



دانشگاه فنی و حرفه‌ای  
دانشکده فنی دکتر شریعتی

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

پایان‌نامه برای دریافت درجه کارشناسی  
در رشته مهندسی کامپیوتر گرایش نرم‌افزار

نرگس فرازان، زهرا نجمی

استاد راهنما:  
دکتر ایمان شریفی

نیم‌سال اول ۱۳۹۷



دانشگاه فنی حرفه‌ای  
دانشکده فنی شریعتی

## اپلیکیشن بهبود سبک زندگی و برنامه‌ریزی با استفاده از بازی انگاری برای سیستم عامل اندروید

پایان‌نامه برای دریافت درجه کارشناسی  
در رشته مهندسی کامپیوتر گرایش نرم‌افزار

نرگس فرازان، زهرا نجمی

استاد راهنما:  
دکتر ایمان شریفی

نیم‌سال اول ۱۳۹۷

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

## تقدیم به:

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

## تشکر و قدردانی:

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

## چکیده

این پروژه، برنامه‌ای تحت سیستم عامل اندروید است که در آن سعی شده با استفاده از مفاهیم gamification (بازی‌انگاری) کاربر را به انجام فعالیت‌های مفید ترغیب کرد. از آنجایی که انجام فعالیت‌ها در قالب بازی و به صورت چالشی برای افراد جذاب‌تر می‌شود، از روش‌های gamification استفاده شده است. در این برنامه کاربر می‌تواند از بین انتخاب‌های موجود در برنامه یا مواردی که خودش تعریف می‌کند، فعالیت‌های مفید و غیرمفید (مضر) را انتخاب و میزان تاثیر آن‌ها را در مهارت‌هایش مشخص کند. با انجام هر فعالیت کاربر XP و HP و SP دریافت می‌کند که در بازه‌های مشخصی، ارتقای سطح<sup>۱</sup> پیدا می‌کند. همچنین با استفاده از المان‌های گرافیکی همچون نمودار سعی می‌شود تا کاربر میزان پیشرفت خود را مشاهده کند.

**واژه‌های کلیدی:** بازی‌انگاری، سبک زندگی، اندروید، اپلیکیشن موبایل، برنامه‌ریزی

## فهرست مطالب

۱	فصل ۱: مقدمه
۱-۱-۱	انگیزه.....
۲-۱-۲	هدف.....
۳-۱-۳	معرفی ابزارها و بستر برنامه نویسی.....
۳-۱-۳-۱	معرفی اجمالی اندروید استودیو و زبان برنامه نویسی جاوا.....
۴-۱-۴	نگاهی به فصل های آینده.....
۵	فصل ۲: تجزیه و تحلیل نیازمندی ها
۶-۱-۲	مقدمه.....
۶-۲-۲	معرفی کاربران برنامه.....
۶-۳-۲	رابط گرافیکی نرم افزار و معرفی صفحات نرم افزار.....
۷-۱-۳-۲	صفحه ی اصلی.....
۸-۲-۳-۲	صفحه ی نمایش اطلاعات کاربر.....
۹-۳-۳-۲	صفحه ی مهارت ها.....
۱۰-۴-۳-۲	صفحه ی ویرایش مهارت ها.....
۱۱-۵-۳-۲	صفحه ی فعالیت ها.....
۱۴-۶-۳-۲	صفحه ویرایش فعالیت.....
۱۶	فصل ۳: ساختار داده ها و بانک اطلاعات
۱۷-۱-۳	مقدمه.....
۱۷-۲-۳	نمودارهای UML.....
۱۸-۱-۲-۳	نمودار Use Case.....
۱۹-۲-۲-۳	نمودار توالی.....
۲۴-۳-۳	سخت افزارهای مورد نیاز.....
۲۵	فصل ۴: جدول ها و بانک اطلاعاتی
۲۶-۱-۴	مقدمه.....
۲۶-۲-۴	پایگاه داده.....
۲۷-۳-۴	نمودار ERD.....
۲۸-۴-۴	جداول پایگاه داده.....

۳۱	فصل ۵: جمع‌بندی و پیشنهادها
۳۲	۵-۱- نتیجه‌گیری.....
۳۲	۵-۲- پیشنهادهایی برای کارهای آتی.....
۳۳	مراجع
۳۵	پیوست‌ها



## فهرست شکل‌ها

شکل (۱-۲) صفحه اصلی برنامه	۷
شکل (۲-۲) صفحه نمایش اطلاعات کاربر	۸
شکل (۳-۲) صفحه مهارت‌ها	۹
شکل (۴-۲) صفحه ویرایش مهارت‌ها	۱۰
شکل (۵-۲) انتخاب عکس جدید برای مهارت	۱۱
شکل (۶-۲) صفحه فعالیت‌ها	۱۲
شکل (۷-۲) نمودار میزان انجام فعالیت	۱۳
شکل (۸-۲) دیالوگ انتخاب میزان ساعات انجام فعالیت	۱۳
شکل (۹-۲) دیالوگ میزان پیشرفت کاربر	۱۴
شکل (۱۰-۲) صفحه ویرایش فعالیت	۱۵
شکل (۱-۳) Use Case نمودار	۱۸
شکل (۲-۳) نمودارهای توالی View Plan و View Quests	۱۹
شکل (۳-۳) نمودارهای توالی View Skills و View Faaliats	۲۰
شکل (۴-۳) نمودارهای توالی Add new Faaliat و Edit Faaliat	۲۱
شکل (۵-۳) نمودارهای توالی View Faaliat Graph و Delete Faaliat	۲۲
شکل (۶-۳) نمودارهای توالی Add New Skill و Edit Skill	۲۳
شکل (۷-۳) نمودار توالی Delete Skill	۲۴
شکل (۱-۴) ER نمودار	۲۷

## فهرست جدول‌ها

۲۸	جدول (۱-۴) User
۲۸	جدول (۲-۴) Skill
۲۹	جدول (۳-۴) Faaliat
۲۹	جدول (۴-۴) Quest
۲۹	جدول (۵-۴) PlanCell
۲۹	جدول (۶-۴) Quest_Skill
۳۰	جدول (۷-۴) Faaliat_Skill
۳۰	جدول (۸-۴) FaaliatRepetitions
۳۰	جدول (۹-۴) Avatar
۳۰	جدول (۱۰-۴) AvatarItem

# فصل ١:

## مقدمه

## ۱-۱- انگیزه

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 فعالیت‌های شغلی، بسیار مورد توجه قرار گرفته است. به این معنا که فرد برای کسب امتیاز بیشتر و جمع کردن دست‌آوردهای مختلف، این نوع فعالیت‌ها را با رغبت بیشتری انجام می‌دهد. بنابراین تصمیم گرفته شد که با ترکیب این دو مقوله یعنی بهبود سبک زندگی و بازی‌انگاری برنامه‌ای ساخته شود تا افراد بتوانند با جذابیت بیشتری زندگی خود را بهبود ببخشند. نمونه‌های مشابهی در فروشگاه‌های نرم‌افزاری موبایل وجود دارند اما این نمونه‌ها ضعف‌هایی دارند که واضح‌ترین آن‌ها رابط کاربری غیردوستانه و غیر جذاب است.

## ۱-۲- هدف

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

### ۱-۳- 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.

#### ۱-۳-۱- 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 لایسنس Apache License 2.0 به صورت رایگان در دسترس قرار دارد. نسخه ۰,۱ 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 ۱,۰ 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 آسان‌تری دارد و از قابلیت‌های سطح پایین کمتری پشتیبانی می‌کند.

## ۴-۱- نگاهی به فصل‌های آینده

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

## **فصل ۲:**

# **تجزیه و تحلیل نیازمندی‌ها**

## ۱-۲- مقدمه

هدف از این فصل تحلیل نیازمندی‌های سیستم تحت طراحی است. سناریوها، نمودارهای تحلیلی مورد کاربرد، کاربرد، توالی و غیره در این فصل شرح داده شده‌اند.

## ۲-۲- معرفی کاربران برنامه

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

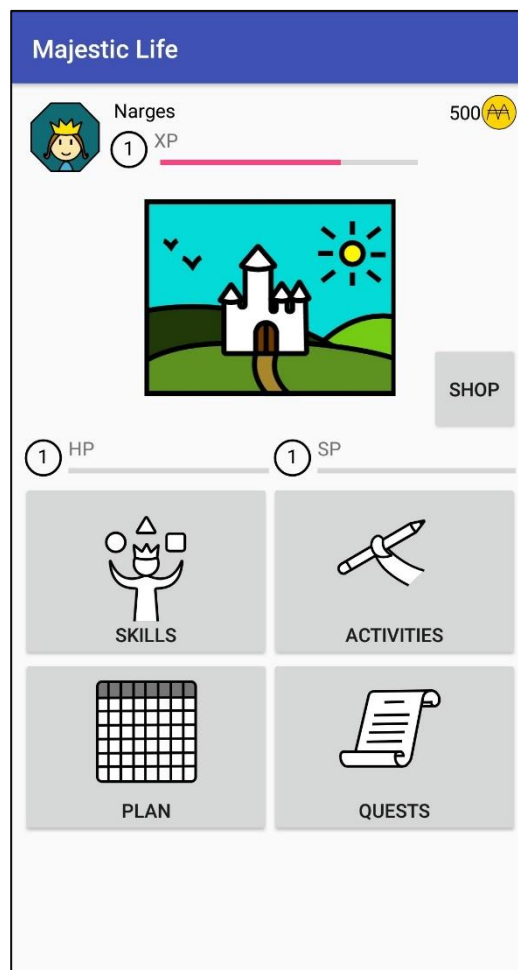
## ۳-۲- رابط گرافیکی نرم‌افزار و معرفی صفحات نرم‌افزار

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



## ۲-۳-۱- صفحه اصلی

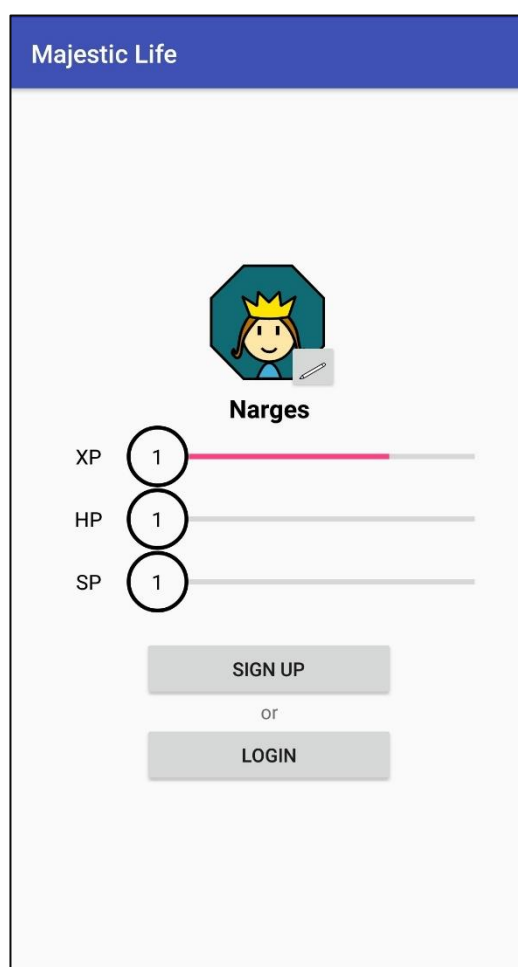
این صفحه، صفحه اصلی و اولیه برنامه می‌باشد. در اینجا، کاربر ۴ دکمه اصلی برنامه که به صفحات "فعالیت‌ها"، "مهارت‌ها"، "برنامه" و "ماموریت‌ها" مسیریابی می‌شوند، مشاهده می‌کند. علاوه بر آن، progress barهایی نیز در این صفحه برای نمایش مقدار و سطح HP، XP و SP کاربر وجود دارند. در بالا سمت راست صفحه نیز، تعداد سکه‌ها و در سمت چپ صفحه هم آدامک کاربر قرار دارد که با زدن روی تصویر آدامک به صفحه ورود (login) مسیریابی می‌شود.



شکل (۲-۱) صفحه اصلی برنامه

## ۲-۳-۲- صفحه‌ی نمایش اطلاعات کاربر

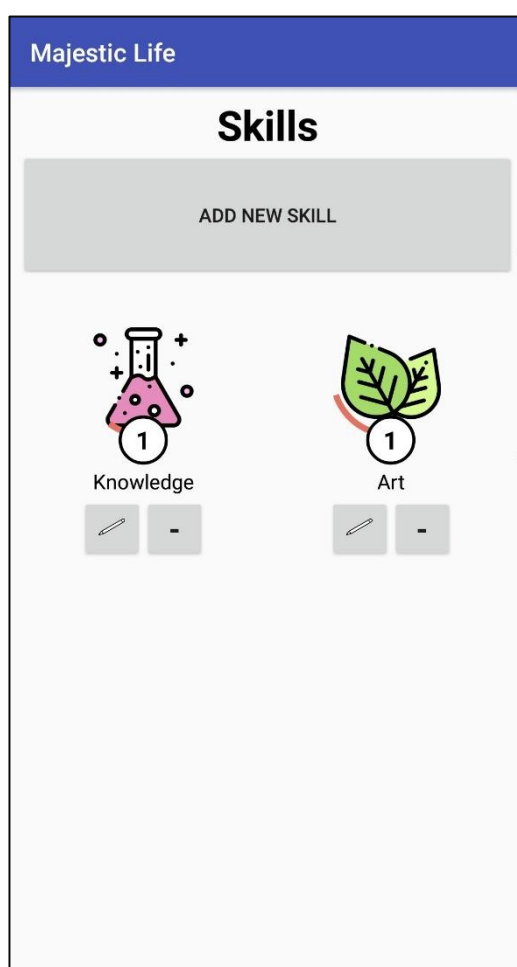
در این صفحه کاربر می‌تواند Sign Up یا Login انجام دهد. هم‌چنین کاربر می‌تواند آدمک خود و میزان پیشرفتش در هریک از XP، HP و SP را مشاهده کند. با زدن روی دکمه کنار آدمک، دکمه ویرایش، کاربر به صفحه فروشگاه آدمک منتقل می‌شود.



شکل (۲-۲) صفحه‌ی نمایش اطلاعات کاربر

## ۲-۳-۳- صفحه‌ی مهارت‌ها

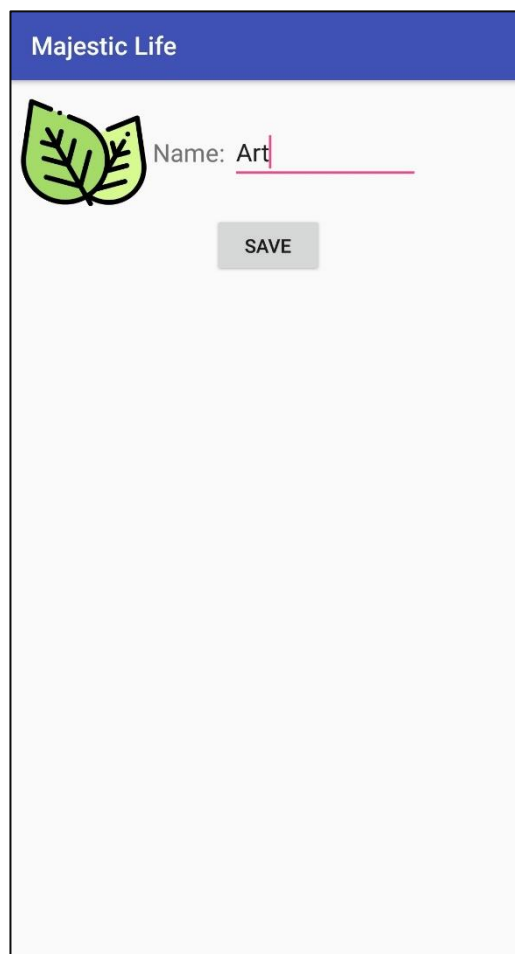
در این صفحه، کاربر لیستی از مهارت‌های خود را مشاهده می‌کند که برای هر کدام از آن‌ها نیز، دکمه‌های حذف و ویرایش آن مهارت قرار دارد. در بالای صفحه دکمه‌ی اضافه کردن مهارت جدید (Add New Skill) وجود دارد که کاربر با کلیک بر روی آن، یک مهارت جدید به مهارت‌های خود افزوده و به صفحه‌ی ویرایش مهارت جدید می‌رود.



