1397



دانشگاه فني حرفه‌اي

دانشكده فني شريعتي

اپلیکیشن بهبود سبک زندگی و برنامه­­­‌ریزی با استفاده از بازی‌انگاری برای سیستم عامل اندروید

پايان‌نامه براي دريافت درجه کارشناسي

در رشته مهندسي کامپيوتر گرايش نرم‌افزار

نرگس فرازان، زهرا نجمی

استاد راهنما:

دكتر ایمان شریفی

نيم‌سال اول 1397



تقديم به:

خانواده­های عزیزمان که در مسیر تحصیل لحظه­ای از کمک به ما دریغ نکردند.

تشكر و قدرداني:

با تشکر از اساتید دوره­ی کارشناسی، استاد راهنما آقای دکتر ایمان شریفی، مدیر گروه دلسوز خانم مهندس خشه­چی و هم­کلاسی­های مهربانمان.

چکيده

این پروژه، برنامه‌ای تحت سیستم عامل اندروید است که در آن سعی شده با استفاده از مفاهیم gamification (بازی‌انگاری) کاربر را به انجام فعالیت‌های مفید ترغیب کرد. از آنجایی که انجام فعالیت‌ها در قالب بازی و به صورت چالشی برای افراد جذاب‌تر می‌شود، از روش‌های gamification استفاده شده است. در این برنامه کاربر می‌تواند از بین انتخاب‌های موجود در برنامه یا مواردی که خودش تعریف می‌کند، فعالیت‌های مفید و غیرمفید (مضر) را انتخاب و میزان تاثیر آن‌ها را در مهارت‌هایش مشخص کند. با انجام هر فعالیت کاربر XP و HP و SP دریافت می‌کند که در بازه‌های مشخصی، ارتقای سطح1 پیدا می‌کند. همچنین با استفاده از المان‌های گرافیکی همچون نموادر سعی می‌شود تا کاربر میزان پیشرفت خود را مشاهده کند.

‌

واژه‌هاي كليدي:بازی­انگاری، سبک زندگی، اندروید، اپلیکیشن موبایل، برنامه­ریزی

فهرست مطالب

[فصل 1: مقدمه 1](#_Toc519244847)

[1-1- انگيزه 2](#_Toc519244848)

[1-2- هدف 2](#_Toc519244849)

[1-3- معرفی ابزارها و بستر برنامه نویسی 3](#_Toc519244850)

[1-3-1- معرفی اجمالی اندروید استودیو و زبان برنامه‌نویسی جاوا 3](#_Toc519244851)

[1-4- نگاهی به فصل‌های آینده 4](#_Toc519244852)

[فصل 2: تجزيه و تحليل نيازمندي‌ها 5](#_Toc519244853)

[2-1- مقدمه 6](#_Toc519244854)

[2-2- معرفی کاربران برنامه 6](#_Toc519244855)

[2-3- رابط گرافیکی نرم‌افزار و معرفی صفحات نرم‌افزار 6](#_Toc519244856)

[2-3-1- صفحه‌‌‌‌‌‌ی اصلی 7](#_Toc519244857)

[2-3-2- صفحه‌ی نمایش اطلاعات کاربر 8](#_Toc519244858)

[2-3-3- صفحه‌ی مهارت‌ها 9](#_Toc519244859)

[2-3-4- صفحه‌ی ویرایش مهارت‌‌ها 10](#_Toc519244860)

[2-3-5- صفحه‌ی فعالیت‌ها 11](#_Toc519244861)

[2-3-6- صفحه ویرایش فعالیت 14](#_Toc519244862)

[فصل 3: ساختار داده‌ها و بانک اطلاعات 16](#_Toc519244863)

[3-1- مقدمه 17](#_Toc519244864)

[3-2- نمودارهای UML 17](#_Toc519244865)

[3-2-1- نمودار Use Case 18](#_Toc519244866)

[3-2-2- نمودار توالی 19](#_Toc519244867)

[3-3- سخت‌افزارهای مورد نیاز 24](#_Toc519244868)

[فصل 4: جدول‌ها و بانک اطلاعاتی 25](#_Toc519244869)

[4-1- مقدمه 26](#_Toc519244870)

[4-2- پایگاه‌ داده 26](#_Toc519244871)

[4-3- نمودار ERD 27](#_Toc519244872)

[4-4- جداول پایگاه داده 28](#_Toc519244873)

[فصل 5: جمع‌‌بندي و پيشنهاد‌ها 31](#_Toc519244874)

[5-1- نتيجه‌گيري 32](#_Toc519244875)

[5-2- پيشنهادهايي براي کارهاي آتي 32](#_Toc519244876)

[مراجع 33](#_Toc519244877)

[پيوست‌ها 35](#_Toc519244878)

**فهرست شکل‌ها**

[شکل (2-1) صفحه اصلی برنامه 7](#_Toc519244879)

[شکل (2-2) صفحه ی نمایش اطلاعات کاربر 8](#_Toc519244880)

[شکل (2-3) صفحه مهارت ها 9](#_Toc519244881)

[شکل (2-4) صفحه ویرایش مهارت ها 10](#_Toc519244882)

[شکل (2-5) انتخاب عکس جدید برای مهارت 11](#_Toc519244883)

[شکل (2-6) صفحه فعالیت ها 12](#_Toc519244884)

[شکل (2-7) نمودار میزان انجام فعالیت 13](#_Toc519244885)

[شکل (2-8) دیالوگ انتخاب میزان ساعات انجام فعالیت 13](#_Toc519244886)

[شکل (2-9) دیالوگ میزان پیشرفت کاربر 14](#_Toc519244887)

[شکل (2-10) صفحه ویرایش فعالیت 15](#_Toc519244888)

[شکل (3-1) نمودار Use Case 18](#_Toc519244889)

[شکل (3-2) نمودارهای توالی View Quests و View Plan 19](#_Toc519244890)

[شکل (3-3) نمودارهای توالی View Faaliats و View Skills 20](#_Toc519244891)

[شکل (3-4) نمودارهای توالی Edit Faaliat و Add new Faaliat 21](#_Toc519244892)

[شکل (3-5) نمودارهای توالی Delete Faaliat و View Faaliat Graph 22](#_Toc519244893)

[شکل (3-6) نمودارهای توالی Edit Skill و Add New Skill 23](#_Toc519244894)

[شکل (3-7) نمودار توالی Delete Skill 24](#_Toc519244895)

[شکل (4-1) نمودار ER 27](#_Toc519244896)

فهرست جدول­ها

[جدول (4-1) جدول User 28](#_Toc519244897)

[جدول (4-2) جدول Skill 28](#_Toc519244898)

[جدول (4-3) جدول Faaliat 29](#_Toc519244899)

[جدول (4-4) جدول Quest 29](#_Toc519244900)

[جدول (4-5) جدول PlanCell 29](#_Toc519244901)

[جدول (4-6) جدول Quest\_Skill 29](#_Toc519244902)

[جدول (4-7) جدول Faaliat\_Skill 30](#_Toc519244903)

[جدول (4-8) جدول FaaliatRepetitions 30](#_Toc519244904)

[جدول (4-9) جدول Avatar 30](#_Toc519244905)

[جدول (4-10) جدول AvatarItem 30](#_Toc519244906)

1. مقدمه
   1. انگيزه

زندگی امروزه­ی بشر با پیشرفت تکنولوژی و ماشینی شدن تفاوت بسیاری با زندگی نسل­های گذشته پیدا کرده است؛ ساعات کاری زیاد، ترافیک، مشغله­های روزمره و مسائل اجتماعی جدید و مختلف، باعث شده است که انسان امروزی از پرداختن به علایق و فعالیت­های مورد­علاقه و چه بسا ضروری خود جا بماند. در این جا است که مسئله­­ی تصحیح و بهبود سبک زندگی و برنامه‌ریزی برای انجام فعالیت‌های مختلف، مسئله­ای ضروری و مهم به نظر می­رسد. همچنین امروزه مسئله بازی‌انگاری کارهای روزمره و یا جدی‌تر مانند فعالیت های شغلی، بسیار مورد توجه قرار گرفته است. به این معنا که فرد برای کسب امتیاز بیشتر و جمع کردن دست‌آوردهای مختلف، این نوع فعالیت‌ها را با رغبت بیشتری انجام می‌دهد. بنابراین تصمیم گرفته شد که با ترکیب این دو مقوله یعنی بهبود سبک زندگی و بازی‌انگاری برنامه‌ای ساخته شود تا افراد بتوانند با جذابیت بیشتری زندگی خود را بهبود ببخشند. نمونه­های مشابهی در فروشگاه­های نرم­افزاری موبایل وجود دارند اما این نمونه ها ضعف­هایی دارند که واضح­ترین آن­ها رابط کاربری غیردوستانه و غیر جذاب است.

* 1. هدف

در این پایان­نامه قصد بر این است که مراحل طراحی و تولید یک نرم­افزار تحت سیستم عامل اندروید با موضوع بهبود سبک زندگی و به صورت بازی‌انگاری را نشان دهیم. به این صورت که فرد یک سری "مهارت"ها و "فعالیت"هایی برای خود تعریف می‌کند که با انتساب آن مهارت‌ها به فعالیت‌هایش و انجام هر کدام از آن‌ها، در آن مهارت‌ها پیشرفت می‌کند و با این پیشرفت‌ها، کاربر ارتقای سطح می‌یابد و برای جایزه، سکه دریافت می‌کند. این سکه‌ها را می‌تواند در قسمت فروشگاه آدمک (avatar) برای تغییر شکل آدمک خود استفاده کند.

* 1. معرفی ابزارها و بستر برنامه ­نویسی

برای توسعه این برنامه از زبان برنامه‌نویسی جاوا(Java) و محیط برنامه‌نویسی اندروید استودیو (Android Studio) استفاده شده است که در ادامه به معرفی هر یک به طور اجمالی می‌پردازیم.

* + 1. معرفی اجمالی اندروید استودیو و زبان برنامه‌نویسی جاوا

نرم افزار اندروید استودیو، یک محیط توسعه یکپارچه رسمی برای توسعه پلتفرم و برنامه‌نویسی اندروید است. این نرم افزار در ۱۶ می سال ۲۰۱۳ در کنفرانس Google I/O معرفی شد. اندروید استودیو تحت لیسانس Apache License 2.0 به صورت رایگان در دسترس قرار دارد.

نسخه 0.1 اندروید استودیو در می سال ۲۰۱۳ به صورت پیش­نمایش در دسترس قرار گرفت، سپس در ژوئن سال ۲۰۱۴ با شروع ورژن ۰.۸ وارد مرحله بتا شد. ورژن 1.0 آن که اولین نسخه ثابت آن بود در دسامبر سال ۲۰۱۴ عرضه شد. این برنامه براساس نرم‌افزار IntelliJ IDEA از شرکت JetBrains، مخصوص توسعه اندروید طراحی شده است. لینک دانلود آن برای ویندوز، Mac OS X و لینوکس قرار دارد و به عنوان IDE (محیط توسعه نرم‌افزار یکپارچه) اصلی گوگل برای توسعه برنامه های اندروید، جایگزین Eclipse Android Development Tools شده است.

اندروید استودیو از build مبتنی بر gradle پشتیبانی می­کند. بازنویسی کد و اصلاحات فوری یکی از ویژگی­های خوب این IDE شمرده می­شود.اندروید استودیو ابزارهای Lint که برای رفع مشکلات عملکرد، کارایی، سازگاری ورژن‌ها و مشکلات دیگر است را در خود جای داده است.

اندروید استودیو یک ویرایشگر رابط (Layout Editor) غنی دارد که به کاربران اجازه می‌دهد مولفه‌های محیط کاربری (UI) را گرفته و رها (Drag and Drop) کنند و گزینه ای برای پیش نمایش رابط‌ها(Layout) در چندین پیکربندی صفحه نمایش وجود دارد.

زبان برنامه‌نویسی مورد استفاده در برنامه اندروید استودیو برای توسعه برنامه تحت سیستم عامل اندروید، زبان جاوا می‌باشد. جاوا یک زبان شی‌گرا است که شبیه به C++ می‌باشد اما [مدل شی‌گرایی](https://fa.wikipedia.org/wiki/%D9%85%D8%AF%D9%84_%D8%B4%DB%8C%D8%A1%DA%AF%D8%B1%D8%A7%DB%8C%DB%8C) آسان‌تری دارد و از [قابلیت‌های سطح پایین](https://fa.wikipedia.org/wiki/%D8%B2%D8%A8%D8%A7%D9%86_%D8%A8%D8%B1%D9%86%D8%A7%D9%85%D9%87%E2%80%8C%D9%86%D9%88%DB%8C%D8%B3%DB%8C_%D8%B3%D8%B7%D8%AD_%D9%BE%D8%A7%DB%8C%DB%8C%D9%86) کمتری پشتیبانی می‌کند.

* 1. نگاهی به فصل‌های آینده

در فصل های آینده با مفاهیم کلی برنامه و نحوه پیاده سازی و ابزارهای مورد استفاده برای پیاده‌سازی آن آشنا می‌شویم؛ همچنین نمودارهای مدل‌سازی UML برای درک بهتر ساختار برنامه آورده شده‌اند.

1. تجزيه و تحليل نيازمندي‌ها
   1. مقدمه

هدف از اين فصل تحليل نيازمندي­هاي سيستم تحت طراحي است. سناريوها، نمودارهاي تحليلي مورد کاربرد، کاربرد، توالي و غيره در این فصل شرح داده شده­اند.

* 1. معرفی کاربران برنامه

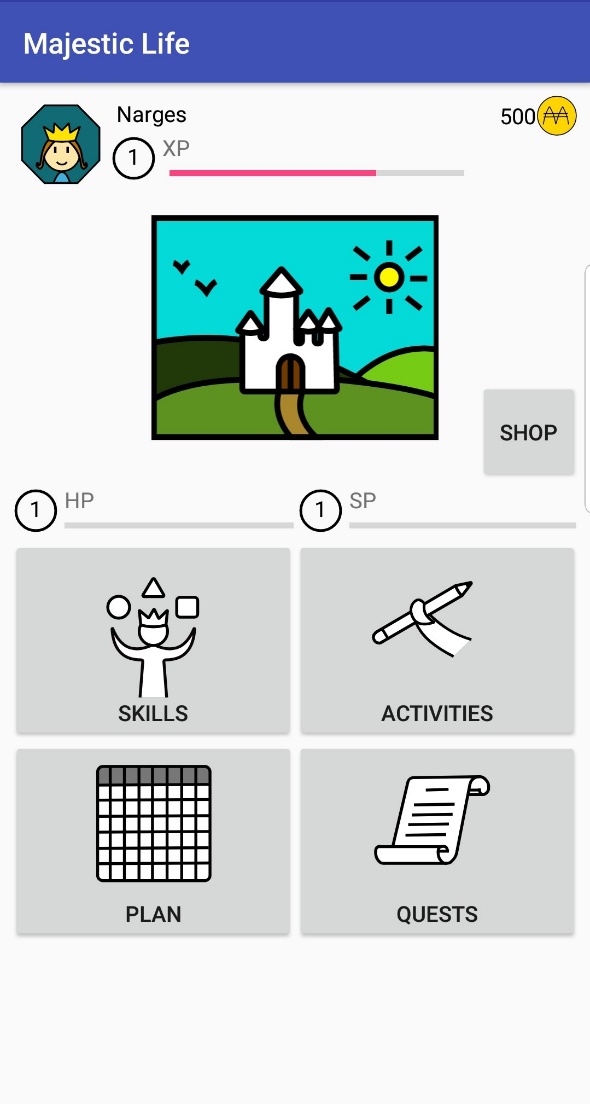
کاربران این برنامه هر طیف و سنی از جامعه می‌توانند باشند و محدودیتی برای بازه­ی کاربران این نرم­افزار وجود ندارد. هر شخصی که به دنبال برنامه‌ریزی فعالیت‌های روزانه خود است می‌تواند از این برنامه استفاده کند.

* 1. رابط گرافیکی نرم‌افزار و معرفی صفحات نرم‌افزار

این نرم­افزار از یک صفحه­ی اصلی و چند صفحه فرعی تشکیل شده است که دسترسی به آن‌ها از طریق صفحه اصلی ممکن است.

* + 1. صفحه‌‌‌‌‌‌ی اصلی

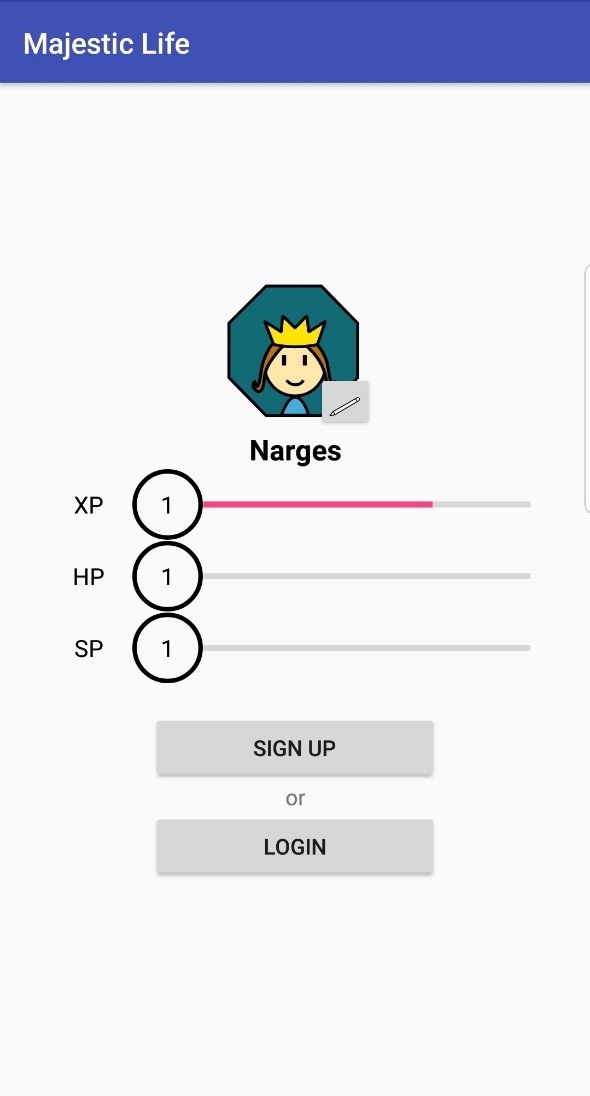
این صفحه، صفحه اصلی و اولیه برنامه می‌باشد. در اینجا، کاربر 4 دکمه اصلی برنامه که به صفحات "فعالیت‌ها"، "مهارت‌ها"، "برنامه" و "ماموریت‌ها" مسیریابی می‌شوند، مشاهده می‌کند. علاوه بر آن، progress barهایی نیز در این صفحه برای نمایش مقدار و سطح XP، HP و SP کاربر وجود دارند. در بالا سمت راست صفحه نیز، تعداد سکه‌ها و در سمت چپ صفحه هم آدمک کاربر قرار دارد که با زدن روی تصویر آدمک به صفحه ورود (login) مسیریابی می‌شود.



