پایان نامه دوره کارشناسی کامپیوتر

دانشکده مهندسی کامپیوتر و فناوری اطلاعات

گرایش مهندسی کامپیوتر

موضوع :

اپلیکیشن موبایل برای یافتن

کوتاهترین مسیر بر روی نقشه شهری برای دو موقعیت داده شده

استاد راهنما :

دکتر رضا شمسائی

نام دانشجو :

محمدمهدی صفرمحمدلو

تیر ماه سال ۱۴۰۰


بسم الله الرحمن الرحيم

تقدیم به

تمام رهپویان علم و معرفت

چکیده

در توسعه این اپلیکیشن تلاش شده است کوتاه ترین مسیر بر روی نقشه شهری برای دو موقعیت داده شده با استفاده از API مسیریابی Open Route Service محاسبه شود و به واسطه نقشه ارائه شده توسط گوگل ، مسیر نشان داده شود.

در این پروژه از ابزار های رسمی و پیشنهادی گوگل برای توسعه ی اپلیکیشن اندروید نظیر Android Studio ، Jetpack Compose ، Kotlin و ... استفاده شده است.

در طراحی ظاهری اپلیکیشن از استاندارد گوگل برای طراحی اپلیکیشن های اندروید به نام زبان طراحی متریال (Material Design) استفاده شده است.

فهرست مطالب

عنوان **صفحه**

پایان نامه دوره کارشناسی کامپیوتر

دانشکده مهندسی کامپیوتر و فناوری اطلاعات

گرایش مهندسی کامپیوتر

موضوع :

اپلیکیشن موبایل برای یافتن

کوتاهترین مسیر بر روی نقشه شهری برای دو موقعیت داده شده

استاد راهنما :

دکتر رضا شمسائی

نام دانشجو :

محمدمهدی صفرمحمدلو

تیر ماه سال ۱۴۰۰


بسم الله الرحمن الرحيم

۱-۱ مقدمه

با گسترش اپلیکیشن های موبایل و افزایش شدید دسترسی مردم به این ابزار ، استفاده از امکانات و سرویس های مختلف برای راحت تر کردن قسمت های مختلف زندگی به موضوع مهمی در این دوران تبدیل شده است.

یکی از مهم ترین موضوعات توی حوزه ی مسافرت درون شهری ، مسیر یابی دقیق مقصد با توجه به فاکتور هایی نظیر مسافت ، ترافیک ، امنیت و ... است.

در مقاله Public Transit Route Mapping for Large-Scale Multimodal Networks[۱] به اهمیت این موضوع اشاره شده است که مسیریابی مناسب چه تاثیراتی بر هزینه ها و زمان دارد.

در این مقاله بیان می کند که شبکه حمل و نقل عمومی بزرگی با ویژگی های مختلف ارتباطی و تعدد مسیر نیازمند یک مسیریابی دقیق برای کاهش هزینه ، افزایش سرعت و بازدهی مناسب زمانی می باشد.

۱-۲ تعریف پروژه

این پروژه بر اساس سرویس نقشه آنلاین گوگل (Google Maps) و سرویس مسیریابی آنلاین OpenRouteService[۲] تعریف شده است.

قصد داریم با طراحی اپلیکیشن و ارائه محیطی برای نمایش نقشه و گرفتن دو موقعیت از کاربر و استفاده از سرویس ها ، کوتاه ترین مسیر بر روی نقشه شهری برای دو موقعیت داده شده را بدست آوریم و در نقشه نشان دهیم.

در فصل سوم جزئیات این پروژه به طور کامل بیان شده است.

[۱] August 2017International Journal of Geo-Information
[۲] openrouteservice.org

۳-۱ هدف پروژه

با بدست آوردن کوتاه ترین مسیر بر روی نقشه شهری برای دو موقعیت داده شده و نشان دادن آن مسیر در نقشه ، میتوان با توجه به سرعت محاسبه مسیر و سهولت استفاده از آن ، از آن برای مسیریابی سفر های درون شهری استفاده کرد.