شکل (۲-۳) صفحه مهارت‌ها

## ۲-۳-۴- 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 Dialog قرار دارند انتخاب کند. با زدن روی دکمه ذخیره (Save) نیز تغییرات خود را اعمال کرده و به صفحه مهارت‌ها باز می‌گردد.



شکل (۲-۴) صفحه ویرایش مهارت‌ها



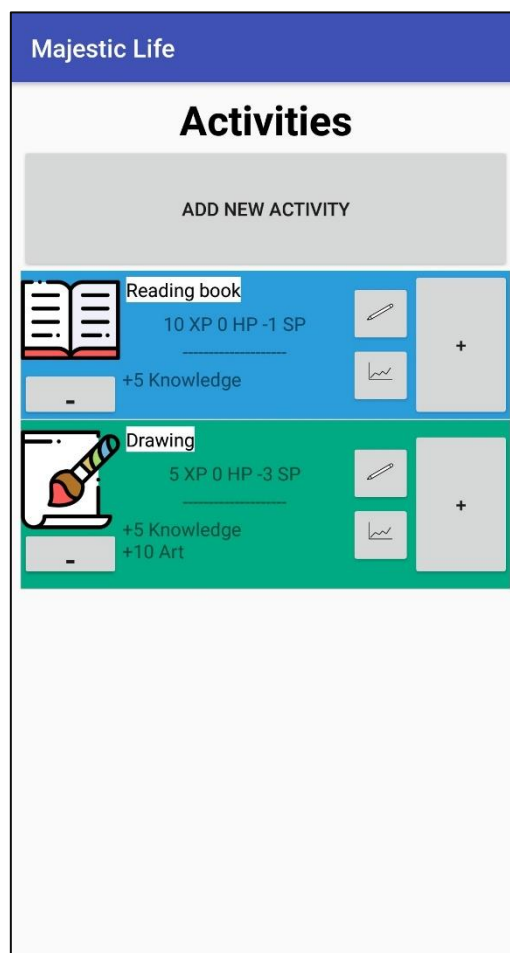
شکل (۵-۲) انتخاب عکس جدید برای مهارت

### ۵-۳-۲- صفحه‌ی فعالیت‌ها

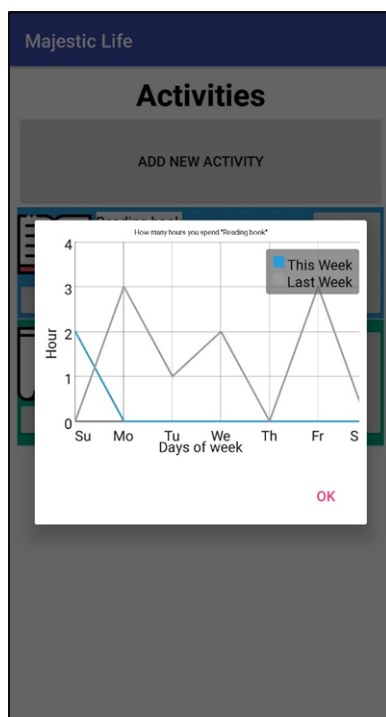
در این صفحه، کاربر لیستی از فعالیت‌های خود را مشاهده می‌کند که برای هر کدام از آن‌ها نیز دکمه‌های حذف، ویرایش، نمودار و پیشرفت (+) آن فعالیت قرار دارد و هر کدام رنگ و تصویر خاص خود را دارند. در بالای صفحه دکمه‌ی اضافه کردن فعالیت جدید (Add New Activity) وجود دارد که کاربر با کلیک بر روی آن، یک فعالیت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

با زدن روی دکمه نمودار هر فعالیت، نموداری از میزان انجام همان فعالیت در روزهای

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



شکل (۲-۶) صفحه فعالیت‌ها



شکل (۲-۷) نمودار میزان انجام فعالیت

How many hours were you doing this activity?

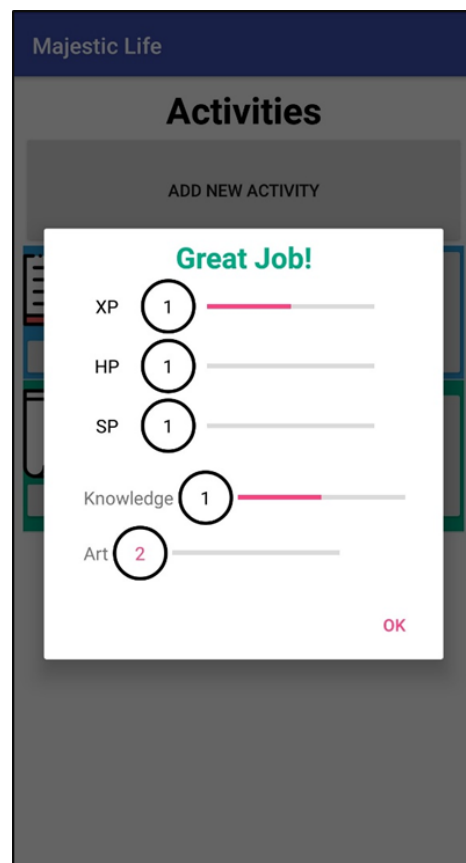
1

2

3

CANCEL OK

شکل (۲-۸) دیالوگ انتخاب میزان ساعات انجام فعالیت



شکل (۹-۲) دیالوگ میزان پیشرفت کاربر

## ۲-۳-۶- 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 Dialog 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 SP را نیز که با انجام این فعالیت به دست می‌آورد، مشخص کند. در قسمت پایینی صفحه، لیستی از مهارت‌هایی قرار دارند که می‌توان به آن‌ها 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.



Majestic Life

SAVE

Name: Reading book 

9

10

-10

XP

-1

0

1

HP

-2


1

0

SP

Skills:

ADD SKILL

Knowledge ▾ X

4

5

6

X

شکل (۲-۱۰) صفحه ویرایش فعالیت

## **فصل ۳:**

# **ساختار داده‌ها و بانک اطلاعات**

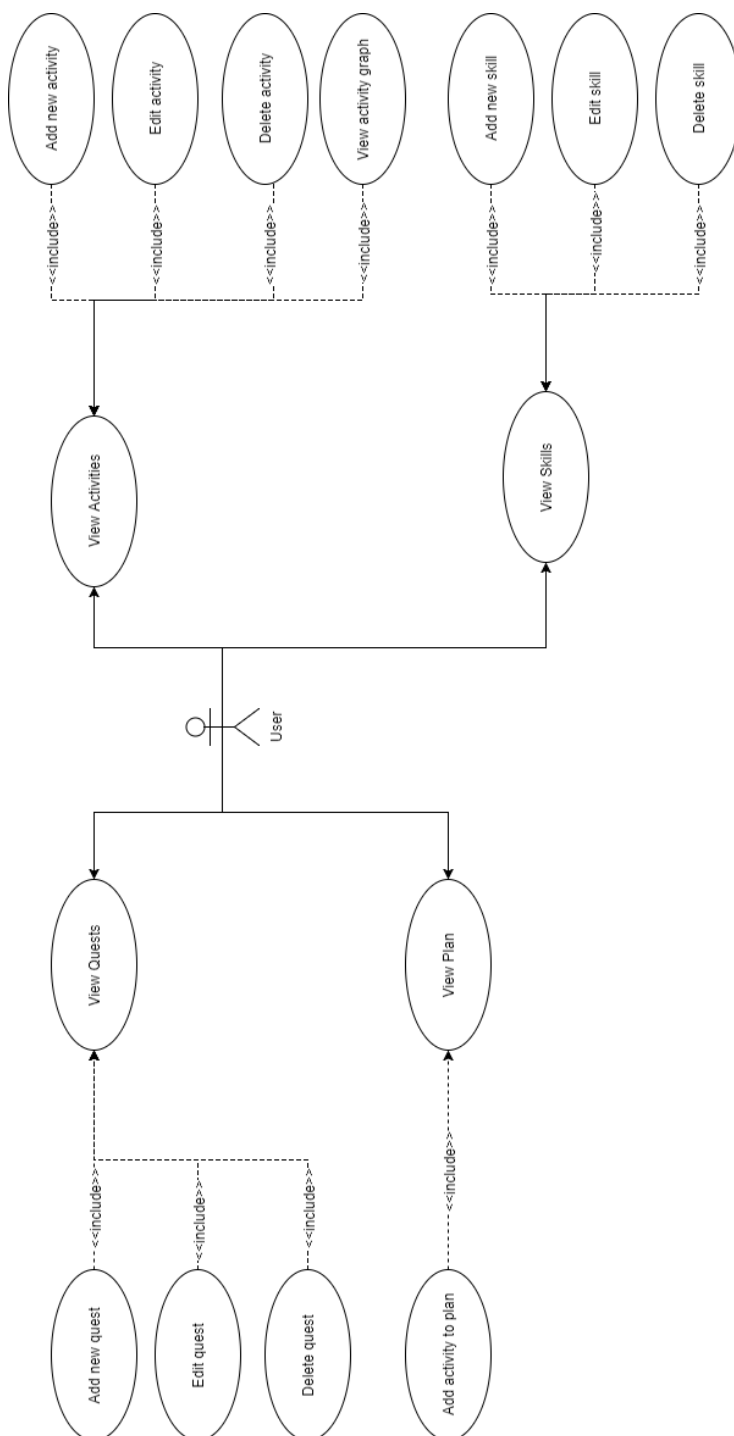
### ۱-۳- مقدمه

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.

### ۲-۳- نمودارهای UML

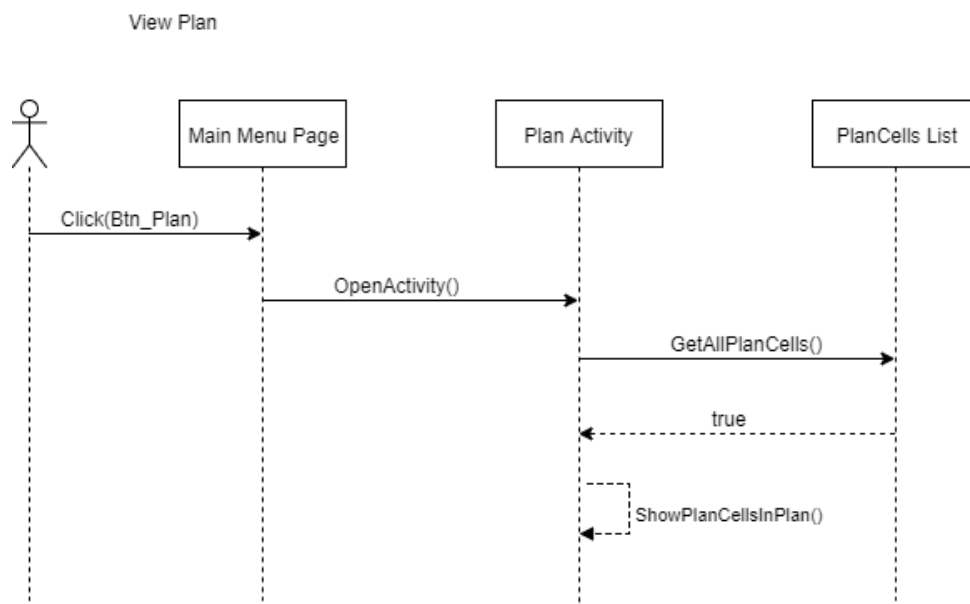
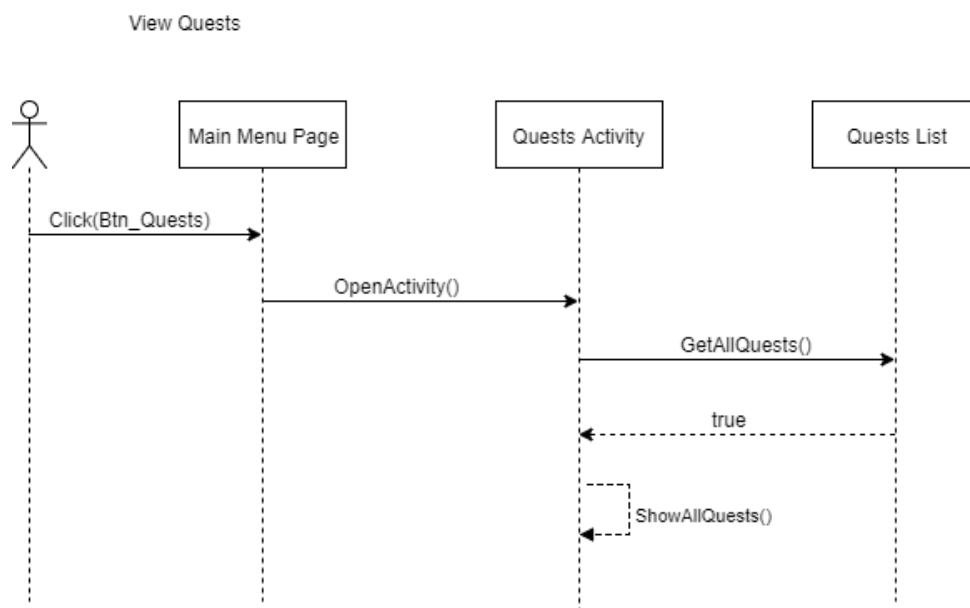
در این بخش نمودارهای Usecase (مورد کاربرد) و نمودارهای توالی مربوط به برنامه رسم شده‌اند.

