

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 است با استفاده از مفاهیم gamification (بازی‌انگاری) کاربر را به انجام فعالیت‌های مفید ترغیب کنیم. 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 پیشرفتش را مشاهده کند.

‌

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

فهرست مطالب

[فصل 1: مقدمه 1](#_Toc518758594)

[1-1- انگيزه 2](#_Toc518758595)

[1-2- هدف 2](#_Toc518758596)

[1-3- معرفی ابزارها و بستر برنامه نویسی 3](#_Toc518758597)

[1-3-1- معرفی اجمالی اندروید استودیو و زبان برنامه‌نویسی جاوا 3](#_Toc518758598)

[1-4- نگاهی به فصل‌های آینده 4](#_Toc518758599)

[فصل 2: تجزيه و تحليل نيازمندي‌ها 5](#_Toc518758600)

[2-1- مقدمه 6](#_Toc518758601)

[2-2- معرفی کاربران برنامه 6](#_Toc518758602)

[2-3- رابط گرافیکی نرم‌افزار و معرفی صفحات نرم‌افزار 6](#_Toc518758603)

[2-3-1- صفحه‌‌‌‌‌‌ی اصلی 7](#_Toc518758604)

[2-3-2- صفحه‌ی نمایش اطلاعات کاربر 8](#_Toc518758605)

[2-3-3- صفحه‌ی مهارت‌ها 9](#_Toc518758606)

[2-3-4- صفحه‌ی ویرایش مهارت‌‌ها 10](#_Toc518758607)

[2-3-5- صفحه‌ی فعالیت‌ها 11](#_Toc518758608)

[2-3-6- صفحه‌ی ویرایش فعالیت‌ 14](#_Toc518758609)

[فصل 3: ساختار داده‌ها و بانک اطلاعات 16](#_Toc518758610)

[3-1- مقدمه 17](#_Toc518758611)

[3-2- نمودارهای UML 17](#_Toc518758612)

[3-2-1- نمودار Use Case 18](#_Toc518758613)

[3-2-2- نمودار توالی 19](#_Toc518758614)

[3-3- سخت‌افزارهای مورد نیاز 24](#_Toc518758615)

[فصل 4: جدول‌ها و بانک اطلاعاتی 25](#_Toc518758616)

[4-1- مقدمه 26](#_Toc518758617)

[4-2- پایگاه‌ داده 26](#_Toc518758618)

[فصل 5: جمع‌‌بندي و پيشنهاد‌ها 29](#_Toc518758619)

[5-1- نتيجه‌گيري 30](#_Toc518758620)

[5-2- پيشنهادهايي براي کارهاي آتي 30](#_Toc518758621)

[مراجع 31](#_Toc518758622)

[پيوست‌ها 33](#_Toc518758623)

**فهرست شکل‌ها**

[شکل (2-1) صفحه اصلی برنامه 7](#_Toc518564241)

[شکل (2-2) صفحه ی نمایش اطلاعات کاربر 8](#_Toc518564242)

[شکل (2-3) صفحه مهارت ها 9](#_Toc518564243)

[شکل (2-4) صفحه ویرایش مهارت ها 10](#_Toc518564244)

[شکل (2-5) انتخاب عکس جدید برای مهارت 11](#_Toc518564245)

[شکل (2-6) صفحه فعالیت ها 12](#_Toc518564246)

[شکل (2-7) نمودار میزان انجام فعالیت 13](#_Toc518564247)

[شکل (2-8) دیالوگ انتخاب میزان ساعات انجام فعالیت 13](#_Toc518564248)

[شکل (2-9) دیالوگ میزان پیشرفت کاربر 14](#_Toc518564249)

[شکل (2-10) صفحه ویرایش فعالیت 15](#_Toc518564250)

[شکل (3-1) نمودار Use Case 18](#_Toc518564251)

[شکل (3-2) نمودارهای توالی Quests و Plan 19](#_Toc518564252)

[شکل (3-3) نمودارهای توالی View Faaliats و View Skills 20](#_Toc518564253)

[شکل (3-4) نمودارهای توالی Edit Faaliat و Add new Faaliat 21](#_Toc518564254)

[شکل (3-5) نمودارهای توالی Delete Faaliat و View Faaliat Graph 22](#_Toc518564255)

[شکل (3-6) نمودارهای توالی Edit Skill و Add New Skill 23](#_Toc518564256)

[شکل (3-7) نمودار توالی Delete Skill 24](#_Toc518564257)

[شکل (4-1) نمودار ER 27](#_Toc518564258)

[شکل (5-1) زيرنويس شکل 30](#_Toc518564259)

فهرست جدول­ها

[جدول (4-1) جدول User 28](#_Toc518825369)

[جدول (4-2) جدول Skill 29](#_Toc518825370)

[جدول (4-3) جدول Faaliat 29](#_Toc518825371)

[جدول (4-4) جدول Quest 30](#_Toc518825372)

[جدول (4-5) جدول PlanCell 30](#_Toc518825373)

[جدول (4-6) جدول Quest\_Skill 30](#_Toc518825374)

[جدول (4-7) جدول Faaliat\_Skill 31](#_Toc518825375)

[جدول (4-8) جدول FaaliatRepetitions 31](#_Toc518825376)

[جدول (4-9) جدول Avatar 32](#_Toc518825377)

[جدول (4-10) جدول AvatarItem 32](#_Toc518825378)

[جدول (5-1) بالانويس جدول 35](#_Toc518825379)

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 رقبت بیشتری انجام دهد. بنابراین تصمیم گرفته شد که با ترکیب این دو مقوله یعنی بهبود سبک زندگی و بازی‌انگاری یک اپلیکیشن 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 با کاربر 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"هایی را برای خود تعریف میکند و نیز "فعالیت‌"هایی که با انتساب توانای‌های خود به این فعالیت ها 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 سکه ها 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 برنامه نویسی 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، نمودار فعالیت، کاربرد، توالي، کلاس و غيره در این فصل شرح داده شده­اند.

* 1. معرفی کاربران برنامه

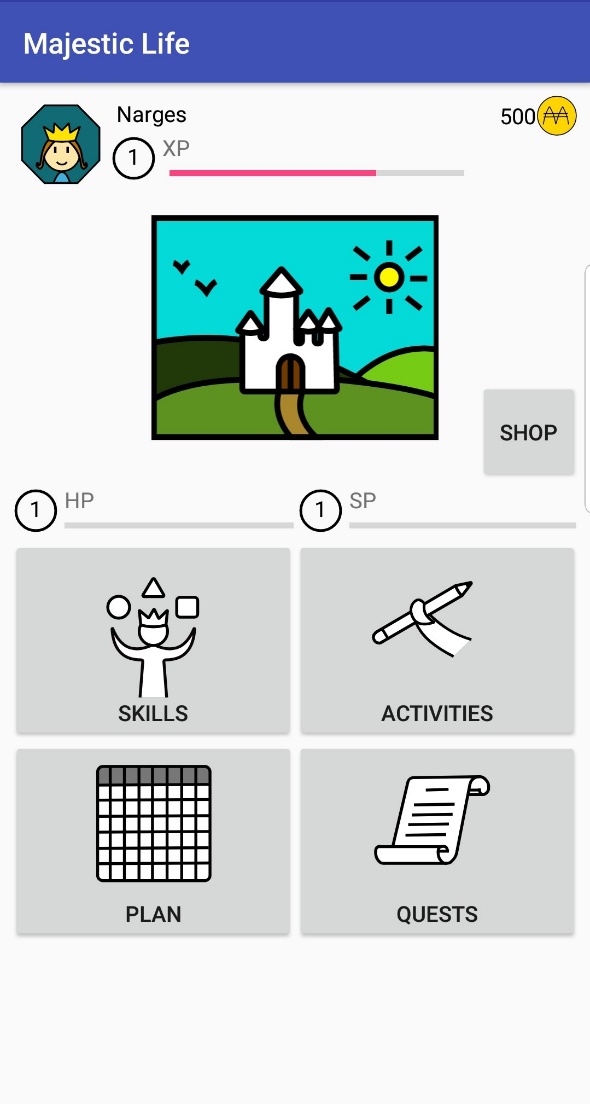
کاربران این برنامه هر طیف و سنی از جامعه میتوانند باشد و محدودیتی برای بازه­ی کاربران این نرم­افزار وجود ندارد. هر شخصی که به دنبال برنامه‌ریزی فعالیت های روزانه خود است میتواند از این برنامه استفاده کند.

* 1. رابط گرافیکی نرم‌افزار و معرفی صفحات نرم‌افزار

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

* + 1. صفحه‌‌‌‌‌‌ی اصلی

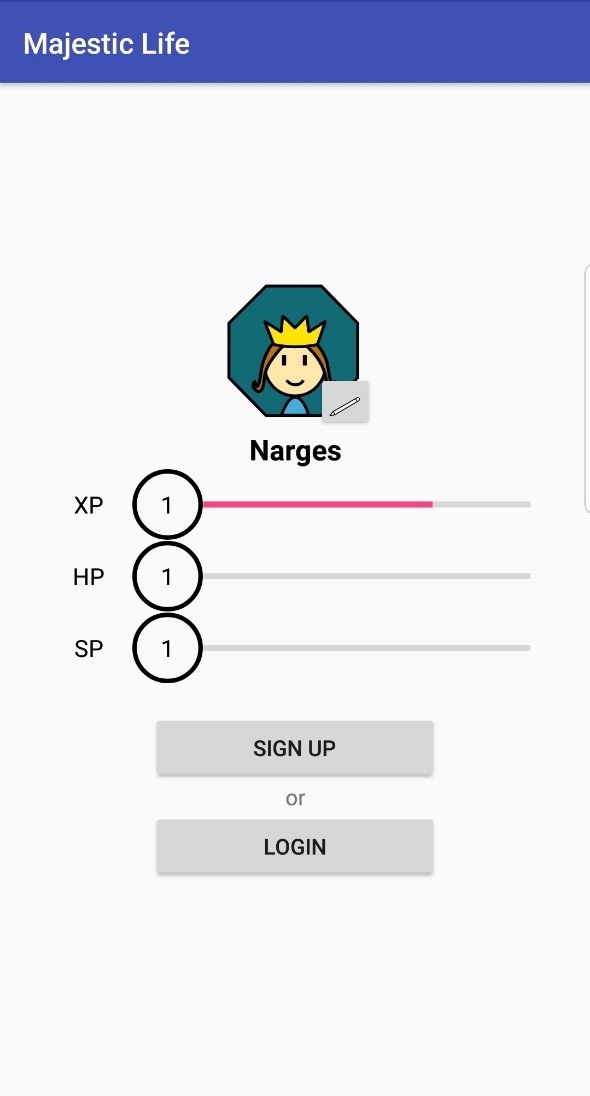
این صفحه، صفحه اصلی و اولیه برنامه می‌باشد. در اینجا، کاربر 4 دکمه اصلی برنامه که به صفحات "فعالیت‌ها"، "مهارت‌ها"، "برنامه" و "ماموریت‌ها" مسیریابی می‌شوند، مشاهده می‌کند. علاوه بر آن، progress barهایی نیز در این صفحه برای نمایش مقدار و سطح XP، HP و SP کاربر وجود دارند. در بالا سمت راست صفحه نیز، تعداد 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

