**Integrate with a RESTful API to fetch real-time currency exchange rates to adjust the values of transactions made in foreign currencies.**

To integrate a RESTful API for real-time currency exchange rates in a **SwiftUI personal budget tracker**, we'll use **Combine** and **URLSession** to fetch exchange rates and apply them to transactions.

**Steps to Implement**

1. **Fetch Exchange Rates** using an API (e.g., [ExchangeRate-API](https://www.exchangerate-api.com/)).
2. **Use Combine to Handle Asynchronous Calls** for fetching exchange rates.
3. **Convert Transaction Amounts** based on real-time rates.
4. **Display Updated Transactions in SwiftUI**.

**1️⃣ Fetch Exchange Rates Using Combine**

Create a CurrencyService to fetch real-time exchange rates.

swift

CopyEdit

import Foundation

import Combine

struct ExchangeRateResponse: Codable {

let rates: [String: Double]

}

class CurrencyService {

static let shared = CurrencyService()

private var cancellables = Set<AnyCancellable>()

func fetchExchangeRates(baseCurrency: String) -> AnyPublisher<[String: Double], Error> {

let urlString = "https://open.er-api.com/v6/latest/\(baseCurrency)?apikey=YOUR\_API\_KEY"

guard let url = URL(string: urlString) else {

return Fail(error: URLError(.badURL)).eraseToAnyPublisher()

}

return URLSession.shared.dataTaskPublisher(for: url)

.map(\.data)

.decode(type: ExchangeRateResponse.self, decoder: JSONDecoder())

.map { $0.rates }

.receive(on: DispatchQueue.main)

.eraseToAnyPublisher()

}

}

**2️⃣ ViewModel to Manage Transactions & Currency Conversion**

Modify the TransactionViewModel to include currency conversion.

swift

CopyEdit

import Foundation

import Combine

import CoreData

class TransactionViewModel: ObservableObject {

@Published var transactions: [Transaction] = []

@Published var exchangeRates: [String: Double] = [:]

private var cancellables = Set<AnyCancellable>()

private let context: NSManagedObjectContext

init(context: NSManagedObjectContext) {

self.context = context

fetchTransactions()

fetchExchangeRates()

}

func fetchTransactions() {

let request: NSFetchRequest<Transaction> = Transaction.fetchRequest()

request.sortDescriptors = [NSSortDescriptor(keyPath: \Transaction.date, ascending: false)]

do {

transactions = try context.fetch(request)

} catch {

print("Error fetching transactions: \(error)")

}

}

func fetchExchangeRates(baseCurrency: String = "USD") {

CurrencyService.shared.fetchExchangeRates(baseCurrency: baseCurrency)

.sink(receiveCompletion: { completion in

if case .failure(let error) = completion {

print("Failed to fetch rates: \(error)")

}

}, receiveValue: { [weak self] rates in

self?.exchangeRates = rates

})

.store(in: &cancellables)

}

func convertAmount(amount: Double, from sourceCurrency: String, to targetCurrency: String) -> Double {

guard let sourceRate = exchangeRates[sourceCurrency],

let targetRate = exchangeRates[targetCurrency] else { return amount }

return (amount / sourceRate) \* targetRate

}

func addTransaction(amount: Double, currency: String, category: String, note: String?) {

let transaction = Transaction(context: context)

transaction.id = UUID()

transaction.amount = amount

transaction.currency = currency

transaction.date = Date()

transaction.category = category

transaction.note = note

saveContext()

fetchTransactions()

}

private func saveContext() {

do {

try context.save()

} catch {

print("Error saving context: \(error)")

}

}

}

**3️⃣ Display Transactions with Converted Amounts in SwiftUI**

Modify TransactionListView to show transactions in a **base currency** (e.g., USD).

swift

CopyEdit

import SwiftUI

struct TransactionListView: View {

@Environment(\.managedObjectContext) private var viewContext

@StateObject private var viewModel: TransactionViewModel

let baseCurrency = "USD"

init(context: NSManagedObjectContext) {

\_viewModel = StateObject(wrappedValue: TransactionViewModel(context: context))

}

var body: some View {

NavigationView {

List {

ForEach(viewModel.transactions, id: \.id) { transaction in

VStack(alignment: .leading) {

Text(transaction.category ?? "Unknown")

.font(.headline)

if let currency = transaction.currency {

let convertedAmount = viewModel.convertAmount(

amount: transaction.amount,

from: currency,

to: baseCurrency

)

Text("\(transaction.amount, specifier: "%.2f") \(currency)")

.font(.subheadline)

Text("≈ \(convertedAmount, specifier: "%.2f") \(baseCurrency)")

.font(.caption)

.foregroundColor(.gray)

}

Text(transaction.date ?? Date(), style: .date)

.font(.caption)

.foregroundColor(.gray)

}

}

.onDelete(perform: deleteTransaction)

}

.navigationTitle("Transactions")

.toolbar {

Button(action: {

viewModel.addTransaction(amount: 100.0, currency: "EUR", category: "Shopping", note: "Shoes")

}) {

Image(systemName: "plus")

}

}

}

}

private func deleteTransaction(at offsets: IndexSet) {

offsets.map { viewModel.transactions[$0] }.forEach(viewModel.deleteTransaction)

}

}

**🚀 Features Implemented**

✅ **Fetch real-time exchange rates** using Combine and URLSession.  
✅ **Store & display transactions with multiple currencies**.  
✅ **Automatically convert foreign currency transactions** into the base currency (USD).  
✅ **Seamlessly integrate with SwiftUI using Combine & Core Data**.