### ۳-۲-۱- نمودار Use Case

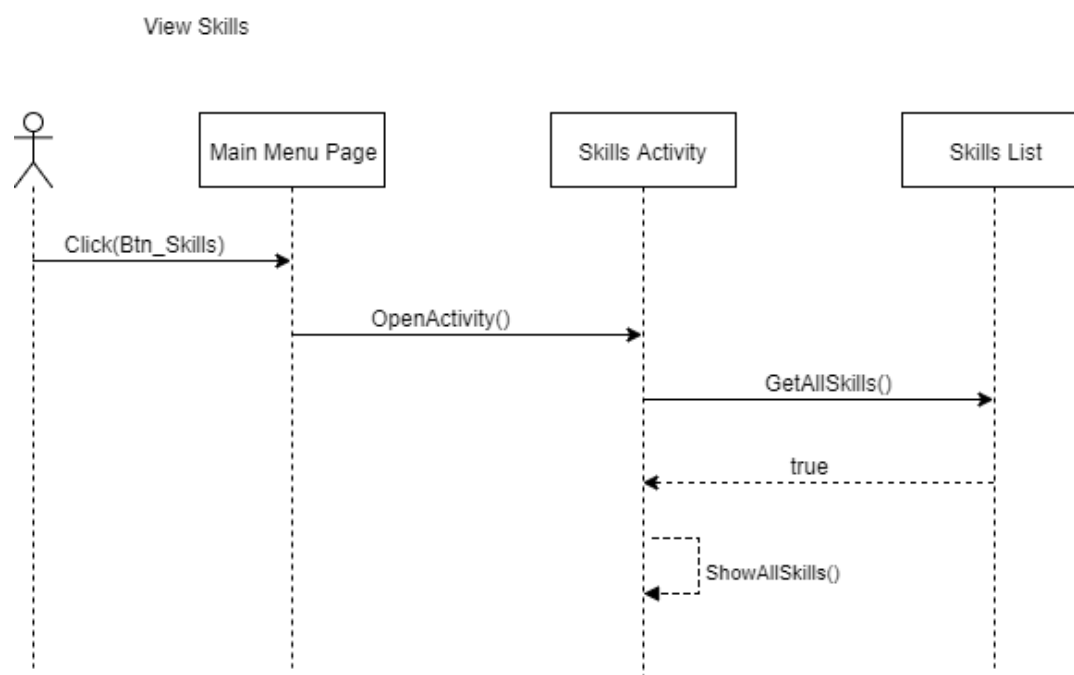
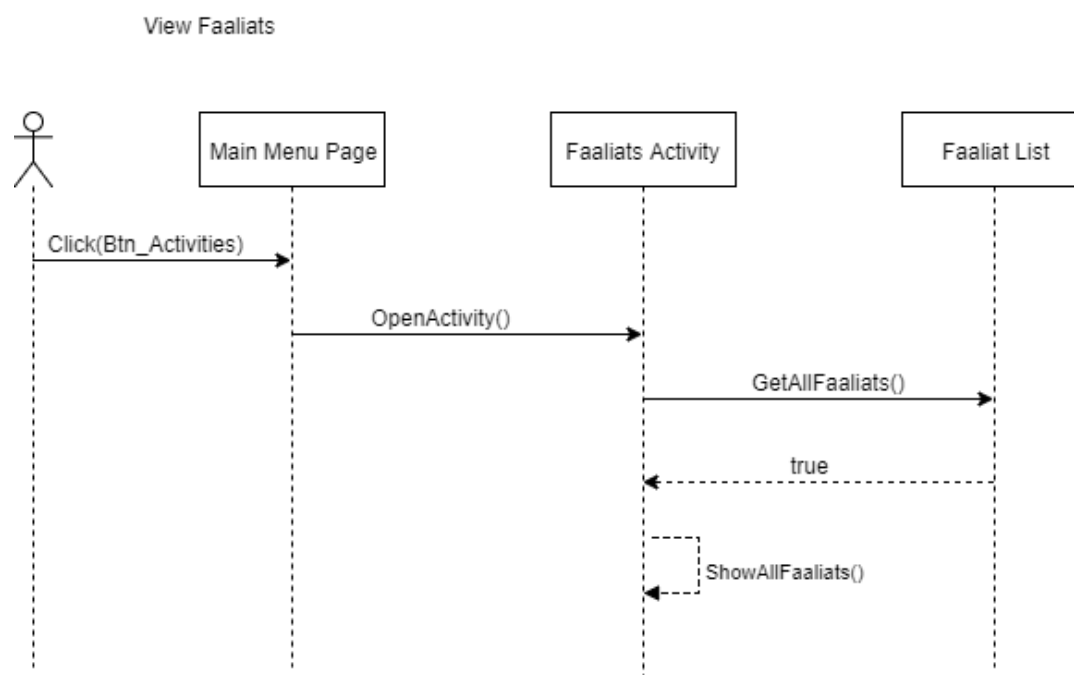


شکل (۳-۱) نمودار Use Case

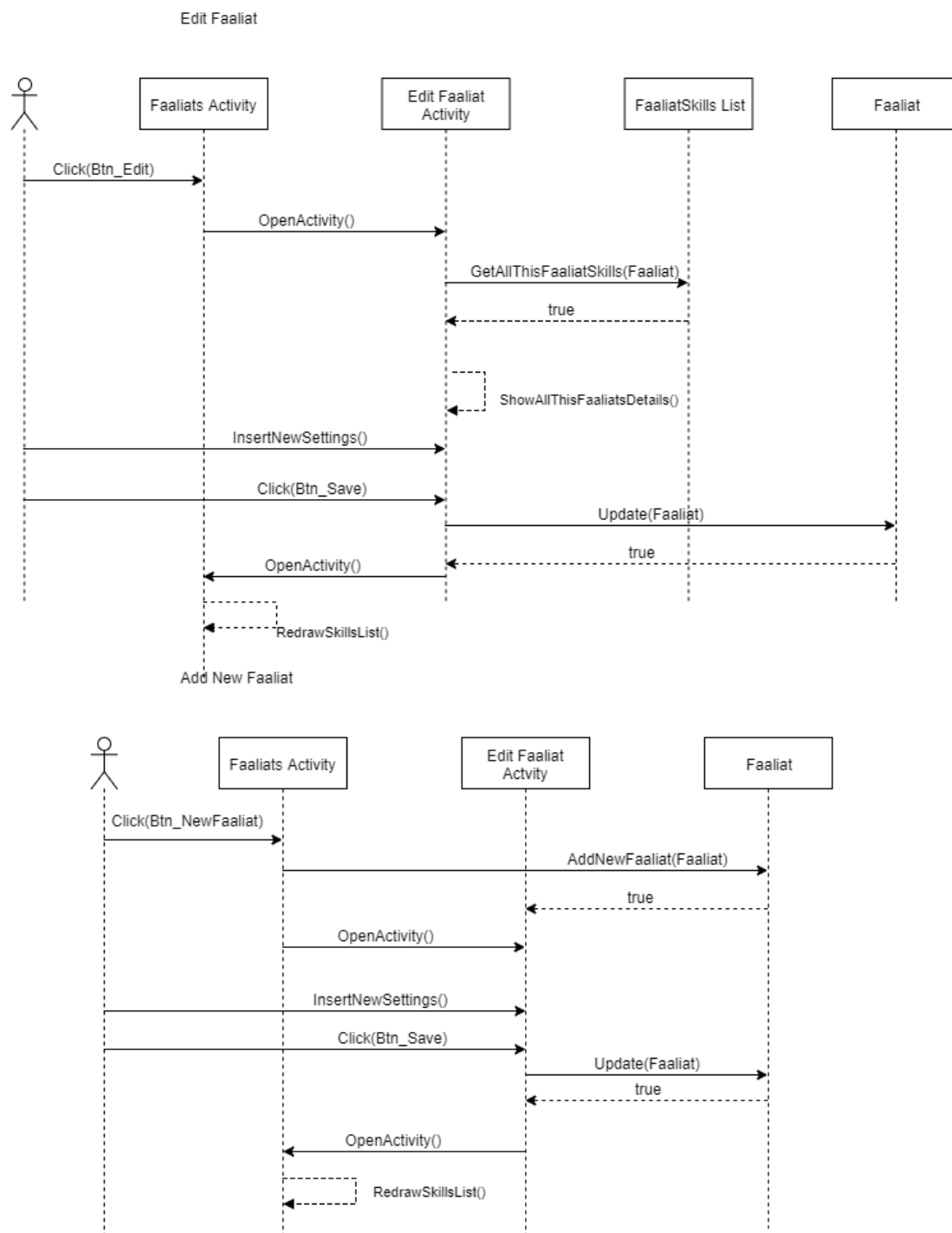
### ۲-۲-۳- نمودار توالی



شکل (۲-۳) نمودارهای توالی View Plan و View Quests

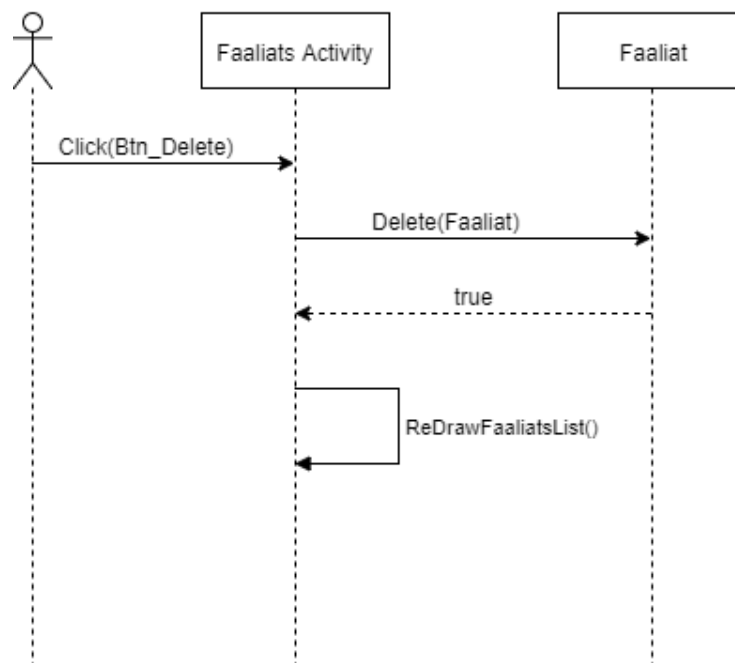


شکل (۳-۳) نمودارهای توالی View Skills و View Faaliats

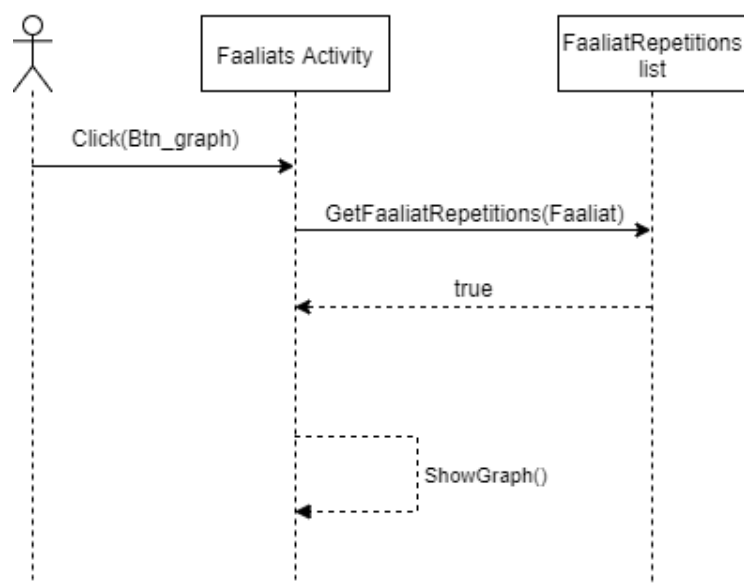


شکل (۳-۴) نمودارهای توالی Edit Faaliat و Add new Faaliat

### Delete Faaliat

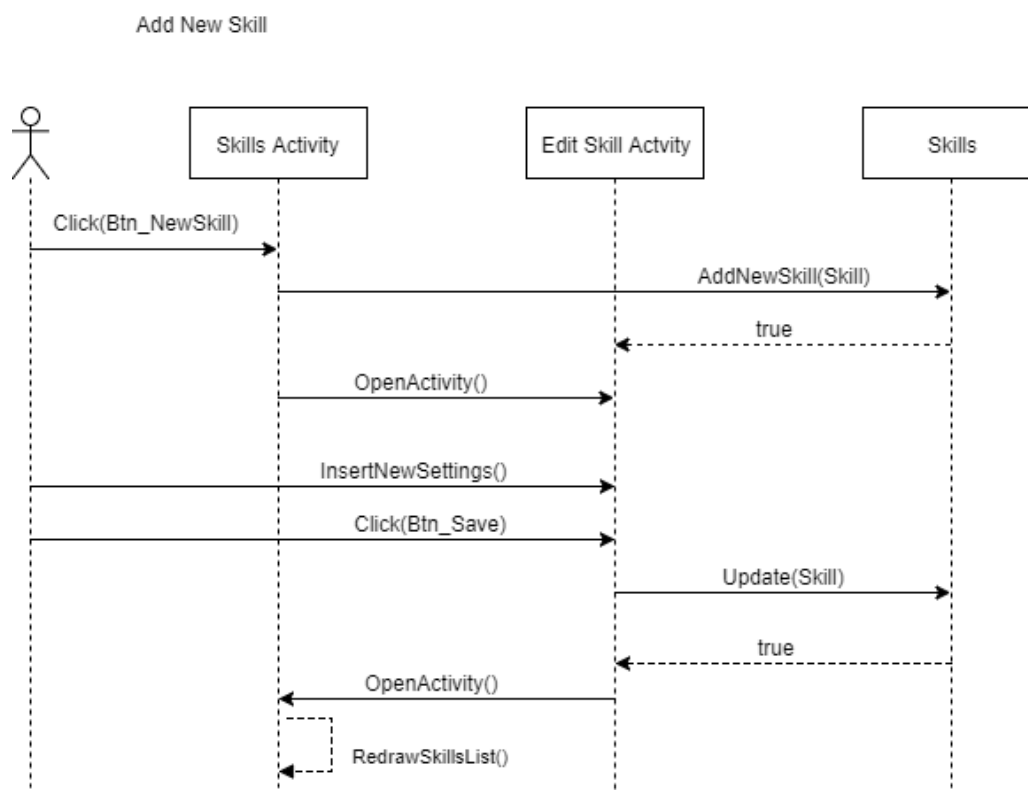
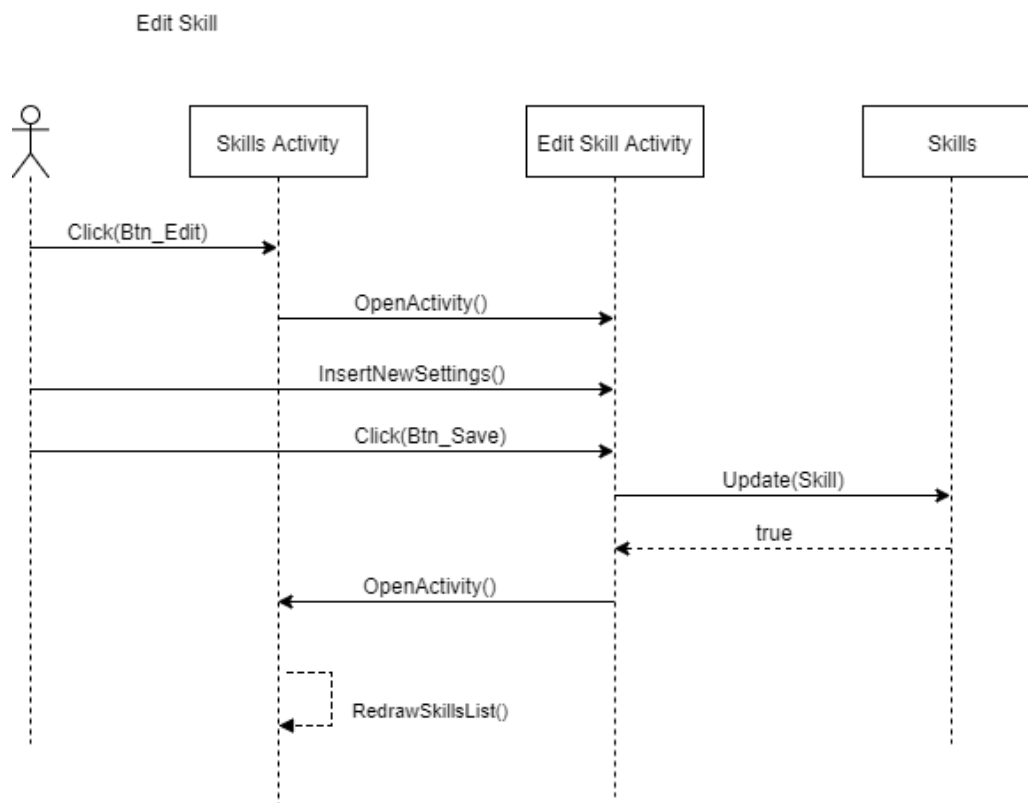


