Yes, I can definitely help you build the code. We can break it down into modules and tackle them one by one. Here's how we can proceed:

1. **Set Up the Project Structure**
2. **Build Individual Modules**
3. **Integrate Modules**
4. **Test and Debug**

**1. Set Up the Project Structure**

We will start by setting up a new SwiftUI project in Xcode.

* **Create Project**: Open Xcode → Create a new project → Choose "App" → Set the project name (e.g., "TempleManagementApp") → Select Swift and SwiftUI as the language and interface.

Once the project is created, we’ll organize the code into different modules like Donation Management, Inventory Management, Cash Book, Accounts, and User Role Management.

**2. Build Individual Modules**

Each module will be created step by step. Let’s start with the **Donation Management Module**.

**Donation Management Module - Step 1**

1. **Create the Data Model for Donations**:

swift

Copy code

import Foundation

import CoreData

struct Donation: Identifiable {

let id: UUID

let donorName: String

let donationType: DonationType

let donationCategory: String

let amount: Double

let date: Date

}

enum DonationType: String {

case cash = "Cash"

case product = "Product"

}

// Core Data Entity (optional)

@objc(DonationEntity)

class DonationEntity: NSManagedObject {

@NSManaged var id: UUID

@NSManaged var donorName: String

@NSManaged var donationType: String

@NSManaged var donationCategory: String

@NSManaged var amount: Double

@NSManaged var date: Date

}

1. **Create the Donation Management View**:

swift

Copy code

import SwiftUI

struct DonationManagementView: View {

@State private var donorName: String = ""

@State private var donationCategory: String = ""

@State private var donationType: DonationType = .cash

@State private var amount: String = ""

@State private var date = Date()

var body: some View {

Form {

Section(header: Text("Donor Information")) {

TextField("Donor Name", text: $donorName)

}

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

TextField("Donation Category", text: $donationCategory)

Picker("Donation Type", selection: $donationType) {

Text("Cash").tag(DonationType.cash)

Text("Product").tag(DonationType.product)

}.pickerStyle(SegmentedPickerStyle())

TextField("Amount", text: $amount)

.keyboardType(.decimalPad)

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

}

Button(action: saveDonation) {

Text("Save Donation")

}

}

.navigationBarTitle("Donation Management")

}

func saveDonation() {

// Save the donation data to Core Data or cloud storage

print("Donation saved!")

}

}

struct DonationManagementView\_Previews: PreviewProvider {

static var previews: some View {

DonationManagementView()

}

}

1. **Handle Saving Donations**: You can implement saving to Core Data or cloud storage based on your chosen backend solution.

swift

Copy code

func saveDonation() {

let newDonation = DonationEntity(context: PersistenceController.shared.container.viewContext)

newDonation.id = UUID()

newDonation.donorName = donorName

newDonation.donationCategory = donationCategory

newDonation.donationType = donationType.rawValue

newDonation.amount = Double(amount) ?? 0.0

newDonation.date = date

do {

try PersistenceController.shared.container.viewContext.save()

print("Donation saved successfully!")

} catch {

print("Failed to save donation: \(error)")

}

}

**3. Integrate Modules**

After building each module (e.g., Inventory Management, Cash Book), we'll integrate them into a unified app structure. Each module will communicate through shared models and services.

**4. Test and Debug**

Testing will include both unit testing and user interface testing. We can use XCTest for unit testing and SwiftUI previews for UI testing.

**Next Steps:**

1. Let me know which module you'd like to build next, or if you need further elaboration on the Donation Management module.
2. We can then proceed with more detailed code implementations.

Feel free to share specific requirements or issues you encounter while coding, and I’ll guide you through them.

Step 1 - Installed xCode with tvOS and iOS simulator