PROG D

ICE TASK 2

Kayla Ferreira
VARSITY COLLEGE | ST10259527

Table of Contents

Check Location

```
1 Usage
private fun checkPermissionsAndRequestLocation() {
    val hasFineLocationPermission =
        ActivityCompat.checkSelfPermission(
            context = this,  permission = "android.permission.ACCESS_FINE_LOCATION"
        )
    val hasCourseLocationPermission =
        ActivityCompat.checkSelfPermission(
            context = this,  permission = "android.permission.ACCESS_COARSE_LOCATION"
        )
    if (hasFineLocationPermission != PackageManager.PERMISSION_GRANTED &&
        hasCourseLocationPermission != PackageManager.PERMISSION_GRANTED
    ) {
        val permissions = arrayOf<String>(
            android.Manifest.permission.ACCESS_FINE_LOCATION,
            android.Manifest.permission.ACCESS_COARSE_LOCATION
        )
        ActivityCompat.requestPermissions( activity = this, permissions,  requestCode = 0)
    } else {
        getLocationAndCreateUI()
    }
}

@RequiresPermission(allOf = [Manifest.permission.ACCESS_FINE_LOCATION, Manifest.permission.ACCESS_COAR
override fun onRequestPermissionsResult(
    requestCode: Int,
    permissions: Array<out String>,
    grantResults: IntArray
) {
    super.onRequestPermissionsResult(
        requestCode, permissions
```

Build Location and Display data function

```
115     private fun buildLocationRequest(): LocationRequest {
116         val locationRequest = LocationRequest()
117         locationRequest.priority = LocationRequest.PRIORITY_HIGH_ACCURACY
118         locationRequest.interval = 5000
119         locationRequest.fastestInterval = 3000
120         return locationRequest
121     }
122
            1 Usage
123     private fun buildLocationCallback(): LocationCallback {
124         return object : LocationCallback() {
125             override fun onLocationResult(locationResult: LocationResult) {
126                 val location = locationResult.lastLocation
127                 Log.i( tag = "LocationResult",  msg = "onLocationResult: $location")
128                 model.setLocation("${location.latitude},${location.longitude}")
129             }
130         }
131     }
132
            1 Usage
133     fun displayData(location: AccuWeatherLocation) {
134         if (location.Key.isNullOrEmpty()) {
135             Log.e( tag = "MainActivity2",  msg = "Cannot display tabs: location key is null")
136             return
137         }
138
139         Log.i( tag = "MainActivity2",  msg = "Creating tabs for key: ${location.Key}")
140
141         val sectionsPagerAdapter = SectionsPagerAdapter(
142             context = this,
```

```
        Log.i( tag = "MainActivity2",  msg = "Creating tabs for key: ${location.Key}")

        val sectionsPagerAdapter = SectionsPagerAdapter(
            context = this,
            fm = supportFragmentManager,
            locationName = location.LocalizedName ?: "Unknown",
            locationKey = location.Key.toString()
        )

        binding.viewPager.adapter = sectionsPagerAdapter
        binding.tabs.setupWithViewPager(binding.viewPager)
    }
```

Fetch Current Weather

```
    private fun fetchCurrentWeather(locationKey: String?) {
        viewModelScope.launch {
            try {
                val response = RetrofitClient.weatherService?.getCurrentWeather(
                    locationKey,
                     apiKey = BuildConfig.ACCUWEATHER_API_KEY
                )
                _currentWeather.value = response ?: emptyList()
            } catch (e: Exception) {
                Log.e( tag = "CurrentWeatherViewModel",  msg = e.message ?: "Error")
            }
        }
    }
```

IAccuweather

```
    2 Usages
    @GET( value = "forecasts/v1/daily/5day/{locationKey}")

    suspend fun getFiveDayForecast(
        @Path( value = "locationKey") locationKey: String?,
        @Query( value = "apikey") apiKey: String?,
        @Query( value = "metric") metric: Boolean
    ): FiveDayForecast?

    2 Usages
    @GET( value = "currentconditions/v1/{locationKey}")
    suspend fun getCurrentWeather(
        @Path( value = "locationKey") locationKey: String?,
        @Query( value = "apikey") apiKey: String?
    ): List<CurrentWeather?>?

    2 Usages
    @GET( value = "locations/v1/cities/geoposition/search")
    suspend fun getLocationByPosition(
        @Query( value = "q") geoposition: String?,
        @Query( value = "apikey") apiKey: String?
    ): AccuWeatherLocation?

}
```

Section Page Adapter

```kotlin
/**
 * A [FragmentPagerAdapter] that returns a fragment corresponding to
 * one of the sections/tabs/pages.
 */
2 Usages
class SectionsPagerAdapter(private val context: Context,
                           fm: FragmentManager,
    private val locationName: String,
    private val locationKey: String) :
    FragmentPagerAdapter(fm) {

    override fun getItem(position: Int): Fragment {
        when (position) {
            0 -> return CurrentWeatherFragment.newInstance(locationName, locationKey)
            1 -> return DailyForecastsFragment()

        }
        // getItem is called to instantiate the fragment for the given page.
        // Return a PlaceholderFragment.
        return PlaceholderFragment.newInstance( sectionNumber = position + 1)
    }

    override fun getPageTitle(position: Int): CharSequence? {
        return context.resources.getString( id = TAB_TITLES[position])
    }

    override fun getCount(): Int {
        // Show 2 total pages.
        return 3
    }
}
```

Current Conditions Model

```
import ...

1 Usage
class CurrentConditionsViewModel : ViewModel() {

    3 Usages
    private val _currentConditions = MutableLiveData<List<CurrentWeather>>()
    1 Usage
    val currentWeather: LiveData<List<CurrentWeather>> = _currentConditions

    1 Usage
    fun getCurrentWeather(locationKey: String) {
        viewModelScope.launch {
            try {
                val weatherData = RetrofitClient.weatherService
                    ?.getCurrentWeather(locationKey,  apiKey = BuildConfig.ACCUWEATHER_API_KEY)
                // weatherData is List<CurrentWeather?>?
                // Assign non-null list or empty list to LiveData
                _currentConditions.value = weatherData?.filterNotNull() ?: emptyList()
            } catch (e: Exception) {
                // Handle errors by clearing or logging
                _currentConditions.value = emptyList()
            }
        }
    }
}
```