### View Faaliat Graph

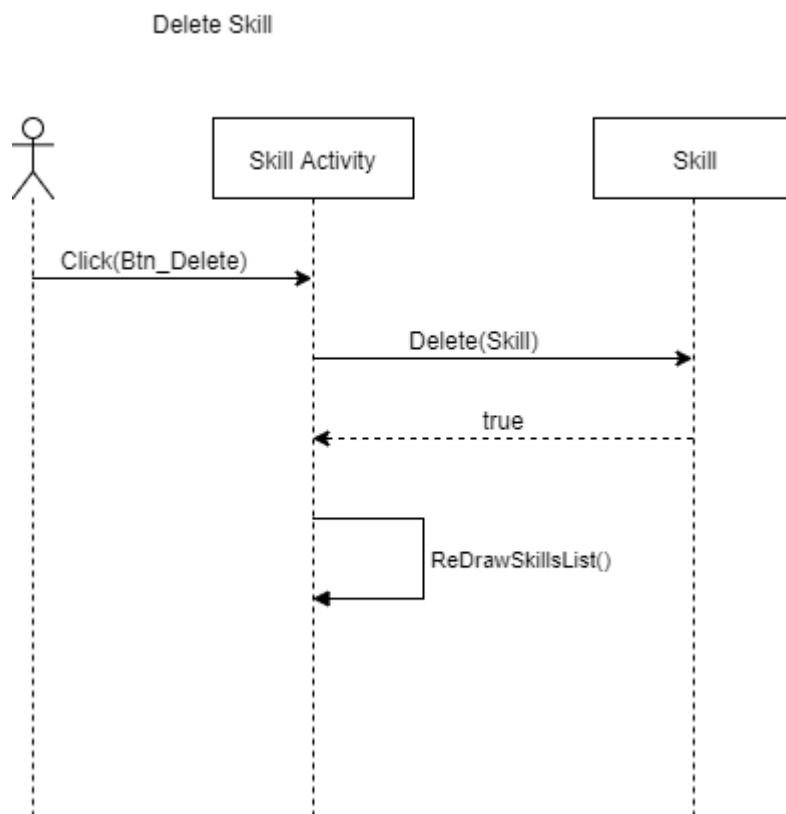


شکل (۳-۵) نمودارهای توالی Delete Faaliat و View Faaliat Graph





شکل (۳-۶) نمودارهای توالی Edit Skill و Add New Skill



شکل (۷-۳) نمودار توالی Delete Skill

### ۳-۳- ساخت افزارهای 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.

## **فصل ۴:**

### **جدول‌ها و بانک اطلاعاتی**

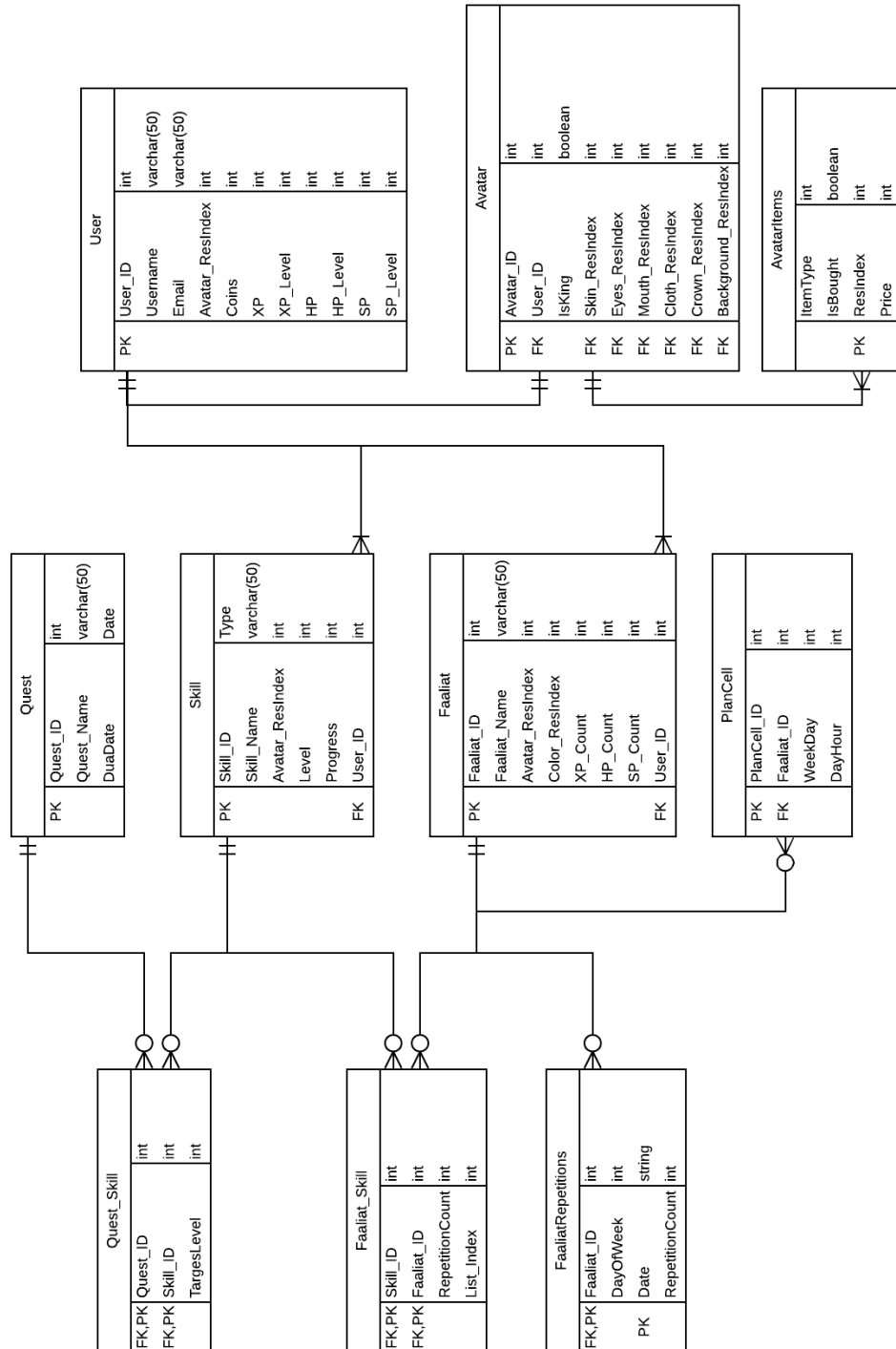
#### ۴-۱- مقدمه

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

#### ۴-۲- پایگاه داده

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

## ۳-۴- نمودار ERD



شکل (۴-۱) نمودار ER

## ۴-۴- جداول پایگاه داده

جدول (۴-۱) User

ویژگی				نام	ردیف
واحد	طول	نوع	کلید اصلی		
بایت	۲	int	PK	User_ID	۱
بایت	۸	string	-	Username	۲
بایت	۸	string	-	Email	۳
بایت	۲	int	-	Coins	۴
بایت	۲	int	-	XP	۵
بایت	۲	int	-	XP_Level	۶
بایت	۲	int	-	HP	۷
بایت	۲	int	-	HP_Level	۸
بایت	۲	int	-	SP	۹
بایت	۲	Int	-	SP_Level	۱۰

جدول (۴-۲) Skill

ویژگی				نام	ردیف
واحد	طول	نوع	کلید اصلی		
بایت	۲	int	PK	Skill_ID	۱
بایت	۸	String	-	Skill_Name	۲
بایت	۲	int	-	Avatar_ResIndex	۳
بایت	۲	int	-	Level	۴
بایت	۲	int	-	Progress	۵
بایت	۲	int	FK	User_ID	۶

جدول (۳-۴) Faaliat

ویژگی				نام	ردیف
واحد	طول	نوع	کلید اصلی		
بایت	۲	int	PK	Faaliat_ID	۱
بایت	۸	String	-	Faaliat_Name	۲
بایت	۲	int	-	Avatar_ResIndex	۳
بایت	۲	int	-	Color_ResIndex	۴
بایت	۲	int	-	XP_Count	۵
بایت	۲	int	-	HP_Count	۶
بایت	۲	int	-	SP_Count	۷
بایت	۲	int	FK	User_ID	۸

جدول (۴-۴) Quest

ویژگی				نام	ردیف
واحد	طول	نوع	کلید اصلی		
بایت	2	int	PK	Quest_ID	۱
بایت	8	String	-	Quest_Name	۲
بایت	8	String	-	DuaDate	۳

جدول (۵-۴) PlanCell

ویژگی				نام	ردیف
واحد	طول	نوع	کلید اصلی		
بایت	2	int	PK	PlanCell_ID	۱
بایت	2	int	FK	Faaliat_ID	۲
بایت	2	int		WeekDay	۳
بایت	2	int		DayHour	۴

جدول (۶-۴) Quest\_Skill

ویژگی				نام	ردیف
واحد	طول	نوع	کلید اصلی		
بایت	2	int	PK, FK	Quest_ID	۱
بایت	2	int	PK, FK	Skill_ID	۲
بایت	2	int	-	TargetLevel	۳

جدول (۴-۷) Faaliat\_Skill

ردیف	نام	ویژگی		
		کلید اصلی	نوع	طول
۱	Skill_ID	FK, PK	int	2
۲	Faaliat_ID	FK, PK	int	2
۳	RepetitionCount	-	int	2
۴	ListIndex	-	int	2

جدول (۴-۸) FaaliatRepetitions

ردیف	نام	ویژگی		
		کلید اصلی	نوع	طول
۱	Faaliat_ID	PK, FK	int	2
۲	DayOfWeek	-	int	2
۳	Date	PK	String	8
۴	RepetitionCount	-	int	2

جدول (۴-۹) Avatar

ردیف	نام	ویژگی		
		کلید اصلی	نوع	طول
۱	Avatar_ID	PK	int	2
۲	User_ID	FK	int	2
۳	IsKing	-	boolean	1
۴	Skin_ResIndex	FK	int	2
۵	Eyes_ResIndex	FK	int	2
۶	Mouth_ResIndex	FK	int	2
۷	Cloth_ResIndex	FK	int	2
۸	Crown_ResIndex	FK	int	2
۹	Background_ResIndex	FK	int	2

جدول (۴-۱۰) AvatarItem

ردیف	نام	ویژگی		
		کلید اصلی	نوع	طول
۱	ItemType	-	int	2
۲	IsBought	-	boolean	1
۳	ResIndex	PK	int	2
۴	Price	-	int	2



## **فصل ۵:**

### **جمع بندي و پيشنهادها**

## ۱-۵- نتیجه‌گیری

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

## ۲-۵- پیشنهادهایی برای کارهای آتی

در این پروژه بخش‌های Quests و Planner، به علت کمبود وقت تکمیل نشده‌اند؛ ولی با توجه به ساختار شی‌گرای برنامه، می‌توان به راحتی این قسمت‌ها را در آینده به برنامه اضافه کرد. همچنین می‌توان بخش‌های مربوط به فروشگاه و تصاویر گرافیکی برنامه را نیز بهبود بخشید. به دلیل اینکه این برنامه با مفاهیم روانشناختی افراد نیز ارتباط نزدیکی می‌تواند داشته باشد، مشورت با متخصصین این حوزه برای بهبود هرچه بهتر برنامه امری ضروری به نظر رسیده و توصیه می‌شود.

## مراجع

## مراجع

- [1] <https://developer.android.com>
- [2] <https://codelabs.developers.google.com/codelabs/android-room-with-a-view/#0>
- [3] <https://android.jlelse.eu/android-architecture-components-room-introduction-4774dd72a1ae>
- [4] <https://medium.com/@srinuraop/database-create-and-open-callbacks-in-room-7ca98c3286ab>
- [5] <https://www.flaticon.com/packs/user-interface-30/2>

[۶] کتاب پایگاه داده ی رانکوهی

[۷] کتاب های مهندسی نرم افزار یک و دو پرس من

# پیوست‌ها

## پیوست الف: کدهای پروژه

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

```
package com.farazannajmi.majesticlife.DataStructures;

import android.arch.persistence.room.ColumnInfo;
import android.arch.persistence.room.Entity;
import android.arch.persistence.room.PrimaryKey;

import com.farazannajmi.majesticlife.R;

import javax.annotation.Nonnull;

@Entity(tableName = "User_table")
public class User
{
    @PrimaryKey
    @Nonnull
    @ColumnInfo(name = "User_ID")
    private int User_ID;

    @ColumnInfo(name = "Username")
    private String Username;

    @ColumnInfo(name = "Email")
    private String Email;

    @ColumnInfo(name = "XP")
    private int XP;
    @ColumnInfo(name = "XpLevel")
    private int XpLevel;

    @ColumnInfo(name = "HP")
    private int HP;
    @ColumnInfo(name = "HpLevel")
    private int HpLevel;

    @ColumnInfo(name = "SP")
    private int SP;
    @ColumnInfo(name = "SpLevel")
    private int SpLevel;

    @ColumnInfo(name = "Coins")
    private int Coins;

    @Nonnull
    public int getUser_ID() {return this.User_ID;}
    public String getUsername() {return this.Username;}
    public String getEmail() {return this.Email;}
    public int getXP() {return this.XP;}
    public int getXpLevel() {return this.XpLevel;}
    public int getHP() {return this.HP;}
    public int getHpLevel() {return this.HpLevel;}
    public int getSP() {return this.SP;}
    public int getSpLevel() {return this.SpLevel;}
    public int getCoins() {return this.Coins;}

    public void setUser_ID(int User_ID) {this.User_ID = User_ID;}
```

```

public void setUsername(String Username){this.Username = Username;}
public void setEmail(String Email){this.Email = Email;}
public void setXP(int XP){this.XP = XP;}
public void setXpLevel(int XpLevel){this.XpLevel = XpLevel;}
public void setHP(int HP){this.HP = HP;}
public void setHpLevel(int HpLevel){this.HpLevel = HpLevel;}
public void setSP(int SP){this.SP = SP;}
public void setSpLevel(int SpLevel){this.SpLevel = SpLevel;}
public void setCoins(int Coins){this.Coins = Coins;}

public User(int User_ID, String Username, String Email,
            int XP, int XpLevel, int HP, int HpLevel, int SP, int
SpLevel, int Coins)
{
    this.User_ID = User_ID;
    this.Username = Username;
    this.Email = Email;
    this.XP = XP;
    this.XpLevel = XpLevel;
    this.HP = HP;
    this.HpLevel = HpLevel;
    this.SP = SP;
    this.SpLevel = SpLevel;
    this.Coins = Coins;
}

```

```

package com.farazannajmi.majesticlife.DataStructures;

import android.arch.persistence.room.ColumnInfo;
import android.arch.persistence.room.Entity;
import android.arch.persistence.room.ForeignKey;
import android.arch.persistence.room.Ignore;
import android.arch.persistence.room.PrimaryKey;

import java.util.ArrayList;

import javax.annotation.Nonnull;

import static android.arch.persistence.room.ForeignKey.CASCADE;

@Entity(tableName = "Faaliat_table",
        foreignKeys = @ForeignKey(entity = User.class,
        parentColumns = "User_ID",
        childColumns = "User_ID",
        onDelete = CASCADE))
public class Faaliat
{
    @PrimaryKey//(autoGenerate = true)
    @Nonnull
    @ColumnInfo(name = "Faaliat_ID")
    private int Faaliat_ID;

    @ColumnInfo(name = "Faaliat_Name")
    private String Faaliat_Name;

    @ColumnInfo(name = "Avatar_ResIndex")
    private int Avatar_ResIndex;
    @ColumnInfo(name = "Color_ResIndex")
    private int Color_ResIndex;

    @ColumnInfo(name = "XpCount")
    private int XpCount;
    @ColumnInfo(name = "HpCount")

```

```

private int HpCount;
@ColumnInfo(name = "SpCount")
private int SpCount;

@ColumnInfo(name = "User_ID")
private int User_ID;

@Ignore
public ArrayList<FaaliatSkill> FaaliatSkills;

@NonNull
public int getFaaliat_ID() {return Faaliat_ID;}
public String getFaaliat_Name() {return Faaliat_Name;}
public int getAvatar_ResIndex() {return Avatar_ResIndex;}
public int getColor_ResIndex() {return Color_ResIndex;}
public int getXpCount() {return XpCount;}
public int getHpCount() {return HpCount;}
public int getSpCount() {return SpCount;}
public int getUser_ID() {return User_ID;}

public void setFaaliat_ID(int Faaliat_ID){this.Faaliat_ID = Faaliat_ID;}
public void setFaaliat_Name(String Faaliat_Name){this.Faaliat_Name =
Faaliat_Name;}
public void setAvatar_ResIndex(int Avatar_ResIndex){this.Avatar_ResIndex
= Avatar_ResIndex;}
public void setColor_ResIndex(int Color_ResIndex){this.Color_ResIndex =
Color_ResIndex;}
public void setXpCount(int XpCount){this.XpCount = XpCount;}
public void setHpCount(int HpCount){this.HpCount = HpCount;}
public void setSpCount(int SpCount){this.SpCount = SpCount;}
public void setUser_ID(int User_ID){this.User_ID = User_ID;}

public Faaliat(@NonNull int Faaliat_ID, String Faaliat_Name, int
Avatar_ResIndex, int Color_ResIndex,
int XpCount, int HpCount, int SpCount, int User_ID)
{
    this.Faaliat_ID = Faaliat_ID;
    this.Faaliat_Name = Faaliat_Name;
    this.Avatar_ResIndex = Avatar_ResIndex;
    this.Color_ResIndex = Color_ResIndex;
    this.XpCount = XpCount;
    this.HpCount = HpCount;
    this.SpCount = SpCount;
    this.User_ID = User_ID;
}
}