صفحه اصلی برنامه

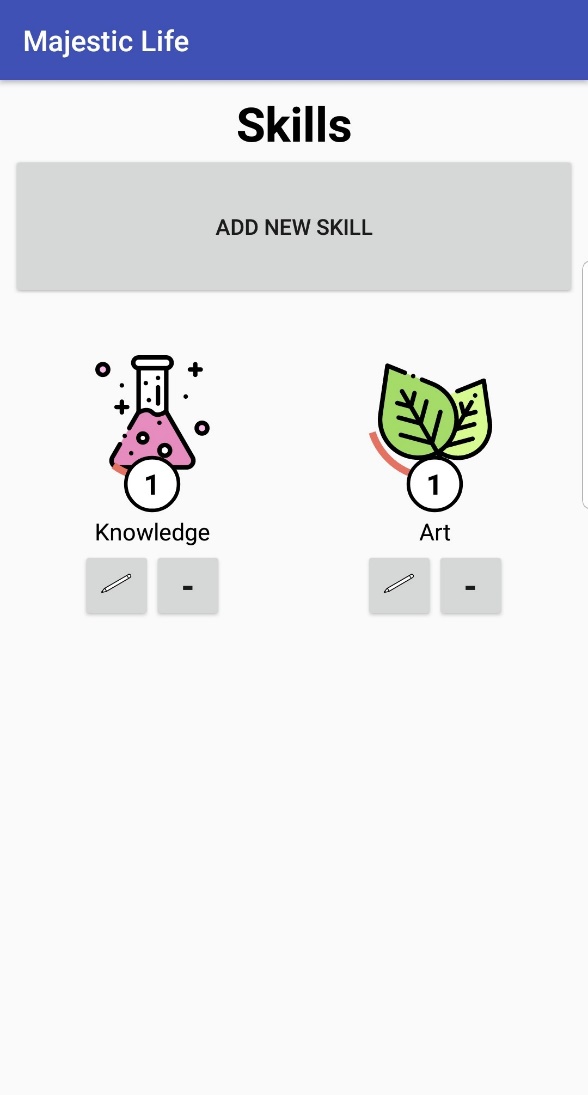
* + 1. صفحه‌ی نمایش اطلاعات کاربر

در این صفحه کاربر می­تواند Sign Up یا Login انجام دهد. هم­چنین کاربر می‌تواند آدمک خود و میزان پیشرفتش در هریک از XP، HP و SP را مشاهده کند. با زدن روی دکمه کنار آدمک، دکمه ویرایش، کاربر به صفحه فروشگاه آدمک منتقل می‌شود.



صفحه ی نمایش اطلاعات کاربر

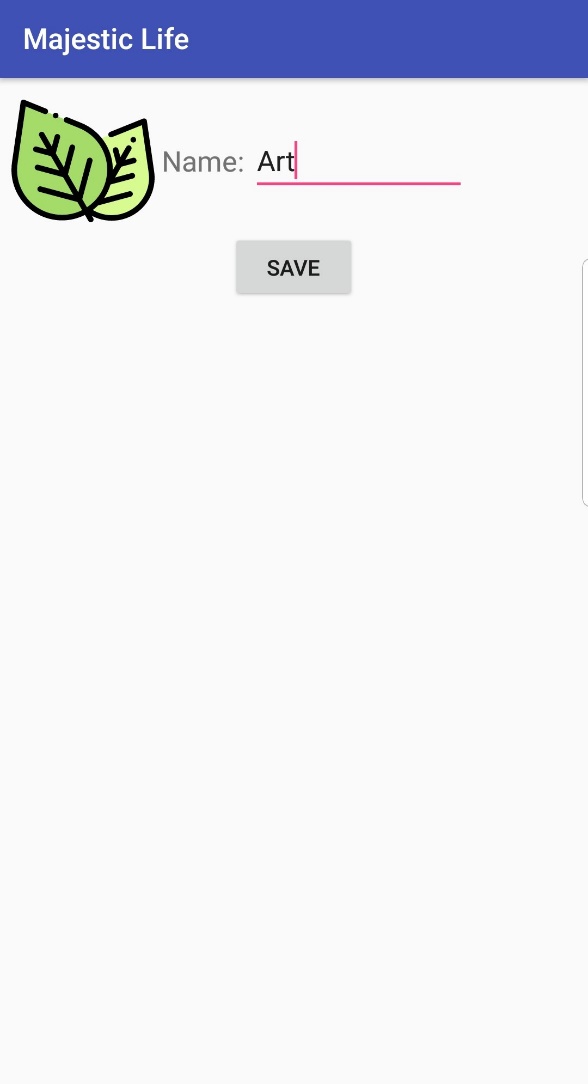
* + 1. صفحه‌ی مهارت‌­ها

در این صفحه، کاربر لیستی از مهارت­های خود را مشاهده می­کند که برای هر کدام از آن‌ها نیز، دکمه‌های حذف و ویرایش آن مهارت قرار دارد. در بالای صفحه دکمه­ی اضافه کردن مهارت جدید(Add New Skill) وجود دارد که کاربر با کلیک بر روی آن، یک مهارت جدید به مهارت‌های خود افزورده و به صفحه­­ی ویرایش مهارت جدید می­رود.

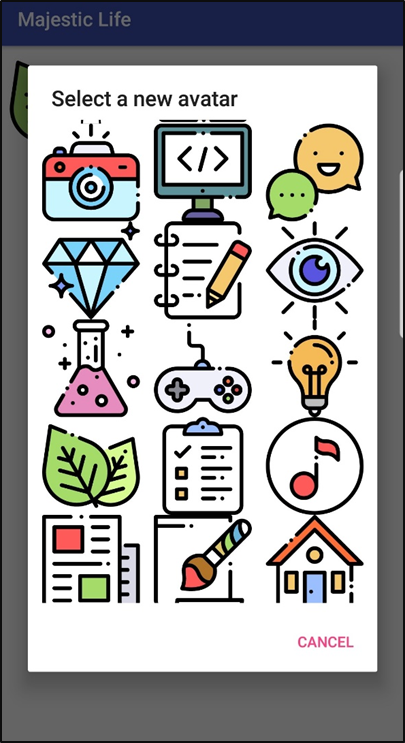
صفحه مهارت ها

* + 1. صفحه‌ی ویرایش مهارت‌‌ها

در این صفحه در قسمت Name، کاربر نام مهارت خود را وارد می‌کند؛ همچنین کاربر می‌تواند با زدن روی تصویر مهارت، تصویر جدیدی را از میان لیستی از تصاویر که در Dialogای قرار دارند انتخاب کند. با زدن روی دکمه ذخیره (Save) نیز تغییرات خود را اعمال کرده و به صفحه مهارت‎ها باز می‌گردد.



صفحه ویرایش مهارت ها

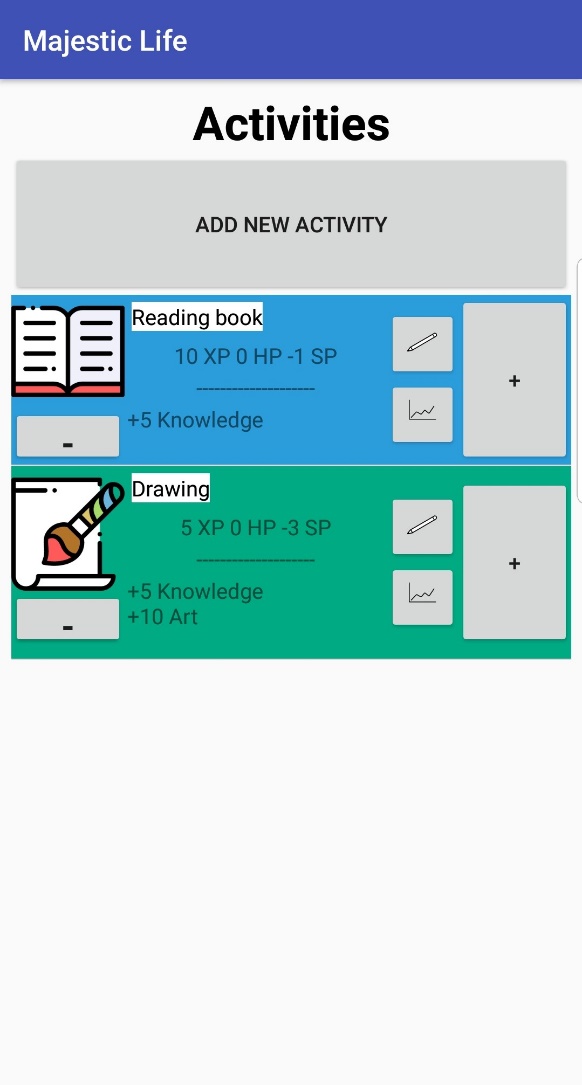


انتخاب عکس جدید برای مهارت

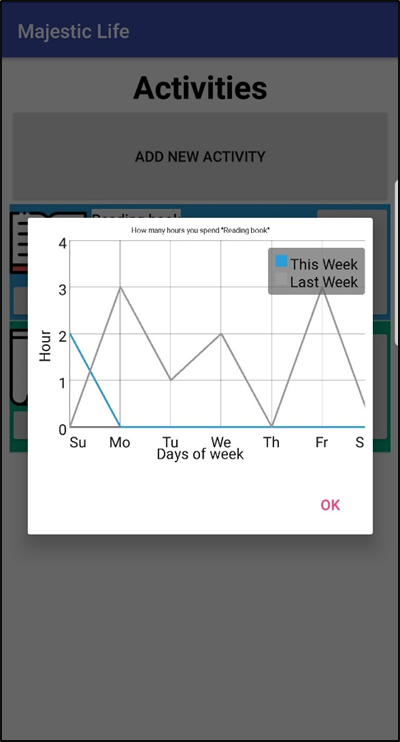
* + 1. صفحه‌ی فعالیت‌­ها

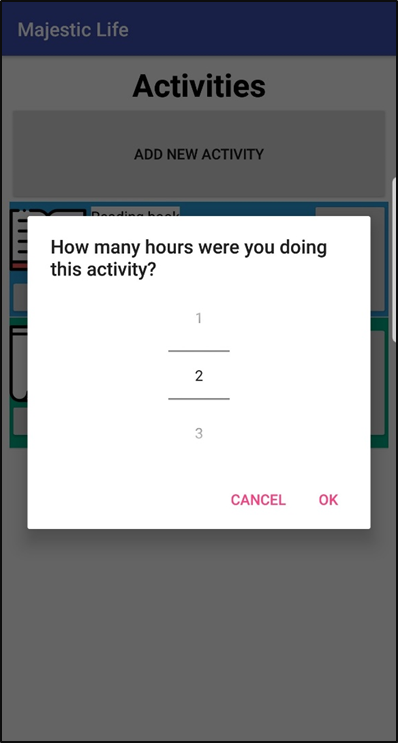
در این صفحه، کاربر لیستی از فعالیت­های خود را مشاهده می­کند که برای هر کدام از آن‌ها نیز دکمه‌های حذف، ویرایش، نمودار و پیشرفت(+) آن فعالیت قرار دارد و هر کدام رنگ و تصویر خاص خود را دارند. در بالای صفحه دکمه­ی اضافه کردن فعالیت جدید(Add New Activity) وجود دارد که کاربر با کلیک بر روی آن، یک فعالیت جدید به فعالیت‌های خود افزورده و به صفحه­­ی ویرایش فعالیت جدید می­رود.

با زدن روی دکمه نمودار هر فعالیت، نموداری از میزان انجام همان فعالیت در روزهای مختلف را می‌توان دید. همچنین با زدن بر روی دکمه پیشرفت (+)، Dialogای برای گرفتن میزان ساعاتی که کاربر مشغول به انجام آن فعالیت بوده است باز می‌شود. بعد از مشخص کردن تعداد ساعات و زدن دکمه OK، دوباره Dialogای باز می‌شود و میزان پیشرفت کاربر را در هر کدام از مشخصه ها و مهارت‌های مربوطه نشان داده می‌شود.

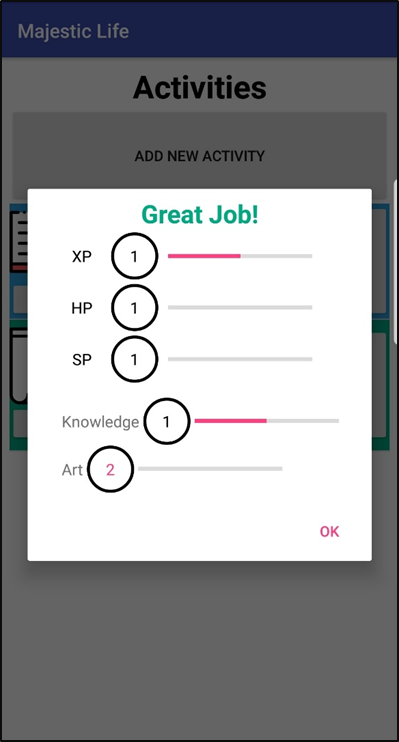


صفحه فعالیت ها

نمودار میزان انجام فعالیت



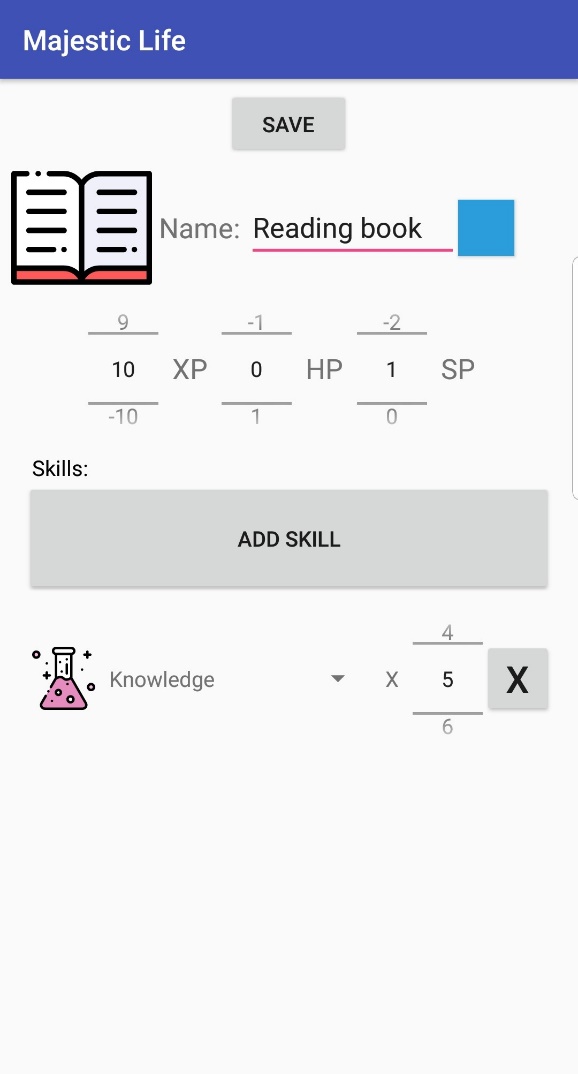
دیالوگ انتخاب میزان ساعات انجام فعالیت

دیالوگ میزان پیشرفت کاربر

* + 1. صفحه ویرایش فعالیت

در این صفحه در قسمت Name، کاربر نام فعالیت خود را وارد می‌کند؛ همچنین کاربر می‌تواند با زدن روی تصویر فعالیت، تصویر جدیدی را از میان لیستی از تصاویر که در Dialogای قرار دارند انتخاب کند. با زدن روی دکمه ذخیره (Save) نیز تغییرات خود را اعمال کرده و به صفحه فعالیت‎ها باز می‌گردد.

هم­چنین در اینجا، کاربر می‌تواند با زدن روی مربع رنگی، رنگ مورد نظر خود را برای این فعالیت انتخاب کند. در پایین قسمت انتخاب نام، می­توان مقدار XP، HP و SPای را نیز که با انجام این فعالیت به دست می‌آورد، مشخص­کند. در قسمت پایینی صفحه، لیستی از مهارت‌هایی قرار دارند که می‌توان به آن ها مهارت جدید افزود و یا مهارت‌های موجود را حذف کرد. در مقابل هر مهارت نیز، میزان پیشرفت آن مهارت با هر بار انجام این فعالیت را نیز می‌توان مشخص کرد.