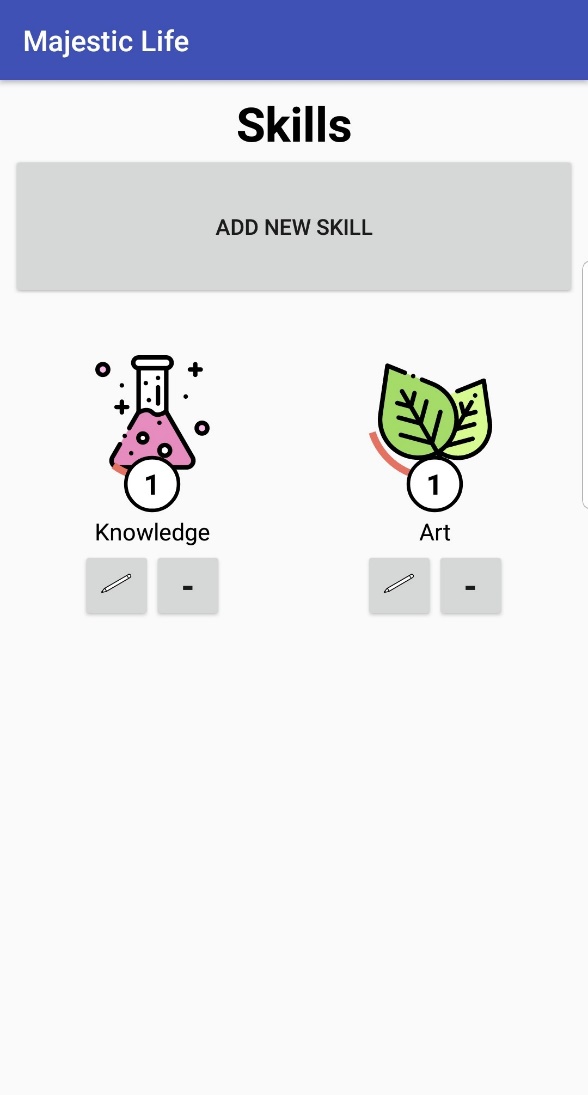
* + 1. صفحه‌ی نمایش اطلاعات کاربر

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، کاربر به صفحه فروشگاه آدمک منتقل می‌شود.



صفحه ی نمایش اطلاعات کاربر

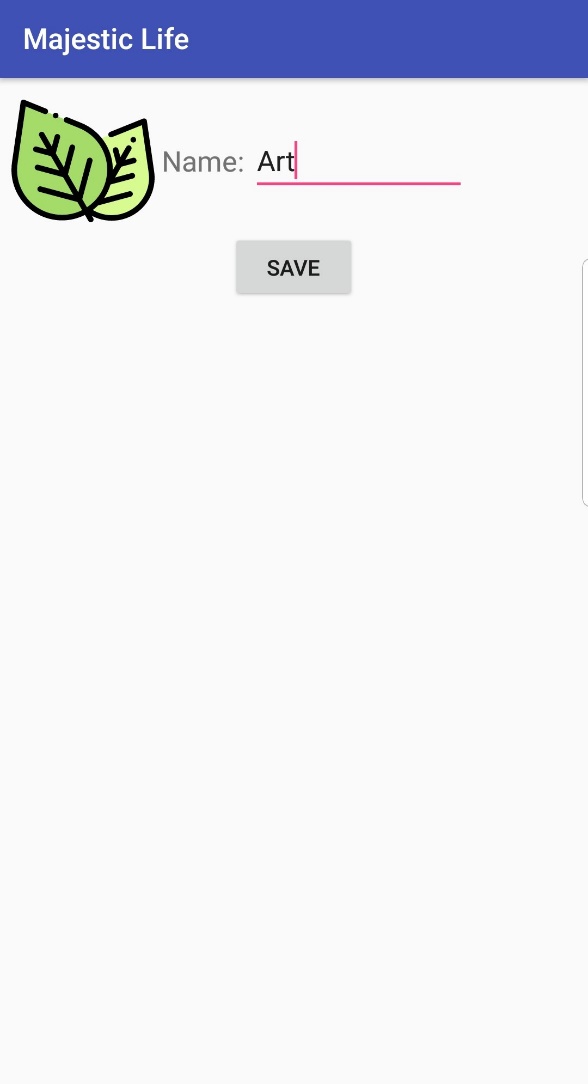
* + 1. صفحه‌ی مهارت‌­ها

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.

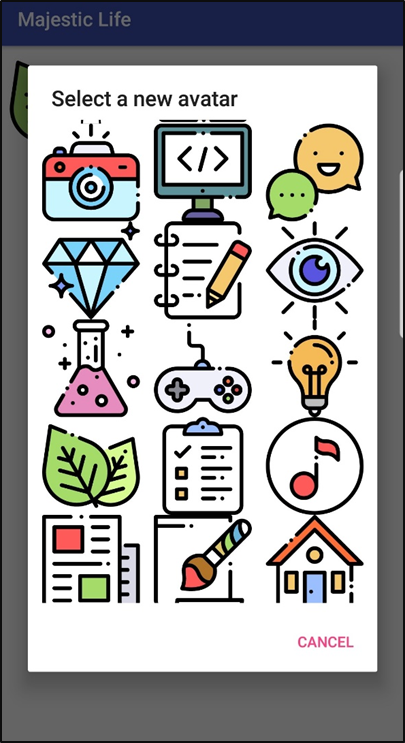
صفحه مهارت ها

* + 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

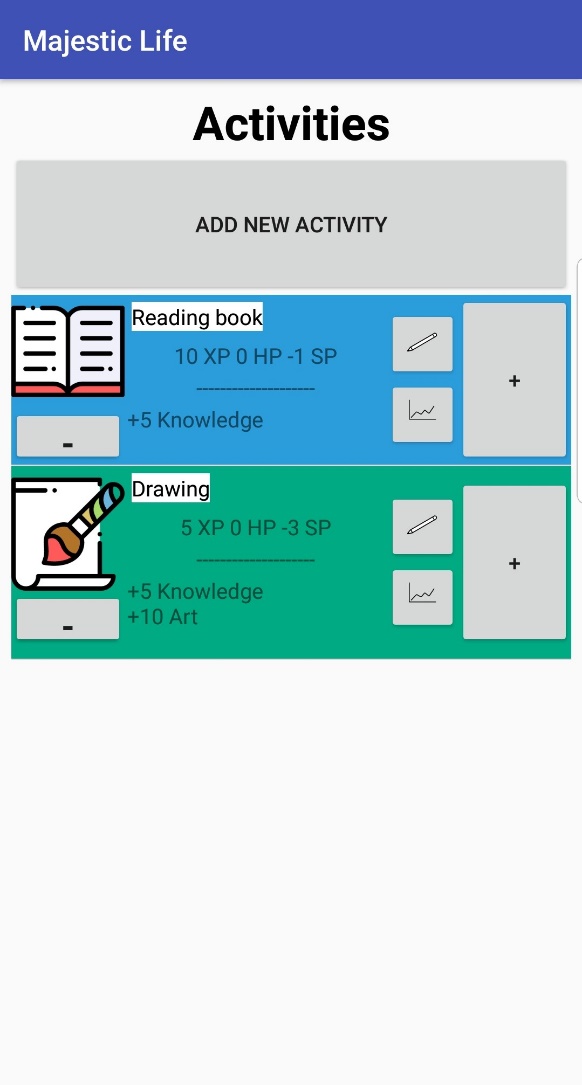


انتخاب عکس جدید برای مهارت

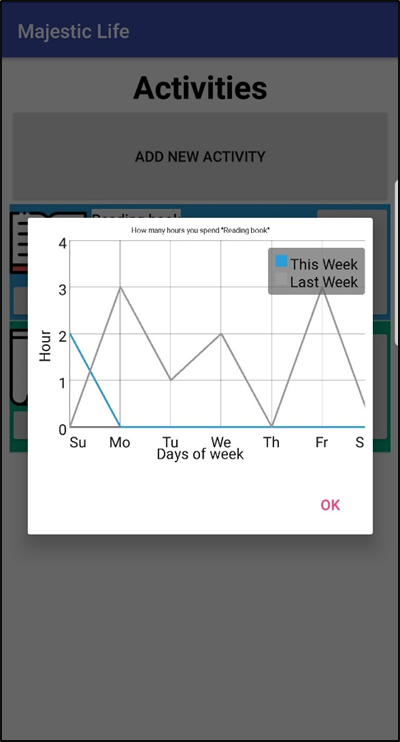
* + 1. صفحه‌ی فعالیت‌­ها

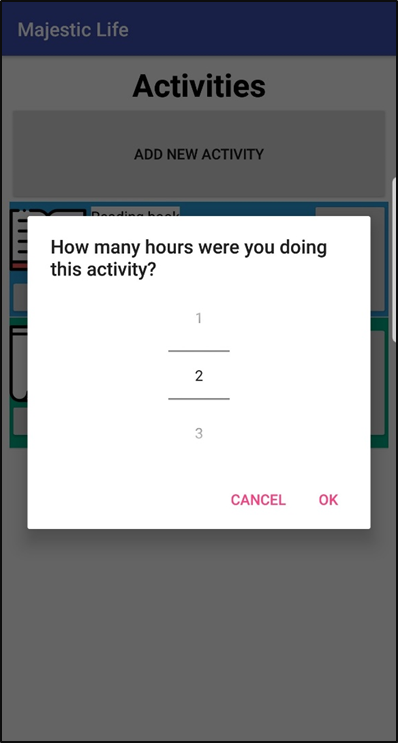
در این صفحه، کاربر لیستی از فعالیت­های خود(Faaliat or Activities1) را مشاهده می­کند که برای هر کدام از آن‌ها نیز دکمه‌های حذف، ویرایش، نمودار و پیشرفت(+) آن فعالیت قرار دارد و هر کدام رنگ و تصویر خاص خود را دارند. در بالای صفحه دکمه­ی اضافه کردن فعالیت جدید(Add New Activity) وجود دارد که کاربر با کلیک 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 را در روزهای مختلف 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 ساعت و زدن دکمه OK، دوباره Dialog ای باز میشود و میزان پیشرفت کاربر را در هر کدام از مشخصه های مربوطه و مهارت ها نشان داده میشود.