```

```

package com.farazannajmi.majesticlife.DataStructures;

import android.arch.persistence.room.ColumnInfo;
import android.arch.persistence.room.Entity;
import android.arch.persistence.room.ForeignKey;
import android.arch.persistence.room.PrimaryKey;

import javax.annotation.NonNull;

import static android.arch.persistence.room.ForeignKey.CASCADE;

@Entity(tableName = "Skill_table",
    foreignKeys = @ForeignKey(entity = User.class,
        parentColumns = "User_ID",
        childColumns = "User_ID",

```



```

        onDelete = CASCADE))
public class Skill
{
    @NonNull
    @PrimaryKey
    @ColumnInfo(name = "Skill_ID")
    private int Skill_ID;

    @ColumnInfo(name = "Skill_Name")
    private String Skill_Name;

    @ColumnInfo(name = "Avatar_ResIndex")
    private int Avatar_ResIndex;

    @ColumnInfo(name = "Level")
    private int Level;

    @ColumnInfo(name = "Progress")
    private int Progress; //the increasing value to reach the next level

    @ColumnInfo(name = "User_ID")
    private int User_ID;

    @NonNull
    public int getSkill_ID(){return this.Skill_ID;}
    public String getSkill_Name(){return this.Skill_Name;}
    public int getAvatar_ResIndex(){return this.Avatar_ResIndex;}
    public int getLevel(){return this.Level;}
    public int getProgress(){return this.Progress;}
    public int getUser_ID(){return this.User_ID;}

    public void setSkill_ID(int Skill_ID){this.Skill_ID = Skill_ID;}
    public void setSkill_Name (String Skill_Name){this.Skill_Name =
Skill_Name;}
    public void setAvatar_ResIndex (int
Avatar_ResIndex){this.Avatar_ResIndex = Avatar_ResIndex;}
    public void setLevel (int Level){this.Level = Level;}
    public void setProgress (int Progress){this.Progress = Progress;}
    public void setUser_ID (int User_ID){this.User_ID = User_ID;}

    public Skill (int Skill_ID, String Skill_Name, int Avatar_ResIndex, int
Level, int Progress, int User_ID)
    {
        this.Skill_ID = Skill_ID;
        this.Skill_Name = Skill_Name;
        this.Avatar_ResIndex = Avatar_ResIndex;
        this.Level = Level;
        this.Progress = Progress;
        this.User_ID = User_ID;
    }
}

```

---

```

package com.farazannajmi.majesticlife.DataStructures;

import android.arch.persistence.room.ColumnInfo;
import android.arch.persistence.room.Entity;
import android.arch.persistence.room.PrimaryKey;

import java.sql.Time;
import java.util.List;

import javax.annotation.NonNull;

```

```

@Entity(tableName = "Quest_table")
public class Quest
{
    @PrimaryKey
    @NonNull
    @ColumnInfo(name = "Quest_ID")
    private int Quest_ID;

    @ColumnInfo(name = "Quest_Name")
    private String Quest_Name;

    @ColumnInfo(name = "DuaDate")
    private String DuaDate;

    @NonNull
    public int getQuest_ID() {return Quest_ID;}
    public String getQuest_Name() {return Quest_Name;}
    public String getDuaDate() {return DuaDate;}

    public void setQuest_ID(@NonNull int quest_ID) {Quest_ID = quest_ID;}
    public void setQuest_Name(String quest_Name) {Quest_Name = quest_Name;}
    public void setDuaDate(String duaDate) {DuaDate = duaDate;}

    public Quest(@NonNull int Quest_ID, String Quest_Name, String DuaDate)
    {
        this.Quest_ID = Quest_ID;
        this.Quest_Name = Quest_Name;
        this.DuaDate = DuaDate;
    }
}

```

```

package com.farazannajmi.majesticlife.DataStructures;

import android.arch.persistence.room.ColumnInfo;
import android.arch.persistence.room.Entity;
import android.arch.persistence.room.ForeignKey;
import android.arch.persistence.room.PrimaryKey;

import javax.annotation.Nonnull;

import static android.arch.persistence.room.ForeignKey.CASCADE;

@Entity(tableName = "PlanCell_table",
        foreignKeys = @ForeignKey(entity = Faaliat.class,
                parentColumns = "Faaliat_ID",
                childColumns = "Faaliat_ID",
                onDelete = CASCADE))
public class PlanCell
{
    @PrimaryKey
    @NonNull
    @ColumnInfo(name = "PlanCell_ID")
    private int PlanCell_ID;

    @NonNull
    @ColumnInfo(name = "Faaliat_ID")
    private int Faaliat_ID;

    @ColumnInfo(name = "WeekDay")
    private int WeekDay;
    @ColumnInfo(name = "DayHour")
    private int DayHour;
}

```

```

@NonNull
public int getPlanCell_ID() {return this.PlanCell_ID;}
public int getFaaliat_ID() {return this.Faaliat_ID;}
public int getWeekDay() {return this.WeekDay;}
public int getDayHour() {return this.DayHour;}

public void setPlanCell_ID(int PlanCell_ID) {this.PlanCell_ID =
PlanCell_ID;}
public void setFaaliat_ID(int Faaliat_ID) {this.Faaliat_ID = Faaliat_ID;}
public void setWeekDay(int WeekDay) {this.WeekDay = WeekDay;}
public void setDayHour(int DayHour) {this.DayHour = DayHour;}

public PlanCell (int PlanCell_ID, int Faaliat_ID, int WeekDay, int
DayHour)
{
    this.PlanCell_ID = PlanCell_ID;
    this.Faaliat_ID = Faaliat_ID;
    this.WeekDay = WeekDay;
    this.DayHour = DayHour;
}
}

```

```

package com.farazannajmi.majesticlife.DataStructures;

import android.arch.persistence.room.ColumnInfo;
import android.arch.persistence.room.Entity;
import android.arch.persistence.room.ForeignKey;
import android.arch.persistence.room.PrimaryKey;

import javax.annotation.NonNull;

import static android.arch.persistence.room.ForeignKey.CASCADE;

@Entity(tableName = "Avatar_table",
        foreignKeys = {
            @ForeignKey(entity = User.class,
                parentColumns = "User_ID",
                childColumns = "User_ID",
                onDelete = CASCADE)})
public class Avatar
{
    @NonNull
    @PrimaryKey
    @ColumnInfo(name = "Avatar_ID")
    private int Avatar_ID;

    @ColumnInfo(name = "User_ID")
    private int User_ID;

    @ColumnInfo(name = "IsKing")
    private boolean IsKing;

    @ColumnInfo(name = "Background_ResIndex")
    private int Background_ResIndex;

    @ColumnInfo(name = "Skin_ResIndex")
    private int Skin_ResIndex;

    @ColumnInfo(name = "Cloth_ResIndex")
    private int Cloth_ResIndex;
}

```

```

@ColumnInfo(name = "Eyes_ResIndex")
private int Eyes_ResIndex;

@ColumnInfo(name = "Mouth_ResIndex")
private int Mouth_ResIndex;

@ColumnInfo(name = "Crown_ResIndex")
private int Crown_ResIndex;

@NonNull
public int getAvatar_ID() {return Avatar_ID;}
public int getUser_ID() {return User_ID;}
public boolean getIsKing() {return IsKing;}
public int getBackground_ResIndex() {return Background_ResIndex;}
public int getSkin_ResIndex() {return Skin_ResIndex;}
public int getCloth_ResIndex() {return Cloth_ResIndex;}
public int getEyes_ResIndex() {return Eyes_ResIndex;}
public int getMouth_ResIndex() {return Mouth_ResIndex;}
public int getCrown_ResIndex() {return Crown_ResIndex;}

public void setAvatar_ID(@NonNull int Avatar_ID) {this.Avatar_ID =
Avatar_ID;}
public void setUser_ID(int User_ID) {this.User_ID = User_ID;}
public void setIsKing(boolean IsKing) {this.IsKing = IsKing;}
public void setBackground_ResIndex(int Background_ResIndex)
{this.Background_ResIndex = Background_ResIndex;}
public void setSkin_ResIndex(int Skin_ResIndex) {this.Skin_ResIndex =
Skin_ResIndex;}
public void setCloth_ResIndex(int Cloth_ResIndex) {this.Cloth_ResIndex =
Cloth_ResIndex;}
public void setEyes_ResIndex(int Eyes_ResIndex) {this.Eyes_ResIndex =
Eyes_ResIndex;}
public void setMouth_ResIndex(int Mouth_ResIndex) {this.Mouth_ResIndex =
Mouth_ResIndex;}
public void setCrown_ResIndex(int Crown_ResIndex) {this.Crown_ResIndex =
Crown_ResIndex;}

public Avatar(@NonNull int Avatar_ID, int User_ID, boolean IsKing, int
Background_ResIndex,
              int Skin_ResIndex, int Cloth_ResIndex, int Eyes_ResIndex,
int Mouth_ResIndex, int Crown_ResIndex)
{
    this.Avatar_ID = Avatar_ID;
    this.User_ID = User_ID;
    this.IsKing = IsKing;
    this.Background_ResIndex = Background_ResIndex;
    this.Skin_ResIndex = Skin_ResIndex;
    this.Cloth_ResIndex = Cloth_ResIndex;
    this.Eyes_ResIndex = Eyes_ResIndex;
    this.Mouth_ResIndex = Mouth_ResIndex;
    this.Crown_ResIndex = Crown_ResIndex;
}
}

```

```
package com.farazannajmi.majesticlife.DataStructures;
```

```
import android.arch.persistence.room.ColumnInfo;
import android.arch.persistence.room.Entity;
import android.arch.persistence.room.PrimaryKey;

import javax.annotation.NonNull;
```

```

@Entity(tableName = "AvatarItem_table")
public class AvatarItem
{
    @NonNull
    @PrimaryKey
    @ColumnInfo(name = "ResourceIndex")
    private int ResourceIndex;

    @ColumnInfo(name = "IsBought")
    private boolean IsBought;

    @ColumnInfo(name = "ItemType")
    private int ItemType;
    /*
    0 => background
    1 => skin
    2 => cloth
    3 => eyes
    4 => mouth
    5 => crown
    */

    @ColumnInfo(name = "Price")
    private int Price;

    @NonNull
    public int getResourceIndex() {return ResourceIndex;}
    public boolean getIsBought() {return IsBought;}
    public int getItemType() {return ItemType;}
    public int getPrice() {return Price;}

    public void setResourceIndex(@NonNull int ResourceIndex)
    {this.ResourceIndex = ResourceIndex;}
    public void setIsBought(boolean IsBought) {this.IsBought = IsBought;}
    public void setItemType(int ItemType) {this.ItemType = ItemType;}
    public void setPrice(int Price) {this.Price = Price;}

    public AvatarItem(@NonNull int ResourceIndex, boolean IsBought, int
    ItemType, int Price)
    {
        this.ResourceIndex = ResourceIndex;
        this.IsBought = IsBought;
        this.ItemType = ItemType;
        this.Price = Price;
    }
}

```

```

package com.farazannajmi.majesticlife.DataStructures;

import android.arch.persistence.room.ColumnInfo;
import android.arch.persistence.room.Entity;
import android.arch.persistence.room.ForeignKey;

import com.farazannajmi.majesticlife.DataStructures.Skill;

import static android.arch.persistence.room.ForeignKey.CASCADE;

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

@Entity(tableName = "FaaliatSkill_table",
        primaryKeys = { "Faaliat_ID", "Skill_ID" },

```

```

        foreignKeys = {
            @ForeignKey(entity = Faaliat.class,
                parentColumns = "Faaliat_ID",
                childColumns = "Faaliat_ID",
                onDelete = CASCADE),
            @ForeignKey(entity = Skill.class,
                parentColumns = "Skill_ID",
                childColumns = "Skill_ID",
                onDelete = CASCADE)
        })
    public class FaaliatSkill
    {
        private int Faaliat_ID;
        private int Skill_ID;

        @ColumnInfo(name = "RepetitionCount")
        private int RepetitionCount;

        public int getFaaliat_ID() {return Faaliat_ID;}
        public int getSkill_ID() {return Skill_ID;}
        public int getRepetitionCount() {return RepetitionCount;}

        public void setFaaliat_ID(int faaliat_ID) {Faaliat_ID = faaliat_ID;}
        public void setSkill_ID(int skill_ID) {Skill_ID = skill_ID;}
        public void setRepetitionCount(int repetitionCount) {RepetitionCount =
repetitionCount;}

        public FaaliatSkill(int Faaliat_ID, int Skill_ID, int RepetitionCount/*,
int ListIndex*/)
        {
            this.Faaliat_ID = Faaliat_ID;
            this.Skill_ID = Skill_ID;
            this.RepetitionCount = RepetitionCount;
        }
    }
}

```

```

package com.farazannajmi.majesticlife.DataStructures;

import android.arch.persistence.room.ColumnInfo;
import android.arch.persistence.room.Entity;
import android.arch.persistence.room.ForeignKey;

import com.farazannajmi.majesticlife.DataStructures.Faaliat;

import static android.arch.persistence.room.ForeignKey.CASCADE;

/**
 * join entity table of Quest and Skill
 */
@Entity(tableName = "QuestSkill_table",
    primaryKeys = { "Quest_ID", "Skill_ID" },
    foreignKeys = {
        @ForeignKey(entity = Quest.class,
            parentColumns = "Quest_ID",
            childColumns = "Quest_ID",
            onDelete = CASCADE),
        @ForeignKey(entity = Skill.class,
            parentColumns = "Skill_ID",
            childColumns = "Skill_ID",
            onDelete = CASCADE)
    })
    public class QuestSkill

```

```

{
    private int Quest_ID;
    private int Skill_ID;

    @ColumnInfo(name = "TargetLevel")
    private int TargetLevel;

    public int getQuest_ID() {return Quest_ID;}
    public int getSkill_ID() {return Skill_ID;}
    public int getTargetLevel() {return TargetLevel;}

    public void setQuest_ID(int quest_ID) {Quest_ID = quest_ID;}
    public void setSkill_ID(int skill_ID) {Skill_ID = skill_ID;}
    public void setTargetLevel(int targetLevel) {TargetLevel = targetLevel;}

    public QuestSkill(int Quest_ID, int Skill_ID, int TargetLevel)
    {
        this.Quest_ID = Quest_ID;
        this.Skill_ID = Skill_ID;
        this.TargetLevel = TargetLevel;
    }
}

```

```

package com.farazannajmi.majesticlife.DataStructures;

import android.arch.persistence.room.ColumnInfo;
import android.arch.persistence.room.Embedded;
import android.arch.persistence.room.Entity;
import android.arch.persistence.room.ForeignKey;
import android.arch.persistence.room.PrimaryKey;
import android.support.annotation.NonNull;

import javax.annotation.Nonnull;

import static android.arch.persistence.room.ForeignKey.CASCADE;

@Entity(tableName = "FaaliatRepetitions_table",
        primaryKeys = {"Faaliat_ID", "FR_Date"},
        foreignKeys = @ForeignKey(entity = Faaliat.class,
                parentColumns = "Faaliat_ID",
                childColumns = "Faaliat_ID",
                onDelete = CASCADE))
public class FaaliatRepetitions
{
    @NonNull
    @ColumnInfo(name = "Faaliat_ID")
    private int Faaliat_ID;

    @NonNull
    @ColumnInfo(name = "FR_Date")
    private String FR_Date;

    @ColumnInfo(name = "DayOfWeek")
    private int DayOfWeek;

    @ColumnInfo(name = "RepetitionCount")
    private int RepetitionCount;

    @NonNull
    public int getFaaliat_ID() {return Faaliat_ID;}

    @NonNull

