پروژه باریوم

شبکه های تلفن همراه

امیر فخار زاده ، هانا هاشمی

۱۵ تیر ۱۴۰۳

مقدمه ابتدا در این داک مقدمه ایی از پروژه هایی که در این organization ساخته شده داده میشود سپس به بررسی ساختار و پیاده سازی بخش های مختلف پروژه می پردازیم.

پروژه شامل دو بخش client ، server است . تعاملی که این دو بابکدیگر دارند به این نحو است که client میخواهد به سرور متصل شود . سپس با توجه به عملکردی که سرور دارد خدماتی از آن دریافت کند . این خدمات میتواند بسته به نیاز مشتری متغیر باشد . مشتری پس از وصل شدن به سرور ، باید مدام سیگنال دریافتی خود را از cell serving ایی که بن آن متصل است داشته باشد و یا لاگ بندازد . اگر از یک treshhold معین ، سیگنال دریافتی کمتر بود آنگاه به سرور در قالب SMS پیغامی از بدهد شامل :

اطلاعات cell serving ، سیگنال دریافتی بر حسب دسی بل و موقعیت مکانی

البته مشتری برای دریافت خدمات از سمت سرور باید از یک لایه امنیتی عبور کند . این لایه امنیتی داخل سرور تعبیه شده تا هر کسی نتواند به آن متصل شود . این لایه امنیتی را در این پروژه از جنس فرستادن یک پسوورد از سمت client به سرور طراحی کردیم که درصورتی که پسوورد درست باشد میتواند به سرور متصل شود.

نکته ایی که برای این پروژه قابل ذکر است این است که برای هر کدام از بخش های پیاده سازی شده ، برای شبیه سازی از دو گوشی استفاده کردیم . یک گوشی در نقش client و یک گوشی در نقش server .حال به جزییات ساختار طراحی می پردازیم:

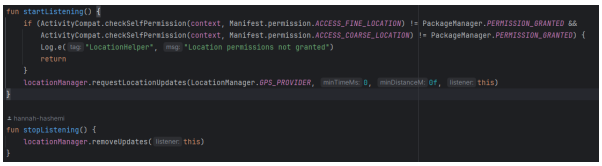
بخش اول برنامه ساخته شده در بخش client شامل چندین بخش مهم است :

- **Location Helper** این بخش همان جایی ست که موقعیت مکانی client ثبت میشود. در این تابع ، با هر تغییر مکانی که در گوشی client اتفاق بیوفتد ، یک لاگ ثبت میشود از lat و long جدید .

```
override fun onLocationChanged(location: Location) {
    Log.d( tag: "LocationHelper", msg: "Location changed: ${location.latitude}, ${location.longitude}")
}
```

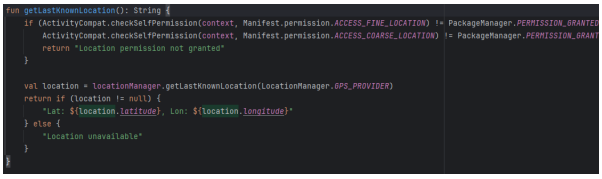
شکل ۱: location onChange function

این دو متد شروع و توقف مانیتورینگ برای آپدیت های موقعیت مکانی می باشد در متد startLis-tening ابتدا مجوز های مورد نظر چک میشود و درصورت دارابودن مجوز درخواست آپدیت موقعیت مکانی داده میشود. در متد stopListening با پاک کردن آپدیت باعث توقف مانیتورینگ تغییر موقعیت مکانی میشود.



شکل ۲: location monitoring

درمتد آخر این کلاس ما داریم که بررسی محوز های لازم، اولین قدم برای دریافت آخرین لوکیشن اعلام شده است که پس از آن با داشتن مجوز های لازم شما میتوانید آخرین لوکیشن ثبت شده را از یک GPS Provider دریافت کنید و در صورت معتبر بودن لت و لانگ آ نرا برگردانید.

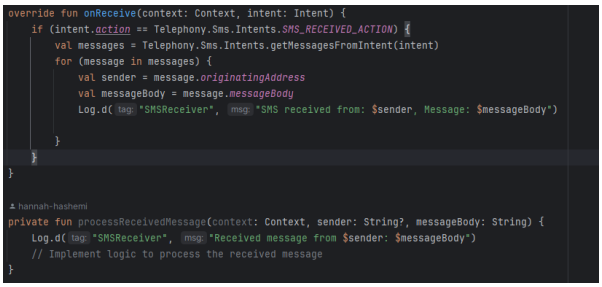


شکل ۳: get Last Location

- **SMS Service and Reciever** این سرویس در واقع همان کار ارسال SMS را برای ما انجام میدهد. دقت کنید که برای استفاده از این سرویس شما نیاز دارید که در فایل Android.xml این سرویس را ایمپورت کنید تا در کل پروژه بتوانید از آن استفاده کنید .