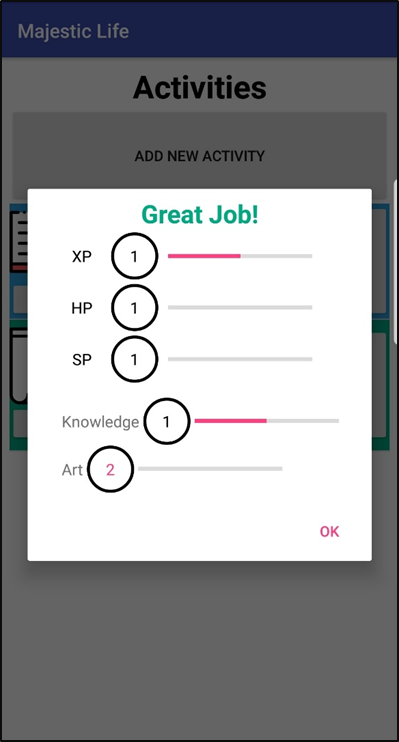


صفحه فعالیت ها

نمودار میزان انجام فعالیت



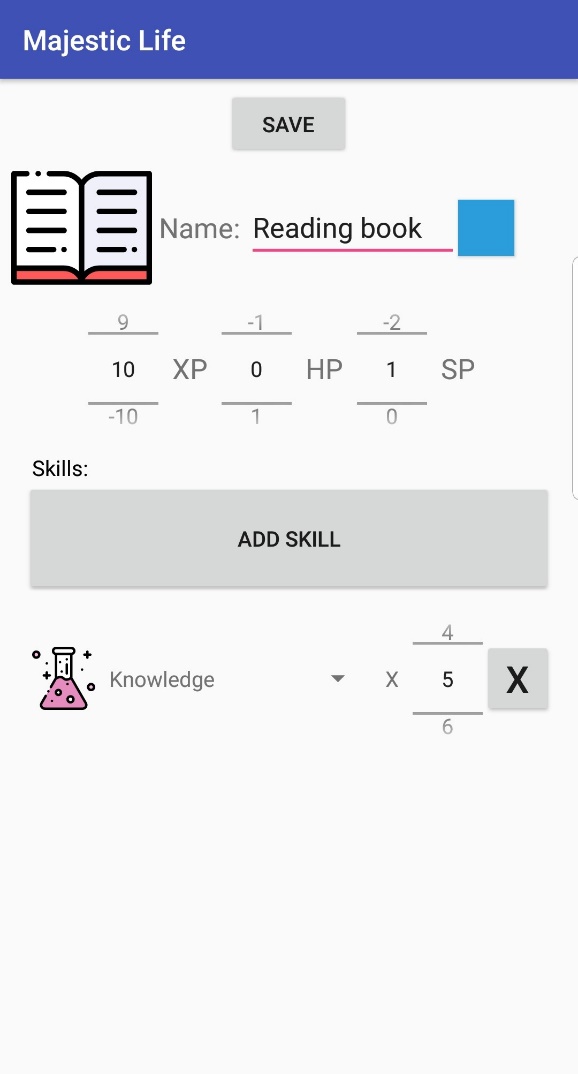
دیالوگ انتخاب میزان ساعات انجام فعالیت

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.



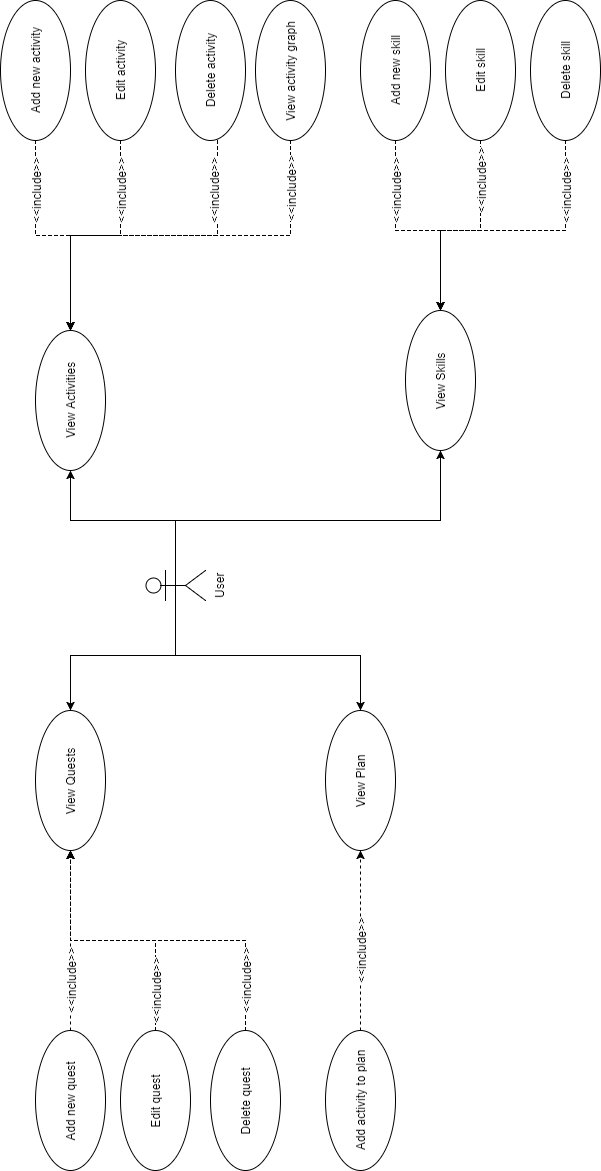
صفحه ویرایش فعالیت

1. ساختار داده‌ها و بانک اطلاعات
   1. مقدمه

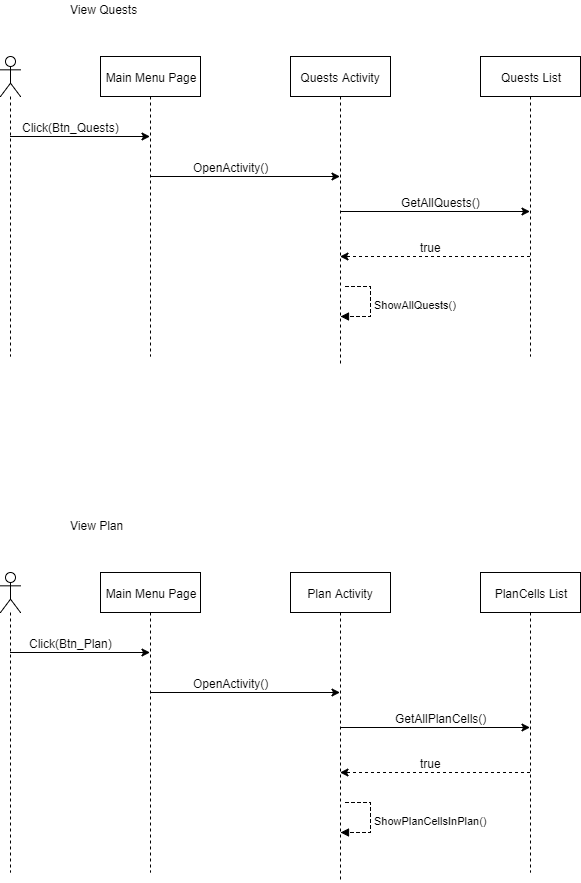
در این فصل نمودارهای زبان مدل‌سازی مورد استفاده در این پروژه، یعنی UML آورده شده است. این نمودارها با استفاده از نرم‌افزار Draw.io که یک نرم‌افزار رایگان برای رسم نمودارهای UML است، کشیده شده‌اند.

* 1. نمودار­های UML

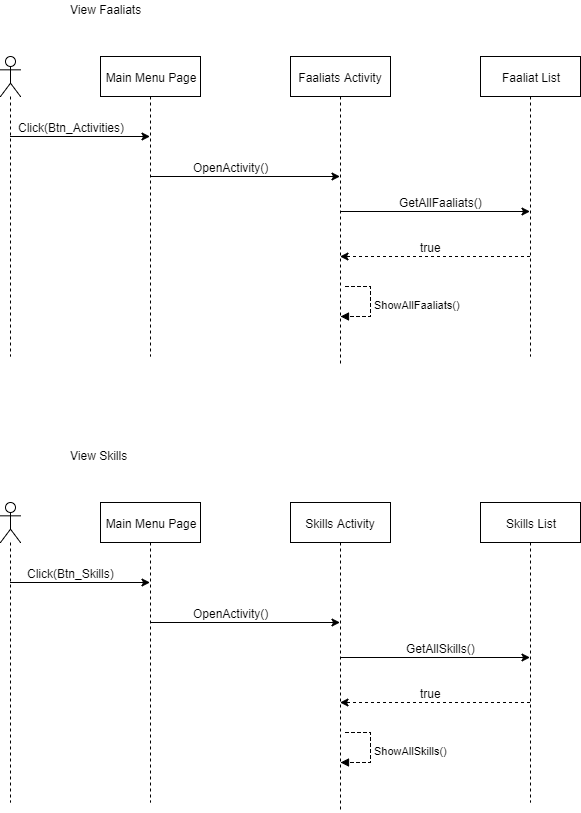
در این بخش نمودارهای Use Case (مورد کاربرد) و نمودارهای توالی مربوط به برنامه رسم شده­اند.