```

```

    public String getFR_Date() {return FR_Date;}
    public int getDayOfWeek() {return DayOfWeek;}
    public int getRepetitionCount() {return RepetitionCount;}

    public void setFaaliat_ID(@NonNull int Faaliat_ID) {this.Faaliat_ID =
Faaliat_ID;}
    public void setFR_Date(@NonNull String FR_Date) {this.FR_Date =
FR_Date;}
    public void setDayOfWeek(int DayOfWeek) {this.DayOfWeek = DayOfWeek;}
    public void setRepetitionCount(int RepetitionCount)
{this.RepetitionCount = RepetitionCount;}

    public FaaliatRepetitions(@NonNull int Faaliat_ID, int DayOfWeek,
@NonNull String FR_Date, int RepetitionCount)
    {
        this.Faaliat_ID = Faaliat_ID;
        this.DayOfWeek = DayOfWeek;
        this.FR_Date = FR_Date;
        this.RepetitionCount = RepetitionCount ;
    }
}

```

DAOی برخی از موجودیت‌ها:

```

package com.farazannajmi.majesticlife.DataStructures;

import android.arch.lifecycle.LiveData;
import android.arch.persistence.room.Dao;
import android.arch.persistence.room.Delete;
import android.arch.persistence.room.Insert;
import android.arch.persistence.room.Query;
import android.arch.persistence.room.Update;

import java.util.List;

@Dao
public interface FaaliatDao
{
    @Insert
    void insert(Faaliat faaliat);

    @Update
    void update(Faaliat faaliat);

    @Delete
    void delete(Faaliat faaliat);

    @Query("DELETE FROM Faaliat_table")
    void deleteAll();

    @Query("SELECT * FROM Faaliat_table ORDER BY Faaliat_ID")
    LiveData<List<Faaliat>> getAllFaaliats();
}

```

```

package com.farazannajmi.majesticlife.DataStructures;

import android.arch.lifecycle.LiveData;
import android.arch.persistence.room.Dao;
import android.arch.persistence.room.Delete;
import android.arch.persistence.room.Insert;
import android.arch.persistence.room.Query;
import android.arch.persistence.room.Update;

```



```
import java.util.List;

@Dao
public interface FaaliatRepetitionsDao
{
    @Insert
    void insert(FaaliatRepetitions faaliatRepetitions);

    @Update
    void update(FaaliatRepetitions faaliatRepetitions);

    @Delete
    void delete(FaaliatRepetitions faaliatRepetitions);

    @Query("DELETE FROM FaaliatRepetitions_table")
    void deleteAll();

    @Query("SELECT * FROM FaaliatRepetitions_table")
    LiveData<List<FaaliatRepetitions>> getAllFaaliatRepetitions();

    @Query("SELECT * FROM FaaliatRepetitions_table " +
            "WHERE FaaliatRepetitions_table.Faaliat_ID=:faaliatID " +
            "ORDER BY DayOfWeek")
    LiveData<List<FaaliatRepetitions>> getAllFaaliatRepsForFaaliat(final int
faaliatID);

    @Query("SELECT * FROM FaaliatRepetitions_table " +
            "WHERE Faaliat_ID=:faaliatID AND FR_Date=:date " +
            "ORDER BY DayOfWeek")
    LiveData<List<FaaliatRepetitions>> getFaaliatRepsForFaaliatDate(final
int faaliatID, final String date);
}
```

### Repository می‌موجودیت فعالیت:

```
package com.farazannajmi.majesticlife.DataStructures;

import android.app.Application;
import android.arch.lifecycle.LiveData;
import android.os.AsyncTask;

import java.util.List;

public class FaaliatRepository
{
    private FaaliatDao mFaaliatDao;
    private LiveData<List<Faaliat>> mAllFaaliats;

    public FaaliatRepository(Application application)
    {
        MajesticLifeRoomDatabase db =
MajesticLifeRoomDatabase.getDatabase(application);
        mFaaliatDao = db.faaliatDao();
        mAllFaaliats = mFaaliatDao.getAllFaaliats();
    }

    //Room executes all queries on a separate thread.
    //Observed LiveData will notify the observer when the data has changed.
    LiveData<List<Faaliat>> getAllFaaliats()
    {
        return mAllFaaliats;
    }

    /*a wrapper for the insert() method. You must call this on a non-UI
thread or your app will crash.
    Room ensures that you don't do any long-running operations on the main
```

```

thread, blocking the UI.*/
    public void insert (Faaliat faaliat)
    {
        new FaaliatRepository.insertAsyncTask(mFaaliatDao).execute(faaliat);
    }

    public void update (Faaliat faaliat)
    {
        new FaaliatRepository.updateAsyncTask(mFaaliatDao).execute(faaliat);
    }

    public void delete (Faaliat faaliat)
    {
        new FaaliatRepository.deleteAsyncTask(mFaaliatDao).execute(faaliat);
    }

    private static class insertAsyncTask extends AsyncTask<Faaliat, Void,
Void>
    {
        private FaaliatDao mAsyncTaskDao;

        insertAsyncTask(FaaliatDao dao)
        {
            mAsyncTaskDao = dao;
        }

        @Override
        protected Void doInBackground(final Faaliat... params)
        {
            mAsyncTaskDao.insert(params[0]);
            return null;
        }
    }

    private static class updateAsyncTask extends AsyncTask<Faaliat, Void,
Void>
    {
        private FaaliatDao mAsyncTaskDao;

        updateAsyncTask(FaaliatDao dao)
        {
            mAsyncTaskDao = dao;
        }

        @Override
        protected Void doInBackground(final Faaliat... params)
        {
            mAsyncTaskDao.update(params[0]);
            return null;
        }
    }

    private static class deleteAsyncTask extends AsyncTask<Faaliat, Void,
Void>
    {
        private FaaliatDao mAsyncTaskDao;

        deleteAsyncTask(FaaliatDao dao)
        {
            mAsyncTaskDao = dao;
        }

        @Override
        protected Void doInBackground(final Faaliat... params)
        {

```

```

        mAsyncTaskDao.delete(params[0]);
        return null;
    }
}

```

### ViewModel ی موجودیت فعالیت:

```

package com.farazannajmi.majesticlife.DataStructures;

import android.app.Application;
import android.arch.lifecycle.AndroidViewModel;
import android.arch.lifecycle.LiveData;

import java.util.List;

public class FaaliatViewModel extends AndroidViewModel
{
    private FaaliatRepository mRepository;
    private LiveData<List<Faaliat>> mAllFaaliats;

    public LiveData<List<Faaliat>> getAllFaaliats() {return mAllFaaliats;}

    public FaaliatViewModel (Application application)
    {
        super(application);
        mRepository = new FaaliatRepository(application);
        mAllFaaliats = mRepository.getAllFaaliats();
    }

    public void insert(Faaliat faaliat)
    {
        mRepository.insert(faaliat);
    }

    public void update(Faaliat faaliat)
    {
        mRepository.update(faaliat);
    }

    public void delete(Faaliat faaliat)
    {
        mRepository.delete(faaliat);
    }
}

```

### کلاس پایگاه داده room:

```

package com.farazannajmi.majesticlife.DataStructures;

import android.arch.persistence.db.SupportSQLiteDatabase;
import android.arch.persistence.room.Database;
import android.arch.persistence.room.Room;
import android.arch.persistence.room.RoomDatabase;
import android.content.Context;
import android.os.AsyncTask;
import android.support.annotation.NonNull;
import android.util.Log;

import com.farazannajmi.majesticlife.DataHolder;
import com.farazannajmi.majesticlife.LoadingActivity;
import com.farazannajmi.majesticlife.R;

```

```

@Database(entities = {User.class, Faaliat.class, Skill.class, Quest.class,
PlanCell.class,
    FaaliatSkill.class, QuestSkill.class, FaaliatRepetitions.class,
    Avatar.class, AvatarItem.class},
    version = 6)
public abstract class MajesticLifeRoomDatabase extends RoomDatabase
{
    private static final String DB_NAME = "MajesticLifeDatabase.db";
    private static volatile MajesticLifeRoomDatabase INSTANCE;

    public abstract UserDao userDao();
    public abstract FaaliatDao faaliatDao();
    public abstract SkillDao skillDao();
    public abstract QuestDao questDao();
    public abstract PlanCellDao planCellDao();
    public abstract FaaliatSkillDao faaliatSkillDao();
    public abstract QuestSkillDao questSkillDao();
    public abstract FaaliatRepetitionsDao faaliatRepetitionsDao();
    public abstract AvatarDao avatarDao();
    public abstract AvatarItemDao avatarItemDao();

    public static MajesticLifeRoomDatabase getDatabase(final Context
context)
    {
        if (INSTANCE == null)
        {
            synchronized (MajesticLifeRoomDatabase.class)
            {
                if (INSTANCE == null)
                {
                    // Create database here
                    INSTANCE =
Room.databaseBuilder(context.getApplicationContext(),
                        MajesticLifeRoomDatabase.class, DB_NAME)
                        .addCallback(sRoomDatabaseCallback) //for
initial data to database
                        .fallbackToDestructiveMigration() //drop and
recreate the whole database if db version goes up
                        .build();
                }
                else
                    LoadingActivity.isDataLoaded = true;
            }
        }
        else
            LoadingActivity.isDataLoaded = true;
        return INSTANCE;
    }

    private static RoomDatabase.Callback sRoomDatabaseCallback = new
RoomDatabase.Callback()
    {
        @Override
        public void onOpen (@NonNull SupportSQLiteDatabase db)
        {
            super.onOpen(db);
            if(!LoadingActivity.isFirstTime)
            {
                Log.d("Data", "Database has been initialed!");
            }
            LoadingActivity.isDataLoaded = true;

            Log.d("Data", "Database opened!");
        }
    }
}

```

```

@Override
public void onCreate(@NonNull SupportSQLiteDatabase db)
{
    super.onCreate(db);
    new PopulateDbAsync(INSTANCE).execute();
    Log.d("Data", "Database created!");
}

};

private static class PopulateDbAsync extends AsyncTask<Void, Void, Void>
{
    private final UserDao userDao;
    private final FaaliatDao faaliatDao;
    private final SkillDao skillDao;
    private final QuestDao questDao;
    private final PlanCellDao planCellDao;
    private final FaaliatSkillDao faaliatSkillDao;
    private final QuestSkillDao questSkillDao;
    private final FaaliatRepetitionsDao faaliatRepetitionsDao;
    private final AvatarDao avatarDao;
    private final AvatarItemDao avatarItemDao;

    PopulateDbAsync(MajesticLifeRoomDatabase db)
    {
        userDao = db.userDao();
        faaliatDao = db.faaliatDao();
        skillDao = db.skillDao();
        questDao = db.questDao();
        planCellDao = db.planCellDao();
        faaliatSkillDao = db.faaliatSkillDao();
        questSkillDao = db.questSkillDao();
        faaliatRepetitionsDao = db.faaliatRepetitionsDao();
        avatarDao = db.avatarDao();
        avatarItemDao = db.avatarItemDao();
    }

    @Override
    protected Void doInBackground(final Void... params)
    {
        if>LoadingActivity.isFirstTime && !DataHolder.IsDatabaseCreated)
        {
            //orders are important!!

            //User:
            userDao.deleteAll();
            User user = new User(0, "NewKing", "king@mail.com",
                                0, 1, 0, 1, 0, 1, 100);
            userDao.insert(user);

            //Faaliats:
            faaliatDao.deleteAll();
            Faaliat faaliat = new Faaliat(0, "Reading book",
                                           R.drawable.ic_majestic_activities,
                                           R.color.faaliatsColor1, 10, 0, 1, 0);
            faaliatDao.insert(faaliat);

            //Skills:
            skillDao.deleteAll();
            Skill skill = new Skill(0, "Knowledge",
                                    R.drawable.ic_skills, 1, 0, 0);
            skillDao.insert(skill);

            //FaaliatSkills:
            faaliatSkillDao.deleteAll();
            FaaliatSkill faaliatSkill = new FaaliatSkill(0, 0, 5);

```

```

faaliatSkillDao.insert(faaliatSkill);

//PlanCells:
planCellDao.deleteAll();
PlanCell planCell = new PlanCell(0, 0, 0, 5);
planCellDao.insert(planCell);

//Quests:
questDao.deleteAll();
Quest quest = new Quest(0, "Pass exam", "2019/1/1");
questDao.insert(quest);

//QuestSkills:
questSkillDao.deleteAll();
QuestSkill questSkill = new QuestSkill(0, 0, 2);
questSkillDao.insert(questSkill);

//region ----- avatarItems -----
//backgrounds:
avatarItemDao.deleteAll();
AvatarItem ava_back1 = new AvatarItem(R.drawable.ava_back1,
true, 0, 0);
avatarItemDao.insert(ava_back1);
AvatarItem ava_back2 = new AvatarItem(R.drawable.ava_back2,
false, 0, 50);
avatarItemDao.insert(ava_back2);
AvatarItem ava_back3 = new AvatarItem(R.drawable.ava_back3,
false, 0, 60);
avatarItemDao.insert(ava_back3);
AvatarItem ava_back4 = new AvatarItem(R.drawable.ava_back4,
false, 0, 60);
avatarItemDao.insert(ava_back4);
AvatarItem ava_back5 = new AvatarItem(R.drawable.ava_back5,
false, 0, 70);
avatarItemDao.insert(ava_back5);
AvatarItem ava_back6 = new AvatarItem(R.drawable.ava_back6,
false, 0, 70);
avatarItemDao.insert(ava_back6);
AvatarItem ava_back7 = new AvatarItem(R.drawable.ava_back7,
false, 0, 70);
avatarItemDao.insert(ava_back7);

//Skins
AvatarItem ava_skin1 = new
AvatarItem(R.drawable.ava_skin_white, true, 0, 0);
avatarItemDao.insert(ava_skin1);

//Clothes
AvatarItem ava_cloth1 = new
AvatarItem(R.drawable.ava_cloth1, true, 0, 0);
avatarItemDao.insert(ava_cloth1);

//Eyes
AvatarItem ava_eyes1 = new AvatarItem(R.drawable.ava_eyes1,
true, 0, 0);
avatarItemDao.insert(ava_eyes1);

//Mouths
AvatarItem ava_mouth1 = new
AvatarItem(R.drawable.ava_mouth1, true, 0, 0);
avatarItemDao.insert(ava_mouth1);

//Crowns
AvatarItem ava_crown1 = new
AvatarItem(R.drawable.ava_crown1, true, 0, 0);

```

```

        avatarItemDao.insert(ava_crown1);
        //todo: insert other crowns
        //endregion -----

        //FaaliatRepetitions
        //test:
        faaliatRepetitionsDao.deleteAll();
        FaaliatRepetitions fr1 = new FaaliatRepetitions(0, 1,
"2018/6/28", 1);
        FaaliatRepetitions fr2 = new FaaliatRepetitions(0, 2,
"2018/6/29", 3);
        FaaliatRepetitions fr3 = new FaaliatRepetitions(0, 3,
"2018/6/30", 2);
        FaaliatRepetitions fr4 = new FaaliatRepetitions(0, 4,
"2018/6/31", 5);
        FaaliatRepetitions fr5 = new FaaliatRepetitions(0, 6,
"2018/7/2", 3);
        FaaliatRepetitions fr6 = new FaaliatRepetitions(0, 1,
"2018/7/4", 1);
        FaaliatRepetitions fr7 = new FaaliatRepetitions(0, 3,
"2018/7/6", 1);
        FaaliatRepetitions fr8 = new FaaliatRepetitions(0, 4,
"2018/7/7", 2);
        faaliatRepetitionsDao.insert(fr1);
        faaliatRepetitionsDao.insert(fr2);
        faaliatRepetitionsDao.insert(fr3);
        faaliatRepetitionsDao.insert(fr4);
        faaliatRepetitionsDao.insert(fr5);
        faaliatRepetitionsDao.insert(fr6);
        faaliatRepetitionsDao.insert(fr7);
        faaliatRepetitionsDao.insert(fr8);

        //Avatar:
        avatarDao.deleteAll();
        Avatar avatar = new Avatar(0,0, true, R.drawable.ava_back1,
            R.drawable.ava_skin_white, R.drawable.ava_cloth1,
            R.drawable.ava_eyes1, R.drawable.ava_mouth1,
R.drawable.ava_crown1);
        avatarDao.insert(avatar);

        DataHolder.IsDatabaseCreated = true;
        Log.d("Data", "Database initialed!");
    }
    return null;
}
}
}