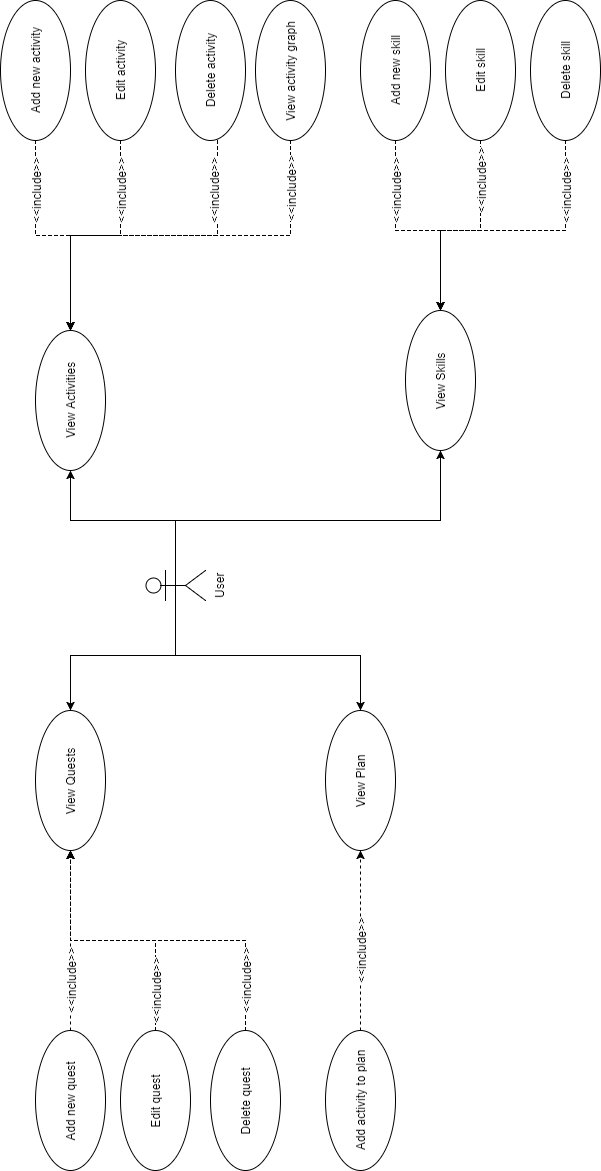
صفحه ویرایش فعالیت

1. ساختار داده‌ها و بانک اطلاعات
   1. مقدمه

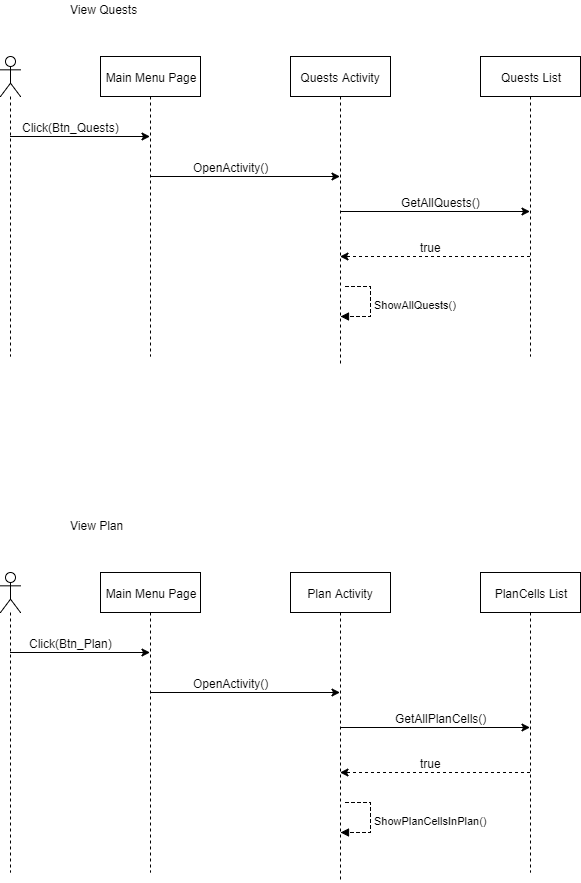
در این فصل نمودارهای زبان مدل‌سازی مورد استفاده در این پروژه، یعنی UML آورده شده است. این نمودارها با استفاده از نرم‌افزار Draw.io که یک نرم‌افزار رایگان برای رسم نمودارهای UML است، کشیده شده‌اند.

* 1. نمودار­های UML

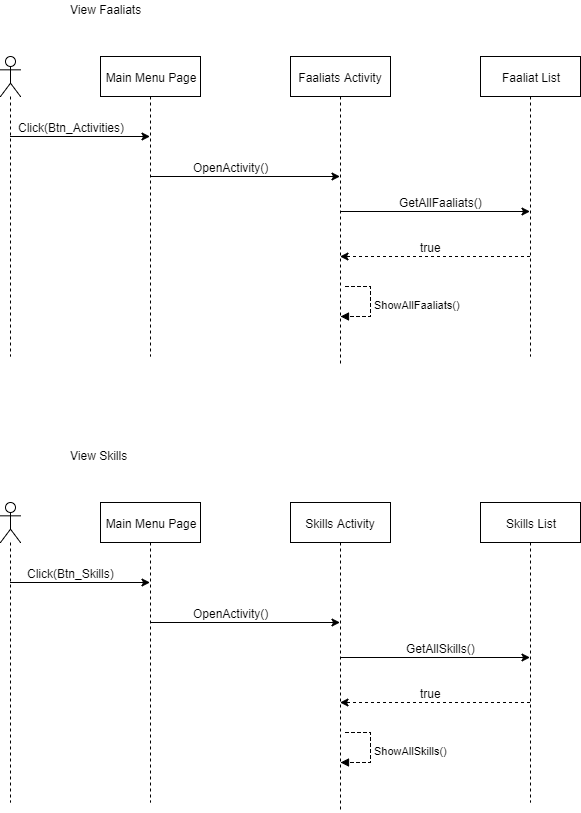
در این بخش نمودارهای Usecase (مورد کاربرد) و نمودارهای توالی مربوط به برنامه رسم شده­اند.

* + 1. نمودار Use Case

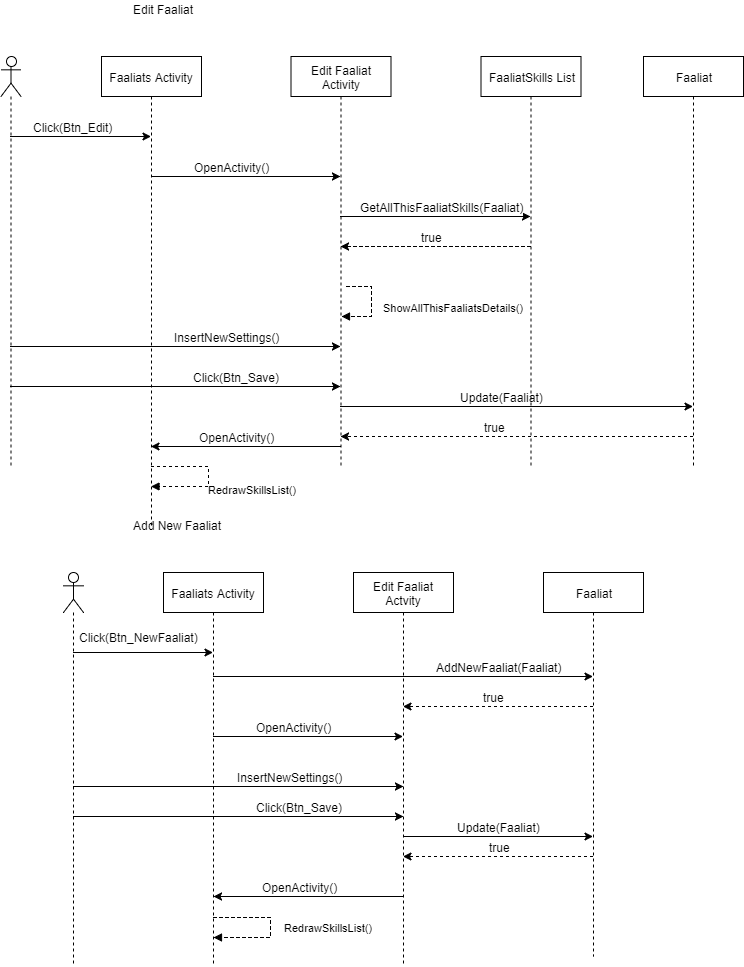
نمودار Use Case

* + 1. نمودار توالی

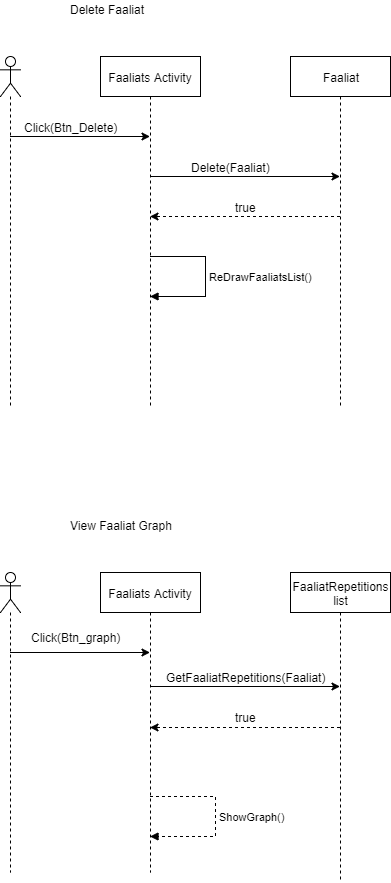
نمودارهای توالی View Quests و View Plan

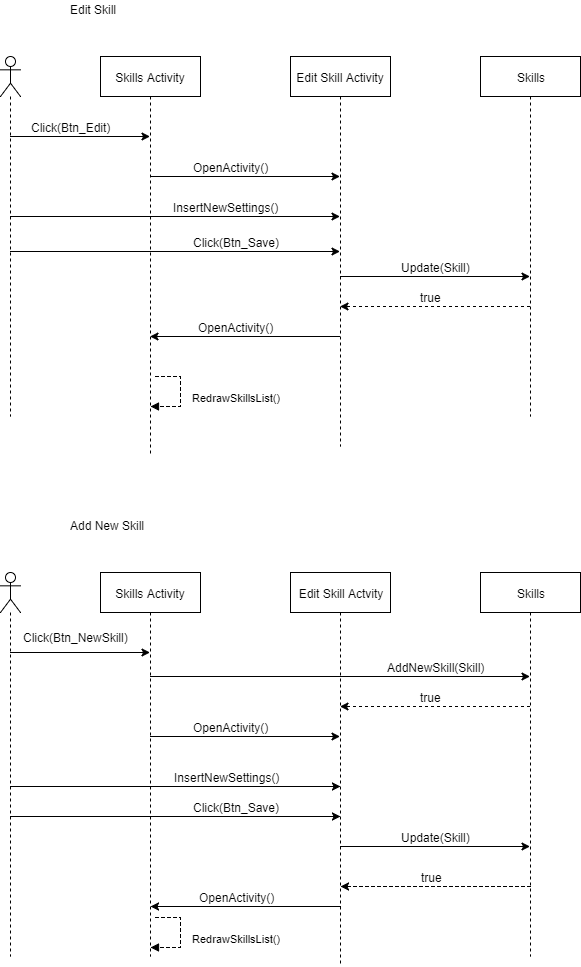


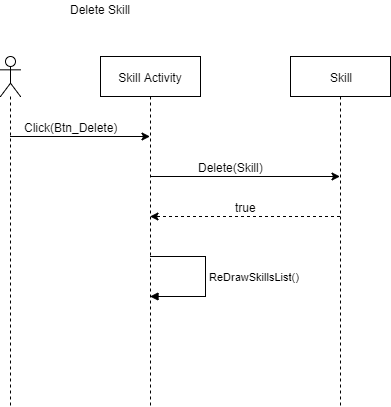
نمودارهای توالی View Faaliats و View Skills



نمودارهای توالی Edit Faaliat و Add new Faaliat

نمودارهای توالی Delete Faaliat و View Faaliat Graph

نمودارهای توالی Edit Skill و Add New Skill

نمودار توالی Delete Skill

* 1. سخت‌­افزار­های مورد نیاز

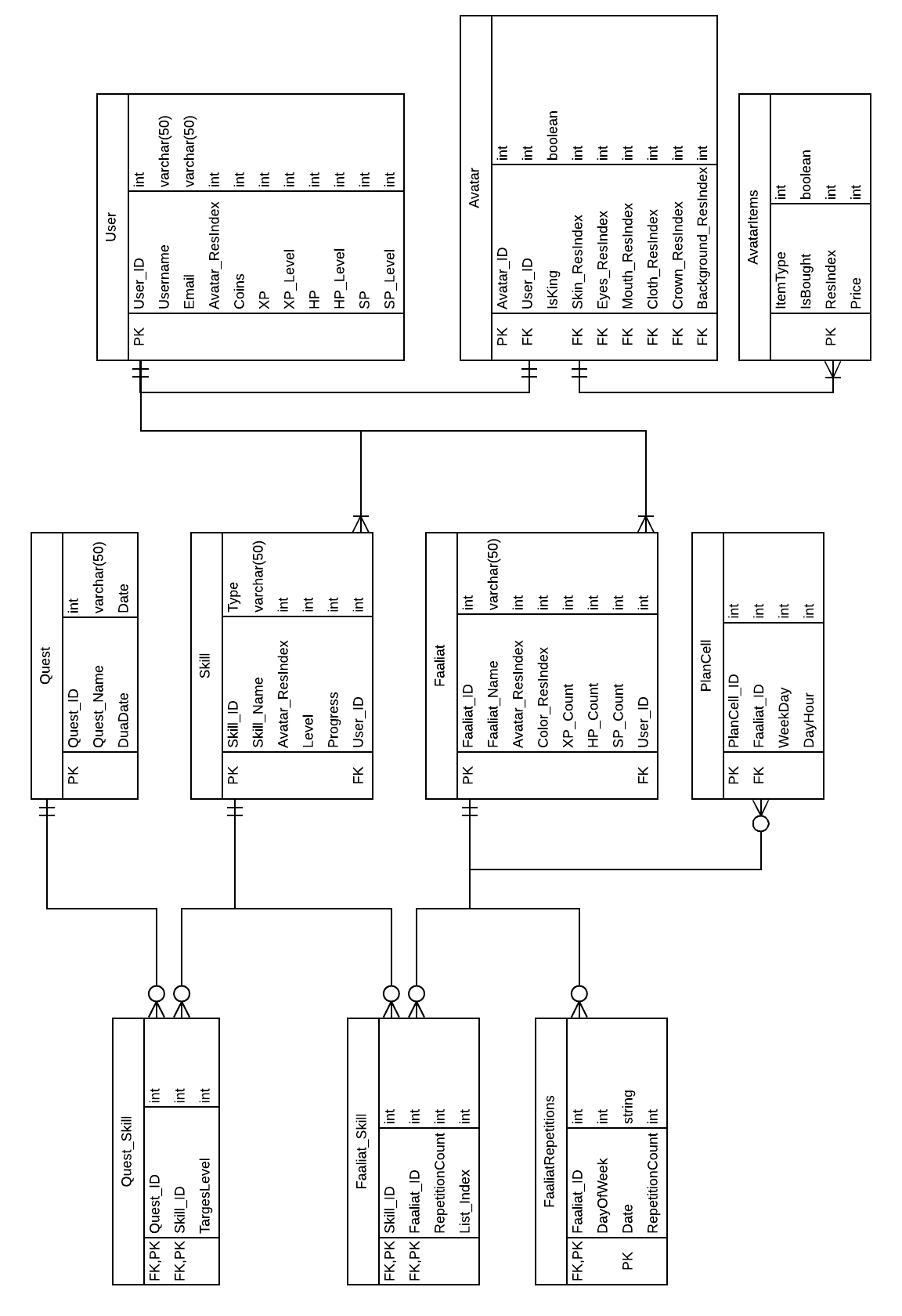
برای اجرای این برنامه، به یک دستگاه با سیستم عامل اندروید با حداقل SDK Version 14 نیاز است. برای ایجاد حساب کاربری نیز، دسترسی به اینترنت لازم است.

1. جدول‌ها و بانک اطلاعاتی
   1. مقدمه

در این فصل، نمودار روابط موجودیت‌های پایگاه داده (Entity Relationship Diagram) آمده است. همچنین توضیحی مختصری در مورد نحوه پیاده‌سازی این پایگاه داده‌ی محلی با استفاده از کتابخانه Room Persistence خود اندورید، بیان شده است.

* 1. پایگاه‌ داده

این اپلیکیشن به زبان جاوا و در محیط Android Studio 3 نوشته شده است. پیگاه داده‌ی محلی این برنامه با استفاده از Room Persistence Library نوشته شده است. کتابخانه‌ی Room Persistence در اصل یک لایه انتزاعی بر روی SQLite موجود در اندروید است که دسترسی به پایگاه‌داده را روان‌تر کرده و در عین حال از تمام توان SQLite استفاده می‌کند. این کتابخانه اطلاعات پایگاه داده برنامه را در دستگاه کاربر، cache می‌کند؛ همچنین باعث درگیری کمتر برنامه‌نویس با نخ‌ها (thread) برای ذخیره و بازیابی اطلاعات می‌شود. این کتابخانه این قابلیت را هم به برنامه‌نویس اندرویدی می‌دهد که در هنگام compile برنامه، اشکالات موجود در پرس و جوهایی (query) که نوشته است را مشاهده کند.