* + 1. نمودار Use Case

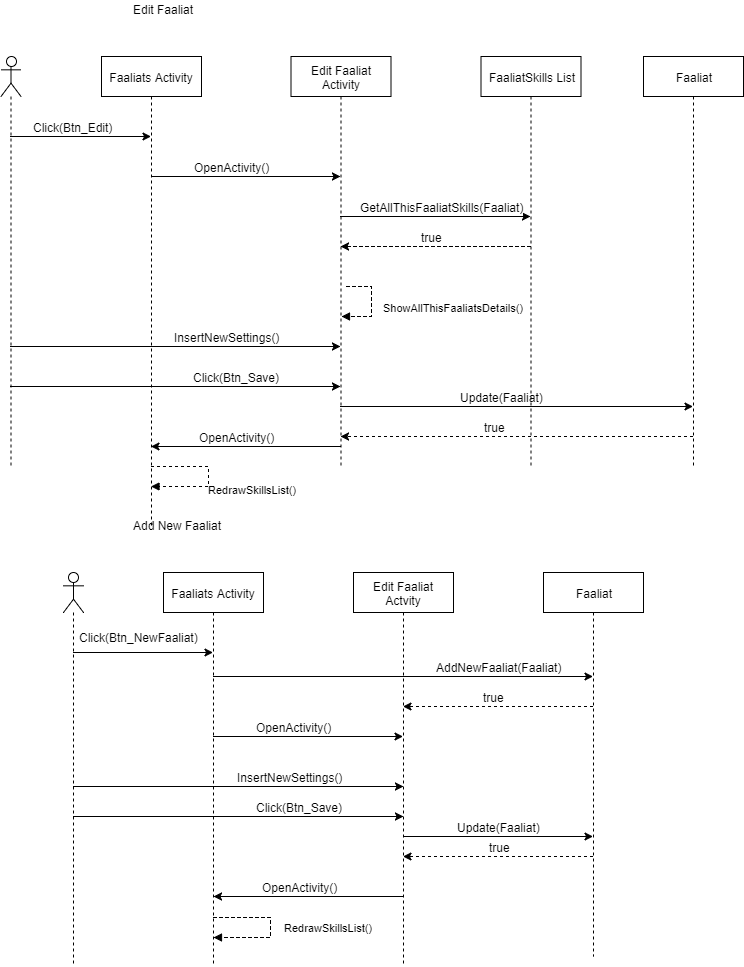
نمودار Use Case

* + 1. نمودار توالی

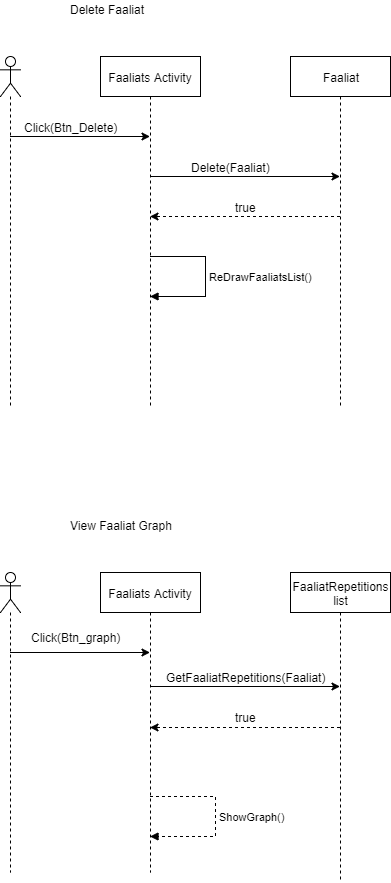
نمودارهای توالی View Quests و View Plan

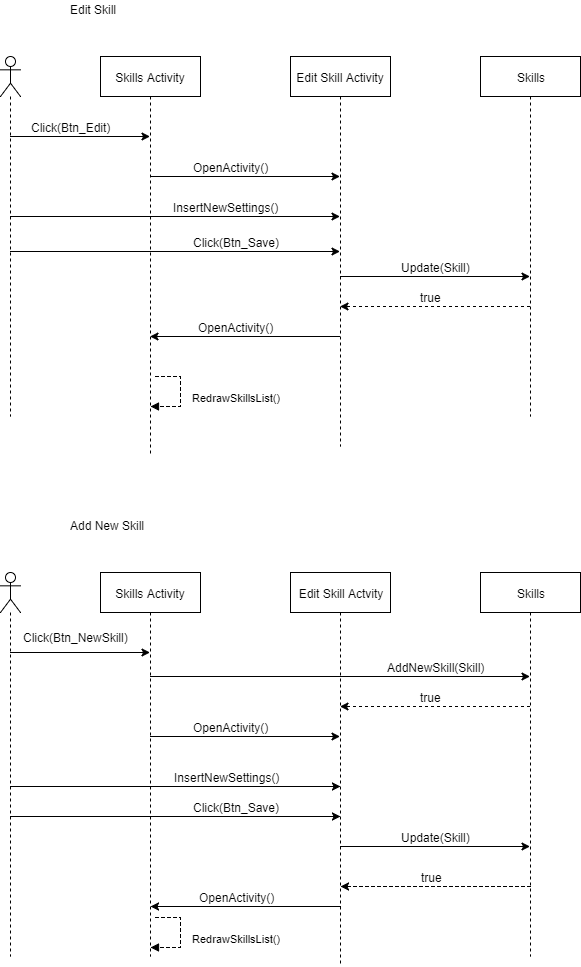


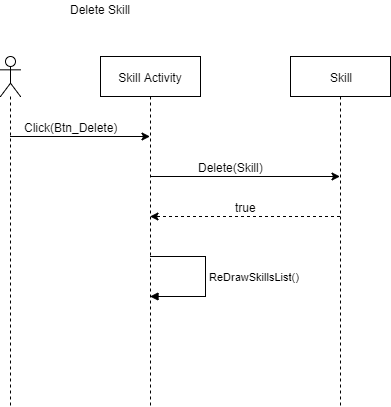
نمودارهای توالی View Faaliats و View Skills



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

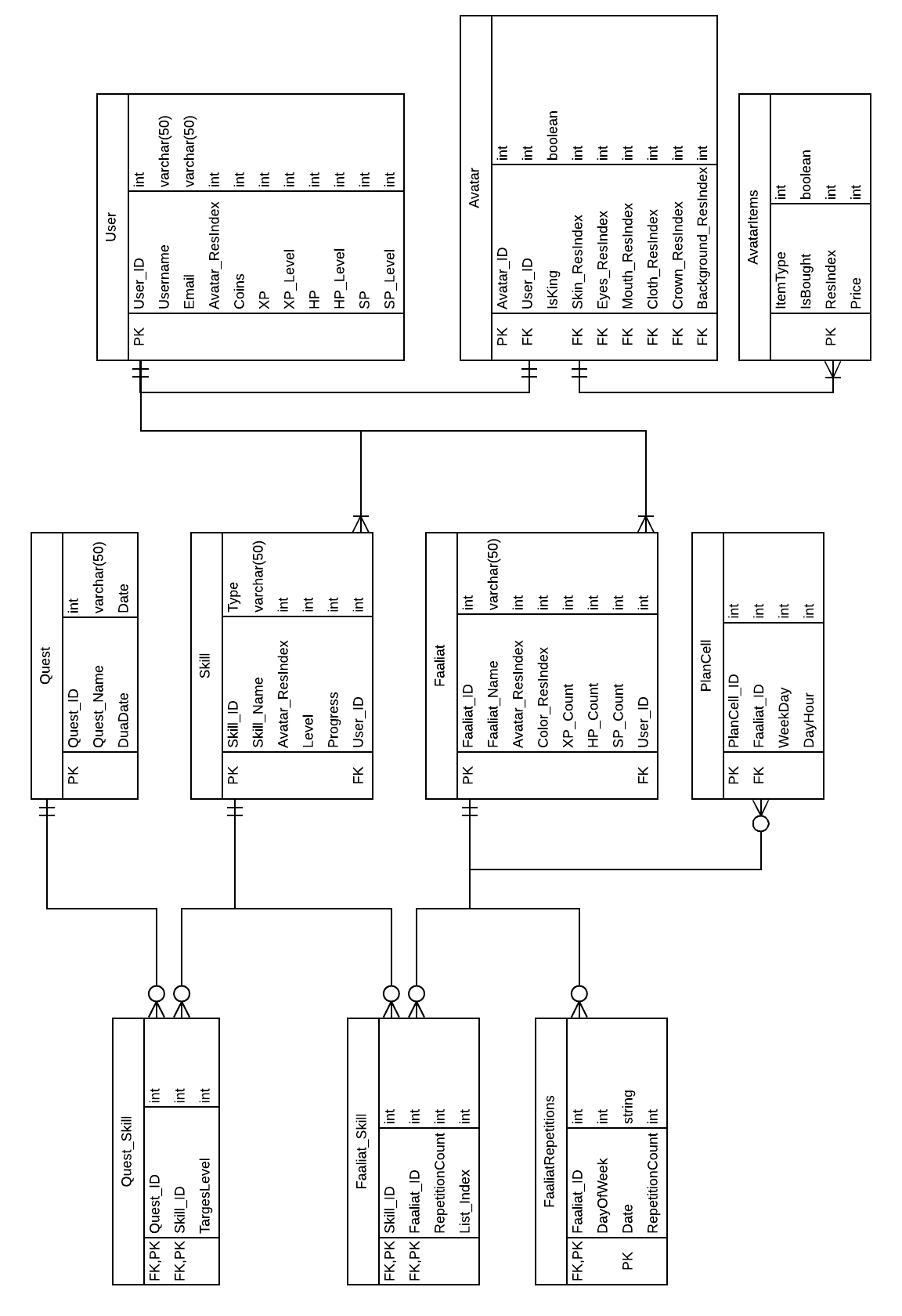
اين فصل حاوي جزئيات پياده­سازي، ابزارها، نرم­افزارها، چارچوب­ها، زبان­ها و تنظيمات است.