شکل ۴: SMS Service

در کلاس SMSReceiver هم متدی تحت عنوان onReceive وجود دارد که هنگامی که گیرنده یکSMS-RECEIVED-ACTION دریافت میکند، اطلاعات مربوط به فرستنده و پیغام را از Intent استخراج میکند. برای پردازش بیشتر یک متد دیگر در نظرگرفته شده است که دارای نام processReceivedMessage میباشد.



شکل ۵: SMS Reciever

- **Signal Strength** می توان گفت اصلی ترین بخش پیاده سازی شده در app client مربوط به این فایل است . کاری که این بخش به صورت کلی انجام میدهد این است که با در نظر گرفتن یک treshhold، میزان سیگنال دریافتی را بررسی میکند . در صورتی که از مقدار آستانه تعیین شده کمتر بود برای سرور یک SMS به صورت خودکار ارسال میشود که حاوی اطلاعاتی ست که بالاتر به آن اشاره کردیم .

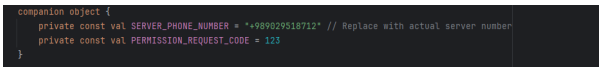
شکل ۶: Cal SignalStrength and send SMS

متد onSignalStrengthsChanged هر زمانی که قدرت سیگنال تغییر کند صدا زده میشود یا با انجام یک سری عملیات به بررسی شرایط بپردازد. ابتدا مند سوپرکلاسش را صدا میزند تا از اجرای رفتار عادی مطمئن شود بر اساس ورژن sdk قدرت سیگنال را برای یک سری از ورژن های جدید به dBm محاسبه میکند و برای ورژن های قدیمی تر این مقدار را با محاسبه ی فرمولی که بخشی از آنرا از GSM گرفته است به دست میاورد پس از ان لاگ آنرا ثبت میکند و در صورتی که strengh سیگنال از ۱۱۰- کمتر باشد آخرین لوکیشن ثبت شده و اطلاعات Cell را به دست میاورد و پیغامی حاوی این اطلاعات میسازد و آنرا برای کاربر SMS میکند.

شکل ۷: Get serving cell information

متد مهم دیگری که در این کلاس وجود دارد getCellServingInfo میباشد که از آن استفاده میکنیم که اطلاعات Serving Cell را بدست بیاوریم. ابتدا بررسی میکند که مجوز های مربوط به موقعیت مکانی را دارد این متد اطلاعات اولین سلول ثبت شده را از بین لیست همه ی سلول هایی که در دسترس دستگاه میباشد، را بر میگرداند. اگر اطلاعات سلول از تایپ CellInfoLte باشد یک سری اطلاعات خاص را استخراج و در خروجی بر میگرداند مثل TAC / CI / MNC / MCC و ... . اما اگر اطلاعات سلول در دسترس نباشد یا تایپ LTE نباشد پیغامی مبنی بر در دسترس نبودن سلول نمایان میشود.

Main حال که مدل های مورد استفاده را خیلی کامل تشریح کردیم به Logic Business پروژه میپردازیم که در فایل MainActivity.kt حضور دارد. دراین بخش در واقع جدای از کد مخصوص دسترسی ، به یک شماره از گوشی نیازمندیم که نقش سرور را ایفا میکند و SMS ها را برای آن میفرستیم.



شکل ۸: properties

بخش اصلی دیگر ، درواقع همان جایی ست که به client اجازه دسترسی داده میشود . درصورتی که دکمه connect-to-server کلیک شود این تابع فراخوانی شده و برای سرور پسوورد داخل input ، SMS میشود.

شکل ۹: properties

مهم ترین اجزای این بخش توضیح داده شد. بخش های دیگر کد مربوط به main بیشتر به مانیتور کردن و ارسال پیام به سرور می پردازد که خروجی آن را در قسمت اجرایی این گزارش مطرح میکنیم.

بخش دوم این بخش مربوط به پیاده سازی همان service بک اندی ست که در phone-server اجرا میشود .

