```
import UIKit
import ArcGIS

class SQViewController: UIViewController, UITableViewDataSource,
UITableViewDelegate  {

    @IBOutlet weak var spinner: UIActivityIndicatorView!
    @IBOutlet weak var mapView: AGSMapView!
    @IBOutlet weak var tableView: UITableView!
    @IBOutlet var searchBar: UISearchBar!

    var map: AGSMap!

    private var MunicipalityFeatureTable: AGSServiceFeatureTable?


    var queriedMunicipalityFeatures = [AGSFeature]()

    var municipalityArray = [[String: String]]()
    var municipalitySearchedArray = [[String: String]]()

    var selectedFeature: AGSFeature!
    var selectedBoundary = ""

    var munName = ""
    var munCode = ""

    var searching = false

    override func viewDidLoad() {
        super.viewDidLoad()

        spinner.startAnimating()

        self.map = AGSMap(basemap: AGSBasemap(baseLayer:
AGSArcGISMapImageLayer(url: .Coast_url)))
        self.mapView.map = self.map
        self.SetMapExtent()

        self.mapView.graphicsOverlays.addObjects(from:
[graphicsOverlay])

 self.Load_Municipality_Boundary()


      }

    func SetMapExtent () {

        let envelope = AGSEnvelope (xMin: 155000.601394999, yMin:
297009.111383, xMax: 246023.662305001, yMax: 495960.810646,
spatialReference: self.mapView.spatialReference)
        self.map.initialViewpoint = AGSViewpoint(targetExtent: envelope)
        self.mapView.map = self.map

        spinner.stopAnimating()
```

```
func Load_Municipality_Boundary() {

        self.MunicipalityFeatureTable = AGSServiceFeatureTable(url:
URL(string: "https://services.gisqatar.org.qa/server/rest/services/
Vector/MunicipalityE/MapServer/0")!)
        self.queryMunicipality ()
        self.selectedBoundary = "MUNICIPALITY"

    }

//MARK: - UITable Delegates

    func tableView(_ tableView: UITableView, numberOfRowsInSection
section: Int) -> Int {

        var boundaryType = 0

            if searching {
                boundaryType = municipalitySearchedArray.count
            } else {
                boundaryType = municipalityArray.count
            }
        }

        return boundaryType

    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath:
IndexPath) -> UITableViewCell {

        let cell = tableView.dequeueReusableCell(withIdentifier: "Cell",
for: indexPath)
        cell.textLabel?.font = UIFont.systemFont(ofSize: 13.0)
        cell.textLabel?.textColor = UIColor.systemBlue

        let searchText = self.searchBar.text
        var resultText: String!
        var displayFieldValue: String!

       if searching{
                let feature =
self.municipalitySearchedArray[indexPath.row]
                resultText = feature["ENAME"]
                cell.textLabel?.attributedText =
boldSearchResult(searchText!, resultString: resultText as String)

            } else {

                let feature = self.municipalityArray[indexPath.row]
                displayFieldValue = feature["ENAME"]
                cell.textLabel?.text = displayFieldValue
            }
        }
```

```
        return cell
    }

    //called the function to make search text Bold in tableview
    func boldSearchResult(_ searchString: String, resultString: String)
-> NSMutableAttributedString {

        let attributedString: NSMutableAttributedString =
NSMutableAttributedString(string: resultString)
        let pattern = searchString.lowercased()
        let range: NSRange = NSMakeRange(0, resultString.count)

        if pattern.count > 0 {
            let regex = try! NSRegularExpression(pattern: pattern,
options:.caseInsensitive)
            regex.enumerateMatches(in: resultString.lowercased(),
options: NSRegularExpression.MatchingOptions(), range: range)
{ (textCheckingResult, matchingFlags, stop) -> Void in
                let subRange = textCheckingResult?.range

attributedString.addAttribute(NSAttributedString.Key.foregroundColor,
value: UIColor.red, range: subRange!)
            }
        }
        return attributedString
    }


private func queryMunicipality() {

    let queryParams = AGSQueryParameters()
    queryParams.whereClause = "1=1"

    self.MunicipalityFeatureTable?.populateFromService(with:
queryParams, clearCache: true, outFields: ["*"]) { [weak self]
(queryResult: AGSFeatureQueryResult?, error: Error?) in

        guard let self = self else { return }

        if let error = error {
            //display error
            self.presentAlert(error: error)

        } else if let features =
queryResult?.featureEnumerator().allObjects {

            if !features.isEmpty {

                self.queriedMunicipalityFeatures = features

                let feature = self.queriedMunicipalityFeatures.first
                let att = feature?.attributes.value(forKey: "ENAME")
                self.selectedFeature = feature
                print("Attr: \(String(describing: att))")

                for feature in features {
```

```
                    let featureDict = ["CODE" :
feature.attributes.value(forKey: "CODE") as! String,
                                       "ENAME" :
feature.attributes.value(forKey: "ENAME") as! String,
                                       "ANAME" :
feature.attributes.value(forKey: "ANAME") as! String,
                                       ]
                        self.municipalityArray.append(featureDict)

                }
                DispatchQueue.main.async {
                  self.tableView.reloadData()
                }

                self.spinner.stopAnimating()

            } else {  //else notify user
                self.presentAlert(message: "NO_RECORD_FOUND")
            }

        }
    }

   }
```