همچنين تصاويري از اجراهاي نمونه و رخدادها....

معرفي فصل در اين قسمت است.

* 1. پایگاه‌ داده

این اپلیکیشن به زبان جاوا و در محیط Android Studio 3 نوشته شده است. در این برنامه از پایگاه داده­ی محلی Room Persistence استفاده شده است. کتابخانه‌ی Room Persistence در اصل یک لایه انتزاعی بر روی SQLite است که دسترسی به پایگاه‌داده را روان‌تر کرده و در عین حال از تمام توان SQLite استفاده می‌کند.



نمودار ER

جدول User

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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

جدول Faaliat

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 Quest

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 PlanCell

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 Quest\_Skill

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 Faaliat\_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 FaaliatRepetitions

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 Avatar

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 AvatarItem

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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 |

1. جمع‌‌بندي و پيشنهاد‌ها
   1. نتيجه‌گيري

ارائه‌ي خلاصه‌اي از کارهاي انجام شده در کل اين پايان­نامه و يافته­ها و مشکلات...

* 1. پيشنهادهايي براي کارهاي آتي

در این پروژه قسمت questها و planner به علت کمبود وقت انجام نشد ولی با توجه به ساختار منظم در برنامه نویسی و شی­گرایی برنامه، می­توان در آینده این قسمت ها را به برنامه اضافه کرد.

همچنین می­توان بخش­های مربوط به فروشگاه و تصاویر گرافیکی برنامه را بهبود بخشید.

بالانويس جدول

جدول

متن

شکل

زيرنويس شکل

متن

فرمول

متن

مراجع

مراجع

1. V. R. Voller, "A Fixed Grid Numerical Modeling Methodology For Convection-Diffusion Mushy Region Phase-Change Problems", Int. J. Heat and Mass Transfer, Vol. 30, No. 8, pp-1709-1719, (1987)
2. سيد حسين سيدين، "مدل‌سازي انتقال حرارت و انجماد در فرايند ريخته‌گري مداوم تک‌غلتکه رول سرب ـ کلسيم"، گزارش قرارداد تحقيقاتي، شهريور 1380
3. کتاب پایگاه داده ی رانکوهی
4. کتاب های مهندسی نرم افزار یک و دو پرس من
5. کتاب های آموزش اندروید یا جاوا یا شی گرایی و....

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 ;  
 }  
}

DAOهای 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 for User entity  
 \*/*@Dao  
**public interface** UserDao  
{  
 @Insert  
 **void** insert(User user);  
  
 @Update  
 **void** update(User user);  
  
 @Delete  
 **void** delete(User user);  
  
 @Query(**"DELETE FROM User\_table"**)  
 **void** deleteAll();  
  
 @Query(**"SELECT \* FROM User\_table"**)  
 LiveData<User> getUser();  
}

**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** SkillDao  
{  
 @Insert  
 **void** insert(Skill skill);  
  
 @Update  
 **void** update(Skill skill);  
  
 @Delete  
 **void** delete(Skill skill);  
  
 @Query(**"DELETE FROM Skill\_table"**)  
 **void** deleteAll();  
  
 @Query(**"SELECT \* FROM Skill\_table ORDER BY Skill\_ID"**)  
 LiveData<List<Skill>> getAllSkills();}

**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** QuestDao  
{  
 @Insert  
 **void** insert(Quest quest);  
  
 @Update  
 **void** update(Quest quest);  
  
 @Delete  
 **void** delete(Quest quest);  
  
 @Query(**"DELETE FROM Quest\_table"**)  
 **void** deleteAll();  
  
 @Query(**"SELECT \* FROM Quest\_table ORDER BY Quest\_ID"**)  
 LiveData<List<Quest>> getAllQuests();  
}

**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** PlanCellDao  
{  
 @Insert  
 **void** insert(PlanCell planCell);  
  
 @Update  
 **void** update(PlanCell planCell);  
  
 @Delete  
 **void** delete(PlanCell planCell);  
  
 @Query(**"DELETE FROM PlanCell\_table"**)  
 **void** deleteAll();  
  
 @Query(**"SELECT \* FROM PlanCell\_table ORDER BY WeekDay, DayHour"**)  
 LiveData<List<PlanCell>> getAllPlanCells();  
}

**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;  
  
*/\*\*  
 \* Created by Narges on 6/25/2018.  
 \*/*@Dao  
**public interface** AvatarDao  
{  
 @Insert  
 **void** insert(Avatar avatar);  
  
 @Update  
 **void** update(Avatar avatar);  
  
 @Delete  
 **void** delete(Avatar avatar);  
  
 @Query(**"DELETE FROM Avatar\_table"**)  
 **void** deleteAll();  
  
 @Query(**"SELECT \* FROM Avatar\_table"**)  
 LiveData<Avatar> getAvatar();  
}

**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** AvatarItemDao  
{  
 @Insert  
 **void** insert(AvatarItem avatarItem);  
  
 @Update  
 **void** update(AvatarItem avatarItem);  
  
 @Delete  
 **void** delete(AvatarItem avatarItem);  
  
 @Query(**"DELETE FROM AvatarItem\_table"**)  
 **void** deleteAll();  
  
 @Query(**"SELECT \* FROM AvatarItem\_table"**)  
 LiveData<List<AvatarItem>> getAllAvatarItem();  
}

**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);  
}

**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;  
  
*/\*\*  
 \* Created by Narges on 6/8/2018.  
 \*/*@Dao  
**public interface** FaaliatSkillDao  
{  
 @Insert  
 **void** insert(FaaliatSkill faaliatSkill);  
  
 @Update  
 **void** update(FaaliatSkill faaliatSkill);  
  
 @Delete  
 **void** delete(FaaliatSkill faaliatSkill);  
  
 @Query(**"DELETE FROM FaaliatSkill\_table"**)  
 **void** deleteAll();  
  
 @Query(**"SELECT \* FROM FaaliatSkill\_table ORDER BY Faaliat\_ID, Skill\_ID"**)  
 LiveData<List<FaaliatSkill>> getAllFaaliatSkills();  
  
 @Query(**"SELECT \* FROM Skill\_table INNER JOIN FaaliatSkill\_table ON "** +  
 **"Skill\_table.Skill\_ID = FaaliatSkill\_table.Skill\_ID "** +  
 **"WHERE FaaliatSkill\_table.Faaliat\_ID=:faaliatID"**)  
 LiveData<List<FaaliatSkill>> getSkillsForFaaliat(**final int** faaliatID);  
}

**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** QuestSkillDao  
{  
 @Insert  
 **void** insert(QuestSkill questSkill);  
  
 @Update  
 **void** update(QuestSkill questSkill);  
  
 @Delete  
 **void** delete(QuestSkill questSkill);  
  
 @Query(**"DELETE FROM QuestSkill\_table"**)  
 **void** deleteAll();  
  
 @Query(**"SELECT \* FROM QuestSkill\_table ORDER BY Quest\_ID, Skill\_ID"**)  
 LiveData<List<QuestSkill>> getAllQuestSkills();  
  
 @Query(**"SELECT \* FROM Skill\_table INNER JOIN QuestSkill\_table ON"** +  
 **" Skill\_table.Skill\_ID = QuestSkill\_table.Skill\_ID "** +  
 **"WHERE QuestSkill\_table.Quest\_ID=:questID"**)  
 List<Skill> getSkillsForQuest(**final int** questID);  
}

Repositoryهای موجودیت‌ها:

**package** com.farazannajmi.majesticlife.DataStructures;  
  
**import** android.app.Application;  
**import** android.arch.lifecycle.LiveData;  
**import** android.os.AsyncTask;  
  
**import** java.util.List;  
**public class** UserRepository  
{  
 **private** UserDao **mUserDao**;  
 **private** LiveData<User> **mUser**;  
  
 LiveData<User> getUser()  
 {  
 **return mUser**;  
 }  
  
 **public** UserRepository(Application application)  
 {  
 MajesticLifeRoomDatabase db = MajesticLifeRoomDatabase.*getDatabase*(application);  
 **mUserDao** = db.userDao();  
 **mUser** = **mUserDao**.getUser();  
 }  
  
 **public void** insert (User user)  
 {  
 **new** UserRepository.insertAsyncTask(**mUserDao**).execute(user);  
 }  
  
 **public void** update (User user)  
 {  
 **new** UserRepository.updateAsyncTask(**mUserDao**).execute(user);  
 }  
  
 **public void** delete (User user)  
 {  
 *//new UserRepository.updateAsyncTask(mUserDao).execute(user);* }  
  
 **private static class** insertAsyncTask **extends** AsyncTask<User, Void, Void>  
 {  
 **private** UserDao **mAsyncTaskDao**;  
  
 insertAsyncTask(UserDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(**final** User... params)  
 {  
 **mAsyncTaskDao**.insert(params[0]);  
 **return null**;  
 }  
 }  
  
 **private static class** updateAsyncTask **extends** AsyncTask<User, Void, Void>  
 {  
 **private** UserDao **mAsyncTaskDao**;  
  
 updateAsyncTask(UserDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(**final** User... params)  
 {  
 **mAsyncTaskDao**.update(params[0]);  
 **return null**;  
 }  
 }  
  
 **private static class** deleteAsyncTask **extends** AsyncTask<User, Void, Void>  
 {  
 **private** UserDao **mAsyncTaskDao**;  
  
 deleteAsyncTask(UserDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(**final** User... params)  
 {  
 **mAsyncTaskDao**.delete(params[0]);  
 **return null**;  
 }  
 }  
}