۲ ابزارهای مورداستفاده

برای توسعه این اپلیکیشن از ابزار های رسمی و پیشنهادی گوگل برای توسعه ی اپلیکیشن های اندروید استفاده شده است.

اندروید استودیو (Android Studio) محیط اصلی توسعه اپلیکیشن اندروید می باشد که تمام کار های برنامه نویسی و مدیریت پروژه در آن صورت می گیرد.

زبان اصلی برنامه نویسی اپلیکیشن های اندروید کاتلین (Kotlin) و ابزار های مخصوص آن نظیر Kotlin Coroutine [۳]برای انجام کار ها یا تسک های ناهمزمان (Asynchronous) و Kotlinx Serialization[۴] برای خواندن و آنالیز داده های با نوع Json در توسعه ی این اپلیکیشن استفاده شده است.

همچنین برای توسعه قسمت ظاهری اپلیکیشن از ابزار جدید گوگل با نام Jetpack Compose[۵] که به تازگی معرفی شده استفاده شده است و در طراحی ظاهری اپلیکیشن ، المان ها و زبان طراحی متریال (Material Design) مورد استفاده قرار گرفته است.

در این اپلیکیشن از سرویس نقشه آنلاین گوگل (Google Maps) و سرویس مسیریابی آنلاین OpenRouteService استفاده شده است که به واسطه HTTP کلاینت Ktor[۶] با اپلیکیشن ارتباط برقرار میکنند.

۳ kotlinlang.org/docs/coroutines-overview.html
٤ github.com/Kotlin/kotlinx.serialization
٥ developer.android.com/jetpack/compose
٦ ktor.io

