توضیح:

بخش های مختلف پروژه:

در بخش GeocodingApiService.kt داریم:

کد زیر یک رابط (interface) برای سرویس Geocoding API از طریق کتابخانه Retrofit در اندروید تعریف می‌کند. این رابط شامل یک تابع برای دریافت آدرس از مختصات جغرافیایی است.

```
package com.tmdsimple.locationapp


import retrofit2.http.GET
import retrofit2.http.Query


interface GeocodingApiService {


    @GET("maps/api/geocode/json")
    suspend fun getAddressFromCoordinates(
        @Query("latlng") latlng : String,
        @Query("key") apiKey : String
    ): GeocodingResponse
{
```

پکیج:

- package com.tmdsimple.locationapp: این خط نشان می‌دهد که کلاس GeocodingApiService در پکیج com.tmdsimple.locationapp قرار دارد.

تعریف رابط (interface):

- interface GeocodingApiService: این خط شروع تعریف یک رابط برای API است.

تابع getAddressFromCoordinates:

- @GET("maps/api/geocode/json"): این آدرس URL نقطه انتهایی API (endpoint) را مشخص می‌کند که از متد HTTP GET استفاده می‌کند.
- suspend fun getAddressFromCoordinates(: این یک تابع معلق (suspend) است که از Coroutines در Kotlin برای انجام عملیات شبکه به صورت غیرهمزمان استفاده می‌کند.
- @Query("latlng") latlng : String,: این پارامتر latlng را به عنوان یک رشته (String) می‌پذیرد که شامل مختصات جغرافیایی (عرض جغرافیایی و طول جغرافیایی) است.
- @Query("key") apiKey : String: این پارامتر apiKey را به عنوان کلید API برای احراز هویت درخواست به سرویس Geocoding ارسال می‌کند.
- GeocodingResponse: (این تابع یک شیء GeocodingResponse را به عنوان نتیجه درخواست برمی‌گرداند.

نتیجه:

این رابط یک درخواست GET به سرویس Google Geocoding API ارسال می‌کند تا آدرس متناظر با مختصات جغرافیایی مشخص شده را دریافت کند. پارامتر latlng باید به فرمت "latitude,longitude" باشد و پارامتر apiKey باید شامل کلید API معتبر برای استفاده از این سرویس باشد.

در بخش LocationData.kt داریم:

LocationData:

- این کلاس داده حاوی اطلاعات مکانی است و شامل دو متغیر است:
  - latitude: عرض جغرافیایی به عنوان یک عدد اعشاری.(Double)
  - longitude: طول جغرافیایی به عنوان یک عدد اعشاری.(Double)

این کلاس معمولاً برای ذخیره و انتقال اطلاعات مختصات جغرافیایی استفاده می‌شود.

GeocodingResponse:

- این کلاس داده نمایانگر پاسخ دریافتی از سرویس Geocoding است و شامل دو متغیر است:
  - results: یک لیست از اشیاء GeocodingResults.
  - status: وضعیت پاسخ به صورت یک رشته.(String)

این کلاس برای مدیریت داده‌های دریافتی از API Geocoding استفاده می‌شود و وضعیت درخواست و نتایج جغرافیایی را ذخیره می‌کند.

GeocodingResults:

- این کلاس داده شامل نتایج جغرافیایی است و شامل یک متغیر است:
  - formatted_address: آدرس فرمت شده به عنوان یک رشته.(String)

این کلاس برای ذخیره آدرس‌های فرمت شده که از API Geocoding دریافت شده‌اند، استفاده می‌شود.

```
package com.tmdsimple.locationapp

data class LocationData(
    val latitude: Double,
    val longitude: Double
)

data class GeocodingResponse(
    val results: List<GeocodingResults>,
    val status: String
)

data class GeocodingResults(
    val formatted_address: String
(
```

در بخش LocationSelectionScreen.kt داریم:

```
package com.tmdsimple.locationapp


import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.Button
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import com.google.android.gms.maps.model.CameraPosition
import com.google.android.gms.maps.model.LatLng
import com.google.maps.android.compose.GoogleMap
import com.google.maps.android.compose.Marker
import com.google.maps.android.compose.MarkerState
import com.google.maps.android.compose.rememberCameraPositionState


@Composable


fun LocationSelectionScreen(
   location : LocationData,
   onLocationSelected: (LocationData) ->Unit)
  {
     val userLocation = remember{
       mutableStateOf(LatLng(location.latitude,location.longitude))
```

```
}

  var cameraPositionState = rememberCameraPositionState{
   position = CameraPosition.fromLatLngZoom(userLocation.value, 10f)
}

Column(modifier = Modifier.fillMaxSize()) {

   GoogleMap(
      modifier = Modifier
         .weight(1f)
         .padding(16.dp),
      cameraPositionState = cameraPositionState,
      onMapClick = {
         userLocation.value = it
      }

   ) {
       Marker(state = MarkerState(userLocation.value))
   }

   var newLocation : LocationData

   Button(onClick = {

      newLocation = (LocationData(userLocation.value.latitude,userLocation.value.longitude))
      onLocationSelected(newLocation)
```

```
    }) {
      Text(text = "Set Location")
    }
{
```

پارامتر locationاطلاعات اولیه مکان که شامل عرض و طول جغرافیایی است.