**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**;  
 }  
 }  
}

**package** com.farazannajmi.majesticlife.DataStructures;  
  
**import** android.app.Application;  
**import** android.arch.lifecycle.LiveData;  
**import** android.os.AsyncTask;  
  
**import** java.util.List;**public class** SkillRepository  
{  
 **private** SkillDao **mSkillDao**;  
 **private** LiveData<List<Skill>> **mAllSkills**;  
  
 **public** SkillRepository(Application application)  
 {  
 MajesticLifeRoomDatabase db = MajesticLifeRoomDatabase.*getDatabase*(application);  
 **mSkillDao** = db.skillDao();  
 **mAllSkills** = **mSkillDao**.getAllSkills();  
 }  
  
 *//Room executes all queries on a separate thread.  
 //Observed LiveData will notify the observer when the data has changed.* LiveData<List<Skill>> getAllSkills()  
 {  
 **return mAllSkills**;  
 }  
  
 */\*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 (Skill skill)  
 {  
 **new** SkillRepository.insertAsyncTask(**mSkillDao**).execute(skill);  
 }  
  
 **public void** update (Skill skill)  
 {  
 **new** SkillRepository.updateAsyncTask(**mSkillDao**).execute(skill);  
 }  
  
 **public void** delete (Skill skill)  
 {  
 **new** SkillRepository.deleteAsyncTask(**mSkillDao**).execute(skill);  
 }  
  
 **private static class** insertAsyncTask **extends** AsyncTask<Skill, Void, Void>  
 {  
 **private** SkillDao **mAsyncTaskDao**;  
  
 insertAsyncTask(SkillDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(**final** Skill... params)  
 {  
 **mAsyncTaskDao**.insert(params[0]);  
 **return null**;  
 }  
 }  
  
 **private static class** updateAsyncTask **extends** AsyncTask<Skill, Void, Void>  
 {  
 **private** SkillDao **mAsyncTaskDao**;  
  
 updateAsyncTask(SkillDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(**final** Skill... params)  
 {  
 **mAsyncTaskDao**.update(params[0]);  
 **return null**;  
 }  
 }  
  
 **private static class** deleteAsyncTask **extends** AsyncTask<Skill, Void, Void>  
 {  
 **private** SkillDao **mAsyncTaskDao**;  
  
 deleteAsyncTask(SkillDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(**final** Skill... params)  
 {  
 **mAsyncTaskDao**.delete(params[0]);  
 **return null**;  
 }  
 }  
}

**package** com.farazannajmi.majesticlife.DataStructures;  
  
**import** android.app.Application;  
**import** android.arch.lifecycle.LiveData;  
**import** android.os.AsyncTask;  
  
**import** java.util.List;**public class** QuestRepository  
{  
 **private** QuestDao **mQuestDao**;  
 **private** LiveData<List<Quest>> **mAllQuests**;  
  
 **public** QuestRepository(Application application)  
 {  
 MajesticLifeRoomDatabase db = MajesticLifeRoomDatabase.*getDatabase*(application);  
 **mQuestDao** = db.questDao();  
 **mAllQuests** = **mQuestDao**.getAllQuests();  
 }  
  
 LiveData<List<Quest>> getAllQuests()  
 {  
 **return mAllQuests**;  
 }  
  
 **public void** insert (Quest quest)  
 {  
 **new** QuestRepository.insertAsyncTask(**mQuestDao**).execute(quest);  
 }  
 **private static class** insertAsyncTask **extends** AsyncTask<Quest, Void, Void>  
 {  
 **private** QuestDao **mAsyncTaskDao**;  
  
 insertAsyncTask(QuestDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(**final** Quest... params)  
 {  
 **mAsyncTaskDao**.insert(params[0]);  
 **return null**;  
 }  
 }  
  
 **public void** update (Quest quest)  
 {  
 **new** QuestRepository.updateAsyncTask(**mQuestDao**).execute(quest);  
 }  
 **private static class** updateAsyncTask **extends** AsyncTask<Quest, Void, Void>  
 {  
 **private** QuestDao **mAsyncTaskDao**;  
  
 updateAsyncTask(QuestDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(**final** Quest... params)  
 {  
 **mAsyncTaskDao**.update(params[0]);  
 **return null**;  
 }  
 }  
  
 **public void** delete (Quest quest)  
 {  
 **new** QuestRepository.deleteAsyncTask(**mQuestDao**).execute(quest);  
 }  
 **private static class** deleteAsyncTask **extends** AsyncTask<Quest, Void, Void>  
 {  
 **private** QuestDao **mAsyncTaskDao**;  
  
 deleteAsyncTask(QuestDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(**final** Quest... params)  
 {  
 **mAsyncTaskDao**.delete(params[0]);  
 **return null**;  
 }  
 }  
  
 **public void** deleteAll ()  
 {  
 **new** QuestRepository.deleteAllAsyncTask(**mQuestDao**).execute();  
 }  
 **private static class** deleteAllAsyncTask **extends** AsyncTask<Void, Void, Void>  
 {  
 **private** QuestDao **mAsyncTaskDao**;  
  
 deleteAllAsyncTask(QuestDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(Void... voids)  
 {  
 **mAsyncTaskDao**.deleteAll();  
 **return null**;  
 }  
 }  
}

**package** com.farazannajmi.majesticlife.DataStructures;  
  
**import** android.app.Application;  
**import** android.arch.lifecycle.LiveData;  
**import** android.os.AsyncTask;  
  
**import** java.util.List;**public class** PlanCellRepository  
{  
 **private** PlanCellDao **mPlanCellDao**;  
 **private** LiveData<List<PlanCell>> **mAllPlanCells**;  
  
 **public** PlanCellRepository(Application application)  
 {  
 MajesticLifeRoomDatabase db = MajesticLifeRoomDatabase.*getDatabase*(application);  
 **mPlanCellDao** = db.planCellDao();  
 **mAllPlanCells** = **mPlanCellDao**.getAllPlanCells();  
 }  
  
 *//Room executes all queries on a separate thread.  
 //Observed LiveData will notify the observer when the data has changed.* LiveData<List<PlanCell>> getAllPlanCells()  
 {  
 **return mAllPlanCells**;  
 }  
  
 */\*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 (PlanCell planCell)  
 {  
 **new** insertAsyncTask(**mPlanCellDao**).execute(planCell);  
 }  
  
 **private static class** insertAsyncTask **extends** AsyncTask<PlanCell, Void, Void>  
 {  
 **private** PlanCellDao **mAsyncTaskDao**;  
  
 insertAsyncTask(PlanCellDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(**final** PlanCell... params)  
 {  
 **mAsyncTaskDao**.insert(params[0]);  
 **return null**;  
 }  
 }  
}

**package** com.farazannajmi.majesticlife.DataStructures;  
  
**import** android.app.Application;  
**import** android.arch.lifecycle.LiveData;  
**import** android.os.AsyncTask;**public class** AvatarRepository  
{  
 **private** AvatarDao **mAvatarDao**;  
 **private** LiveData<Avatar> **mAvatar**;  
  
 LiveData<Avatar> getAvatar()  
 {  
 **return mAvatar**;  
 }  
  
 **public** AvatarRepository(Application application)  
 {  
 MajesticLifeRoomDatabase db = MajesticLifeRoomDatabase.*getDatabase*(application);  
 **mAvatarDao** = db.avatarDao();  
 **mAvatar** = **mAvatarDao**.getAvatar();  
 }  
  
 **public void** insert (Avatar avatar)  
 {  
 **new** AvatarRepository.insertAsyncTask(**mAvatarDao**).execute(avatar);  
 }  
  
 **public void** update (Avatar avatar)  
 {  
 **new** AvatarRepository.updateAsyncTask(**mAvatarDao**).execute(avatar);  
 }  
  
 **public void** delete (Avatar avatar)  
 {  
 **new** AvatarRepository.deleteAsyncTask(**mAvatarDao**).execute(avatar);  
 }  
  
 **private static class** insertAsyncTask **extends** AsyncTask<Avatar, Void, Void>  
 {  
 **private** AvatarDao **mAsyncTaskDao**;  
  
 insertAsyncTask(AvatarDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(**final** Avatar... params)  
 {  
 **mAsyncTaskDao**.insert(params[0]);  
 **return null**;  
 }  
 }  
  
 **private static class** updateAsyncTask **extends** AsyncTask<Avatar, Void, Void>  
 {  
 **private** AvatarDao **mAsyncTaskDao**;  
  
 updateAsyncTask(AvatarDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(**final** Avatar... params)  
 {  
 **mAsyncTaskDao**.update(params[0]);  
 **return null**;  
 }  
 }  
  
 **private static class** deleteAsyncTask **extends** AsyncTask<Avatar, Void, Void>  
 {  
 **private** AvatarDao **mAsyncTaskDao**;  
  
 deleteAsyncTask(AvatarDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(**final** Avatar... params)  
 {  
 **mAsyncTaskDao**.delete(params[0]);  
 **return null**;  
 }  
 }  
}

**package** com.farazannajmi.majesticlife.DataStructures;  
  
**import** android.app.Application;  
**import** android.arch.lifecycle.LiveData;  
**import** android.os.AsyncTask;  
  
**import** java.util.List;**public class** AvatarItemRepository  
{  
 **private** AvatarItemDao **mAvatarItemDao**;  
 **private** LiveData<List<AvatarItem>> **mAllAvatarItems**;  
  
 LiveData<List<AvatarItem>> getAvatarItems()  
 {  
 **return mAllAvatarItems**;  
 }  
  
 **public** AvatarItemRepository(Application application)  
 {  
 MajesticLifeRoomDatabase db = MajesticLifeRoomDatabase.*getDatabase*(application);  
 **mAvatarItemDao** = db.avatarItemDao();  
 **mAllAvatarItems** = **mAvatarItemDao**.getAllAvatarItem();  
 }  
  
 **public void** insert (AvatarItem avatarItem)  
 {  
 **new** AvatarItemRepository.insertAsyncTask(**mAvatarItemDao**).execute(avatarItem);  
 }  
  
 **public void** update (AvatarItem avatarItem)  
 {  
 **new** AvatarItemRepository.updateAsyncTask(**mAvatarItemDao**).execute(avatarItem);  
 }  
  