* 1. نمودار ERD

نمودار ER

* 1. جداول پایگاه داده

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ردیف** | **نام** | **ویژگی** | | | |
|  |  | کلید اصلی | نوع | طول | واحد |
| 1 | User\_ID | PK | int | 2 | بایت |
| 2 | Username | - | string | 8 | بایت |
| 3 | Email | - | string | 8 | بایت |
| 4 | Coins | - | int | 2 | بایت |
| 5 | XP | - | int | 2 | بایت |
| 6 | XP\_Level | - | int | 2 | بایت |
| 7 | HP | - | int | 2 | بایت |
| 8 | HP\_Level | - | int | 2 | بایت |
| 9 | SP | - | int | 2 | بایت |
| 10 | SP\_Level | - | int | 2 | بایت |

جدول User

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ردیف** | **نام** | **ویژگی** | | | |
|  |  | کلید اصلی | نوع | طول | واحد |
| 1 | Skill\_ID | PK | int | 2 | بایت |
| 2 | Skill\_Name | - | String | 8 | بایت |
| 3 | Avatar\_ResIndex | - | int | 2 | بایت |
| 4 | Level | - | int | 2 | بایت |
| 5 | Progress | - | int | 2 | بایت |
| 6 | User\_ID | FK | int | 2 | بایت |

جدول Skill

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ردیف** | **نام** | **ویژگی** | | | |
|  |  | کلید اصلی | نوع | طول | واحد |
| 1 | Faaliat\_ID | PK | int | 2 | بایت |
| 2 | Faaliat\_Name | - | String | 8 | بایت |
| 3 | Avatar\_ResIndex | - | int | 2 | بایت |
| 4 | Color\_ResIndex | - | int | 2 | بایت |
| 5 | XP\_Count | - | int | 2 | بایت |
| 6 | HP\_Count | - | int | 2 | بایت |
| 7 | SP\_Count | - | int | 2 | بایت |
| 8 | User\_ID | FK | int | 2 | بایت |

جدول Faaliat

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ردیف** | **نام** | **ویژگی** | | | |
|  |  | کلید اصلی | نوع | طول | واحد |
| 1 | Quest\_ID | PK | int | 2 | بایت |
| 2 | Quest\_Name | - | String | 8 | بایت |
| 3 | DuaDate | - | String | 8 | بایت |

جدول Quest

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ردیف | نام | ویژگی | | | |
|  |  | کلید اصلی | نوع | طول | واحد |
| 1 | PlanCell\_ID | PK | int | 2 | بایت |
| 2 | Faaliat\_ID | FK | int | 2 | بایت |
| 3 | WeekDay |  | int | 2 | بایت |
| 4 | DayHour |  | int | 2 | بایت |

جدول PlanCell

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ردیف** | **نام** | **ویژگی** | | | |
|  |  | کلید اصلی | نوع | طول | واحد |
| 1 | Quest\_ID | PK, FK | int | 2 | بایت |
| 2 | Skill\_ID | PK, FK | int | 2 | بایت |
| 3 | TargetLevel | - | int | 2 | بایت |

جدول Quest\_Skill

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ردیف** | **نام** | **ویژگی** | | | |
|  |  | کلید اصلی | نوع | طول | واحد |
| 1 | Skill\_ID | PK، FK | int | 2 | بایت |
| 2 | Faaliat\_ID | PK، FK | int | 2 | بایت |
| 3 | RepetitionCount | - | int | 2 | بایت |
| 4 | ListIndex | - | int | 2 | بایت |

جدول Faaliat\_Skill

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ردیف** | **نام** | **ویژگی** | | | |
|  |  | کلید اصلی | نوع | طول | واحد |
| 1 | Faaliat\_ID | PK, FK | int | 2 | بایت |
| 2 | DayOfWeek | - | int | 2 | بایت |
| 3 | Date | PK | String | 8 | بایت |
| 4 | RepetitionCount | - | int | 2 | بایت |

جدول FaaliatRepetitions

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ردیف** | **نام** | **ویژگی** | | | |
|  |  | کلید اصلی | نوع | طول | واحد |
| 1 | Avatar\_ID | PK | int | 2 | بایت |
| 2 | User\_ID | FK | int | 2 | بایت |
| 3 | IsKing | - | boolean | 1 | بیت |
| 4 | Skin\_ResIndex | FK | int | 2 | بایت |
| 5 | Eyes\_ResIndex | FK | int | 2 | بایت |
| 6 | Mouth\_ResIndex | FK | int | 2 | بایت |
| 7 | Cloth\_ResIndex | FK | int | 2 | بایت |
| 8 | Crown\_ResIndex | FK | int | 2 | بایت |
| 9 | Background\_ResIndex | FK | int | 2 | بایت |

جدول Avatar

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ردیف** | **نام** | **ویژگی** | | | |
|  |  | کلید اصلی | نوع | طول | واحد |
| 1 | ItemType | - | int | 2 | بایت |
| 2 | IsBought | - | boolean | 1 | بیت |
| 3 | ResIndex | PK | int | 2 | بایت |
| 4 | Price | - | int | 2 | بایت |

جدول AvatarItem

1. جمع‌‌بندي و پيشنهاد‌ها
   1. نتيجه‌گيري

در این پروژه دوره کارشناسی، در ابتدا با استفاده از مفاهیم اولیه‌ی بازی‌انگاری، ساختار کلی برنامه و سپس ساختار برنامه سازی آن طراحی شده است. چالش اصلی در این برنامه ذخیره درست اطلاعات به صورت محلی بود که با استفاده از Room Persistence Library این چالش برطرف شد.

* 1. پيشنهادهايي براي کارهاي آتي

در این پروژه بخش‌های Quests و Planner، به علت کمبود وقت تکمیل نشده‌اند؛ ولی با توجه به ساختار شی­گرای برنامه، می­توان به راحتی این قسمت‌ها را در آینده به برنامه اضافه کرد. همچنین می­توان بخش­های مربوط به فروشگاه و تصاویر گرافیکی برنامه را نیز بهبود بخشید.

به دلیل اینکه این برنامه با مفاهیم روانشناختی افراد نیز ارتباط نزدیکی می‌تواند داشته باشد، مشورت با متخصصین این حوزه برای بهبود هرچه بهتر برنامه امری ضروری به نظر رسیده و توصیه می‌شود.

مراجع

مراجع

1. <https://developer.android.com>
2. <https://codelabs.developers.google.com/codelabs/android-room-with-a-view/#0>
3. <https://android.jlelse.eu/android-architecture-components-room-introduction-4774dd72a1ae>
4. <https://medium.com/@srinuraop/database-create-and-open-callbacks-in-room-7ca98c3286ab>
5. <https://www.flaticon.com/packs/user-interface-30/2>
6. کتاب پایگاه داده ی رانکوهی
7. کتاب های مهندسی نرم افزار یک و دو پرس من

پيوست‌ها

پيوست الف: کدهای پروژه

کدهایی که در ادامه آمده اند، Entityها یا موجودیت‌های پایگاه داده هستند.

**package** com.farazannajmi.majesticlife.DataStructures;  
  
**import** android.arch.persistence.room.ColumnInfo;  
**import** android.arch.persistence.room.Entity;  
**import** android.arch.persistence.room.PrimaryKey;  
  
**import** com.farazannajmi.majesticlife.R;  
  
**import** javax.annotation.Nonnull;

@Entity(tableName = **"User\_table"**)  
**public class** User  
{  
 @PrimaryKey  
 @Nonnull  
 @ColumnInfo(name = **"User\_ID"**)  
 **private int User\_ID**;  
  
 @ColumnInfo(name = **"Username"**)  
 **private** String **Username**;  
  
 @ColumnInfo(name = **"Email"**)  
 **private** String **Email**;  
  
 @ColumnInfo(name = **"XP"**)  
 **private int XP**;  
 @ColumnInfo(name = **"XpLevel"**)  
 **private int XpLevel**;  
  
 @ColumnInfo(name = **"HP"**)  
 **private int HP**;  
 @ColumnInfo(name = **"HpLevel"**)  
 **private int HpLevel**;  
  
 @ColumnInfo(name = **"SP"**)  
 **private int SP**;  
 @ColumnInfo(name = **"SpLevel"**)  
 **private int SpLevel**;  
  
 @ColumnInfo(name = **"Coins"**)  
 **private int Coins**;  
  
 @Nonnull  
 **public int** getUser\_ID(){**return this**.**User\_ID**;}  
 **public** String getUsername(){**return this**.**Username**;}  
 **public** String getEmail(){**return this**.**Email**;}  
 **public int** getXP(){**return this**.**XP**;}  
 **public int** getXpLevel(){**return this**.**XpLevel**;}  
 **public int** getHP(){**return this**.**HP**;}  
 **public int** getHpLevel(){**return this**.**HpLevel**;}  
 **public int** getSP(){**return this**.**SP**;}  
 **public int** getSpLevel(){**return this**.**SpLevel**;}  
 **public int** getCoins(){**return this**.**Coins**;}  
  
 **public void** setUser\_ID(**int** User\_ID){**this**.**User\_ID** = User\_ID;}  
 **public void** setUsername(String Username){**this**.**Username** = Username;}  
 **public void** setEmail(String Email){**this**.**Email** = Email;}  
 **public void** setXP(**int** XP){**this**.**XP** = XP;}  
 **public void** setXpLevel(**int** XpLevel){**this**.**XpLevel** = XpLevel;}  
 **public void** setHP(**int** HP){**this**.**HP** = HP;}  
 **public void** setHpLevel(**int** HpLevel){**this**.**HpLevel** = HpLevel;}  
 **public void** setSP(**int** SP){**this**.**SP** = SP;}  
 **public void** setSpLevel(**int** SpLevel){**this**.**SpLevel** = SpLevel;}  
 **public void** setCoins(**int** Coins){**this**.**Coins** = Coins;}  
  
 **public** User(**int** User\_ID, String Username, String Email,  
 **int** XP, **int** XpLevel, **int** HP, **int** HpLevel, **int** SP, **int** SpLevel, **int** Coins)  
 {  
 **this**.**User\_ID** = User\_ID;  
 **this**.**Username** = Username;  
 **this**.**Email** = Email;  
 **this**.**XP** = XP;  
 **this**.**XpLevel** = XpLevel;  
 **this**.**HP** = HP;  
 **this**.**HpLevel** = HpLevel;  
 **this**.**SP** = SP;  
 **this**.**SpLevel** = SpLevel;  
 **this**.**Coins** = Coins;  
 }

**package** com.farazannajmi.majesticlife.DataStructures;  
  
**import** android.arch.persistence.room.ColumnInfo;  
**import** android.arch.persistence.room.Entity;  
**import** android.arch.persistence.room.ForeignKey;  
**import** android.arch.persistence.room.Ignore;  
**import** android.arch.persistence.room.PrimaryKey;  
  
**import** java.util.ArrayList;  
  
**import** javax.annotation.Nonnull;  
  
**import static** android.arch.persistence.room.ForeignKey.***CASCADE***;  
  
@Entity(tableName = **"Faaliat\_table"**,  
 foreignKeys = @ForeignKey(entity = User.**class**,  
 parentColumns = **"User\_ID"**,  
 childColumns = **"User\_ID"**,  
 onDelete = ***CASCADE***))  
**public class** Faaliat  
{  
 @PrimaryKey*//(autoGenerate = true)* @Nonnull  
 @ColumnInfo(name = **"Faaliat\_ID"**)  
 **private int Faaliat\_ID**;  
  
 @ColumnInfo(name = **"Faaliat\_Name"**)  
 **private** String **Faaliat\_Name**;  
  
 @ColumnInfo(name = **"Avatar\_ResIndex"**)  
 **private int Avatar\_ResIndex**;  
 @ColumnInfo(name = **"Color\_ResIndex"**)  
 **private int Color\_ResIndex**;  
  
 @ColumnInfo(name = **"XpCount"**)  
 **private int XpCount**;  
 @ColumnInfo(name = **"HpCount"**)  
 **private int HpCount**;  
 @ColumnInfo(name = **"SpCount"**)  
 **private int SpCount**;  
  
 @ColumnInfo(name = **"User\_ID"**)  
 **private int User\_ID**;  
  
 @Ignore  
 **public** ArrayList<FaaliatSkill> **FaaliatSkills**;  
  
 @Nonnull  
 **public int** getFaaliat\_ID() {**return Faaliat\_ID**;}  
 **public** String getFaaliat\_Name() {**return Faaliat\_Name**;}  
 **public int** getAvatar\_ResIndex() {**return Avatar\_ResIndex**;}  
 **public int** getColor\_ResIndex() {**return Color\_ResIndex**;}  
 **public int** getXpCount() {**return XpCount**;}  
 **public int** getHpCount() {**return HpCount**;}  
 **public int** getSpCount() {**return SpCount**;}  
 **public int** getUser\_ID() {**return User\_ID**;}  
  
 **public void** setFaaliat\_ID(**int** Faaliat\_ID){**this**.**Faaliat\_ID** = Faaliat\_ID;}  
 **public void** setFaaliat\_Name(String Faaliat\_Name){**this**.**Faaliat\_Name** = Faaliat\_Name;}  
 **public void** setAvatar\_ResIndex(**int** Avatar\_ResIndex){**this**.**Avatar\_ResIndex** = Avatar\_ResIndex;}  
 **public void** setColor\_ResIndex(**int** Color\_ResIndex){**this**.**Color\_ResIndex** = Color\_ResIndex;}  
 **public void** setXpCount(**int** XpCount){**this**.**XpCount** = XpCount;}  
 **public void** setHpCount(**int** HpCount){**this**.**HpCount** = HpCount;}  
 **public void** setSpCount(**int** SpCount){**this**.**SpCount** = SpCount;}  
 **public void** setUser\_ID(**int** User\_ID){**this**.**User\_ID** = User\_ID;}  
  
 **public** Faaliat(@Nonnull **int** Faaliat\_ID, String Faaliat\_Name, **int** Avatar\_ResIndex, **int** Color\_ResIndex,  
 **int** XpCount, **int** HpCount, **int** SpCount, **int** User\_ID)  
 {  
 **this**.**Faaliat\_ID** = Faaliat\_ID;  
 **this**.**Faaliat\_Name** = Faaliat\_Name;  
 **this**.**Avatar\_ResIndex** = Avatar\_ResIndex;  
 **this**.**Color\_ResIndex** = Color\_ResIndex;  
 **this**.**XpCount** = XpCount;  
 **this**.**HpCount** = HpCount;  
 **this**.**SpCount** = SpCount;  
 **this**.**User\_ID** = User\_ID;  
 }  
}

**package** com.farazannajmi.majesticlife.DataStructures;  
  
**import** android.arch.persistence.room.ColumnInfo;  
**import** android.arch.persistence.room.Entity;  
**import** android.arch.persistence.room.ForeignKey;  
**import** android.arch.persistence.room.PrimaryKey;  
  
**import** javax.annotation.Nonnull;  
  
**import static** android.arch.persistence.room.ForeignKey.***CASCADE***;  
@Entity(tableName = **"Skill\_table"**,  
 foreignKeys = @ForeignKey(entity = User.**class**,  
 parentColumns = **"User\_ID"**,  
 childColumns = **"User\_ID"**,  
 onDelete = ***CASCADE***))  
**public class** Skill  
{  
 @Nonnull  
 @PrimaryKey  
 @ColumnInfo(name = **"Skill\_ID"**)  
 **private int Skill\_ID**;  
  
 @ColumnInfo(name = **"Skill\_Name"**)  
 **private** String **Skill\_Name**;  
  
 @ColumnInfo(name = **"Avatar\_ResIndex"**)  
 **private int Avatar\_ResIndex**;  
  
 @ColumnInfo(name = **"Level"**)  
 **private int Level**;  
  
 @ColumnInfo(name = **"Progress"**)  
 **private int Progress**; *//the increasing value to reach the next level* @ColumnInfo(name = **"User\_ID"**)  
 **private int User\_ID**;  
  
 @Nonnull  
 **public int** getSkill\_ID(){**return this**.**Skill\_ID**;}  
 **public** String getSkill\_Name(){**return this**.**Skill\_Name**;}  
 **public int** getAvatar\_ResIndex(){**return this**.**Avatar\_ResIndex**;}  
 **public int** getLevel(){**return this**.**Level**;}  
 **public int** getProgress(){**return this**.**Progress**;}  
 **public int** getUser\_ID(){**return this**.**User\_ID**;}  
  
 **public void** setSkill\_ID(**int** Skill\_ID){**this**.**Skill\_ID** = Skill\_ID;}  
 **public void** setSkill\_Name (String Skill\_Name){**this**.**Skill\_Name** = Skill\_Name;}  
 **public void** setAvatar\_ResIndex (**int** Avatar\_ResIndex){**this**.**Avatar\_ResIndex** = Avatar\_ResIndex;}  
 **public void** setLevel (**int** Level){**this**.**Level** = Level;}  
 **public void** setProgress (**int** Progress){**this**.**Progress** = Progress;}  
 **public void** setUser\_ID (**int** User\_ID){**this**.**User\_ID** = User\_ID;}  
  
 **public** Skill (**int** Skill\_ID, String Skill\_Name, **int** Avatar\_ResIndex, **int** Level, **int** Progress, **int** User\_ID)  
 {  
 **this**.**Skill\_ID** = Skill\_ID;  
 **this**.**Skill\_Name** = Skill\_Name;  
 **this**.**Avatar\_ResIndex** = Avatar\_ResIndex;  
 **this**.**Level** = Level;  
 **this**.**Progress** = Progress;  
 **this**.**User\_ID** = User\_ID;  
 }  
}

**package** com.farazannajmi.majesticlife.DataStructures;  
  
**import** android.arch.persistence.room.ColumnInfo;  
**import** android.arch.persistence.room.Entity;  
**import** android.arch.persistence.room.PrimaryKey;  
  
**import** java.sql.Time;  
**import** java.util.List;  
  
**import** javax.annotation.Nonnull;@Entity(tableName = **"Quest\_table"**)  
**public class** Quest  
{@PrimaryKey  
 @Nonnull  
 @ColumnInfo(name = **"Quest\_ID"**)  
 **private int Quest\_ID**;  
  
 @ColumnInfo(name = **"Quest\_Name"**)  
 **private** String **Quest\_Name**;  
  
 @ColumnInfo(name = **"DuaDate"**)  
 **private** String **DuaDate**;  
  
 @Nonnull  
 **public int** getQuest\_ID() {**return Quest\_ID**;}  
 **public** String getQuest\_Name() {**return Quest\_Name**;}  
 **public** String getDuaDate() {**return DuaDate**;}  
  
 **public void** setQuest\_ID(@Nonnull **int** quest\_ID) {**Quest\_ID** = quest\_ID;}  
 **public void** setQuest\_Name(String quest\_Name) {**Quest\_Name** = quest\_Name;}  
 **public void** setDuaDate(String duaDate) {**DuaDate** = duaDate;}  
  
 **public** Quest(@Nonnull **int** Quest\_ID, String Quest\_Name, String DuaDate)  
 {  
 **this**.**Quest\_ID** = Quest\_ID;  
 **this**.**Quest\_Name** = Quest\_Name;  
 **this**.**DuaDate** = DuaDate;  
 }  
}