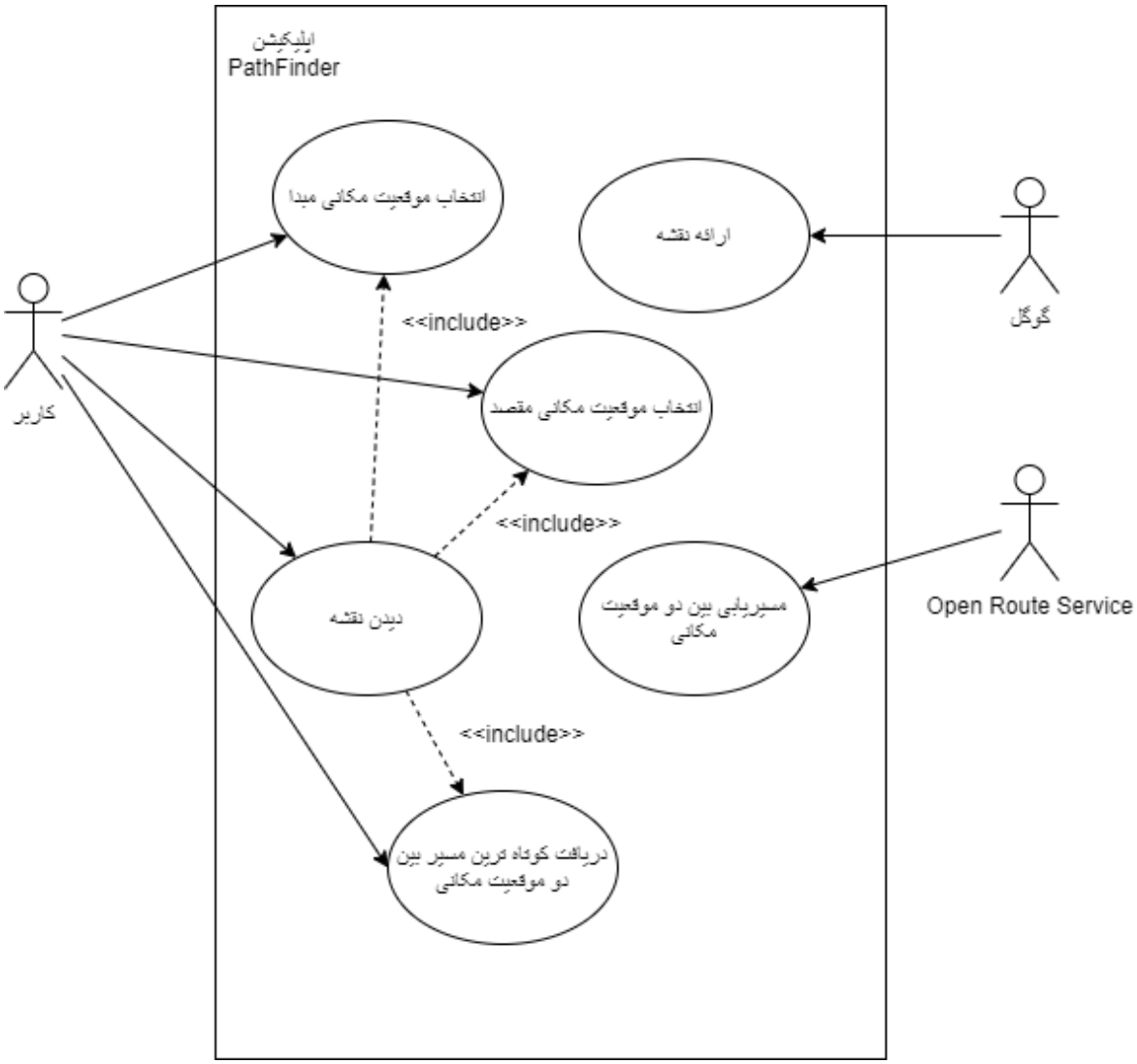
۳ تحلیل پروژه

در این فصل قصد داریم به معرفی کوتاهی از آزمون های بیان مورد بحث در فصول گذشته بپردازیم.

۳-۱ نمودار Use Case :

این نمودار نشان‌دهنده‌ی روابط بین کنشگرها و موارد کاربرد است. یک نمودار مورد کاربرد، کاربرهای مختلف سیستم را در ارتباط با موارد کاربرد متفاوت نمایش می‌دهد. هدف از استفاده از نمودارUse Case ، نمایش گرافیکی ابعاد دینامیکی یک سیستم است.

