**//CoreData**

//array of type CoreData Entity

var sportsList = [Sports]()

//needed objects

let context = (UIApplication.shared.delegate as! AppDelegate).persistentContainer.viewContext

let save = (UIApplication.shared.delegate as! AppDelegate).saveContext

//fetching data

func fetchingSportsData() {

let result: NSFetchRequest<Sports> = Sports.fetchRequest()

do {

sportsList = try context.fetch(result)

}catch {

print(error)

}

tableView.reloadData()

}

//add new sport

@IBAction func addNewSport(\_ sender: UIBarButtonItem) {

let alert = UIAlertController(title: "Add New Sport", message: nil, preferredStyle: .alert)

alert.addTextField { textField in

textField.placeholder = "Enter Sport Name"

}

alert.addAction(UIAlertAction(title: "Save", style: .default, handler: { \_ in

let name = alert.textFields![0].text!

if name.isEmpty {

return

}

let newSport = Sports(context: self.context)

newSport.name = name

self.save()

self.fetchingSportsData()

}))

alert.addAction(UIAlertAction(title: "Cancel", style: .cancel, handler: .none))

present(alert, animated: true)

}

//delete

override func tableView(\_ tableView: UITableView, commit editingStyle: UITableViewCell.EditingStyle, forRowAt indexPath: IndexPath) {

context.delete(sportsList[indexPath.row])

save()

fetchingSportsData()

}

**//image picker**

// variable

var imageView = UIImageView()

//Picker calling variable

let picker = UIImagePickerController()

picker.delegate = self

picker.sourceType = .photoLibrary

present(picker, animated: true)

//extension

extension ……: UIImagePickerControllerDelegate, UINavigationControllerDelegate {

func imagePickerController(\_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info: [UIImagePickerController.InfoKey : Any]) {

guard let userPickedImage = info[.originalImage] as? UIImage else { return }

imageView.image = userPickedImage

picker.dismiss(animated: true, completion: nil)

}

}

**//Delete from API**

//Delete functionality

static func deleteTaskWithObjective(id: String, completionHandler: @escaping(\_ data: Data?, \_ response: URLResponse?, \_ error: Error?) -> Void) {

// Create the url to request

if let urlToReq = URL(string: "http://127.0.0.1:8080/tasks/\(id)") {

// Create an NSMutableURLRequest using the url. This Mutable Request will allow us to modify the headers.

var request = URLRequest(url: urlToReq)

// Set the method to DELETE

request.httpMethod = "DELETE"

// Create the session

let session = URLSession.shared

let task = session.dataTask(with: request as URLRequest, completionHandler: completionHandler)

task.resume()

}

}

//call deleting function using tableView

override func tableView(\_ tableView: UITableView, commit editingStyle: UITableViewCell.EditingStyle, forRowAt indexPath: IndexPath) {

let id = list[indexPath.section].id

TaskModel.deleteTaskWithObjective(id: id) { data, response, error in

if let error = error {

print(error)

}else {

print("Deleted Successfully")

}

DispatchQueue.main.sync {

self.fetchAllItems()

}

}

}