پارامتر onLocationSelectedیک تابع که زمانی که مکان انتخاب شد، فراخوانی می‌شود.

userLocation یک وضعیت قابل‌تغییر که مکان فعلی کاربر را نگه می‌دارد.

cameraPositionState وضعیت دوربین نقشه که موقعیت اولیه و زوم را تنظیم می‌کند.

Column: برای چیدمان عمودی عناصر.

GoogleMap: نقشه Google را نمایش می‌دهد و به کلیک‌های نقشه واکنش نشان می‌دهد.

- **modifier**تنظیمات ظاهری نقشه.
- **cameraPositionState**وضعیت دوربین نقشه.
- **onMapClick**تابعی که وقتی کاربر روی نقشه کلیک می‌کند، مکان کلیک را به userLocationاختصاص می‌دهد.

Marker: نشانگر مکان فعلی کاربر روی نقشه.

Button: دکمه‌ای که مکان انتخاب شده را تنظیم و تابع onLocationSelectedرا فراخوانی می‌کند.

- **Text**متن دکمه که "Set Location" را نشان می‌دهد.

در بخش LocationViewModel.kt داریم:

```
package com.tmdsimple.locationapp


import android.util.Log
import androidx.compose.runtime.State
import androidx.compose.runtime.mutableStateOf
import androidx.lifecycle.ViewModel
```

```
import androidx.lifecycle.viewModelScope
import kotlinx.coroutines.launch

class LocationViewModel: ViewModel() {

    private val _location = mutableStateOf<LocationData?>(null)
    val location: State<LocationData?> = _location

    private val _address = mutableStateOf(listOf<GeocodingResults>())
    val address : State<List<GeocodingResults>> = _address

    fun updateLocation(newLocation : LocationData){
        _location.value = newLocation
    }

    fun fetchAddress(latlng:String) {
        try {
            viewModelScope.launch {
                val result = RetrofitClient.create().getAddressFromCoordinates(
                    latlng,
                    "AIzaSyBcrUW6OoEeCOs9sRctj59UNWwGjUZTVRY"
                )
                _address.value =result.results
                //Log.d("err1", " ${_address.value} ")
            }
        } catch (e:Exception){
            Log.d("err1", " ${e.cause} ${e.message}")
        }
```

}

{

کلاس LocationViewModel

این کلاس از ViewModel ارث‌بری می‌کند که بخشی از معماری Jetpack است و برای مدیریت داده‌های UI به صورت ایمن استفاده می‌شود.

متغیرهای وضعیت

location: یک متغیر خصوصی قابل تغییر که مکان فعلی را نگه می‌دارد. مقدار اولیه آن nullاست.

location: یک وضعیت عمومی فقط خواندنی که به UI امکان مشاهده مکان فعلی را می‌دهد.

_address: یک متغیر خصوصی قابل تغییر که نتایج جغرافیایی را نگه می‌دارد.

address: یک وضعیت عمومی فقط خواندنی که به UI امکان مشاهده آدرس‌های جغرافیایی را می‌دهد.

متد updateLocation

این متد مکان فعلی را با یک مکان جدید به‌روزرسانی می‌کند.

متد fetchAddress

fetchAddress: یک متد که آدرس مربوط به مختصات جغرافیایی داده شده را دریافت می‌کند.

- try/catch: برای مدیریت استثناها و لاگ کردن خطاها استفاده می‌شود.
- viewModelScope.launch: یک Coroutine جدید در محدوده ViewModel راه‌اندازی می‌کند.
- RetrofitClient.create().getAddressFromCoordinates: یک درخواست به API Geocoding با استفاده از Retrofitارسال می‌کند.
- _address.value = result.results: نتایج جغرافیایی را در وضعیت _address ذخیره می‌کند.
- Log.d: برای لاگ کردن خطاها استفاده می‌شود.

در بخش mainactivity.kt داریم:

MainActivity

این کلاس فعالیت اصلی برنامه‌ی اندروید را تعریف می‌کند. در اینجا ViewModel برای مدیریت داده‌ها و رابط کاربری استفاده شده است و تابع setContentاز Compose برای تنظیم رابط کاربری استفاده شده است.

Navigation Composable

این تابع شامل یک NavHostاست که مسیرهای مختلف را مدیریت می‌کند. دو مسیر اصلی داریم:

- mainscreen:نمایش موقعیت فعلی و امکان انتخاب موقعیت جدید.
- locationselectionscreen:نمایش صفحه‌ای برای انتخاب موقعیت جدید توسط کاربر.

DisplayLocation Composable

این تابع برای نمایش اطلاعات مربوط به موقعیت فعلی و امکان انتخاب موقعیت جدید توسط کاربر طراحی شده است. از MyLocationUtilsبرای دسترسی به ابزارهای مکان‌یابی استفاده می‌شود. همچنین از requestPermissionLauncherبرای درخواست مجوزهای لازم برای دسترسی به موقعیت استفاده شده است.