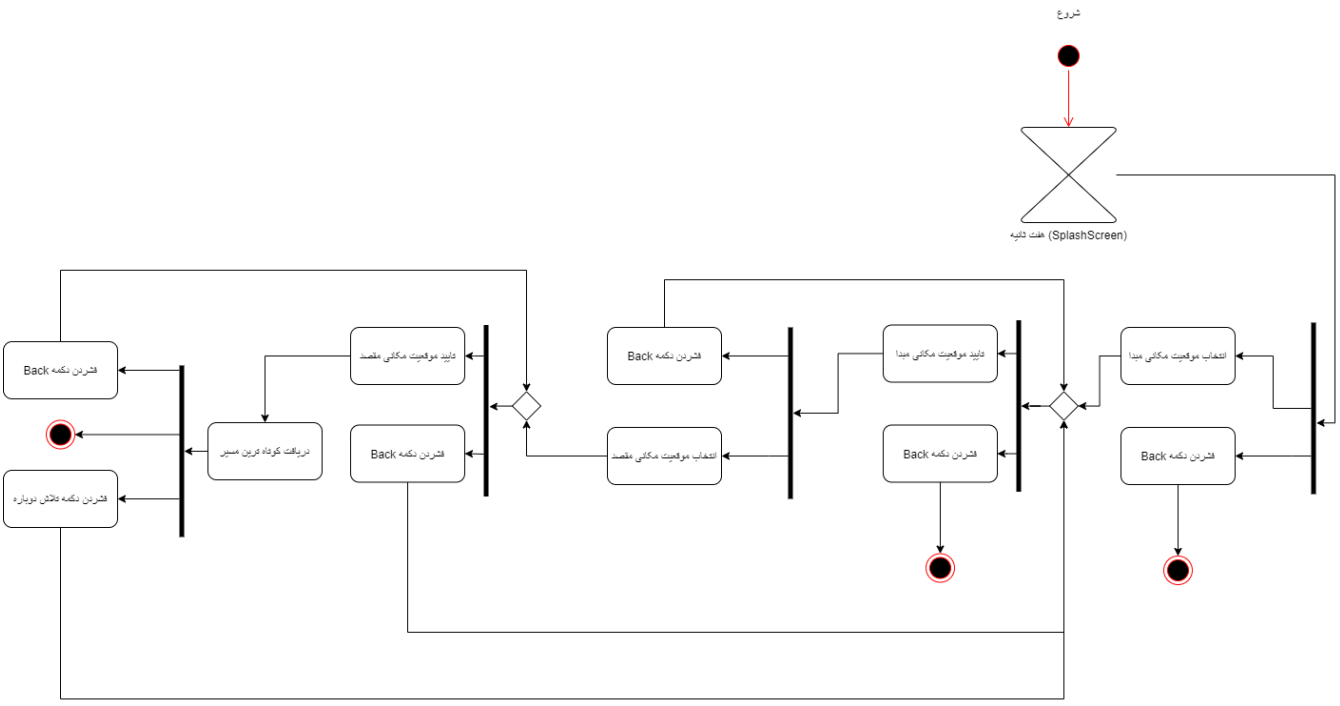
تصویر ۳-۱ نمودار UseCase



۲-۳ نمودار Activity :

این نمودار نمایش گرافیکی از گردش کار در فعالیت‌ها و اقدامات در یک سناریوی مشخص می‌باشد.

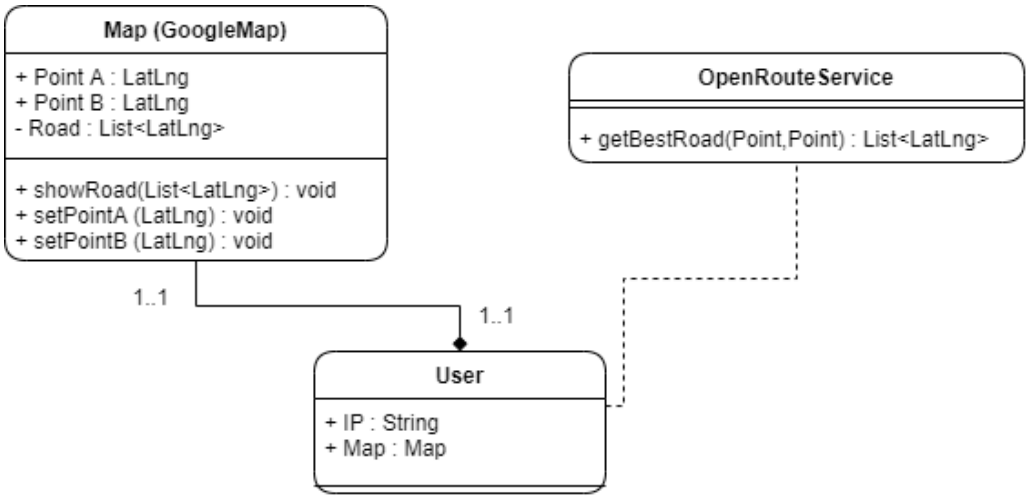
تصویر ۳-۲ نمودار Activity



۳-۳ نمودار Class :

نمودار کلاس یک نوع از نمودارهای ساختاری ایستاست که ساختار یک سیستم را با نمایش کلاسهای سیستم، خصوصیات آنها و روابط بین آنها توصیف می‌کند.