**package** com.farazannajmi.majesticlife.DataStructures;  
  
**import** android.arch.persistence.room.ColumnInfo;  
**import** android.arch.persistence.room.Entity;  
**import** android.arch.persistence.room.ForeignKey;  
**import** android.arch.persistence.room.PrimaryKey;  
  
**import** javax.annotation.Nonnull;  
  
**import static** android.arch.persistence.room.ForeignKey.***CASCADE***;@Entity(tableName = **"PlanCell\_table"**,  
 foreignKeys = @ForeignKey(entity = Faaliat.**class**,  
 parentColumns = **"Faaliat\_ID"**,  
 childColumns = **"Faaliat\_ID"**,  
 onDelete = ***CASCADE***))  
**public class** PlanCell  
{  
 @PrimaryKey  
 @Nonnull  
 @ColumnInfo(name = **"PlanCell\_ID"**)  
 **private int PlanCell\_ID**;  
  
 @Nonnull  
 @ColumnInfo(name = **"Faaliat\_ID"**)  
 **private int Faaliat\_ID**;  
  
 @ColumnInfo(name = **"WeekDay"**)  
 **private int WeekDay**;  
 @ColumnInfo(name = **"DayHour"**)  
 **private int DayHour**;  
  
 @Nonnull  
 **public int** getPlanCell\_ID(){**return this**.**PlanCell\_ID**;}  
 **public int** getFaaliat\_ID(){**return this**.**Faaliat\_ID**;}  
 **public int** getWeekDay(){**return this**.**WeekDay**;}  
 **public int** getDayHour(){**return this**.**DayHour**;}  
  
 **public void** setPlanCell\_ID(**int** PlanCell\_ID){**this**.**PlanCell\_ID** = PlanCell\_ID;}  
 **public void** setFaaliat\_ID(**int** Faaliat\_ID){**this**.**Faaliat\_ID** = Faaliat\_ID;}  
 **public void** setWeekDay(**int** WeekDay){**this**.**WeekDay** = WeekDay;}  
 **public void** setDayHour(**int** DayHour){**this**.**DayHour** = DayHour;}  
  
 **public** PlanCell (**int** PlanCell\_ID, **int** Faaliat\_ID, **int** WeekDay, **int** DayHour)  
 {  
 **this**.**PlanCell\_ID** = PlanCell\_ID;  
 **this**.**Faaliat\_ID** = Faaliat\_ID;  
 **this**.**WeekDay** = WeekDay;  
 **this**.**DayHour** = DayHour;  
 }  
}

**package** com.farazannajmi.majesticlife.DataStructures;  
  
**import** android.arch.persistence.room.ColumnInfo;  
**import** android.arch.persistence.room.Entity;  
**import** android.arch.persistence.room.ForeignKey;  
**import** android.arch.persistence.room.PrimaryKey;  
  
**import** javax.annotation.Nonnull;  
  
**import static** android.arch.persistence.room.ForeignKey.***CASCADE***;  
@Entity(tableName = **"Avatar\_table"**,  
 foreignKeys = {  
 @ForeignKey(entity = User.**class**,  
 parentColumns = **"User\_ID"**,  
 childColumns = **"User\_ID"**,  
 onDelete = ***CASCADE***)})  
**public class** Avatar  
{  
 @Nonnull  
 @PrimaryKey  
 @ColumnInfo(name = **"Avatar\_ID"**)  
 **private int Avatar\_ID**;  
  
 @ColumnInfo(name = **"User\_ID"**)  
 **private int User\_ID**;  
  
 @ColumnInfo(name = **"IsKing"**)  
 **private boolean IsKing**;  
  
  
 @ColumnInfo(name = **"Background\_ResIndex"**)  
 **private int Background\_ResIndex**;  
  
 @ColumnInfo(name = **"Skin\_ResIndex"**)  
 **private int Skin\_ResIndex**;  
  
 @ColumnInfo(name = **"Cloth\_ResIndex"**)  
 **private int Cloth\_ResIndex**;  
  
 @ColumnInfo(name = **"Eyes\_ResIndex"**)  
 **private int Eyes\_ResIndex**;  
  
 @ColumnInfo(name = **"Mouth\_ResIndex"**)  
 **private int Mouth\_ResIndex**;  
  
 @ColumnInfo(name = **"Crown\_ResIndex"**)  
 **private int Crown\_ResIndex**;  
  
 @Nonnull  
 **public int** getAvatar\_ID() {**return Avatar\_ID**;}  
 **public int** getUser\_ID() {**return User\_ID**;}  
 **public boolean** getIsKing() {**return IsKing**;}  
 **public int** getBackground\_ResIndex() {**return Background\_ResIndex**;}  
 **public int** getSkin\_ResIndex() {**return Skin\_ResIndex**;}  
 **public int** getCloth\_ResIndex() {**return Cloth\_ResIndex**;}  
 **public int** getEyes\_ResIndex() {**return Eyes\_ResIndex**;}  
 **public int** getMouth\_ResIndex() {**return Mouth\_ResIndex**;}  
 **public int** getCrown\_ResIndex() {**return Crown\_ResIndex**;}  
  
 **public void** setAvatar\_ID(@Nonnull **int** Avatar\_ID) {**this**.**Avatar\_ID** = Avatar\_ID;}  
 **public void** setUser\_ID(**int** User\_ID) {**this**.**User\_ID** = User\_ID;}  
 **public void** setIsKing(**boolean** IsKing) {**this**.**IsKing** = IsKing;}  
 **public void** setBackground\_ResIndex(**int** Background\_ResIndex) {**this**.**Background\_ResIndex** = Background\_ResIndex;}  
 **public void** setSkin\_ResIndex(**int** Skin\_ResIndex) {**this**.**Skin\_ResIndex** = Skin\_ResIndex;}  
 **public void** setCloth\_ResIndex(**int** Cloth\_ResIndex) {**this**.**Cloth\_ResIndex** = Cloth\_ResIndex;}  
 **public void** setEyes\_ResIndex(**int** Eyes\_ResIndex) {**this**.**Eyes\_ResIndex** = Eyes\_ResIndex;}  
 **public void** setMouth\_ResIndex(**int** Mouth\_ResIndex) {**this**.**Mouth\_ResIndex** = Mouth\_ResIndex;}  
 **public void** setCrown\_ResIndex(**int** Crown\_ResIndex) {**this**.**Crown\_ResIndex** = Crown\_ResIndex;}  
  
 **public** Avatar(@Nonnull **int** Avatar\_ID, **int** User\_ID, **boolean** IsKing, **int** Background\_ResIndex,  
 **int** Skin\_ResIndex, **int** Cloth\_ResIndex, **int** Eyes\_ResIndex, **int** Mouth\_ResIndex, **int** Crown\_ResIndex)  
 {  
 **this**.**Avatar\_ID** = Avatar\_ID;  
 **this**.**User\_ID** = User\_ID;  
 **this**.**IsKing** = IsKing;  
 **this**.**Background\_ResIndex** = Background\_ResIndex;  
 **this**.**Skin\_ResIndex** = Skin\_ResIndex;  
 **this**.**Cloth\_ResIndex** = Cloth\_ResIndex;  
 **this**.**Eyes\_ResIndex** = Eyes\_ResIndex;  
 **this**.**Mouth\_ResIndex** = Mouth\_ResIndex;  
 **this**.**Crown\_ResIndex** = Crown\_ResIndex;  
 }  
}

**package** com.farazannajmi.majesticlife.DataStructures;  
  
**import** android.arch.persistence.room.ColumnInfo;  
**import** android.arch.persistence.room.Entity;  
**import** android.arch.persistence.room.PrimaryKey;  
  
**import** javax.annotation.Nonnull;  
@Entity(tableName = **"AvatarItem\_table"**)  
**public class** AvatarItem  
{  
 @Nonnull  
 @PrimaryKey  
 @ColumnInfo(name = **"ResourceIndex"**)  
 **private int ResourceIndex**;  
  
 @ColumnInfo(name = **"IsBought"**)  
 **private boolean IsBought**;  
  
 @ColumnInfo(name = **"ItemType"**)  
 **private int ItemType**;  
 */\*  
 0 => background  
 1 => skin  
 2 => cloth  
 3 => eyes  
 4 => mouth  
 5 => crown  
 \*/* @ColumnInfo(name = **"Price"**)  
 **private int Price**;  
  
 @Nonnull  
 **public int** getResourceIndex() {**return ResourceIndex**;}  
 **public boolean** getIsBought() {**return IsBought**;}  
 **public int** getItemType() {**return ItemType**;}  
 **public int** getPrice() {**return Price**;}  
  
 **public void** setResourceIndex(@Nonnull **int** ResourceIndex) {**this**.**ResourceIndex** = ResourceIndex;}  
 **public void** setIsBought(**boolean** IsBought) {**this**.**IsBought** = IsBought;}  
 **public void** setItemType(**int** ItemType) {**this**.**ItemType** = ItemType;}  
 **public void** setPrice(**int** Price) {**this**.**Price** = Price;}  
  
 **public** AvatarItem(@Nonnull **int** ResourceIndex, **boolean** IsBought, **int** ItemType, **int** Price)  
 {  
 **this**.**ResourceIndex** = ResourceIndex;  
 **this**.**IsBought** = IsBought;  
 **this**.**ItemType** = ItemType;  
 **this**.**Price** = Price;  
 }  
}

**package** com.farazannajmi.majesticlife.DataStructures;  
  
**import** android.arch.persistence.room.ColumnInfo;  
**import** android.arch.persistence.room.Entity;  
**import** android.arch.persistence.room.ForeignKey;  
  
**import** com.farazannajmi.majesticlife.DataStructures.Skill;  
  
**import static** android.arch.persistence.room.ForeignKey.***CASCADE***;

*/\*\*  
 \* join entity table of Faaliat and Skill  
 \*/*

@Entity(tableName = **"FaaliatSkill\_table"**,  
 primaryKeys = { **"Faaliat\_ID"**, **"Skill\_ID"** },  
 foreignKeys = {  
 @ForeignKey(entity = Faaliat.**class**,  
 parentColumns = **"Faaliat\_ID"**,  
 childColumns = **"Faaliat\_ID"**,  
 onDelete = ***CASCADE***),  
 @ForeignKey(entity = Skill.**class**,  
 parentColumns = **"Skill\_ID"**,  
 childColumns = **"Skill\_ID"**,  
 onDelete = ***CASCADE***)  
 })  
**public class** FaaliatSkill  
{  
 **private int Faaliat\_ID**;  
 **private int Skill\_ID**;  
  
 @ColumnInfo(name = **"RepetitionCount"**)  
 **private int RepetitionCount**;  
  
 **public int** getFaaliat\_ID() {**return Faaliat\_ID**;}  
 **public int** getSkill\_ID() {**return Skill\_ID**;}  
 **public int** getRepetitionCount() {**return RepetitionCount**;}  
  
 **public void** setFaaliat\_ID(**int** faaliat\_ID) {**Faaliat\_ID** = faaliat\_ID;}  
 **public void** setSkill\_ID(**int** skill\_ID) {**Skill\_ID** = skill\_ID;}  
 **public void** setRepetitionCount(**int** repetitionCount) {**RepetitionCount** = repetitionCount;}  
  
 **public** FaaliatSkill(**int** Faaliat\_ID, **int** Skill\_ID, **int** RepetitionCount*/\*, int ListIndex\*/*)  
 {  
 **this**.**Faaliat\_ID** = Faaliat\_ID;  
 **this**.**Skill\_ID** = Skill\_ID;  
 **this**.**RepetitionCount** = RepetitionCount;  
 }  
}

**package** com.farazannajmi.majesticlife.DataStructures;  
  
**import** android.arch.persistence.room.ColumnInfo;  
**import** android.arch.persistence.room.Entity;  
**import** android.arch.persistence.room.ForeignKey;  
  
**import** com.farazannajmi.majesticlife.DataStructures.Faaliat;  
  
**import static** android.arch.persistence.room.ForeignKey.***CASCADE***;  
  
*/\*\*  
 \* join entity table of Quest and Skill  
 \*/*@Entity(tableName = **"QuestSkill\_table"**,  
 primaryKeys = { **"Quest\_ID"**, **"Skill\_ID"** },  
 foreignKeys = {  
 @ForeignKey(entity = Quest.**class**,  
 parentColumns = **"Quest\_ID"**,  
 childColumns = **"Quest\_ID"**,  
 onDelete = ***CASCADE***),  
 @ForeignKey(entity = Skill.**class**,  
 parentColumns = **"Skill\_ID"**,  
 childColumns = **"Skill\_ID"**,  
 onDelete = ***CASCADE***)  
 })  
**public class** QuestSkill  
{  
 **private int Quest\_ID**;  
 **private int Skill\_ID**;  
  
 @ColumnInfo(name = **"TargetLevel"**)  
 **private int TargetLevel**;  
  
 **public int** getQuest\_ID() {**return Quest\_ID**;}  
 **public int** getSkill\_ID() {**return Skill\_ID**;}  
 **public int** getTargetLevel() {**return TargetLevel**;}  
  
 **public void** setQuest\_ID(**int** quest\_ID) {**Quest\_ID** = quest\_ID;}  
 **public void** setSkill\_ID(**int** skill\_ID) {**Skill\_ID** = skill\_ID;}  
 **public void** setTargetLevel(**int** targetLevel) {**TargetLevel** = targetLevel;}  
  
 **public** QuestSkill(**int** Quest\_ID, **int** Skill\_ID, **int** TargetLevel)  
 {  
 **this**.**Quest\_ID** = Quest\_ID;  
 **this**.**Skill\_ID** = Skill\_ID;  
 **this**.**TargetLevel** = TargetLevel;  
 }  
}

**package** com.farazannajmi.majesticlife.DataStructures;  
  
**import** android.arch.persistence.room.ColumnInfo;  
**import** android.arch.persistence.room.Embedded;  
**import** android.arch.persistence.room.Entity;  
**import** android.arch.persistence.room.ForeignKey;  
**import** android.arch.persistence.room.PrimaryKey;  
**import** android.support.annotation.NonNull;  
  
**import** javax.annotation.Nonnull;  
  
**import static** android.arch.persistence.room.ForeignKey.***CASCADE***;@Entity(tableName = **"FaaliatRepetitions\_table"**,  
 primaryKeys = {**"Faaliat\_ID"**, **"FR\_Date"**},  
 foreignKeys = @ForeignKey(entity = Faaliat.**class**,  
 parentColumns = **"Faaliat\_ID"**,  
 childColumns = **"Faaliat\_ID"**,  
 onDelete = ***CASCADE***))  
**public class** FaaliatRepetitions  
{  
 @Nonnull  
 @ColumnInfo(name = **"Faaliat\_ID"**)  
 **private int Faaliat\_ID**;  
  
 @NonNull  
 @ColumnInfo(name = **"FR\_Date"**)  
 **private** String **FR\_Date**;  
  
 @ColumnInfo(name = **"DayOfWeek"**)  
 **private int DayOfWeek**;  
  
 @ColumnInfo(name = **"RepetitionCount"**)  
 **private int RepetitionCount**;  
  
  
 @Nonnull  
 **public int** getFaaliat\_ID() {**return Faaliat\_ID**;}  
  
 @Nonnull  
 **public** String getFR\_Date() {**return FR\_Date**;}  
 **public int** getDayOfWeek() {**return DayOfWeek**;}  
 **public int** getRepetitionCount() {**return RepetitionCount**;}  
  
 **public void** setFaaliat\_ID(@Nonnull **int** Faaliat\_ID) {**this**.**Faaliat\_ID** = Faaliat\_ID;}  
 **public void** setFR\_Date(@Nonnull String FR\_Date) {**this**.**FR\_Date** = FR\_Date;}  
 **public void** setDayOfWeek(**int** DayOfWeek) {**this**.**DayOfWeek** = DayOfWeek;}  
 **public void** setRepetitionCount(**int** RepetitionCount) {**this**.**RepetitionCount** = RepetitionCount;}  
  
 **public** FaaliatRepetitions(@Nonnull **int** Faaliat\_ID, **int** DayOfWeek, @Nonnull String FR\_Date, **int** RepetitionCount)  
 {  
 **this**.**Faaliat\_ID** = Faaliat\_ID;  
 **this**.**DayOfWeek** = DayOfWeek;  
 **this**.**FR\_Date** = FR\_Date;  
 **this**.**RepetitionCount** = RepetitionCount ;  
 }  
}

DAOی برخی از موجودیت‌‌ها:

**package** com.farazannajmi.majesticlife.DataStructures;  
  
**import** android.arch.lifecycle.LiveData;  
**import** android.arch.persistence.room.Dao;  
**import** android.arch.persistence.room.Delete;  
**import** android.arch.persistence.room.Insert;  
**import** android.arch.persistence.room.Query;  
**import** android.arch.persistence.room.Update;  
  
**import** java.util.List;@Dao  
**public interface** FaaliatDao  
{  
 @Insert  
 **void** insert(Faaliat faaliat);  
  
 @Update  
 **void** update(Faaliat faaliat);  
  
 @Delete  
 **void** delete(Faaliat faaliat);  
  
 @Query(**"DELETE FROM Faaliat\_table"**)  
 **void** deleteAll();  
  
 @Query(**"SELECT \* FROM Faaliat\_table ORDER BY Faaliat\_ID"**)  
 LiveData<List<Faaliat>> getAllFaaliats();  
}

**package** com.farazannajmi.majesticlife.DataStructures;  
  
**import** android.arch.lifecycle.LiveData;  
**import** android.arch.persistence.room.Dao;  
**import** android.arch.persistence.room.Delete;  
**import** android.arch.persistence.room.Insert;  
**import** android.arch.persistence.room.Query;  
**import** android.arch.persistence.room.Update;  
  
**import** java.util.List;  
@Dao  
**public interface** FaaliatRepetitionsDao  
{  
 @Insert  
 **void** insert(FaaliatRepetitions faaliatRepetitions);  
  
 @Update  
 **void** update(FaaliatRepetitions faaliatRepetitions);  
  
 @Delete  
 **void** delete(FaaliatRepetitions faaliatRepetitions);  
  