```

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

```

package com.farazannajmi.majesticlife;

import android.arch.lifecycle.LifecycleOwner;
import android.arch.lifecycle.Observer;
import android.arch.lifecycle.ViewModelProviders;
import android.support.annotation.Nullable;
import android.support.v4.app.FragmentActivity;
import android.util.Log;

import com.backtory.java.internal.BacktoryUser;
import com.farazannajmi.majesticlife.DataStructures.Avatar;
import com.farazannajmi.majesticlife.DataStructures.AvatarItem;
import com.farazannajmi.majesticlife.DataStructures.AvatarItemViewModel;

```

```

import com.farazannajmi.majesticlife.DataStructures.AvatarViewModel;
import com.farazannajmi.majesticlife.DataStructures.Faaliat;
import com.farazannajmi.majesticlife.DataStructures.FaaliatRepetitions;
import com.farazannajmi.majesticlife.DataStructures.FaaliatRepetitionsViewModel;
import com.farazannajmi.majesticlife.DataStructures.FaaliatSkill;
import com.farazannajmi.majesticlife.DataStructures.FaaliatSkillViewModel;
import com.farazannajmi.majesticlife.DataStructures.FaaliatViewModel;
import com.farazannajmi.majesticlife.DataStructures.PlanCell;
import com.farazannajmi.majesticlife.DataStructures.PlanCellViewModel;
import com.farazannajmi.majesticlife.DataStructures.Quest;
import com.farazannajmi.majesticlife.DataStructures.QuestViewModel;
import com.farazannajmi.majesticlife.DataStructures.Skill;
import com.farazannajmi.majesticlife.DataStructures.SkillViewModel;
import com.farazannajmi.majesticlife.DataStructures.User;
import com.farazannajmi.majesticlife.DataStructures.UserViewModel;

import java.util.ArrayList;
import java.util.List;

/**
 * <div>Icons made by <a href="https://www.flaticon.com/authors/freepik"
title="leaf">leaf</a> from <a href="https://www.flaticon.com/"
title="Flaticon">www.flaticon.com</a> is licensed by <a
href="http://creativecommons.org/licenses/by/3.0/" title="Creative
Commons BY 3.0" target=" blank">CC 3.0 BY</a></div>
 * https://www.flaticon.com/packs/user-interface-30/2
 */

public class DataHolder
{
    public static int LevelUpReward = 50;

    public static User ThisUser;
    public static BacktoryUser CurrentBacktoryUser;
    public static ArrayList<Faaliat> Faaliats;
    public static ArrayList<Skill> Skills;
    public static ArrayList<PlanCell> PlanCells;
    public static ArrayList<Quest> Quests;
    public static int FaaliatRepetitionsIdCounter;
    public static Avatar UserAvatar;
    public static ArrayList<AvatarItem> AvatarItems;

    public static ArrayList<Integer> FaaliatAvatars;
    public static ArrayList<Integer> SkillAvatars;

    public static boolean IsDatabaseCreated = false;

    public static void InitialDataStructures()
    {
        IsDatabaseCreated = false;

        FaaliatAvatars = new ArrayList<Integer>();
        FaaliatAvatars.add(R.drawable.ic_majestic_activities);
        FaaliatAvatars.add(R.drawable.ic_bag);
        FaaliatAvatars.add(R.drawable.ic_book);
        FaaliatAvatars.add(R.drawable.ic_camera);
        FaaliatAvatars.add(R.drawable.ic_computer);
        FaaliatAvatars.add(R.drawable.ic_conversation);
        FaaliatAvatars.add(R.drawable.ic_diamond);
        FaaliatAvatars.add(R.drawable.ic_edit);
        FaaliatAvatars.add(R.drawable.ic_eye);
        FaaliatAvatars.add(R.drawable.ic_flask);
        FaaliatAvatars.add(R.drawable.ic_gamepad);
        FaaliatAvatars.add(R.drawable.ic_idea);
    }
}

```



```

    FaaliatAvatars.add(R.drawable.ic_leaf);
    FaaliatAvatars.add(R.drawable.ic_list);
    FaaliatAvatars.add(R.drawable.ic_music);
    FaaliatAvatars.add(R.drawable.ic_news);
    FaaliatAvatars.add(R.drawable.ic_paint);
    FaaliatAvatars.add(R.drawable.ic_real_state);
    FaaliatAvatars.add(R.drawable.ic_rocket);
    FaaliatAvatars.add(R.drawable.ic_search);
    FaaliatAvatars.add(R.drawable.ic_speed);
    FaaliatAvatars.add(R.drawable.ic_target);

    SkillAvatars = new ArrayList<Integer>();
    SkillAvatars.add(R.drawable.ic_skills);
    SkillAvatars.add(R.drawable.ic_bag);
    SkillAvatars.add(R.drawable.ic_book);
    SkillAvatars.add(R.drawable.ic_camera);
    SkillAvatars.add(R.drawable.ic_computer);
    SkillAvatars.add(R.drawable.ic_conversation);
    SkillAvatars.add(R.drawable.ic_diamond);
    SkillAvatars.add(R.drawable.ic_edit);
    SkillAvatars.add(R.drawable.ic_eye);
    SkillAvatars.add(R.drawable.ic_flask);
    SkillAvatars.add(R.drawable.ic_gamepad);
    SkillAvatars.add(R.drawable.ic_idea);
    SkillAvatars.add(R.drawable.ic_leaf);
    SkillAvatars.add(R.drawable.ic_list);
    SkillAvatars.add(R.drawable.ic_music);
    SkillAvatars.add(R.drawable.ic_news);
    SkillAvatars.add(R.drawable.ic_paint);
    SkillAvatars.add(R.drawable.ic_real_state);
    SkillAvatars.add(R.drawable.ic_rocket);
    SkillAvatars.add(R.drawable.ic_search);
    SkillAvatars.add(R.drawable.ic_speed);
    SkillAvatars.add(R.drawable.ic_target);
}

public static void LoadUser(FragmentActivity activity, LifecycleOwner
owner)
{
    Log.d("Data", "Loading User from Database");

    UserViewModel userViewModel =
    ViewModelProviders.of(activity).get(UserViewModel.class);
    userViewModel.getUser().observe(owner, new Observer<User>() {
        @Override
        public void onChanged(@Nullable User user) {
            DataHolder.ThisUser = user;
        }
    });
}

public static void LoadData(FragmentActivity activity, LifecycleOwner
owner)
{
    Log.d("Data", "Loading data from Database");

    //getting user:
    UserViewModel userViewModel =
    ViewModelProviders.of(activity).get(UserViewModel.class);
    userViewModel.getUser().observe(owner, new Observer<User>() {
        @Override
        public void onChanged(@Nullable User user) {
            DataHolder.ThisUser = user;
        }
    });
}

```

```

//getting Faaliats:
FaaliatViewModel faaliatViewModel =
ViewModelProviders.of(activity).get(FaaliatViewModel.class);
faaliatViewModel.getAllFaaliats().observe(owner, new
Observer<List<Faaliat>>() {
    @Override
    public void onChanged(@Nullable List<Faaliat> faaliats) {
        DataHolder.Faaliats = (ArrayList) faaliats;
    }
});

//getting Skills:
SkillViewModel skillViewModel =
ViewModelProviders.of(activity).get(SkillViewModel.class);
skillViewModel.getAllSkills().observe(owner, new
Observer<List<Skill>>() {
    @Override
    public void onChanged(@Nullable List<Skill> skills) {
        DataHolder.Skills = (ArrayList) skills;
    }
});

//getting PlanCells:
PlanCellViewModel planCellViewModel =
ViewModelProviders.of(activity).get(PlanCellViewModel.class);
planCellViewModel.getAllPlanCells().observe(owner, new
Observer<List<PlanCell>>() {
    @Override
    public void onChanged(@Nullable List<PlanCell> planCells) {
        DataHolder.PlanCells = (ArrayList) planCells;
    }
});

//getting Quests:
QuestViewModel questViewModel =
ViewModelProviders.of(activity).get(QuestViewModel.class);
questViewModel.getAllQuests().observe(owner, new
Observer<List<Quest>>() {
    @Override
    public void onChanged(@Nullable List<Quest> quests) {
        DataHolder.Quests = (ArrayList) quests;
    }
});

//getting FaaliatRepetitions
FaaliatRepetitionsViewModel faaliatRepetitionsViewModel =
ViewModelProviders.of(activity).get(FaaliatRepetitionsViewModel.class);

faaliatRepetitionsViewModel.getAllFaaliatRepetitions().observe(owner, new
Observer<List<FaaliatRepetitions>>() {
    @Override
    public void onChanged(@Nullable List<FaaliatRepetitions>
faaliatRepetitions) {
        FaaliatRepetitionsIdCounter = faaliatRepetitions.size();
    }
});

//getting AvatarItems
AvatarItemViewModel avatarItemViewModel =
ViewModelProviders.of(activity).get(AvatarItemViewModel.class);
avatarItemViewModel.getAvatarItems().observe(owner, new
Observer<List<AvatarItem>>() {
    @Override
    public void onChanged(@Nullable List<AvatarItem> avatarItems) {

```

```

        AvatarItems = (ArrayList) avatarItems;
    }
});

//getting UserAvatar
AvatarViewModel avatarViewModel =
ViewModelProviders.of(activity).get(AvatarViewModel.class);
avatarViewModel.getAvatar().observe(owner, new Observer<Avatar>() {
    @Override
    public void onChanged(@Nullable Avatar avatar) {
        UserAvatar = avatar;
    }
});
}

public static void LoadFaaliatSkills(FragmentActivity activity,
LifecycleOwner owner)
{
    FaaliatSkillViewModel faaliatSkillViewModel =
ViewModelProviders.of(activity).get(FaaliatSkillViewModel.class);

    for(int faaliatCounter = 0; faaliatCounter < Faaliats.size();
faaliatCounter++)
    {

faaliatSkillViewModel.getSkillsForFaaliat(Faaliats.get(faaliatCounter)).obse
rve(owner, new Observer<List<FaaliatSkill>>() {
    @Override
    public void onChanged(@Nullable List<FaaliatSkill>
faaliatSkills)
    {
        if(faaliatSkills.size() != 0)
        {
            for (int i = 0; i < Faaliats.size(); i++)
            {
                if (Faaliats.get(i).getFaaliat_ID() ==
faaliatSkills.get(0).getFaaliat_ID())
                    Faaliats.get(i).FaaliatSkills = (ArrayList)
faaliatSkills;
            }
        }
    }
});
    }
}
}
}

```

کلاس مربوط به صفحه اصلی برنامه:

```

package com.farazannajmi.majesticlife;

import android.arch.lifecycle.ViewModelProviders;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.ImageView;

```

```

import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import com.backtory.java.internal.BacktoryCallBack;
import com.backtory.java.internal.BacktoryResponse;
import com.backtory.java.internal.BacktoryUser;
import
com.farazannajmi.majesticlife.AccountPackage.AccountManagementActivity;
import com.farazannajmi.majesticlife.DataStructures.UserViewModel;
import com.farazannajmi.majesticlife.FaaliatPackage.FaaliatsActivity;
import com.farazannajmi.majesticlife.PlanPackage.PlanActivity;
import com.farazannajmi.majesticlife.QuestPackage.QuestsActivity;
import com.farazannajmi.majesticlife.SkillPackage.SkillsActivity;

public class MainMenuActivity extends AppCompatActivity
{
    public static AppManager TheAppManager;

    public TextView Username_txt;
    public TextView Coins_txt;

    public TextView XpLevel_txt;
    public TextView HpLevel_txt;
    public TextView SpLevel_txt;
    public ProgressBar Xp_progBar;
    public ProgressBar Hp_progBar;
    public ProgressBar Sp_progBar;

    public ImageView Avatar_back_img;
    public ImageView Avatar_skin_img;
    public ImageView Avatar_cloth_img;
    public ImageView Avatar_eyes_img;
    public ImageView Avatar_mouth_img;
    public ImageView Avatar_crown_img;
    public ImageView Avatar_hair_img;

    public static boolean FirstTime;
    private boolean _isConnected;
    public static boolean GuestNotLoggedIn;

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

        final AppCompatActivity activity = this;

        TheAppManager = (AppManager) getApplicationContext();

        //region ----- getting UI elements -----
        Username_txt = (TextView) findViewById(R.id.Main_UserName_txt);
        Coins_txt = (TextView) findViewById(R.id.Main_Coins_txt);
        XpLevel_txt = findViewById(R.id.Main_XpLevel_txt);
        HpLevel_txt = findViewById(R.id.Main_HpLevel_txt);
        SpLevel_txt = findViewById(R.id.Main_SpLevel_txt);
        Xp_progBar = findViewById(R.id.Main_Xp_progBar);
        Hp_progBar = findViewById(R.id.Main_HP_progBar);
        Sp_progBar = findViewById(R.id.Main_SP_progBar);

        Avatar_back_img = findViewById(R.id.MainM_Back_img);
        Avatar_skin_img = findViewById(R.id.MainM_skin_img);
        Avatar_cloth_img = findViewById(R.id.MainM_cloth_img);
        Avatar_eyes_img = findViewById(R.id.MainM_eyes_img);

```

```

Avatar_mouth_img = findViewById(R.id.MainM_mouth_img);
Avatar_crown_img = findViewById(R.id.MainM_crown_img);
Avatar_hair_img = findViewById(R.id.MainM_hair_img);
//endregion -----

//for checking internet connection:
ConnectivityManager cm =
(ConnectivityManager)getApplicationContext().getSystemService(Context.CONNEC
TIVITY_SERVICE);
NetworkInfo activeNetwork = cm.getActiveNetworkInfo();
_isConnected = activeNetwork != null &&
activeNetwork.isConnectedOrConnecting();

//region ----- load or initial data -----

DataHolder.LoadData(this, this);
DataHolder.LoadFaaliatSkills(this, this);

//if it is the first time app runs, the FirstTime value becomes
true:
SharedPreferences sharedPref = getPreferences(Context.MODE_PRIVATE);
FirstTime = sharedPref.getBoolean("FirstTime", true);
GuestNotLoggedIn = sharedPref.getBoolean("GuestNotLoggedIn", true);

if(FirstTime)
{
    //region first time
    Log.d("WorkFlow", "First Time running app.");

    if(_isConnected)
    {
        Toast.makeText(getApplicationContext(), "Connected to the
Internet!", Toast.LENGTH_SHORT).show();

        // Request a guest user from backtory:
        BacktoryUser.loginAsGuestInBackground(new
BacktoryCallBack<Void>() {
            @Override
            public void onResponse(BacktoryResponse<Void> response)
            {
                if (response.isSuccessful()) {
                    // Getting new username and password from
CURRENT user
                    String newUsername =
BacktoryUser.getCurrentUser().getUsername();
                    String newPassword =
BacktoryUser.getCurrentUser().getGuestPassword();

                    Log.d("Backtory", "Logged in as guest, username:
" + newUsername +
                    " password: " + newPassword);

                    // saving user info in sharedPreferences:
                    SharedPreferences sharedPref =
getPreferences(Context.MODE_PRIVATE);
                    SharedPreferences.Editor editor =
sharedPref.edit();

                    editor.putBoolean("GuestNotLoggedIn", false);
                    editor.putString("UserName", newUsername);
                    editor.putString("UserPassword", newPassword);
                    editor.commit();

                    //saving username in database:
                    DataHolder.ThisUser.setUsername(newUsername);
                    UserViewModel userViewModel =