این بخش از برنامه UI خاصی ندارد و بیشتر به عنوان یه سرور عمل میکند. ما برای شبیه سازی این بخش از یک گوشی دیگری استفاده کردیم . برنامه که در گوشی اجرا میشود ، حکم سروری ست که client به آن متصل است . کد بلاک مربوط به این سرویس اینگونه است :

شکل ۱۰: properties

در این قسمت ما دو property مهم را داریم :

در property ، چون از گوشی به عنوان سرور استفاده میکنیم که SMS های دریافتی از Clinet را بگیرد ، شماره سرور را تعیین کردیم. در property دوم ،پسوورد را به صورت hardcode قرار دادیم. این پسوورد باید با پسووردی که client برای دسترسی به سرور میفرستد یکسان باشد درغیر این صورت اجازه دسترسی به سرور را نخواهد داشت.

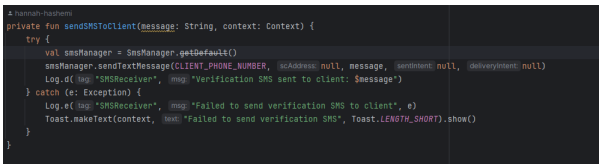
شکل ۱۱: BroadcastReceiver

عملکرد این تیکه از کد به شرح زیر است :

کامپوننت BroadcastReceiver متعلق به کتابخانه خود اندروید است و به این منظور استفاده میشود اگر در سیستم اندرویدی یاهمان گوشی ، اعلانیه هایی مثل کم شدن باتری ، بوت شدن سیستم و یا زمانی که SMS ایی دریافت کند. اگر از هرکدام از این موارد اتفاق بیوفتد ، onReceive فراخوانی میشود .

بعد از آن چک میشود که اعلانیه که در سیستم وجود دارد آیا از نوع SMS است یا نه . اگر این شرط برقرار بود شروع به چک کردن هر کدام از پیام های دریافتی میکند .

در هر کدام از پیام ها اگر SMS دریافتی متعلق به شماره client بود ، آنگار محتویات داخل مسیج ها را چک میکند . سرور در محتوی پیام به دنبال همان کلمه رمزی که client فرستاده میگردد . اگر کلمه عبور وجود داشت و درست بود ، به client این sms را میدهد که verify شده و میتواند از سرور استفاده کند.

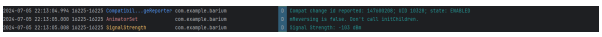


شکل ۱۲: server sends SMS to verify client

در آخر نیز وقتی client از سوی سرور verify شد ، برای SMS client، ایی ارسال میکند که بیانگر verify شدن موفقیت آمیز client است .

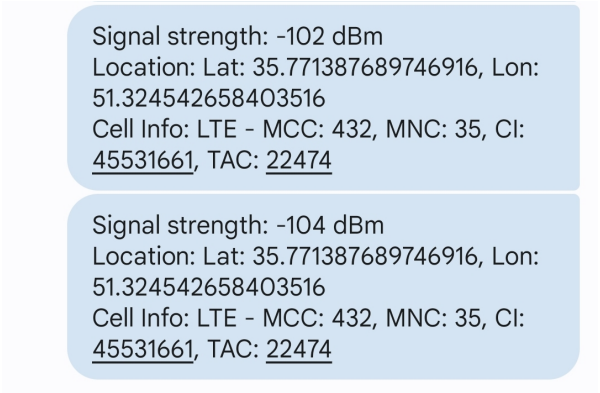
همانجور که از پیاده سازی ها واضح است ، این قسمت پیاده سازی در سرور بیشتر جنبه سرویس امنیتی را دارد که برای هر client ایی اجازه دسترسی به دیگر اجزای سرور را ندهد .

مرحله اجرایی ابتدا باید به این امر توجه کرد که هر کدام از اپ های بخش client و server باید در گوشی های جداگانه و به صورت همزمان ران شوند که بتوان خروجی مد نظر را مشاهده کرد . وقتی اپ سمت client را اجرا میگیریم ، از شما پسووردی میخواهد که بتوانید به سرور متصل شوید . پس از دریافت پسوورد از سمت سرور مراحل مانیتور کردن client شروع میشود .



شکل ۱۳: SMS Reciever

برای تست کردن ارسال SMS، مقدار treshhold را ۱۰۰db- قرار میدهیم تا شرط فرستادن SMS برقرار شود . SMS ایی که از سمت client به سرور فرستاده میشود به این شکل خواهد بود :



شکل ۱۴: SMS Reciever

همان طور که مشاهده میشود، تمامی اطلاعات از جمله ناحیه TAC نیز برای سرور فرستاده میشود . اطلاعاتی دیگری مثل اپراتوری که به آن متصل است نیز در این پیغام آمده که MNC=۳۵ مربوط به اپراتور ایرانسل است .