 @Query(**"DELETE FROM FaaliatRepetitions\_table"**)  
 **void** deleteAll();  
  
 @Query(**"SELECT \* FROM FaaliatRepetitions\_table"**)  
 LiveData<List<FaaliatRepetitions>> getAllFaaliatRepetitions();  
  
 @Query(**"SELECT \* FROM FaaliatRepetitions\_table "** +  
 **"WHERE FaaliatRepetitions\_table.Faaliat\_ID=:faaliatID "** +  
 **"ORDER BY DayOfWeek"**)  
 LiveData<List<FaaliatRepetitions>> getAllFaaliatRepsForFaaliat(**final int** faaliatID);  
  
 @Query(**"SELECT \* FROM FaaliatRepetitions\_table "** +  
 **"WHERE Faaliat\_ID=:faaliatID AND FR\_Date=:date "** +  
 **"ORDER BY DayOfWeek"**)  
 LiveData<List<FaaliatRepetitions>> getFaaliatRepsForFaaliatDate(**final int** faaliatID, **final** String date);  
}

Repositoryی موجودیت فعالیت:

**package** com.farazannajmi.majesticlife.DataStructures;  
  
**import** android.app.Application;  
**import** android.arch.lifecycle.LiveData;  
**import** android.os.AsyncTask;  
  
**import** java.util.List;**public class** FaaliatRepository  
{  
 **private** FaaliatDao **mFaaliatDao**;  
 **private** LiveData<List<Faaliat>> **mAllFaaliats**;  
  
 **public** FaaliatRepository(Application application)  
 {  
 MajesticLifeRoomDatabase db = MajesticLifeRoomDatabase.*getDatabase*(application);  
 **mFaaliatDao** = db.faaliatDao();  
 **mAllFaaliats** = **mFaaliatDao**.getAllFaaliats();  
 }  
  
 *//Room executes all queries on a separate thread.  
 //Observed LiveData will notify the observer when the data has changed.* LiveData<List<Faaliat>> getAllFaaliats()  
 {  
 **return mAllFaaliats**;  
 }  
  
 */\*a wrapper for the insert() method. You must call this on a non-UI thread or your app will crash.  
 Room ensures that you don't do any long-running operations on the main thread, blocking the UI.\*/* **public void** insert (Faaliat faaliat)  
 {  
 **new** FaaliatRepository.insertAsyncTask(**mFaaliatDao**).execute(faaliat);  
 }  
  
 **public void** update (Faaliat faaliat)  
 {  
 **new** FaaliatRepository.updateAsyncTask(**mFaaliatDao**).execute(faaliat);  
 }  
  
 **public void** delete (Faaliat faaliat)  
 {  
 **new** FaaliatRepository.deleteAsyncTask(**mFaaliatDao**).execute(faaliat);  
 }  
  
 **private static class** insertAsyncTask **extends** AsyncTask<Faaliat, Void, Void>  
 {  
 **private** FaaliatDao **mAsyncTaskDao**;  
  
 insertAsyncTask(FaaliatDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(**final** Faaliat... params)  
 {  
 **mAsyncTaskDao**.insert(params[0]);  
 **return null**;  
 }  
 }  
  
 **private static class** updateAsyncTask **extends** AsyncTask<Faaliat, Void, Void>  
 {  
 **private** FaaliatDao **mAsyncTaskDao**;  
  
 updateAsyncTask(FaaliatDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(**final** Faaliat... params)  
 {  
 **mAsyncTaskDao**.update(params[0]);  
 **return null**;  
 }  
 }  
  
 **private static class** deleteAsyncTask **extends** AsyncTask<Faaliat, Void, Void>  
 {  
 **private** FaaliatDao **mAsyncTaskDao**;  
  
 deleteAsyncTask(FaaliatDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(**final** Faaliat... params)  
 {  
 **mAsyncTaskDao**.delete(params[0]);  
 **return null**;  
 }  
 }  
}

ViewModelی موجودیت فعالیت:

**package** com.farazannajmi.majesticlife.DataStructures;  
  
**import** android.app.Application;  
**import** android.arch.lifecycle.AndroidViewModel;  
**import** android.arch.lifecycle.LiveData;  
  
**import** java.util.List;**public class** FaaliatViewModel **extends** AndroidViewModel  
{  
 **private** FaaliatRepository **mRepository**;  
 **private** LiveData<List<Faaliat>> **mAllFaaliats**;  
  
 **public** LiveData<List<Faaliat>> getAllFaaliats() {**return mAllFaaliats**;}  
  
 **public** FaaliatViewModel (Application application)  
 {  
 **super**(application);  
 **mRepository** = **new** FaaliatRepository(application);  
 **mAllFaaliats** = **mRepository**.getAllFaaliats();  
 }  
  
 **public void** insert(Faaliat faaliat)  
 {  
 **mRepository**.insert(faaliat);  
 }  
  
 **public void** update(Faaliat faaliat)  
 {  
 **mRepository**.update(faaliat);  
 }  
  
 **public void** delete(Faaliat faaliat)  
 {  
 **mRepository**.delete(faaliat);  
 }  
}

کلاس پایگاه داده room:

**package** com.farazannajmi.majesticlife.DataStructures;  
  
**import** android.arch.persistence.db.SupportSQLiteDatabase;  
**import** android.arch.persistence.room.Database;  
**import** android.arch.persistence.room.Room;  
**import** android.arch.persistence.room.RoomDatabase;  
**import** android.content.Context;  
**import** android.os.AsyncTask;  
**import** android.support.annotation.NonNull;  
**import** android.util.Log;  
  
**import** com.farazannajmi.majesticlife.DataHolder;  
**import** com.farazannajmi.majesticlife.LoadingActivity;  
**import** com.farazannajmi.majesticlife.R;  
@Database(entities = {User.**class**, Faaliat.**class**, Skill.**class**, Quest.**class**, PlanCell.**class**,  
 FaaliatSkill.**class**, QuestSkill.**class**, FaaliatRepetitions.**class**,  
 Avatar.**class**, AvatarItem.**class**},  
 version = 6)  
**public abstract class** MajesticLifeRoomDatabase **extends** RoomDatabase  
{  
 **private static final** String DB\_NAME = **"MajesticLifeDatabase.db"**;  
 **private static volatile** MajesticLifeRoomDatabase INSTANCE;  
  
 **public abstract** UserDao userDao();  
 **public abstract** FaaliatDao faaliatDao();  
 **public abstract** SkillDao skillDao();  
 **public abstract** QuestDao questDao();  
 **public abstract** PlanCellDao planCellDao();  
 **public abstract** FaaliatSkillDao faaliatSkillDao();  
 **public abstract** QuestSkillDao questSkillDao();  
 **public abstract** FaaliatRepetitionsDao faaliatRepetitionsDao();  
 **public abstract** AvatarDao avatarDao();  
 **public abstract** AvatarItemDao avatarItemDao();  
  
 **public static** MajesticLifeRoomDatabase getDatabase(**final** Context context)  
 {  
 **if** (INSTANCE == **null**)  
 {  
 **synchronized** (MajesticLifeRoomDatabase.**class**)  
 {  
 **if** (INSTANCE == **null**)  
 {  
 *// Create database here* INSTANCE = Room.databaseBuilder(context.getApplicationContext(),  
 MajesticLifeRoomDatabase.**class**, DB\_NAME)  
 .addCallback(sRoomDatabaseCallback) *//for initial data to database* .fallbackToDestructiveMigration() *//drop and recreate the whole database if db version goes up* .build();  
 }  
 **else** LoadingActivity.isDataLoaded = **true**;  
 }  
 }  
 **else** LoadingActivity.isDataLoaded = **true**;  
 **return** INSTANCE;  
 }  
  
 **private static** RoomDatabase.Callback sRoomDatabaseCallback = **new** RoomDatabase.Callback()  
 {  
 @Override  
 **public void** onOpen (@NonNull SupportSQLiteDatabase db)  
 {  
 **super**.onOpen(db);  
 **if**(!LoadingActivity.isFirstTime)  
 {  
 Log.d(**"Data"**, **"Database has been initialed!"**);  
 }  
 LoadingActivity.isDataLoaded = **true**;  
  
 Log.d(**"Data"**, **"Database opened!"**);  
 }  
  
 @Override  
 **public void** onCreate(@NonNull SupportSQLiteDatabase db)  
 {  
 **super**.onCreate(db);  
 **new** PopulateDbAsync(INSTANCE).execute();  
 Log.d(**"Data"**, **"Database created!"**);  
 }  
 };  
  
 **private static class** PopulateDbAsync **extends** AsyncTask<Void, Void, Void>  
 {  
 **private final** UserDao userDao;  
 **private final** FaaliatDao faaliatDao;  
 **private final** SkillDao skillDao;  
 **private final** QuestDao questDao;  
 **private final** PlanCellDao planCellDao;  
 **private final** FaaliatSkillDao faaliatSkillDao;  
 **private final** QuestSkillDao questSkillDao;  
 **private final** FaaliatRepetitionsDao faaliatRepetitionsDao;  
 **private final** AvatarDao avatarDao;  
 **private final** AvatarItemDao avatarItemDao;  
  
 PopulateDbAsync(MajesticLifeRoomDatabase db)  
 {  
 userDao = db.userDao();  
 faaliatDao = db.faaliatDao();  
 skillDao = db.skillDao();  
 questDao = db.questDao();  
 planCellDao = db.planCellDao();  
 faaliatSkillDao = db.faaliatSkillDao();  
 questSkillDao = db.questSkillDao();  
 faaliatRepetitionsDao = db.faaliatRepetitionsDao();  
 avatarDao = db.avatarDao();  
 avatarItemDao = db.avatarItemDao();  
 }  
  
 @Override  
 **protected** Void doInBackground(**final** Void... params)  
 {  
 **if**(LoadingActivity.isFirstTime && !DataHolder.IsDatabaseCreated)  
 {  
 *//orders are important!!  
  
 //User:* userDao.deleteAll();  
 User user = **new** User(0, **"NewKing"**, **"king@mail.com"**,  
 0, 1,0, 1, 0, 1, 100);  
 userDao.insert(user);  
  
 *//Faaliats:* faaliatDao.deleteAll();  
 Faaliat faaliat = **new** Faaliat(0, **"Reading book"**,  
 R.drawable.ic\_majestic\_activities, R.color.faaliatsColor1, 10,0,1,0);  
 faaliatDao.insert(faaliat);  
  
 *//Skills:* skillDao.deleteAll();  
 Skill skill = **new** Skill(0, **"Knowledge"**, R.drawable.ic\_skills, 1, 0, 0);  
 skillDao.insert(skill);  
  
 *//FaaliatSkills:* faaliatSkillDao.deleteAll();  
 FaaliatSkill faaliatSkill = **new** FaaliatSkill(0, 0, 5);  
 faaliatSkillDao.insert(faaliatSkill);  
  
 *//PlanCells:* planCellDao.deleteAll();  
 PlanCell planCell = **new** PlanCell(0, 0, 0,5);  
 planCellDao.insert(planCell);  
  
 *//Quests:* questDao.deleteAll();  
 Quest quest = **new** Quest(0, **"Pass exam"**, **"2019/1/1"**);questDao.insert(quest);  
  
 *//QuestSkills:* questSkillDao.deleteAll();  
 QuestSkill questSkill = **new** QuestSkill(0, 0, 2);  
 questSkillDao.insert(questSkill);  
  
 *//region ---------- avatarItems ----------  
 //backgrounds:* avatarItemDao.deleteAll();  
 AvatarItem ava\_back1 = **new** AvatarItem(R.drawable.ava\_back1, **true**, 0, 0);  
 avatarItemDao.insert(ava\_back1);  
 AvatarItem ava\_back2 = **new** AvatarItem(R.drawable.ava\_back2, **false**, 0, 50);  
 avatarItemDao.insert(ava\_back2);  
 AvatarItem ava\_back3 = **new** AvatarItem(R.drawable.ava\_back3, **false**, 0, 60);  
 avatarItemDao.insert(ava\_back3);  
 AvatarItem ava\_back4 = **new** AvatarItem(R.drawable.ava\_back4, **false**, 0, 60);  
 avatarItemDao.insert(ava\_back4);  
 AvatarItem ava\_back5 = **new** AvatarItem(R.drawable.ava\_back5, **false**, 0, 70);  
 avatarItemDao.insert(ava\_back5);  
 AvatarItem ava\_back6 = **new** AvatarItem(R.drawable.ava\_back6, **false**, 0, 70);  
 avatarItemDao.insert(ava\_back6);  
 AvatarItem ava\_back7 = **new** AvatarItem(R.drawable.ava\_back7, **false**, 0, 70);  
 avatarItemDao.insert(ava\_back7);  
  
 *//Skins* AvatarItem ava\_skin1 = **new** AvatarItem(R.drawable.ava\_skin\_white, **true**, 0, 0);  
 avatarItemDao.insert(ava\_skin1); *//Clothes* AvatarItem ava\_cloth1 = **new** AvatarItem(R.drawable.ava\_cloth1, **true**, 0, 0);  
 **avatarItemDao**.insert(ava\_cloth1); *//Eyes* AvatarItem ava\_eyes1 = **new** AvatarItem(R.drawable.ava\_eyes1, **true**, 0, 0);  
 avatarItemDao.insert(ava\_eyes1); *//Mouths* AvatarItem ava\_mouth1 = **new** AvatarItem(R.drawable.ava\_mouth1, **true**, 0, 0);  
 avatarItemDao.insert(ava\_mouth1);  *//Crowns* AvatarItem ava\_crown1 = **new** AvatarItem(R.drawable.ava\_crown1, **true**, 0, 0);  
 avatarItemDao.insert(ava\_crown1);  
 *//todo: insert other crowns  
 //endregion ------------------------------  
  
 //FaaliatRepetitions  
 //test:* faaliatRepetitionsDao.deleteAll();  
 FaaliatRepetitions fr1 = **new** FaaliatRepetitions(0, 1, **"2018/6/28"**, 1);  
 FaaliatRepetitions fr2 = **new** FaaliatRepetitions(0, 2, **"2018/6/29"**, 3);  
 FaaliatRepetitions fr3 = **new** FaaliatRepetitions(0, 3, **"2018/6/30"**, 2);  
 FaaliatRepetitions fr4 = **new** FaaliatRepetitions(0, 4, **"2018/6/31"**, 5);  
 FaaliatRepetitions fr5 = **new** FaaliatRepetitions(0, 6, **"2018/7/2"**, 3);  
 FaaliatRepetitions fr6 = **new** FaaliatRepetitions(0, 1, **"2018/7/4"**, 1);  
 FaaliatRepetitions fr7 = **new** FaaliatRepetitions(0, 3, **"2018/7/6"**, 1);  
 FaaliatRepetitions fr8 = **new** FaaliatRepetitions(0, 4, **"2018/7/7"**, 2);  
 faaliatRepetitionsDao.insert(fr1);  
 faaliatRepetitionsDao.insert(fr2);  
 faaliatRepetitionsDao.insert(fr3);  
 faaliatRepetitionsDao.insert(fr4);  
 faaliatRepetitionsDao.insert(fr5);  
 faaliatRepetitionsDao.insert(fr6);  
 faaliatRepetitionsDao.insert(fr7);  
 faaliatRepetitionsDao.insert(fr8);  
  
 *//Avatar:* avatarDao.deleteAll();  
 Avatar avatar = **new** Avatar(0,0, **true**, R.drawable.ava\_back1,  
 R.drawable.ava\_skin\_white, R.drawable.ava\_cloth1,  
 R.drawable.ava\_eyes1, R.drawable.ava\_mouth1, R.drawable.ava\_crown1);  
 avatarDao.insert(avatar);  
  
 DataHolder.IsDatabaseCreated = **true**;  
 Log.d(**"Data"**, **"Database initialed!"**);  
 }  
 **return null**;  
 }  
 }  
}

کلاس DataHolder که برای مدیریت و cache کردن داده‌های برنامه نوشته شده است:

**package** com.farazannajmi.majesticlife;  
  
**import** android.arch.lifecycle.LifecycleOwner;  
**import** android.arch.lifecycle.Observer;  
**import** android.arch.lifecycle.ViewModelProviders;  
**import** android.support.annotation.Nullable;  
**import** android.support.v4.app.FragmentActivity;  
**import** android.util.Log;  
  
**import** com.backtory.java.internal.BacktoryUser;  
**import** com.farazannajmi.majesticlife.DataStructures.Avatar;  
**import** com.farazannajmi.majesticlife.DataStructures.AvatarItem;  
**import** com.farazannajmi.majesticlife.DataStructures.AvatarItemViewModel;  
**import** com.farazannajmi.majesticlife.DataStructures.AvatarViewModel;  
**import** com.farazannajmi.majesticlife.DataStructures.Faaliat;  
**import** com.farazannajmi.majesticlife.DataStructures.FaaliatRepetitions;  
**import** com.farazannajmi.majesticlife.DataStructures.FaaliatRepetitionsViewModel;  
**import** com.farazannajmi.majesticlife.DataStructures.FaaliatSkill;  
**import** com.farazannajmi.majesticlife.DataStructures.FaaliatSkillViewModel;  
**import** com.farazannajmi.majesticlife.DataStructures.FaaliatViewModel;  
**import** com.farazannajmi.majesticlife.DataStructures.PlanCell;  
**import** com.farazannajmi.majesticlife.DataStructures.PlanCellViewModel;  
**import** com.farazannajmi.majesticlife.DataStructures.Quest;  
**import** com.farazannajmi.majesticlife.DataStructures.QuestViewModel;  
**import** com.farazannajmi.majesticlife.DataStructures.Skill;  
**import** com.farazannajmi.majesticlife.DataStructures.SkillViewModel;  
**import** com.farazannajmi.majesticlife.DataStructures.User;  
**import** com.farazannajmi.majesticlife.DataStructures.UserViewModel;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
*/\*\*  
 \* <div>Icons made by <a href="https://www.flaticon.com/authors/freepik" title="leaf">leaf</a> from <a href="https://www.flaticon.com/" title="Flaticon">www.flaticon.com</a> is licensed by <a href="http://creativecommons.org/licenses/by/3.0/" title="Creative Commons BY 3.0" target="\_blank">CC 3.0 BY</a></div>  
 \* https://www.flaticon.com/packs/user-interface-30/2  
 \*/***public class** DataHolder  
{  
 **public static int** *LevelUpReward* = 50;  
  