 **public void** delete (AvatarItem avatarItem)  
 {  
 **new** AvatarItemRepository.deleteAsyncTask(**mAvatarItemDao**).execute(avatarItem);  
 }  
  
 **private static class** insertAsyncTask **extends** AsyncTask<AvatarItem, Void, Void>  
 {  
 **private** AvatarItemDao **mAsyncTaskDao**;  
  
 insertAsyncTask(AvatarItemDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(**final** AvatarItem... params)  
 {  
 **mAsyncTaskDao**.insert(params[0]);  
 **return null**;  
 }  
 }  
  
 **private static class** updateAsyncTask **extends** AsyncTask<AvatarItem, Void, Void>  
 {  
 **private** AvatarItemDao **mAsyncTaskDao**;  
  
 updateAsyncTask(AvatarItemDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(**final** AvatarItem... params)  
 {  
 **mAsyncTaskDao**.update(params[0]);  
 **return null**;  
 }  
 }  
  
 **private static class** deleteAsyncTask **extends** AsyncTask<AvatarItem, Void, Void>  
 {  
 **private** AvatarItemDao **mAsyncTaskDao**;  
  
 deleteAsyncTask(AvatarItemDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(**final** AvatarItem... params)  
 {  
 **mAsyncTaskDao**.delete(params[0]);  
 **return null**;  
 }  
 }  
}

**package** com.farazannajmi.majesticlife.DataStructures;  
  
**import** android.app.Application;  
**import** android.arch.lifecycle.LiveData;  
**import** android.os.AsyncTask;  
  
**import** java.util.List;**public class** FaaliatSkillRepository  
{  
 **private** FaaliatSkillDao **mFaaliatSkillDao**;  
 **private** LiveData<List<FaaliatSkill>> **mAllFaaliatSkills**;  
  
 **public** FaaliatSkillRepository(Application application)  
 {  
 MajesticLifeRoomDatabase db = MajesticLifeRoomDatabase.*getDatabase*(application);  
 **mFaaliatSkillDao** = db.faaliatSkillDao();  
 **mAllFaaliatSkills** = **mFaaliatSkillDao**.getAllFaaliatSkills();  
 }  
LiveData<List<FaaliatSkill>> getAllFaaliatSkills()  
 {  
 **return mAllFaaliatSkills**;  
 }

**public void** insert (FaaliatSkill faaliatSkill)  
 {  
 **new** FaaliatSkillRepository.insertAsyncTask(**mFaaliatSkillDao**).execute(faaliatSkill);  
 }  
 **private static class** insertAsyncTask **extends** AsyncTask<FaaliatSkill, Void, Void>  
 {  
 **private** FaaliatSkillDao **mAsyncTaskDao**;  
  
 insertAsyncTask(FaaliatSkillDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(**final** FaaliatSkill... params)  
 {  
 **mAsyncTaskDao**.insert(params[0]);  
 **return null**;  
 }  
 }  
  
 **public void** update (FaaliatSkill faaliatSkill)  
 {  
 **new** FaaliatSkillRepository.updateAsyncTask(**mFaaliatSkillDao**).execute(faaliatSkill);  
 }  
 **private static class** updateAsyncTask **extends** AsyncTask<FaaliatSkill, Void, Void>  
 {  
 **private** FaaliatSkillDao **mAsyncTaskDao**;  
  
 updateAsyncTask(FaaliatSkillDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(**final** FaaliatSkill... params)  
 {  
 **mAsyncTaskDao**.update(params[0]);  
 **return null**;  
 }  
 }  
  
 **public void** delete (FaaliatSkill faaliatSkill)  
 {  
 **new** FaaliatSkillRepository.deleteAsyncTask(**mFaaliatSkillDao**).execute(faaliatSkill);  
 }  
 **private static class** deleteAsyncTask **extends** AsyncTask<FaaliatSkill, Void, Void>  
 {  
 **private** FaaliatSkillDao **mAsyncTaskDao**;  
  
 deleteAsyncTask(FaaliatSkillDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(**final** FaaliatSkill... params)  
 {  
 **mAsyncTaskDao**.delete(params[0]);  
 **return null**;  
 }  
 }  
  
 **public void** deleteAll ()  
 {  
 **new** FaaliatSkillRepository.deleteAllAsyncTask(**mFaaliatSkillDao**).execute();  
 }  
 **private static class** deleteAllAsyncTask **extends** AsyncTask<Void, Void, Void>  
 {  
 **private** FaaliatSkillDao **mAsyncTaskDao**;  
  
 deleteAllAsyncTask(FaaliatSkillDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(Void... voids)  
 {  
 **mAsyncTaskDao**.deleteAll();  
 **return null**;  
 }  
 }  
  
 **public** LiveData<List<FaaliatSkill>> getSkillsForFaaliat (Faaliat faaliat)  
 {  
 **return mFaaliatSkillDao**.getSkillsForFaaliat(faaliat.getFaaliat\_ID());  
 }  
}

**package** com.farazannajmi.majesticlife.DataStructures;  
  
**import** android.app.Application;  
**import** android.arch.lifecycle.LiveData;  
**import** android.os.AsyncTask;  
  
**import** java.util.List;  
  
*/\*\*  
 \* Created by Narges on 6/9/2018.  
 \*/***public class** QuestSkillRepository  
{  
 **private** QuestSkillDao **mQuestSkillDao**;  
 **private** LiveData<List<QuestSkill>> **mAllQuestSkills**;  
  
 **public** QuestSkillRepository(Application application)  
 {  
 MajesticLifeRoomDatabase db = MajesticLifeRoomDatabase.*getDatabase*(application);  
 **mQuestSkillDao** = db.questSkillDao();  
 **mAllQuestSkills** = **mQuestSkillDao**.getAllQuestSkills();  
 }  
  
 *//Room executes all queries on a separate thread.  
 //Observed LiveData will notify the observer when the data has changed.* LiveData<List<QuestSkill>> getAllQuestSkills()  
 {  
 **return mAllQuestSkills**;  
 }  
  
 */\*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 (QuestSkill questSkill)  
 {  
 **new** QuestSkillRepository.insertAsyncTask(**mQuestSkillDao**).execute(questSkill);  
 }  
 **private static class** insertAsyncTask **extends** AsyncTask<QuestSkill, Void, Void>  
 {  
 **private** QuestSkillDao **mAsyncTaskDao**;  
  
 insertAsyncTask(QuestSkillDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(**final** QuestSkill... params)  
 {  
 **mAsyncTaskDao**.insert(params[0]);  
 **return null**;  
 }  
 }  
  
 **public void** update (QuestSkill questSkill)  
 {  
 **new** QuestSkillRepository.updateAsyncTask(**mQuestSkillDao**).execute(questSkill);  
 }  
 **private static class** updateAsyncTask **extends** AsyncTask<QuestSkill, Void, Void>  
 {  
 **private** QuestSkillDao **mAsyncTaskDao**;  
  
 updateAsyncTask(QuestSkillDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(**final** QuestSkill... params)  
 {  
 **mAsyncTaskDao**.update(params[0]);  
 **return null**;  
 }  
 }  
  
 **public void** delete (QuestSkill questSkill)  
 {  
 **new** QuestSkillRepository.deleteAsyncTask(**mQuestSkillDao**).execute(questSkill);  
 }  
 **private static class** deleteAsyncTask **extends** AsyncTask<QuestSkill, Void, Void>  
 {  
 **private** QuestSkillDao **mAsyncTaskDao**;  
  
 deleteAsyncTask(QuestSkillDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(**final** QuestSkill... params)  
 {  
 **mAsyncTaskDao**.delete(params[0]);  
 **return null**;  
 }  
 }  
  
 **public void** deleteAll ()  
 {  
 **new** QuestSkillRepository.deleteAllAsyncTask(**mQuestSkillDao**).execute();  
 }  
 **private static class** deleteAllAsyncTask **extends** AsyncTask<Void, Void, Void>  
 {  
 **private** QuestSkillDao **mAsyncTaskDao**;  
  
 deleteAllAsyncTask(QuestSkillDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(Void... voids)  
 {  
 **mAsyncTaskDao**.deleteAll();  
 **return null**;  
 }  
 }  
  
 **public void** getSkillsForQuest (Quest quest)  
 {  
 **new** QuestSkillRepository.getSkillsForQuestAllAsyncTask(**mQuestSkillDao**).execute(quest);  
 }  
 **private static class** getSkillsForQuestAllAsyncTask **extends** AsyncTask<Quest, Void, Void>  
 {  
 **private** QuestSkillDao **mAsyncTaskDao**;  
  
 getSkillsForQuestAllAsyncTask(QuestSkillDao dao)  
 {  
 **mAsyncTaskDao** = dao;  
 }  
  
 @Override  
 **protected** Void doInBackground(Quest... params)  
 {  
 **mAsyncTaskDao**.getSkillsForQuest(params[0].getQuest\_ID());  
 **return null**;  
 }  
 }  
}

ViewModelهای موجودیت‌ها:

**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** UserViewModel **extends** AndroidViewModel  
{  
 **private** UserRepository mRepository;  
 **private** LiveData<User> mUser;  
  
 **public** LiveData<User> getUser() {**return** mUser;}  
  
 **public** UserViewModel (Application application)  
 {  
 **super**(application);  
 mRepository = **new** UserRepository(application);  
 mUser = mRepository.getUser();  
 }  
  
 **public void** insert(User user)  
 {  
 mRepository.insert(user);  
 }  
  
 **public void** update(User user)  
 {  
 mRepository.update(user);  
 }  
  
 **public void** delete(User user)  
 {  
 mRepository.delete(user);  
 }}

**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);  
 }  
}

**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** SkillViewModel **extends** AndroidViewModel  
{  
 **private** SkillRepository **mRepository**;  
 **private** LiveData<List<Skill>> **mAllSkills**;  
  
 **public** LiveData<List<Skill>> getAllSkills() {**return mAllSkills**;}  
  
 **public** SkillViewModel (Application application)  
 {  
 **super**(application);  
 **mRepository** = **new** SkillRepository(application);  
 **mAllSkills** = **mRepository**.getAllSkills();  
 }  
  
