**Core Data in SwiftUI: Real-Time Case Study Example**

**Scenario: Personal Task Manager App**

This real-time example focuses on a **Task Manager App** where users can:

1. **Add tasks** with a title, priority level, and due date.
2. **Edit or delete tasks** directly from the task list.
3. **Display tasks sorted by priority and due date.**

This app utilizes **Core Data** for persistent storage, allowing tasks to remain available even after the app restarts.

**Step 1: Create Core Data Model**

1. In Xcode, add a Core Data model (.xcdatamodeld file).
2. Add an entity called Task with attributes:
   * title: String
   * dueDate: Date
   * priority: String
   * isCompleted: Boolean

**Step 2: PersistentController Setup**

Create a Core Data manager (PersistentController) for database interactions.

swift

CopyEdit

import CoreData

class PersistentController {

static let shared = PersistentController()

let container: NSPersistentContainer

private init() {

container = NSPersistentContainer(name: "TaskManagerModel")

container.loadPersistentStores { description, error in

if let error = error {

fatalError("Failed to load Core Data stack: \(error)")

}

}

}

func saveContext() {

let context = container.viewContext

if context.hasChanges {

do {

try context.save()

} catch {

let nsError = error as NSError

fatalError("Unresolved error \(nsError), \(nsError.userInfo)")

}

}

}

}

**Step 3: Task List View**

This view displays the list of tasks and allows the user to mark tasks as complete.

swift

CopyEdit

import SwiftUI

struct TaskListView: View {

@Environment(\.managedObjectContext) private var viewContext

@FetchRequest(

sortDescriptors: [NSSortDescriptor(keyPath: \Task.dueDate, ascending: true)],

animation: .default

) private var tasks: FetchedResults<Task>

var body: some View {

NavigationView {

List {

ForEach(tasks) { task in

HStack {

VStack(alignment: .leading) {

Text(task.title ?? "Untitled")

.font(.headline)

Text("Due: \(task.dueDate ?? Date(), formatter: dateFormatter)")

.font(.subheadline)

}

Spacer()

Text(task.priority ?? "Medium")

.padding(8)

.background(task.priority == "High" ? Color.red : Color.green)

.foregroundColor(.white)

.cornerRadius(4)

}

}

.onDelete(perform: deleteTasks)

}

.navigationTitle("Tasks")

.toolbar {

ToolbarItem(placement: .navigationBarTrailing) {

NavigationLink(destination: AddTaskView()) {

Label("Add Task", systemImage: "plus")

}

}

}

}

}

private func deleteTasks(offsets: IndexSet) {

withAnimation {

offsets.map { tasks[$0] }.forEach(viewContext.delete)

do {

try viewContext.save()

} catch {

let nsError = error as NSError

fatalError("Unresolved error \(nsError), \(nsError.userInfo)")

}

}

}

private let dateFormatter: DateFormatter = {

let formatter = DateFormatter()

formatter.dateStyle = .short

formatter.timeStyle = .none

return formatter

}()

}

**Step 4: Add Task View**

A form that allows users to create a new task and save it to Core Data.

swift

CopyEdit

struct AddTaskView: View {

@Environment(\.managedObjectContext) private var viewContext

@Environment(\.dismiss) private var dismiss

@State private var title: String = ""

@State private var dueDate = Date()

@State private var priority: String = "Medium"

let priorities = ["High", "Medium", "Low"]

var body: some View {

Form {

Section(header: Text("Task Details")) {

TextField("Task Title", text: $title)

DatePicker("Due Date", selection: $dueDate, displayedComponents: .date)

Picker("Priority", selection: $priority) {

ForEach(priorities, id: \.self) { priority in

Text(priority)

}

}

}

Button("Save Task") {

addTask()

}

}

.navigationTitle("Add Task")

}

private func addTask() {

withAnimation {

let newTask = Task(context: viewContext)

newTask.title = title

newTask.dueDate = dueDate

newTask.priority = priority

newTask.isCompleted = false

do {

try viewContext.save()

dismiss()

} catch {

let nsError = error as NSError

fatalError("Unresolved error \(nsError), \(nsError.userInfo)")

}

}

}

}

**Step 5: Enable Core Data in the Main App**

Modify the app entry point to inject Core Data context.

swift

CopyEdit

@main

struct TaskManagerApp: App {

let persistentController = PersistentController.shared

var body: some Scene {

WindowGroup {

TaskListView()

.environment(\.managedObjectContext, persistentController.container.viewContext)

}

}

}

**Features Demonstrated**

* **Data Persistence:** Tasks remain available across app launches using Core Data.
* **Real-Time Data Updates:** FetchRequest automatically updates the UI when tasks are added or deleted.
* **Navigation Flows:** Smooth navigation between task list and task creation form.