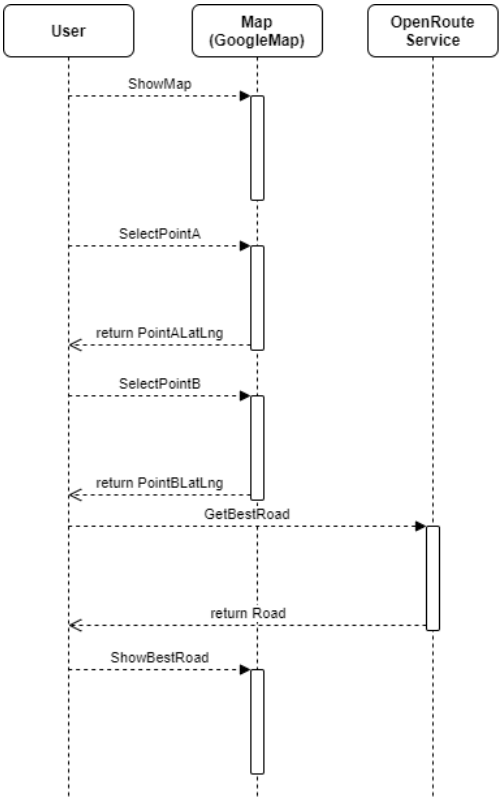
تصویر ۳-۲ نمودار Class



۳-۴ نمودار Sequence :

نمودار توالی (Sequence Diagram) یکی از نمودارهای زبان مدل‌سازی یکپارچه است که روندی در یک پروژه را مرحله به مرحله نشان می‌دهد.

تصویر ۴-۲ نمودار Sequence



٤ شرح پروژه

برای راحتی بیشتر و افزایش سرعت و کارایی اپلکیشن و استفاده از طراحی بهتر UX [۷]، این اپلیکیشن در دو صفحه طراحی شده است.

در ادامه این فصل به صورت کامل صفحهات اپلیکیشن شرح داده شده است.

٤-١ صفحه Splash Screen :

صفحه اسپلش به یک استاندارد برای اپلکیشن تبدیل شده است و استفاده های متعددی دارد نظیر نمایش Loading ، چک کردن اینترنت کاربر ، معرفی و تبلیغات و ... .

در اینجا هدف استفاده از صفحه اسپلش یا SplashScreen معرفی سازنده بوده است.

شکل ۴-۱ Screenshot از صفحه SplashScreen

[۷] *User experience*

با توجه به شکل ۴-۱ ، صفحه شامل ۴ قسمت لوگو اپلیکیشن ، نام اپلیکیشن ، لوگو سازنده و یک Progress Bar به صورت دایره ای می باشد.

۴-۲ صفحه اصلی :

در صفحه اصلی تمام اعمال اصلی اپلیکیشن انجام می شود.

با توجه به ساختار اپلیکیشن و نمودار Activity و طراحی UX متناسب با سادگی ، این اپلیکیشن دارای ۶ حالت یا State می باشد.