 **public void** insert(Skill skill)  
 {  
 **mRepository**.insert(skill);  
 }  
  
 **public void** update(Skill skill)  
 {  
 **mRepository**.update(skill);  
 }  
  
 **public void** delete(Skill skill)  
 {  
 **mRepository**.delete(skill);  
 }  
}

**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** QuestViewModel **extends** AndroidViewModel  
{  
 **private** QuestRepository **mRepository**;  
 **private** LiveData<List<Quest>> **mAllQuests**;  
  
 **public** LiveData<List<Quest>> getAllQuests() {**return mAllQuests**;}  
  
 **public** QuestViewModel (Application application)  
 {  
 **super**(application);  
 **mRepository** = **new** QuestRepository(application);  
 **mAllQuests** = **mRepository**.getAllQuests();  
 }  
  
 **public void** insert(Quest quest)  
 {  
 **mRepository**.insert(quest);  
 }  
  
 **public void** update(Quest quest)  
 {  
 **mRepository**.update(quest);  
 }  
  
 **public void** delete(Quest quest)  
 {  
 **mRepository**.delete(quest);  
 }  
  
 **public void** deleteAll()  
 {  
 **mRepository**.deleteAll();  
 }  
}

**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** PlanCellViewModel **extends** AndroidViewModel  
{  
 **private** PlanCellRepository **mRepository**;  
 **private** LiveData<List<PlanCell>> **mAllPlanCells**;  
  
 **public** LiveData<List<PlanCell>> getAllPlanCells() {**return mAllPlanCells**;}  
  
 **public** PlanCellViewModel (Application application)  
 {  
 **super**(application);  
 **mRepository** = **new** PlanCellRepository(application);  
 **mAllPlanCells** = **mRepository**.getAllPlanCells();  
 }  
  
 **public void** insert(PlanCell planCell)  
 {  
 **mRepository**.insert(planCell);  
 }  
}

**package** com.farazannajmi.majesticlife.DataStructures;  
  
**import** android.app.Application;  
**import** android.arch.lifecycle.AndroidViewModel;  
**import** android.arch.lifecycle.LiveData;**public class** AvatarViewModel **extends** AndroidViewModel  
{  
 **private** AvatarRepository **mRepository**;  
 **private** LiveData<Avatar> **mAvatar**;  
  
 **public** LiveData<Avatar> getAvatar() {**return mAvatar**;}  
  
 **public** AvatarViewModel (Application application)  
 {  
 **super**(application);  
 **mRepository** = **new** AvatarRepository(application);  
 **mAvatar** = **mRepository**.getAvatar();  
 }  
  
 **public void** insert(Avatar avatar)  
 {  
 **mRepository**.insert(avatar);  
 }  
  
 **public void** update(Avatar avatar)  
 {  
 **mRepository**.update(avatar);  
 }  
  
 **public void** delete(Avatar avatar)  
 {  
 **mRepository**.delete(avatar);  
 }  
}

**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** AvatarItemViewModel **extends** AndroidViewModel  
{  
 **private** AvatarItemRepository **mRepository**;  
 **private** LiveData<List<AvatarItem>> **mAvatarItem**;  
  
 **public** LiveData<List<AvatarItem>> getAvatarItems() {**return mAvatarItem**;}  
  
 **public** AvatarItemViewModel (Application application)  
 {  
 **super**(application);  
 **mRepository** = **new** AvatarItemRepository(application);  
 **mAvatarItem** = **mRepository**.getAvatarItems();  
 }  
  
 **public void** insert(AvatarItem avatarItem) {**mRepository**.insert(avatarItem);}  
  
 **public void** update(AvatarItem avatarItem) {**mRepository**.update(avatarItem);}  
  
 **public void** delete(AvatarItem avatarItem) { **mRepository**.delete(avatarItem);}  
}

**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** FaaliatRepetitionsViewModel **extends** AndroidViewModel  
{  
 **private** FaaliatRepetitionsRepository **mRepository**;  
 **private** LiveData<List<FaaliatRepetitions>> **mAllFaaliatRepetitions**;  
  
 **public** LiveData<List<FaaliatRepetitions>> getAllFaaliatRepetitions() {**return mAllFaaliatRepetitions**;}  
  
 **public** FaaliatRepetitionsViewModel (Application application)  
 {  
 **super**(application);  
 **mRepository** = **new** FaaliatRepetitionsRepository(application);  
 **mAllFaaliatRepetitions** = **mRepository**.getAllFaaliatRepetitions();  
 }  
  
 **public void** insert(FaaliatRepetitions faaliatRepetitions)  
 {  
 **mRepository**.insert(faaliatRepetitions);  
 }  
  
 **public void** update(FaaliatRepetitions faaliatRepetitions)  
 {  
 **mRepository**.update(faaliatRepetitions);  
 }  
  
 **public void** delete(FaaliatRepetitions faaliatRepetitions)  
 {  
 **mRepository**.delete(faaliatRepetitions);  
 }  
  
 **public void** deleteAll()  
 {  
 **mRepository**.deleteAll();  
 }  
  
 **public** LiveData<List<FaaliatRepetitions>> getAllFaaliatRepsForFaaliat(**int** faaliat\_ID)  
 {  
 **return mRepository**.getAllFaaliatRepsForFaaliat(faaliat\_ID);  
 }  
  
 **public** LiveData<List<FaaliatRepetitions>> getFaaliatRepsForFaaliatDate(**int** faaliat\_ID, String date)  
 {  
 **return mRepository**.getFaaliatRepsForFaaliatDate(faaliat\_ID, date);  
 }  
}

**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** FaaliatSkillViewModel **extends** AndroidViewModel  
{  
 **private** FaaliatSkillRepository **mRepository**;  
 **private** LiveData<List<FaaliatSkill>> **mAllFaaliatSkills**;  
  
 **public** LiveData<List<FaaliatSkill>> getAllFaaliatSkills() {**return mAllFaaliatSkills**;}  
  
 **public** FaaliatSkillViewModel (Application application)  
 {  
 **super**(application);  
 **mRepository** = **new** FaaliatSkillRepository(application);  
 **mAllFaaliatSkills** = **mRepository**.getAllFaaliatSkills();  
 }  
  
 **public void** insert(FaaliatSkill faaliatSkill)  
 {  
 **mRepository**.insert(faaliatSkill);  
 }  
  
 **public void** update(FaaliatSkill faaliatSkill)  
 {  
 **mRepository**.update(faaliatSkill);  
 }  
  
 **public void** delete(FaaliatSkill faaliatSkill)  
 {  
 **mRepository**.delete(faaliatSkill);  
 }  
  
 **public void** deleteAll()  
 {  
 **mRepository**.deleteAll();  
 }  
  
 **public** LiveData<List<FaaliatSkill>> getSkillsForFaaliat(Faaliat faaliat)  
 {  
 **return mRepository**.getSkillsForFaaliat(faaliat);  
 }  
}

**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** PlanCellViewModel **extends** AndroidViewModel  
{  
 **private** PlanCellRepository **mRepository**;  
 **private** LiveData<List<PlanCell>> **mAllPlanCells**;  
  
 **public** LiveData<List<PlanCell>> getAllPlanCells() {**return mAllPlanCells**;}  
  
 **public** PlanCellViewModel (Application application)  
 {  
 **super**(application);  
 **mRepository** = **new** PlanCellRepository(application);  
 **mAllPlanCells** = **mRepository**.getAllPlanCells();  
 }  
  
 **public void** insert(PlanCell planCell)  
 {  
 **mRepository**.insert(planCell);  
 }  
}

**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** QuestSkillViewModel **extends** AndroidViewModel  
{  
 **private** QuestSkillRepository **mRepository**;  
 **private** LiveData<List<QuestSkill>> **mAllQuestSkills**;  
  
 **public** LiveData<List<QuestSkill>> getAllQuestSkills() {**return mAllQuestSkills**;}  
  
 **public** QuestSkillViewModel (Application application)  
 {  
 **super**(application);  
 **mRepository** = **new** QuestSkillRepository(application);  
 **mAllQuestSkills** = **mRepository**.getAllQuestSkills();  
 }  
  
 **public void** insert(QuestSkill questSkill)  
 {  
 **mRepository**.insert(questSkill);  
 }  
  
 **public void** update(QuestSkill questSkill)  
 {  
 **mRepository**.update(questSkill);  
 }  
  
 **public void** delete(QuestSkill questSkill)  
 {  
 **mRepository**.delete(questSkill);  
 }  
  
 **public void** deleteAll()  
 {  
 **mRepository**.deleteAll();  
 }  
  
 **public void** getSkillsForQuest(Quest quest)  
 {  
 **mRepository**.getSkillsForQuest(quest);  
 }  
}

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)  
 {  
 *//order is 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"**); *//todo: wrong DuaDate param* 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);  
 *//todo: insert other clothes  
  
 //Clothes* AvatarItem ava\_cloth1 = **new** AvatarItem(R.drawable.ava\_cloth1, **true**, 0, 0);  
 **avatarItemDao**.insert(ava\_cloth1);  
 *//todo: insert other clothes  
  
 //Eyes* AvatarItem ava\_eyes1 = **new** AvatarItem(R.drawable.ava\_eyes1, **true**, 0, 0);  
 avatarItemDao.insert(ava\_eyes1);  
 *//todo: insert other eyes  
  
 //Mouths* AvatarItem ava\_mouth1 = **new** AvatarItem(R.drawable.ava\_mouth1, **true**, 0, 0);  
 avatarItemDao.insert(ava\_mouth1);  
 *//todo: insert other mouths  
  
 //Crowns* AvatarItem ava\_crown1 = **new** AvatarItem(R.drawable.ava\_crown1, **true**, 0, 0);  
 avatarItemDao.insert(ava\_crown1);  
 *//todo: insert other crowns  
 //endregion ------------------------------  
  
 //no initial FaaliatRepetitions  
 //todo 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**;  
 }  
 }  
}

**Abstract:**

This application is about gamifying daily activities and improving in your skills. By level upping 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