// Single View App

mainStoryboard → tableViewController → check is initial view controller → Identifier → PlacesTableViewController

// PlacesTableViewController.swift (initial View)

```
import Foundation
import UIKit

class PlacesTableViewController: UITableViewController {
    private let places = ["Coffee", "Bars", "Fast Food", "Banks", "Hospitals"]

    override func viewDidLoad(){
        super.viewDidLoad()
    }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int{
        return self.places.count
    }

    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell{
        let cell = tableView.dequeueReusableCell(withIdentifier: "Cell", for: indexPath)
        cell.textLabel?.text = self.places[indexPath.row]
        return cell
    }

    override func prepare(for segue: UIStoryboardSegue, sender: Any?){
        let indexPath = (self.tableView.indexPathForSelectedRow)!
        let place = self.places[indexPath.row]
        let vc = segue.destination as! ViewController
        vc.place = place
    }
```

↳ change the identifier to cell in storyboard

↳ cell → Accessory → Disclosure Indicator

// ViewController.swift

```
import UIKit
import CoreLocation
import MapKit
import ARCL
class ViewController: UIViewController, CLLocationManagerDelegate{
    var place: String!

    var sceneLocationView = SceneLocationView() ← part of ARCL

    lazy private var locationManager = CLLocationManager()
    override func viewDidLoad{
        super.viewDidLoad()

        sceneLocationView.run()
        self.view.addSubView(sceneLocationView)
```

// MLLocalSearchRequest

// ARKit - CoreLocation

// CocoaPods app

File → new Podfile from Xcode project

↳ select the .xcodeproj file

↳ close the xcode software

↳ # Pods for CoreLocation ARKit

pod 'ARCL

↳ click the install button

↳ then use .xcworkspace file

only initialized when it is needed

```
        self.title = self.place
        self.locationManager.delegate = self
        self.locationManager.desiredAccuracy = kCLLocationAccuracyBest
        self.locationManager.requestAlwaysAuthorization() // must set up the p-list for both Always and while in use
        self.locationManager.startUpdatingLocation()

        findLocalPlaces()
    }

    override func viewDidLayoutSubviews(){
        super.viewDidLayoutSubviews()
        sceneLocationView.frame = self.view.bounds
    }

    private func findLocalPlaces(){
        guard let location = self.locationManager.location else{
            return
        }
        let request = MKLocalSearchRequest()
        request.naturalLanguageQuery = place

        var region = MKCoordinateRegion()
        region.center = CLLocationCoordinate2D(latitude: location.coordinate.latitude,
                                               longitude: location.coordinate.longitude)
        request.region = region
        let search = MKLocalSearch(request: request)
        search.start{ response, error in
            if error != nil{
                return
            }
            guard let response = response else{
                return
            }
            for item in response.mapItems{
                let placeLocation = (item.placemark.location)!
                //let image = UIImage(named: "pin")!
                let placeAnnotationNode = PlaceAnnotation(location: placeLocation, title: item.placemark.name!)
                                                                  // from the ARCL
                //let annotationNode = LocationAnnotationNode(location: placeLocation, image: image)
                //annotationNode.scaleRelativeToDistance = false
                DispatchQueue.main.async{                                  // placeAnnotationNode
                    self.sceneLocationView.addLocationNodeWithConfirmedLocation(locationNode: annotationNode)
                }
            }
        }
    }
```

```
// PlaceAnnotation.swift
import Foundation
import ARCL
import CoreLocation
import SceneKit

class PlaceAnnotation : LocationNode{
    var title : String
    var annotationNode : SCNNode

    init(location : CLLocation?, title:String){
        self.annotationNode = SCNNode()
        self.title = title
        super.init(location : location)
        initialiseUI()
    }
    private func center(node: SCNNode){
        let (min,max) = node.boundingBox
        let dx = min.x + 0.5 * (max.x - min.x)
        let dy = min.y + 0.5 * (max.y - min.y)
        let dz = min.z + 0.5 * (max.z - min.z)
        node.pivot = SCNMatrix4MakeTranslation(dx, dy, dz)
    }

    private func initialiseUI(){
        let plane = SCNPlane(width:5, height:3)
        plane.cornerRadius = 0.2
        plane.firstMaterial?.diffuse.contents = UIColor(fromHexString:"3498db")
        let text = SCNText(string: self.title, extrusionDepth:0)
        text.containerFrame = CGRect(x:0, y:0, width:5, height:3)
        text.isWrapped = true
        text.font = UIFont(name:"Futura", size:1.0)
        text.alignmentMode = kCAAlignmentCenter
        text.truncationMode = kCATruncationMiddle
        let textNode = SCNNode(geometry:text)
        textNode.position = SCNVector3(0,0,0.2)
        center(node:textNode) // user Defined function
        let planeNode = SCNNode(geometry:plane)
        planeNode.addChildNode(textNode)

        self.annotationNode.scale = SCNVector(3,3,3)
        self.annotationNode.addChildNode(planeNode)

        let billboardConstraint = SCNBillboardConstraint()
        billboardConstraint.freeAxes = SCNBillboardAxis.Y
        constrains = [billboardConstraint]

        self.addChildNode(self.annotationNode)
    }
```

```
required init?(coder aDecoder: NSCoder) {
    fatalError("init(coder:) has not been implemented")
}
```