 **public static** User *ThisUser*;  
 **public static** BacktoryUser *CurrentBacktoryUser*;  
 **public static** ArrayList<Faaliat> *Faaliats*;  
 **public static** ArrayList<Skill> *Skills*;  
 **public static** ArrayList<PlanCell> *PlanCells*;  
 **public static** ArrayList<Quest> *Quests*;  
 **public static int** *FaaliatRepetitionsIdCounter*;  
 **public static** Avatar *UserAvatar*;  
 **public static** ArrayList<AvatarItem> *AvatarItems*;  
  
 **public static** ArrayList<Integer> *FaaliatAvatars*;  
 **public static** ArrayList<Integer> *SkillAvatars*;  
  
 **public static boolean** *IsDatabaseCreated* = **false**;  
  
 **public static void** InitialDataStructures()  
 {  
 *IsDatabaseCreated* = **false**;  
  
 *FaaliatAvatars* = **new** ArrayList<Integer>();  
 *FaaliatAvatars*.add(R.drawable.***ic\_majestic\_activities***);  
 *FaaliatAvatars*.add(R.drawable.***ic\_bag***);  
 *FaaliatAvatars*.add(R.drawable.***ic\_book***);  
 *FaaliatAvatars*.add(R.drawable.***ic\_camera***);  
 *FaaliatAvatars*.add(R.drawable.***ic\_computer***);  
 *FaaliatAvatars*.add(R.drawable.***ic\_conversation***);  
 *FaaliatAvatars*.add(R.drawable.***ic\_diamond***);  
 *FaaliatAvatars*.add(R.drawable.***ic\_edit***);  
 *FaaliatAvatars*.add(R.drawable.***ic\_eye***);  
 *FaaliatAvatars*.add(R.drawable.***ic\_flask***);  
 *FaaliatAvatars*.add(R.drawable.***ic\_gamepad***);  
 *FaaliatAvatars*.add(R.drawable.***ic\_idea***);  
 *FaaliatAvatars*.add(R.drawable.***ic\_leaf***);  
 *FaaliatAvatars*.add(R.drawable.***ic\_list***);  
 *FaaliatAvatars*.add(R.drawable.***ic\_music***);  
 *FaaliatAvatars*.add(R.drawable.***ic\_news***);  
 *FaaliatAvatars*.add(R.drawable.***ic\_paint***);  
 *FaaliatAvatars*.add(R.drawable.***ic\_real\_state***);  
 *FaaliatAvatars*.add(R.drawable.***ic\_rocket***);  
 *FaaliatAvatars*.add(R.drawable.***ic\_search***);  
 *FaaliatAvatars*.add(R.drawable.***ic\_speed***);  
 *FaaliatAvatars*.add(R.drawable.***ic\_target***);  
  
 *SkillAvatars* = **new** ArrayList<Integer>();  
 *SkillAvatars*.add(R.drawable.***ic\_skills***);  
 *SkillAvatars*.add(R.drawable.***ic\_bag***);  
 *SkillAvatars*.add(R.drawable.***ic\_book***);  
 *SkillAvatars*.add(R.drawable.***ic\_camera***);  
 *SkillAvatars*.add(R.drawable.***ic\_computer***);  
 *SkillAvatars*.add(R.drawable.***ic\_conversation***);  
 *SkillAvatars*.add(R.drawable.***ic\_diamond***);  
 *SkillAvatars*.add(R.drawable.***ic\_edit***);  
 *SkillAvatars*.add(R.drawable.***ic\_eye***);  
 *SkillAvatars*.add(R.drawable.***ic\_flask***);  
 *SkillAvatars*.add(R.drawable.***ic\_gamepad***);  
 *SkillAvatars*.add(R.drawable.***ic\_idea***);  
 *SkillAvatars*.add(R.drawable.***ic\_leaf***);  
 *SkillAvatars*.add(R.drawable.***ic\_list***);  
 *SkillAvatars*.add(R.drawable.***ic\_music***);  
 *SkillAvatars*.add(R.drawable.***ic\_news***);  
 *SkillAvatars*.add(R.drawable.***ic\_paint***);  
 *SkillAvatars*.add(R.drawable.***ic\_real\_state***);  
 *SkillAvatars*.add(R.drawable.***ic\_rocket***);  
 *SkillAvatars*.add(R.drawable.***ic\_search***);  
 *SkillAvatars*.add(R.drawable.***ic\_speed***);  
 *SkillAvatars*.add(R.drawable.***ic\_target***);  
 }  
  
 **public static void** LoadUser(FragmentActivity activity, LifecycleOwner owner)  
 {  
 Log.*d*(**"Data"**, **"Loading User from Database"**);  
UserViewModel userViewModel = ViewModelProviders.*of*(activity).get(UserViewModel.**class**);  
 userViewModel.getUser().observe(owner, **new** Observer<User>() {  
 @Override  
 **public void** onChanged(@Nullable User user) {  
 DataHolder.*ThisUser* = user;  
 }  
 });  
 }  
  
 **public static void** LoadData(FragmentActivity activity, LifecycleOwner owner)  
 {  
 Log.*d*(**"Data"**, **"Loading data from Database"**);  
  
 *//getting user:* UserViewModel userViewModel = ViewModelProviders.*of*(activity).get(UserViewModel.**class**);  
 userViewModel.getUser().observe(owner, **new** Observer<User>() {  
 @Override  
 **public void** onChanged(@Nullable User user) {  
 DataHolder.*ThisUser* = user;  
 }  
 });  
  
 *//getting Faaliats:* FaaliatViewModel faaliatViewModel = ViewModelProviders.*of*(activity).get(FaaliatViewModel.**class**);  
 faaliatViewModel.getAllFaaliats().observe(owner, **new** Observer<List<Faaliat>>() {  
 @Override  
 **public void** onChanged(@Nullable List<Faaliat> faaliats) {  
 DataHolder.*Faaliats* = (ArrayList) faaliats;  
 }  
 });  
  
 *//getting Skills:* SkillViewModel skillViewModel = ViewModelProviders.*of*(activity).get(SkillViewModel.**class**);  
 skillViewModel.getAllSkills().observe(owner, **new** Observer<List<Skill>>() {  
 @Override  
 **public void** onChanged(@Nullable List<Skill> skills) {  
 DataHolder.*Skills* = (ArrayList) skills;  
 }  
 });  
  
 *//getting PlanCells:* PlanCellViewModel planCellViewModel = ViewModelProviders.*of*(activity).get(PlanCellViewModel.**class**);  
 planCellViewModel.getAllPlanCells().observe(owner, **new** Observer<List<PlanCell>>() {  
 @Override  
 **public void** onChanged(@Nullable List<PlanCell> planCells) {  
 DataHolder.*PlanCells* = (ArrayList) planCells;  
 }  
 });  
  
 *//getting Quests:* QuestViewModel questViewModel = ViewModelProviders.*of*(activity).get(QuestViewModel.**class**);  
 questViewModel.getAllQuests().observe(owner, **new** Observer<List<Quest>>() {  
 @Override  
 **public void** onChanged(@Nullable List<Quest> quests) {  
 DataHolder.*Quests* = (ArrayList) quests;  
 }  
 });  
  
 *//getting FaaliatRepetitions* FaaliatRepetitionsViewModel faaliatRepetitionsViewModel = ViewModelProviders.*of*(activity).get(FaaliatRepetitionsViewModel.**class**);  
 faaliatRepetitionsViewModel.getAllFaaliatRepetitions().observe(owner, **new** Observer<List<FaaliatRepetitions>>() {  
 @Override  
 **public void** onChanged(@Nullable List<FaaliatRepetitions> faaliatRepetitions) {  
 *FaaliatRepetitionsIdCounter* = faaliatRepetitions.size();  
 }  
 });  
  
 *//getting AvatarItems* AvatarItemViewModel avatarItemViewModel = ViewModelProviders.*of*(activity).get(AvatarItemViewModel.**class**);  
 avatarItemViewModel.getAvatarItems().observe(owner, **new** Observer<List<AvatarItem>>() {  
 @Override  
 **public void** onChanged(@Nullable List<AvatarItem> avatarItems) {  
 *AvatarItems* = (ArrayList) avatarItems;  
 }  
 });  
  
 *//getting UserAvatar* AvatarViewModel avatarViewModel = ViewModelProviders.*of*(activity).get(AvatarViewModel.**class**);  
 avatarViewModel.getAvatar().observe(owner, **new** Observer<Avatar>() {  
 @Override  
 **public void** onChanged(@Nullable Avatar avatar) {  
 *UserAvatar* = avatar;  
 }  
 });  
 }  
  
 **public static void** LoadFaaliatSkills(FragmentActivity activity, LifecycleOwner owner)  
 {  
 FaaliatSkillViewModel faaliatSkillViewModel = ViewModelProviders.*of*(activity).get(FaaliatSkillViewModel.**class**);  
  
 **for**(**int** faaliatCounter = 0; faaliatCounter < *Faaliats*.size(); faaliatCounter++)  
 {  
 faaliatSkillViewModel.getSkillsForFaaliat(*Faaliats*.get(faaliatCounter)).observe(owner, **new** Observer<List<FaaliatSkill>>() {  
 @Override  
 **public void** onChanged(@Nullable List<FaaliatSkill> faaliatSkills)  
 {  
 **if**(faaliatSkills.size() != 0)  
 {  
 **for** (**int** i = 0; i < *Faaliats*.size(); i++)  
 {  
 **if** (*Faaliats*.get(i).getFaaliat\_ID() == faaliatSkills.get(0).getFaaliat\_ID())  
 *Faaliats*.get(i).**FaaliatSkills** = (ArrayList) faaliatSkills;  
 }  
 }  
 }  
 });  
 }  
 }  
}

کلاس مربوط به صفحه اصلی برنامه:

**package** com.farazannajmi.majesticlife;  
  
**import** android.arch.lifecycle.ViewModelProviders;  
**import** android.content.Context;  
**import** android.content.Intent;  
**import** android.content.SharedPreferences;  
**import** android.net.ConnectivityManager;  
**import** android.net.NetworkInfo;  
**import** android.support.v4.app.ActivityCompat;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.util.Log;  
**import** android.view.View;  
**import** android.widget.ImageView;  
**import** android.widget.ProgressBar;  
**import** android.widget.TextView;  
**import** android.widget.Toast;  
  
**import** com.backtory.java.internal.BacktoryCallBack;  
**import** com.backtory.java.internal.BacktoryResponse;  
**import** com.backtory.java.internal.BacktoryUser;  
**import** com.farazannajmi.majesticlife.AccountPackage.AccountManagementActivity;  
**import** com.farazannajmi.majesticlife.DataStructures.UserViewModel;  
**import** com.farazannajmi.majesticlife.FaaliatPackage.FaaliatsActivity;  
**import** com.farazannajmi.majesticlife.PlanPackage.PlanActivity;  
**import** com.farazannajmi.majesticlife.QuestPackage.QuestsActivity;  
**import** com.farazannajmi.majesticlife.SkillPackage.SkillsActivity;  
  
**public class** MainMenuActivity **extends** AppCompatActivity  
{  
 **public static** AppManager *TheAppManager*;  
  
 **public** TextView **Username\_txt**;  
 **public** TextView **Coins\_txt**;  
  
 **public** TextView **XpLevel\_txt**;  
 **public** TextView **HpLevel\_txt**;  
 **public** TextView **SpLevel\_txt**;  
 **public** ProgressBar **Xp\_progBar**;  
 **public** ProgressBar **Hp\_progBar**;  
 **public** ProgressBar **Sp\_progBar**;  
  
 **public** ImageView **Avatar\_back\_img**;  
 **public** ImageView **Avatar\_skin\_img**;  
 **public** ImageView **Avatar\_cloth\_img**;  
 **public** ImageView **Avatar\_eyes\_img**;  
 **public** ImageView **Avatar\_mouth\_img**;  
 **public** ImageView **Avatar\_crown\_img**;  
 **public** ImageView **Avatar\_hair\_img**;  
  
 **public static boolean** *FirstTime*;  
 **private boolean \_isConnected**;  
 **public static boolean** *GuestNotLoggedIn*;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState)  
 {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main\_menu***);  
  
 **final** AppCompatActivity activity = **this**;  
  
 *TheAppManager* = (AppManager) getApplicationContext();  
  
 *//region ----------- getting UI elements -----------* **Username\_txt** = (TextView) findViewById(R.id.***Main\_UserName\_txt***);  
 **Coins\_txt** = (TextView) findViewById(R.id.***Main\_Coins\_txt***);  
 **XpLevel\_txt** = findViewById(R.id.***Main\_XpLevel\_txt***);  
 **HpLevel\_txt** = findViewById(R.id.***Main\_HpLevel\_txt***);  
 **SpLevel\_txt** = findViewById(R.id.***Main\_SpLevel\_txt***);  
 **Xp\_progBar** = findViewById(R.id.***Main\_Xp\_progBar***);  
 **Hp\_progBar** = findViewById(R.id.***Main\_HP\_progBar***);  
 **Sp\_progBar** = findViewById(R.id.***Main\_SP\_progBar***);  
  
 **Avatar\_back\_img** = findViewById(R.id.***MainM\_Back\_img***);  
 **Avatar\_skin\_img** = findViewById(R.id.***MainM\_skin\_img***);  
 **Avatar\_cloth\_img** = findViewById(R.id.***MainM\_cloth\_img***);  
 **Avatar\_eyes\_img** = findViewById(R.id.***MainM\_eyes\_img***);  
 **Avatar\_mouth\_img** = findViewById(R.id.***MainM\_mouth\_img***);  
 **Avatar\_crown\_img** = findViewById(R.id.***MainM\_crown\_img***);  
 **Avatar\_hair\_img** = findViewById(R.id.***MainM\_hair\_img***);  
 *//endregion ---------------------------------  
  
 //for checking internet connection:* ConnectivityManager cm = (ConnectivityManager)getApplicationContext().getSystemService(Context.***CONNECTIVITY\_SERVICE***);  
 NetworkInfo activeNetwork = cm.getActiveNetworkInfo();  
 **\_isConnected** = activeNetwork != **null** && activeNetwork.isConnectedOrConnecting();  
  
 *//region ----------- load or initial data -----------* DataHolder.*LoadData*(**this**, **this**);  
 DataHolder.*LoadFaaliatSkills*(**this**, **this**);  
  
 *//if it is the first time app runs, the FirstTime value becomes true:* SharedPreferences sharedPref = getPreferences(Context.***MODE\_PRIVATE***);  
 *FirstTime* = sharedPref.getBoolean(**"FirstTime"**, **true**);  
 *GuestNotLoggedIn* = sharedPref.getBoolean(**"GuestNotLoggedIn"**, **true**);  
  
 **if**(*FirstTime*)  
 {  
 *//region first time* Log.*d*(**"WorkFlow"**, **"First Time running app."**);  
  
 **if**(**\_isConnected**)  
 {  
 Toast.*makeText*(getApplicationContext(), **"Connected to the Internet!"**, Toast.***LENGTH\_SHORT***).show();  
  
 *// Request a guest user from backtory:* BacktoryUser.*loginAsGuestInBackground*(**new** BacktoryCallBack<Void>() {  
 @Override  
 **public void** onResponse(BacktoryResponse<Void> response) {  
 **if** (response.isSuccessful()) {  
 *// Getting new username and password from CURRENT user* String newUsername = BacktoryUser.*getCurrentUser*().getUsername();  
 String newPassword = BacktoryUser.*getCurrentUser*().getGuestPassword();  
  
 Log.*d*(**"Backtory"**, **"Logged in as guest, username: "** + newUsername +  
 **" password: "** + newPassword);  
  
 *// saving user info in sharedPreference:* SharedPreferences sharedPref = getPreferences(Context.***MODE\_PRIVATE***);  
 SharedPreferences.Editor editor = sharedPref.edit();  
 editor.putBoolean(**"GuestNotLoggedIn"**, **false**);  
 editor.putString(**"UserName"**, newUsername);  
 editor.putString(**"UserPassword"**, newPassword);  
 editor.commit();  
  
 *//saving username in database:* DataHolder.*ThisUser*.setUsername(newUsername);  
 UserViewModel userViewModel = ViewModelProviders.*of*(activity).get(UserViewModel.**class**);  
 userViewModel.update(DataHolder.*ThisUser*);  
  
 DataHolder.*CurrentBacktoryUser* = BacktoryUser.*getCurrentUser*();  
 } **else** {  
 Log.*d*(**"Backtory"**, **"Failed log in as guest: "** + response.code());  
 }  
 }  
 });  
 }  
 **else** *//if is not connected to internet* {Toast.*makeText*(getApplicationContext(), **"No internet connection!"**, Toast.***LENGTH\_SHORT***).show();  
  
 SharedPreferences.Editor editor = sharedPref.edit();  
 editor.putBoolean(**"GuestNotLoggedIn"**, **true**);  
 editor.putString(**"UserName"**, **"NewKing"**);  
 editor.putString(**"UserPassword"**, **"1111"**);  
 editor.commit();  
 }  
  
 SharedPreferences.Editor editor = sharedPref.edit();  
 editor.putBoolean(**"FirstTime"**, **false**);  
 editor.commit();  
 *//endregion* }  
 **else** *//not the first time opening the application* {  
 *//region not first time* Log.*d*(**"WorkFlow"**, **"Not the first Time running app."**);  
  
 **if**(**\_isConnected**)  
 {  
 **if**(*GuestNotLoggedIn*)  
 {  
 *// Request a guest user from backtory:* BacktoryUser.*loginAsGuestInBackground*(**new** BacktoryCallBack<Void>() {  
 @Override  
 **public void** onResponse(BacktoryResponse<Void> response) {  
 **if** (response.isSuccessful()) {  
 *// Getting new username and password from CURRENT user* String newUsername = BacktoryUser.*getCurrentUser*().getUsername();  
 String newPassword = BacktoryUser.*getCurrentUser*().getGuestPassword();  
  