شکل ۴-۲ قسمت آپدیت کردن State

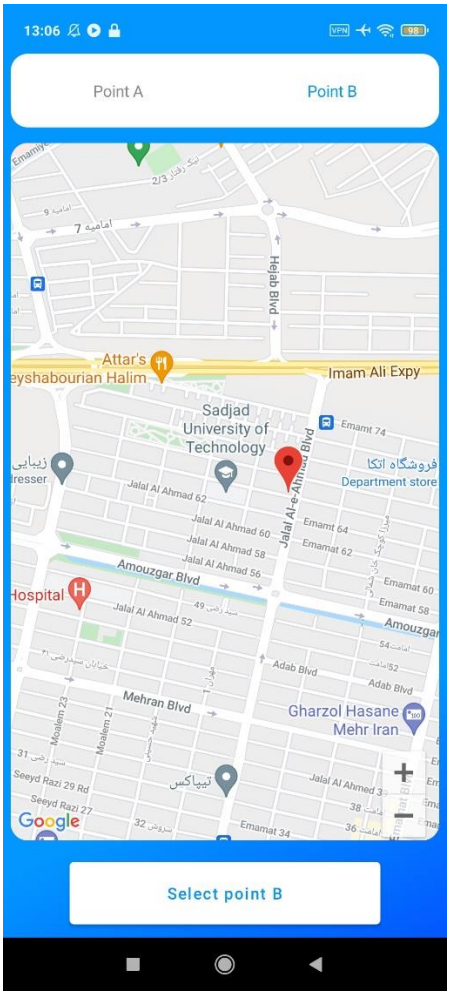
```
fun setState (nextState:Int){
    when (nextState){
        0 -> { // start
            state = 0
            mapWeightState = 0.9f
            buttonTextState = Strings.button0
        }
        1 -> { // a
            state = 1
            mapWeightState = 0.8f
            buttonTextState = Strings.button1
            title1ColorState = Colors.Main
            title2ColorState = Colors.Gray
        }
        2 -> { // b
            state = 2
            mapWeightState = 0.8f
            buttonTextState = Strings.button2
            title1ColorState = Colors.Gray
            title2ColorState = Colors.Main
        }
        3 -> { // loading
            state = 3
            mapWeightState = 0.9f
        }
        4 -> { // failed
            state = 4
            mapWeightState = 0.9f
            buttonTextState = Strings.button4
        }
        5 -> { // success
            state = 5
            mapWeightState = 0.9f
            buttonTextState = Strings.button5
        }
    }
```

با کمک ابزار و روش جدید برنامه نویسی اندروید توسط گوگل به نام Jetpack Compose با تغییر حالت ها UI به صورت اتوماتیک تغییر میکند و نیازی به صدا زدن تگ ها و آی دی های مختلف از XML[۸] ها نمی باشد.

البته در این روش جدید کلا XML هم وجود ندارد و فقط در کاتلین طراحی انجام می شود.

وقتی نقطه ای در نقشه تاچ میشود گوگل مپ موقعیت مکانی آن نقطه رو به صورت Latitude Longitude برمیگرداند.

شکل ۴-۳ قسمت انتخاب موقعیت مکانی B



همانطور که در شکل ۳-۴ مشاهده می کنید تمام رنگ ها و المان ها و دکمه ها با توجه به استاندارد گوگل برای طراحی اپلیکیشن های اندرویدی با عنوان Material Design انتخاب شده است.

۸ Extensible Markup Language

Kotlin Coroutine این قابلیت رو در اختیار ما قرار می دهد تا به سادگی بدون آن که Thread UI درگیر یا بلاک بشود (که باعث هنگ کردن سیستم می شود) بتوانیم یک یا چند عملیات را هم زمان اجرا کنیم ، AsyncTask روشی بود که قبلا استفاده میشد و مشکلات بسیار زیادی داشت اما Coroutine بسیار ساده است.

با تایید نقطه دوم ، یک Coroutine کاتلین اجرا می شود تا از سرویس open route service با توجه به نقاطی که گوگل مپ ارائه کرده است کوتاه ترین مسیر را درخواست کند ، این درخواست توسط Ktor صورت می گیرد ، ابزار های مشابهی همچون retrofit[۹] و okhttp[۱۰] نیز وجود داشتند اما سازگاری بهتر ktor با زبان برنامه نویسی کاتلین و همچنین همگام بودن با Kotlin Coroutine باعث شد ktor به عنوان http client در نظر گرفته شود.

در جواب این درخواست ، تعداد نقاطی راه که یک داده به صورت json گرفته می شود که با استفاده از kotlinx serialization آن را parse می کنیم.

سپس آن نقاط گرفته شده را در گوگل مپ به سادگی رسم می کنیم.