```

```

ViewModelProviders.of(activity).get(UserViewModel.class);
userViewModel.update(DataHolder.ThisUser);

DataHolder.CurrentBacktoryUser =
BacktoryUser.getCurrentUser();
} else {
    Log.d("Backtory", "Failed log in as guest: " +
response.code());
}
}
});
}
else //if is not connected to internet
{
    Toast.makeText(getApplicationContext(), "No internet
connection!", Toast.LENGTH_SHORT).show();

    SharedPreferences.Editor editor = sharedPref.edit();
    editor.putBoolean("GuestNotLoggedIn", true);
    editor.putString("UserName", "NewKing");
    editor.putString("UserPassword", "1111");
    editor.commit();
}

SharedPreferences.Editor editor = sharedPref.edit();
editor.putBoolean("FirstTime", false);
editor.commit();
//endregion
}
else //not the first time opening the application
{
    //region not first time
    Log.d("WorkFlow", "Not the first Time running app.");

    if(_isConnected)
    {
        if(GuestNotLoggedIn)
        {
            // Request a guest user from backtory:
            BacktoryUser.loginAsGuestInBackground(new
BacktoryCallBack<Void>() {
                @Override
                public void onResponse(BacktoryResponse<Void>
response) {
                    if (response.isSuccessful()) {
                        // Getting new username and password from
CURRENT user
                        String newUsername =
BacktoryUser.getCurrentUser().getUsername();
                        String newPassword =
BacktoryUser.getCurrentUser().getGuestPassword();

                        Log.d("Backtory", "Logged in as guest,
username: " + newUsername +
                        " password: " + newPassword);

                        // saving user info in sharedPreference:
                        SharedPreferences sharedPref =
getPreferences(Context.MODE_PRIVATE);
                        SharedPreferences.Editor editor =
sharedPref.edit();

                        editor.putBoolean("GuestNotLoggedIn",
false);

                        editor.putString("UserName", newUsername);
                        editor.putString("UserPassword",

```

```

newPassword);

        editor.commit();

        //saving username in database:

DataHolder.ThisUser.setUsername(newUsername);
        UserViewModel userViewModel =
ViewModelProviders.of(activity).get(UserViewModel.class);
        userViewModel.update(DataHolder.ThisUser);

        DataHolder.CurrentBacktoryUser =
BacktoryUser.getCurrentUser();
        } else {
            Log.d("Backtory", "Failed log in as guest: "
+ response.code());
        }
    }
}

else //guest has been logged in
{
    // loading username and password from SharedPreferences
    String username = sharedPreferences.getString("UserName", "");
    String password = sharedPreferences.getString("UserPassword",
"");

    Log.d("WorkFlow", "Username: " + username);
}

else //if not connected to internet
{
    Toast.makeText(getApplicationContext(), "No internet
connection!", Toast.LENGTH_SHORT).show();

    // loading username and password from SharedPreferences
    String username = sharedPreferences.getString("UserName", "");
    String password = sharedPreferences.getString("UserPassword", "");
    Log.d("WorkFlow", "Username: " + username);

    SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.putBoolean("GuestNotLoggedIn", true);
    editor.commit();
}

DataHolder.LoadData(this, this);
//endregion
}
//endregion -----

InitialUIElements();
}

@Override
protected void onResume()
{
    super.onResume();
    InitialUIElements();
}

@Override
public void onBackPressed()
{
    try
    {
        finishAffinity();
    }
}

```

```

    }
    catch (Exception e)
    {
        ActivityCompat.finishAffinity(MainMenuActivity.this);
    }
}

public void UiElementsOnClick(View view)
{
    switch (view.getId())
    {
        case R.id.Main_UserAvatar_img:
        {
            Intent AccountIntent = new Intent(MainMenuActivity.this,
AccountManagementActivity.class);
            startActivity(AccountIntent);
            break;
        }
        case R.id.Main_Skills_btn:
        {
            Intent skillsIntent = new Intent(MainMenuActivity.this,
SkillsActivity.class);
            startActivity(skillsIntent);
            break;
        }
        case R.id.Main_Faaliat_btn:
        {
            Intent activitiesIntent = new Intent(MainMenuActivity.this,
FaaliatsActivity.class);
            startActivity(activitiesIntent);
            break;
        }
        case R.id.Main_Plan_btn:
        {
            Intent planIntent = new Intent(MainMenuActivity.this,
PlanActivity.class);
            startActivity(planIntent);
            break;
        }
        case R.id.Main_Quests_btn:
        {
            Intent questsIntent = new Intent(MainMenuActivity.this,
QuestsActivity.class);
            startActivity(questsIntent);
            break;
        }
        case R.id.Main_Shop_btn:
        {
            Intent shopIntent = new Intent(MainMenuActivity.this,
ShopActivity.class);
            startActivity(shopIntent);

            break;
        }
        default:
            break;
    }
}

public void InitialUIElements()
{
    Username_txt.setText(DataHolder.ThisUser.getUsername());
    Coins_txt.setText(Integer.toString(DataHolder.ThisUser.getCoins()));
}

```



```

XpLevel_txt.setText(Integer.toString(DataHolder.ThisUser.getXpLevel()));

HpLevel_txt.setText(Integer.toString(DataHolder.ThisUser.getHpLevel()));

SpLevel_txt.setText(Integer.toString(DataHolder.ThisUser.getSpLevel()));
Xp_progBar.setProgress(DataHolder.ThisUser.getXP());
Hp_progBar.setProgress(DataHolder.ThisUser.getHP());
Sp_progBar.setProgress(DataHolder.ThisUser.getSP());

//setting user avatar:

Avatar_back_img.setImageResource(DataHolder.UserAvatar.getBackground_ResIndex());

Avatar_skin_img.setImageResource(DataHolder.UserAvatar.getSkin_ResIndex());

Avatar_cloth_img.setImageResource(DataHolder.UserAvatar.getCloth_ResIndex());

Avatar_eyes_img.setImageResource(DataHolder.UserAvatar.getEyes_ResIndex());

Avatar_mouth_img.setImageResource(DataHolder.UserAvatar.getMouth_ResIndex());

Avatar_crown_img.setImageResource(DataHolder.UserAvatar.getCrown_ResIndex());

if(DataHolder.UserAvatar.getIsKing())
{
    Avatar_hair_img.setVisibility(View.INVISIBLE);
}
else
{
    Avatar_hair_img.setVisibility(View.VISIBLE);
}
}

```

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

```

package com.farazannajmi.majesticlife.SkillPackage;

import android.arch.lifecycle.ViewModelProviders;
import android.content.Context;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.GridView;

import com.farazannajmi.majesticlife.DataHolder;
import com.farazannajmi.majesticlife.DataStructures.Skill;
import com.farazannajmi.majesticlife.DataStructures.SkillViewModel;
import com.farazannajmi.majesticlife.R;

public class SkillsActivity extends AppCompatActivity
{
    public static AppCompatActivity appCompatActivity;
    public static Context context;
    public static ArrayAdapter<Skill> skill_gridview_adapter;

```

```

private GridView skills_gridview;

@Override
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_skills);
    context = getApplicationContext();
    appCompatActivity = this;

    //getting UI elements
    skills_gridview = (GridView)
findViewById(R.id.SkillsActivity_gridview);

    //create our new array adapter
    skill_gridview_adapter = new SkillsListItemArrayAdapter(this,
DataHolder.Skills);
    skills_gridview.setAdapter(skill_gridview_adapter);
}

public void UiElementsOnClick(View view)
{
    if (view.getId() == R.id.SkillsActivity_addSkill_btn)
    {
        Skill newSkill = new Skill(DataHolder.Skills.size(), "new
skill", DataHolder.SkillAvatars.get(0), 1, 0,
DataHolder.ThisUser.getUser_ID());
        DataHolder.Skills.add(newSkill);

        //adding skill in database:
        SkillViewModel skillViewModel =
ViewModelProviders.of(this).get(SkillViewModel.class);
        skillViewModel.insert(newSkill);

        //opening new activity for editing this new faaliat:
        Intent intent = new Intent(SkillsActivity.this,
EditSkillActivity.class);
        intent.putExtra("The_Skill", DataHolder.Skills.size() - 1);
        startActivity(intent);
    }
}

public static void DeleteSkill(int skillIndex)
{
    //deleting skill from database:
    SkillViewModel skillViewModel =
ViewModelProviders.of(appCompatActivity).get(SkillViewModel.class);
    skillViewModel.delete(DataHolder.Skills.get(skillIndex));

    DataHolder.Skills.remove(skillIndex);
    skill_gridview_adapter.notifyDataSetChanged();
}
}

```

کلاس SkillsListItemArrayAdapter که برای مقداردهی لیست مهارت‌ها استفاده شده است:

```

package com.farazannajmi.majesticlife.SkillPackage;

import android.animation.ObjectAnimator;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;

```

```

import android.support.annotation.NonNull;
import android.support.v7.app.AlertDialog;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.animation.DecelerateInterpolator;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.TextView;
import android.widget.Toast;

import com.farazannajmi.majesticlife.DataStructures.Skill;
import com.farazannajmi.majesticlife.FaaliatPackage.FaaliatsActivity;
import com.farazannajmi.majesticlife.R;

import java.util.ArrayList;

public class SkillsListItemArrayAdapter extends ArrayAdapter<Skill>
{
    private Context context;
    private ArrayList<Skill> skillsList;

    public SkillsListItemArrayAdapter(@NonNull Context context, @NonNull
ArrayList<Skill> objects)
    {
        super(context, R.layout.listitem_faaliat, objects);

        this.context = context;
        this.skillsList = objects;
    }

    //called when rendering the list
    public View getView(final int position, View convertView, ViewGroup
parent)
    {
        //get the property we are displaying
        Skill skill = getItem(position);

        //get the inflater and inflate the XML layout for each item
        LayoutInflater inflater = LayoutInflater.from(getContext());
        convertView = inflater.inflate(R.layout.listitem_skill, parent,
false);

        TextView name_txt = (TextView)
convertView.findViewById(R.id.listItem_skill_name_txt);
        TextView level_txt = (TextView)
convertView.findViewById(R.id.listItem_skill_level_txt);
        ImageView avatar_img = (ImageView)
convertView.findViewById(R.id.listItem_skill_avatar_img);
        ProgressBar progress_Bar = (ProgressBar)
convertView.findViewById(R.id.listItem_skill_progressBar);
        Button edit_btn = (Button)
convertView.findViewById(R.id.listItem_skill_edit_btn);
        Button delete_btn = (Button)
convertView.findViewById(R.id.listItem_skill_delete_btn);

        //setting info in UI:
        name_txt.setText(skill.getSkill_Name());
        level_txt.setText(Integer.toString(skill.getLevel()));

        progress_Bar.setMax(100);
        progress_Bar.setProgress(skill.getProgress());
        ObjectAnimator animation = ObjectAnimator.ofInt(progress_Bar,

```

```

"progress", 0, skill.getProgress()); // see this max value coming back here,
we animate towards that value
    animation.setDuration(1000); // in milliseconds
    animation.setInterpolator(new DecelerateInterpolator());
    animation.start();

    //get the image associated with this property
    avatar_img.setImageResource(skill.getAvatar_ResIndex());

    edit_btn.setOnClickListener(new View.OnClickListener()
    {
        @Override
        public void onClick(View v)
        {
            //opening new activity for editing this faaliat:
            Intent intent = new Intent(getApplicationContext(),
EditSkillActivity.class);
            intent.putExtra("The_Skill", position);
            context.startActivity(intent);
        }
    });

    delete_btn.setOnClickListener(new View.OnClickListener()
    {
        @Override
        public void onClick(View v)
        {
            //show are you sure dialogue popup:
            AlertDialog.Builder builder = new
AlertDialog.Builder(context);
            // Add action buttons
            builder.setPositiveButton(R.string.yes, new
DialogInterface.OnClickListener()
            {
                @Override
                public void onClick(DialogInterface dialog, int id)
                {
                    SkillsActivity.DeleteSkill(position);
                    Toast.makeText(context, "Skill deleted
successfully.", Toast.LENGTH_SHORT).show();
                    dialog.dismiss();
                }
            });
            builder.setNegativeButton(R.string.no, new
DialogInterface.OnClickListener()
            {
                public void onClick(DialogInterface dialog, int id)
                {
                    dialog.cancel();
                }
            });

            builder.setTitle("Are you sure you want to delete this
skill?");

            builder.create();
            builder.show();
        }
    });

    return convertView;
}
}

```

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

```
package com.farazannajmi.majesticlife.AccountPackage;

import android.app.Activity;
import android.app.Dialog;
import android.app.DialogFragment;
import android.content.DialogInterface;
import android.os.Bundle;
import android.support.v7.app.AlertDialog;
import android.view.LayoutInflater;
import android.widget.EditText;

import com.farazannajmi.majesticlife.R;

public class SignupDialogFragment extends DialogFragment
{
    private EditText Username_editText;
    private EditText Email_editText;
    private EditText Password_editText;

    /* The activity that creates an instance of this dialog fragment must
     * implement this interface in order to receive event callbacks.
     * Each method passes the DialogFragment in case the host needs to query
     it. */
    public interface SignupDialogListener
    {
        public void onDialogPositiveClick(DialogFragment dialog, String
username, String email, String password);
        //public void onDialogNegativeClick(DialogFragment dialog);
    }

    // Use this instance of the interface to deliver action events
    SignupDialogListener mListener;

    // Override the Fragment.onAttach() method to instantiate the
    NoticeDialogListener
    @Override
    public void onAttach(Activity activity)
    {
        super.onAttach(activity);

        // Verify that the host activity implements the callback interface
        try
        {
            // Instantiate the NoticeDialogListener so we can send events to
the host
            mListener = (SignupDialogListener) activity;
        }
        catch (ClassCastException e) {
            // The activity doesn't implement the interface, throw exception
            throw new ClassCastException(activity.toString()
+ " must implement SignupDialogListener");
        }
    }

    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState)
    {
        Username_editText = new EditText(getActivity());
```

```

        AlertDialog.Builder builder = new
AlertDialog.Builder(getActivity());

        // Get the layout inflater
        LayoutInflater inflater = getActivity().getLayoutInflater();

        // Inflate and set the layout for the dialog
        // Pass null as the parent view because its going in the dialog
        layout
        builder.setView(inflater.inflate(R.layout.dialog_signup, null));
        // Add action buttons
        builder.setPositiveButton(R.string.sign_up, new
DialogInterface.OnClickListener()
        {
            @Override
            public void onClick(DialogInterface dialog, int id)
            {
                Username_editText =
getDialog().findViewById(R.id.SignupD_username_editText);
                Email_editText =
getDialog().findViewById(R.id.SignupD_email_editText);
                Password_editText =
getDialog().findViewById(R.id.SignupD_password_editText);

                mListener.onDialogPositiveClick(SignupDialogFragment.this,
                    Username_editText.getText().toString(),
                    Email_editText.getText().toString(),
                    Password_editText.getText().toString());
            }
        });
        builder.setNegativeButton(R.string.cancel, new
DialogInterface.OnClickListener()
        {
            public void onClick(DialogInterface dialog, int id)
            {
                SignupDialogFragment.this.getDialog().cancel();
            }
        });
        builder.setTitle("Please enter your info for sign up.");

        // Create the AlertDialog object and return it
        return builder.create();
    }
}

```

**Abstract:**

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

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



**Technical and Vocational University  
Shariaty Technical College**

# **Life style and planning application using gamification for Android OS**

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

**By:  
Narges Farazan, Zahra Najmi**

**Supervisor:  
Dr. Iman Sharifi**

**July 2018**