 Log.*d*(**"Backtory"**, **"Logged in as guest, username: "** + newUsername +  
 **" password: "** + newPassword);  
  
 *// saving user info in sharedPreference:* SharedPreferences sharedPref = getPreferences(Context.***MODE\_PRIVATE***);  
 SharedPreferences.Editor editor = sharedPref.edit();  
 editor.putBoolean(**"GuestNotLoggedIn"**, **false**);  
 editor.putString(**"UserName"**, newUsername);  
 editor.putString(**"UserPassword"**, newPassword);  
 editor.commit();  
  
 *//saving username in database:* DataHolder.*ThisUser*.setUsername(newUsername);  
 UserViewModel userViewModel = ViewModelProviders.*of*(activity).get(UserViewModel.**class**);  
 userViewModel.update(DataHolder.*ThisUser*);  
  
 DataHolder.*CurrentBacktoryUser* = BacktoryUser.*getCurrentUser*();  
 } **else** {  
 Log.*d*(**"Backtory"**, **"Failed log in as guest: "** + response.code());  
 }  
 }  
 });  
 }  
 **else** *//guest has been logged in* {  
 *// loading username and password from SharedPref* String username = sharedPref.getString(**"UserName"**, **""**);  
 String password = sharedPref.getString(**"UserPassword"**, **""**);  
  
 Log.*d*(**"WorkFlow"**, **"Username: "** + username);  
}  
 }  
 **else** *//if not connected to internet* {Toast.*makeText*(getApplicationContext(), **"No internet connection!"**, Toast.***LENGTH\_SHORT***).show();  
  
 *// loading username and password from SharedPref* String username = sharedPref.getString(**"UserName"**, **""**);  
 String password = sharedPref.getString(**"UserPassword"**, **""**);  
 Log.*d*(**"WorkFlow"**, **"Username: "** + username);  
  
 SharedPreferences.Editor editor = sharedPref.edit();  
 editor.putBoolean(**"GuestNotLoggedIn"**, **true**);  
 editor.commit();  
 }  
  
 DataHolder.*LoadData*(**this**, **this**);  
 *//endregion* }  
 *//endregion ---------------------------------* InitialUIElements();  
 }  
  
 @Override  
 **protected void** onResume()  
 {  
 **super**.onResume();  
 InitialUIElements();  
 }  
  
 @Override  
 **public void** onBackPressed()  
 {  
 **try** {  
 finishAffinity();  
 }  
 **catch** (Exception e)  
 {  
 ActivityCompat.*finishAffinity*(MainMenuActivity.**this**);  
 }  
 }  
  
 **public void** UiElementsOnClick(View view)  
 {  
 **switch** (view.getId())  
 {  
 **case** R.id.***Main\_UserAvatar\_img***:  
 {  
 Intent AccountIntent = **new** Intent(MainMenuActivity.**this**, AccountManagementActivity.**class**);  
 startActivity(AccountIntent);  
 **break**;  
 }  
 **case** R.id.***Main\_Skills\_btn***:  
 {  
 Intent skillsIntent = **new** Intent(MainMenuActivity.**this**, SkillsActivity.**class**);  
 startActivity(skillsIntent);  
 **break**;  
 }  
 **case** R.id.***Main\_Faaliat\_btn***:  
 {  
 Intent activitiesIntent = **new** Intent(MainMenuActivity.**this**, FaaliatsActivity.**class**);  
 startActivity(activitiesIntent);  
 **break**;  
 }  
 **case** R.id.***Main\_Plan\_btn***:  
 {  
 Intent planIntent = **new** Intent(MainMenuActivity.**this**, PlanActivity.**class**);  
 startActivity(planIntent);  
 **break**;  
 }  
 **case** R.id.***Main\_Quests\_btn***:  
 {  
 Intent questsIntent = **new** Intent(MainMenuActivity.**this**, QuestsActivity.**class**);  
 startActivity(questsIntent);  
 **break**;  
 }  
 **case** R.id.***Main\_Shop\_btn***:  
 {  
 Intent shopIntent = **new** Intent(MainMenuActivity.**this**, ShopActivity.**class**);  
 startActivity(shopIntent);  
**break**;  
 }  
 **default**:  
 **break**;  
 }  
 }  
  
 **public void** InitialUIElements()  
 {  
 **Username\_txt**.setText(DataHolder.*ThisUser*.getUsername());  
 **Coins\_txt**.setText(Integer.*toString*(DataHolder.*ThisUser*.getCoins()));  
  
 **XpLevel\_txt**.setText(Integer.*toString*(DataHolder.*ThisUser*.getXpLevel()));  
 **HpLevel\_txt**.setText(Integer.*toString*(DataHolder.*ThisUser*.getHpLevel()));  
 **SpLevel\_txt**.setText(Integer.*toString*(DataHolder.*ThisUser*.getSpLevel()));  
 **Xp\_progBar**.setProgress(DataHolder.*ThisUser*.getXP());  
 **Hp\_progBar**.setProgress(DataHolder.*ThisUser*.getHP());  
 **Sp\_progBar**.setProgress(DataHolder.*ThisUser*.getSP());  
  
 *//setting user avatar:* **Avatar\_back\_img**.setImageResource(DataHolder.*UserAvatar*.getBackground\_ResIndex());  
 **Avatar\_skin\_img**.setImageResource(DataHolder.*UserAvatar*.getSkin\_ResIndex());  
 **Avatar\_cloth\_img**.setImageResource(DataHolder.*UserAvatar*.getCloth\_ResIndex());  
 **Avatar\_eyes\_img**.setImageResource(DataHolder.*UserAvatar*.getEyes\_ResIndex());  
 **Avatar\_mouth\_img**.setImageResource(DataHolder.*UserAvatar*.getMouth\_ResIndex());  
 **Avatar\_crown\_img**.setImageResource(DataHolder.*UserAvatar*.getCrown\_ResIndex());  
 **if**(DataHolder.*UserAvatar*.getIsKing())  
 {  
 **Avatar\_hair\_img**.setVisibility(View.***INVISIBLE***);  
 }  
 **else** {  
 **Avatar\_hair\_img**.setVisibility(View.***VISIBLE***);  
 }  
 }  
}

کلاس صفحه مهارت‌ها که شامل لیستی از مهارت‌هاست که با استفاده از ArrayAdaptorای که نوشته شده، مقداردهی می‌شود:

**package** com.farazannajmi.majesticlife.SkillPackage;  
  
**import** android.arch.lifecycle.ViewModelProviders;  
**import** android.content.Context;  
**import** android.content.Intent;  
**import** android.support.v7.app.AppCompatActivity;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.GridView;  
  
**import** com.farazannajmi.majesticlife.DataHolder;  
**import** com.farazannajmi.majesticlife.DataStructures.Skill;  
**import** com.farazannajmi.majesticlife.DataStructures.SkillViewModel;  
**import** com.farazannajmi.majesticlife.R;  
  
**public class** SkillsActivity **extends** AppCompatActivity  
{  
 **public static** AppCompatActivity *appCompatActivity*;  
 **public static** Context *context*;  
 **public static** ArrayAdapter<Skill> *skill\_gridview\_adapter*;  
  
 **private** GridView **skills\_gridview**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState)  
 {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_skills***);  
 *context* = getApplicationContext();  
 *appCompatActivity* = **this**;  
  
 *//getting UI elements* **skills\_gridview** = (GridView) findViewById(R.id.***SkillsActivity\_gridview***);  
  
 *//create our new array adapter  
 skill\_gridview\_adapter* = **new** SkillsListItemArrayAdapter(**this**, DataHolder.*Skills*);  
 **skills\_gridview**.setAdapter(*skill\_gridview\_adapter*);  
 }  
  
 **public void** UiElementsOnClick(View view)  
 {  
 **if** (view.getId() == R.id.***SkillsActivity\_addSkill\_btn***)  
 {  
 Skill newSkill = **new** Skill(DataHolder.*Skills*.size(), **"new skill"**, DataHolder.*SkillAvatars*.get(0),1,0, DataHolder.*ThisUser*.getUser\_ID());  
 DataHolder.*Skills*.add(newSkill);  
  
 *//adding skill in database:* SkillViewModel skillViewModel = ViewModelProviders.*of*(**this**).get(SkillViewModel.**class**);  
 skillViewModel.insert(newSkill);  
  
 *//opening new activity for editing this new faaliat:* Intent intent = **new** Intent(SkillsActivity.**this**, EditSkillActivity.**class**);  
 intent.putExtra(**"The\_Skill"**, DataHolder.*Skills*.size() - 1);  
 startActivity(intent);  
 }  
 }  
  
 **public static void** DeleteSkill(**int** skillIndex)  
 {  
 *//deleting skill from database:* SkillViewModel skillViewModel = ViewModelProviders.*of*(*appCompatActivity*).get(SkillViewModel.**class**);  
 skillViewModel.delete(DataHolder.*Skills*.get(skillIndex));  
  
 DataHolder.*Skills*.remove(skillIndex);  
 *skill\_gridview\_adapter*.notifyDataSetChanged();  
 }  
}

کلاس SkillsListItemArrayAdapter که برای مقداردهی لیست مهارت‌ها استفاده شده است:

**package** com.farazannajmi.majesticlife.SkillPackage;  
  
**import** android.animation.ObjectAnimator;  
**import** android.content.Context;  
**import** android.content.DialogInterface;  
**import** android.content.Intent;  
**import** android.support.annotation.NonNull;  
**import** android.support.v7.app.AlertDialog;  
**import** android.view.LayoutInflater;  
**import** android.view.View;  
**import** android.view.ViewGroup;  
**import** android.view.animation.DecelerateInterpolator;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.Button;  
**import** android.widget.ImageView;  
**import** android.widget.ProgressBar;  
**import** android.widget.TextView;  
**import** android.widget.Toast;  
  
**import** com.farazannajmi.majesticlife.DataStructures.Skill;  
**import** com.farazannajmi.majesticlife.FaaliatPackage.FaaliatsActivity;  
**import** com.farazannajmi.majesticlife.R;  
  
**import** java.util.ArrayList;**public class** SkillsListItemArrayAdapter **extends** ArrayAdapter<Skill>  
{  
 **private** Context **context**;  
 **private** ArrayList<Skill> **skillsList**;  
  
 **public** SkillsListItemArrayAdapter(@NonNull Context context, @NonNull ArrayList<Skill> objects)  
 {  
 **super**(context, R.layout.***listitem\_faaliat***, objects);  
  
 **this**.**context** = context;  
 **this**.**skillsList** = objects;  
 }  
  
 *//called when rendering the list* **public** View getView(**final int** position, View convertView, ViewGroup parent)  
 {  
 *//get the property we are displaying* Skill skill = getItem(position);  
  
 *//get the inflater and inflate the XML layout for each item* LayoutInflater inflater = LayoutInflater.*from*(getContext());  
 convertView = inflater.inflate(R.layout.***listitem\_skill***, parent, **false**);  
  
 TextView name\_txt = (TextView) convertView.findViewById(R.id.***listItem\_skill\_name\_txt***);  
 TextView level\_txt = (TextView) convertView.findViewById(R.id.***listItem\_skill\_level\_txt***);  
 ImageView avatar\_img = (ImageView) convertView.findViewById(R.id.***listItem\_skill\_avatar\_img***);  
 ProgressBar progress\_Bar = (ProgressBar) convertView.findViewById(R.id.***listItem\_skill\_progressBar***);  
 Button edit\_btn = (Button) convertView.findViewById(R.id.***listItem\_skill\_edit\_btn***);Button delete\_btn = (Button) convertView.findViewById(R.id.***listItem\_skill\_delete\_btn***);  
  
 *//setting info in UI:* name\_txt.setText(skill.getSkill\_Name());  
 level\_txt.setText(Integer.*toString*(skill.getLevel()));  
  
 progress\_Bar.setMax(100);  
 progress\_Bar.setProgress(skill.getProgress());  
 ObjectAnimator animation = ObjectAnimator.*ofInt*(progress\_Bar, **"progress"**, 0, skill.getProgress()); *// see this max value coming back here, we animate towards that value* animation.setDuration(1000); *// in milliseconds* animation.setInterpolator(**new** DecelerateInterpolator());  
 animation.start();  
  
 *//get the image associated with this property* avatar\_img.setImageResource(skill.getAvatar\_ResIndex());  
  
 edit\_btn.setOnClickListener(**new** View.OnClickListener()  
 {  
 @Override  
 **public void** onClick(View v)  
 {  
 *//opening new activity for editing this faaliat:* Intent intent = **new** Intent(getContext(), EditSkillActivity.**class**);  
 intent.putExtra(**"The\_Skill"**, position);  
 **context**.startActivity(intent);  
 }  
 });delete\_btn.setOnClickListener(**new** View.OnClickListener()  
 {  
 @Override  
 **public void** onClick(View v)  
 {  
 *//show are you sure dialogue popup:* AlertDialog.Builder builder = **new** AlertDialog.Builder(**context**);  
 *// Add action buttons* builder.setPositiveButton(R.string.***yes***, **new** DialogInterface.OnClickListener()  
 {  
 @Override  
 **public void** onClick(DialogInterface dialog, **int** id)  
 {  
 SkillsActivity.*DeleteSkill*(position);  
 Toast.*makeText*(**context**, **"Skill deleted successfully."**, Toast.***LENGTH\_SHORT***).show();  
 dialog.dismiss();  
 }  
 });  
 builder.setNegativeButton(R.string.***no***, **new** DialogInterface.OnClickListener()  
 {  
 **public void** onClick(DialogInterface dialog, **int** id)  
 {  
 dialog.cancel();  
 }  
 });  
  
 builder.setTitle(**"Are you sure you want to delete this skill?"**);  
  
 builder.create();  
 builder.show();  
 }  
 });  
  
 **return** convertView;  
 }  
}

کلاس SignupDialogFragment که هنگامی که کاربر بر روی دکمه ایجاد حساب (Signup) کلیک می‌کند، این Dialog نشان داده می‌شود تا کاربر نام کاربری، ایمیل و گذرواژه خود را وارد کند:

**package** com.farazannajmi.majesticlife.AccountPackage;  
  
**import** android.app.Activity;  
**import** android.app.Dialog;  
**import** android.app.DialogFragment;  
**import** android.content.DialogInterface;  
**import** android.os.Bundle;  
**import** android.support.v7.app.AlertDialog;  
**import** android.view.LayoutInflater;  
**import** android.widget.EditText;  
  
**import** com.farazannajmi.majesticlife.R;**public class** SignupDialogFragment **extends** DialogFragment  
{  
 **private** EditText **Username\_editText**;  
 **private** EditText **Email\_editText**;  
 **private** EditText **Password\_editText**;  
  
 */\* The activity that creates an instance of this dialog fragment must  
 \* implement this interface in order to receive event callbacks.  
 \* Each method passes the DialogFragment in case the host needs to query it. \*/* **public interface** SignupDialogListener  
 {  
 **public void** onDialogPositiveClick(DialogFragment dialog, String username, String email, String password);  
 *//public void onDialogNegativeClick(DialogFragment dialog);* }  
  
 *// Use this instance of the interface to deliver action events* SignupDialogListener **mListener**;  
  
 *// Override the Fragment.onAttach() method to instantiate the NoticeDialogListener* @Override  
 **public void** onAttach(Activity activity)  
 {  
 **super**.onAttach(activity);  
  
 *// Verify that the host activity implements the callback interface* **try** {  
 *// Instantiate the NoticeDialogListener so we can send events to the host* **mListener** = (SignupDialogListener) activity;  
 }  
 **catch** (ClassCastException e) {  
 *// The activity doesn't implement the interface, throw exception* **throw new** ClassCastException(activity.toString()  
 + **" must implement SignupDialogListener"**);  
 }  
 }  
  
 @Override  
 **public** Dialog onCreateDialog(Bundle savedInstanceState)  
 {  
 **Username\_editText** = **new** EditText(getActivity());  
  
 AlertDialog.Builder builder = **new** AlertDialog.Builder(getActivity());  
  
 *// Get the layout inflater* LayoutInflater inflater = getActivity().getLayoutInflater();  
  
 *// Inflate and set the layout for the dialog  
 // Pass null as the parent view because its going in the dialog layout* builder.setView(inflater.inflate(R.layout.***dialog\_signup***, **null**));  
 *// Add action buttons* builder.setPositiveButton(R.string.***sign\_up***, **new** DialogInterface.OnClickListener()  
 {  
 @Override  
 **public void** onClick(DialogInterface dialog, **int** id)  
 {  
 **Username\_editText** = getDialog().findViewById(R.id.***SignupD\_username\_editText***);  
 **Email\_editText** = getDialog().findViewById(R.id.***SignupD\_email\_editText***);  
 **Password\_editText** = getDialog().findViewById(R.id.***SignupD\_password\_editText***);  
  
**mListener**.onDialogPositiveClick(SignupDialogFragment.**this**,  
 **Username\_editText**.getText().toString(),  
 **Email\_editText**.getText().toString(),  
 **Password\_editText**.getText().toString());  
 }  
 });  
 builder.setNegativeButton(R.string.***cancel***, **new** DialogInterface.OnClickListener()  
 {  
 **public void** onClick(DialogInterface dialog, **int** id)  
 {  
 SignupDialogFragment.**this**.getDialog().cancel();  
}  
 });  
 builder.setTitle(**"Please enter your info for sign up."**);  
  
 *// Create the AlertDialog object and return it* **return** builder.create();  
 }  
}

**Abstract:**

This application is about gamifying daily activities and improving in your skills. By level upping in your skills you earn some coins that you can buy avatar customization items with them.

**Keywords:** Gamification, life style, android, mobile application, planning



Technical and Vocational University

Shariaty Technical College

Life style and planning application using gamification for Android OS

A Thesis Submitted in Partial Fulfillment of the Requirement for the Degree of Bachelor of Science in Software Engineering

By:

Narges Farazan, Zahra Najmi

Supervisor:

Dr. Iman Sharifi

July 2018