شکل ۴-۴ Parse کردن Json و نمایش آن

```
val json = Json {
    isLenient = true
    prettyPrint = true
    ignoreUnknownKeys = true
}


val element = json.parseToJsonElement(stringBody)

element
    .jsonObject["features"]!!
    .jsonArray[0]
    .jsonObject["geometry"]!!
    .jsonObject["coordinates"]!!
    .jsonArray.forEach {
        road.add(LatLng(it.jsonArray[1].jsonPrimitive.double,
            it.jsonArray[0].jsonPrimitive.double
        ))
    }

road.add(pointBLatlng!!)

val polylineOptions = PolylineOptions()
road.forEach {
    polylineOptions.add(it)
}
polyLine = map?.addPolyline(polylineOptions)
```

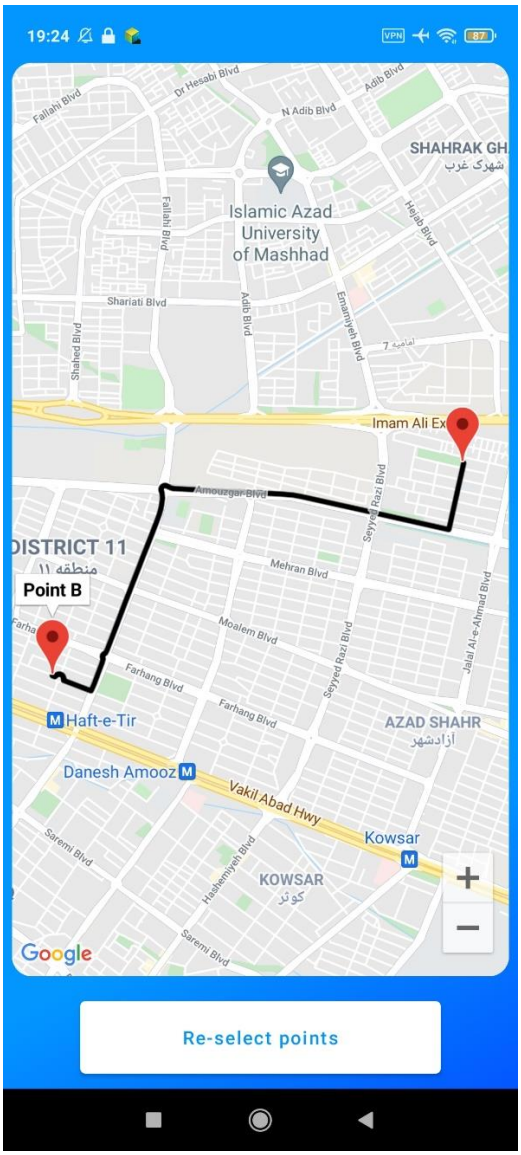
۸ square.github.io/retrofit
۹ square.github.io/okhttp

۳-٤ ارزیابی :

با توجه به این که سیستم مسیریابی گوگل مپ برای برنامه نویسان ایرانی تحریم شده است این اپلیکیشن نشان داد که می توان با استفاده از سرویس های جایگزین و جهانی (بر خلاف سرویس های ایرانی نظیر بلد و نشان و map.ir) و دقت بالای سرویس OpenRouteService بطور اندک این محدودیت ها را جبران کرد.

در دو شکل ۴-۵ و ۴-۶ مقایسه ای بین مسیریابی گوگل مپ و این اپلیکیشن را مشاهده می کنید که یک مسیر مشابه را به عنوان کوتاه ترین مسیر بین دو نقطه یکسان ، معرفی کرده اند.

شکل ۴-۵ مقایسه مسیریابی گوگل مپ و اپلیکیشن توسعه داده شده (PathFinder)



شکل ۴-۵ مقایسه مسیریابی گوگل مپ و اپلیکیشن توسعه داده شده (GoogleMap)