Enable Location Service to The Application

Date	09 NOV 2022
Team ID	PNT2022TMID19660
Project Name	CONTAINMENT ZONE ALERTING APPLICATION
Team Members:	Vibin U, Vigneshwar C.U, Srinath R, Satheeshkumar A

Code:

```
package com.example.containmentzone


import android.Manifest
import android.content.pm.PackageManager
import android.location.Address
import android.location.Geocoder
import android.os.Bundle
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import androidx.appcompat.widget.SearchView
import androidx.core.app.ActivityCompat
import androidx.fragment.app.Fragment
import androidx.navigation.fragment.findNavController
import com.google.android.gms.location.LocationServices
import com.google.android.gms.maps.CameraUpdateFactory
import com.google.android.gms.maps.GoogleMap
import com.google.android.gms.maps.OnMapReadyCallback
import com.google.android.gms.maps.SupportMapFragment
import com.google.android.gms.maps.model.LatLng
import com.google.android.gms.maps.model.MarkerOptions


class LocationFragment : Fragment(), OnMapReadyCallback,
GoogleMap.OnMapClickListener {

    lateinit var binding: FragmentLocationBinding
    lateinit var gMap: GoogleMap
    var chosenLocation: LatLng? = null

    companion object {
        private const val LOCATION_REQ_CODE = 10001;
        private const val TAG = "MapsActivity"
    }

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        binding =
FragmentLocationBinding.inflate(inflater,container,false)
        return binding.root
    }

    override fun onViewCreated(view: View, savedInstanceState:
Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        val mapFragment =
childFragmentManager.findFragmentById(R.id.map) as
SupportMapFragment?
        mapFragment?.getMapAsync(this)

        binding.searchView.setOnQueryTextListener(object :
SearchView.OnQueryTextListener{
            override fun onQueryTextSubmit(query: String?):
Boolean {
                if (query == null)
                    return false
                moveToSearchedLocation(query)
                return false
            }

            override fun onQueryTextChange(newText: String?):
Boolean = false
```

```
    })

    binding.confirmloc.setOnClickListener {
        var address: Address? = null
        if (chosenLocation == null)
            return@setOnClickListener
        try {
            val geocoder = Geocoder(requireContext())
            val addresses = geocoder.getFromLocation(
                chosenLocation!!.latitude,
                chosenLocation!!.longitude,
                1
            )
            if(addresses.isNotEmpty()){
                address = addresses[0]
            }
        } catch (e: Exception){
            e.printStackTrace()
        }
        findNavController().navigate(action)
    }

}
```

```
    override fun onMapReady(googleMap: GoogleMap) {
        gMap = googleMap
        gMap.setOnMapClickListener(this)
        if (ActivityCompat.checkSelfPermission(
                requireContext(),
                Manifest.permission.ACCESS_FINE_LOCATION
            ) == PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(
                requireContext(),
                Manifest.permission.ACCESS_COARSE_LOCATION
            ) == PackageManager.PERMISSION_GRANTED
        ) {
            googleMap.isMyLocationEnabled = true
        }

    }

    override fun onMapClick(latLng: LatLng) {
        Log.d(TAG, "onMapClick: $latLng")
        gMap.clear()

gMap.addMarker(MarkerOptions().position(latLng).title("Your
postion"))

gMap.animateCamera(CameraUpdateFactory.newLatLngZoom(l
atLng,16f))
```

```
        chosenLocation = latLng
    }

    override fun onStart() {
        super.onStart()
        when {
PermissionUtils.isAccessFineLocationGranted(requireContext())
                    &&
PermissionUtils.isAccessCoarseLocationGranted(requireContext(
))
            -> {
                when {
                    PermissionUtils.isLocationEnabled(requireContext())
-> {
                        getLastLocation()
                    }
                    else -> {

PermissionUtils.showGPSNotEnabledDialog(requireContext())
                    }
                }
            }
            else -> {
                PermissionUtils.requestAccessFineLocationPermission(
                    requireActivity(),
                    LOCATION_REQ_CODE
                )

PermissionUtils.requestAccessCoarseLocationPermission(
                    requireActivity(),
                    LOCATION_REQ_CODE
                )
            }
        }
    }

    override fun onResume() {
        super.onResume()
        when {

PermissionUtils.isAccessFineLocationGranted(requireContext())
                    && 
PermissionUtils.isAccessCoarseLocationGranted(requireContext(
))
            -> {
                when {
                    PermissionUtils.isLocationEnabled(requireContext()) 
-> {
                        getLastLocation()
                    }
                    else -> {
```

```
PermissionUtils.showGPSNotEnabledDialog(requireContext())
                    }
                }
            }
            else -> {
                PermissionUtils.requestAccessFineLocationPermission(
                    requireActivity(),
                    LOCATION_REQ_CODE
                )

PermissionUtils.requestAccessCoarseLocationPermission(
                    requireActivity(),
                    LOCATION_REQ_CODE
                )
            }
        }
    }



    private fun getLastLocation(){
        try {
            val fusedLocationProviderClient =

LocationServices.getFusedLocationProviderClient(requireContex
t())
```

```
            if (ActivityCompat.checkSelfPermission(
                    requireContext(),
                    Manifest.permission.ACCESS_FINE_LOCATION
                ) == PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(
                    requireContext(),

Manifest.permission.ACCESS_COARSE_LOCATION
                ) == PackageManager.PERMISSION_GRANTED
            ) {

fusedLocationProviderClient.lastLocation.addOnSuccessListener
{ location ->
                        if(location == null)
                            return@addOnSuccessListener
                        val loc = LatLng(location.latitude,
location.longitude)

gMap.animateCamera(CameraUpdateFactory.newLatLngZoom(l
oc, 15f))
                    }
            }
        } catch (e: Exception){


        }
```

```
    }

    private fun moveToSearchedLocation(location: String) {
        gMap.clear()
        val geocoder = Geocoder(requireContext())
        try {
            val addresses =
geocoder.getFromLocationName(location,1)
            if(addresses.isNotEmpty()){
                val address = addresses[0]
                val position =
LatLng(address.latitude,address.longitude)

gMap.addMarker(MarkerOptions().position(position).title(addre
ss.featureName))

gMap.animateCamera(CameraUpdateFactory.newLatLngZoom(
position,16f))
                chosenLocation = position
            }
        } catch (e: Exception){
            e.printStackTrace()
        }
    }



}
```