AP CSP CPT Final Submission

CSP 2025

```
// Input View
// Project creation aided by AI using Chat GPT 4.1 (for debugging & parts of code generation)
// PDF of Code created with CodePrint.org

import SwiftUI

struct Subscription: Identifiable {
    let id = UUID()
    var name: String
    var monthlyCost: Double
    var type: String
}

struct ContentView: View {
    @State private var subscriptionName: String = ""
    @State private var monthlyCost: String = ""
    @State private var subscriptionType: String = "Streaming"
    @State private var totalBudget: String = ""
    @State private var subscriptions: [Subscription] = []
    @State private var totalCost: Double = 0.0
    @State private var displayText: String = ""
    @State private var displayColor: Color = .red
    @State private var showAlert: Bool = false
    @State private var navigateToResults: Bool = false

    let subscriptionTypes = ["Streaming","Music", "Gaming", "Software" , "News", "Fitness", "Other"]

    var body: some View {
        NavigationStack {
            VStack(spacing: 10) {
                TextField("Subscription Name", text: $subscriptionName)
                    .textFieldStyle(RoundedBorderTextFieldStyle())
                    .padding()

                TextField("Monthly Cost (e.g., 9.99)", text: $monthlyCost)
                    .textFieldStyle(RoundedBorderTextFieldStyle())
                    .keyboardType(.decimalPad)
                    .padding()

                Picker("Subscription Type", selection: $subscriptionType) {
                    ForEach(subscriptionTypes, id: \.self) {
                        Text($0)
                    }
                }
                .pickerStyle(.wheel)
                .padding()

                TextField("Enter Your Monthly Budget", text: $totalBudget)
                    .textFieldStyle(RoundedBorderTextFieldStyle())
                    .keyboardType(.decimalPad)
                    .padding()

                Button(action: {
                    addSubscription()
                }) {
                    Text("Add Subscription")
                        .frame(maxWidth: .infinity)
                        .padding()
                        .foregroundColor(.white)
                        .background(Color.blue)
                        .cornerRadius(8)
                }
                .padding(.horizontal)
```

```
                Button(action: {
                    calculateTotalCost(subscriptions: subscriptions)
                    navigateToResults = true
                }) {
                    Text("See your results!")
                        .frame(maxWidth: .infinity)
                        .padding()
                        .foregroundColor(.white)
                        .background(Color.green)
                        .cornerRadius(8)
                }
                .padding(.horizontal)

                .alert(isPresented: $showAlert) {
                    Alert(title: Text("Error"), message: Text("Please enter valid subscription details."),
                          dismissButton: .default(Text("OK")))
                }
            }
            .navigationTitle("Subscription Tracker")
            .padding()
            .navigationDestination(isPresented: $navigateToResults) {
                OutputView(
                    subscriptions: subscriptions,
                    totalCost: totalCost,
                    totalBudget: totalBudget,
                    displayText: displayText,
                    displayColor: displayColor
                )
            }
        }
    }

    func addSubscription() {
        guard !subscriptionName.isEmpty, let cost = Double(monthlyCost) else {
            showAlert = true
            return
        }

        let newSubscription = Subscription(name: subscriptionName, monthlyCost: cost, type: subscriptionType)
        subscriptions.append(newSubscription)

        subscriptionName = ""
        monthlyCost = ""
        subscriptionType = "Streaming"

        organizeSubscriptions(subscriptions: &subscriptions)
    }

    func calculateTotalCost(subscriptions: [Subscription]) {
        totalCost = 0.0

        for subscription in subscriptions {
            totalCost += subscription.monthlyCost
        }

        let budget = Double(totalBudget) ?? 0

        if budget == 0 {
            displayText = "No budget set. Add a budget to track your subscriptions."
            displayColor = .gray
        } else if totalCost > budget {
            displayText = "You are over budget! Consider removing some subscriptions!"
            displayColor = .red
        } else if totalCost == budget {
            displayText = "You are exactly on budget! Great job!"
            displayColor = .green
        } else {
            displayText = "You are under budget! Great job!"
```

```
                displayColor = .green
            }
        }

        func organizeSubscriptions(subscriptions: inout [Subscription]) {
            let typeOrder = ["Streaming", "Music", "Gaming", "Software", "News", "Fitness", "Other"]

            var groupedSubscriptions: [String: [Subscription]] = [:]
            for subscription in subscriptions {
                groupedSubscriptions[subscription.type, default: []].append(subscription)
            }

            for type in groupedSubscriptions.keys {
                if var group = groupedSubscriptions[type] {
                    for currentIndex in 0..<group.count {
                        var maxIndex = currentIndex
                        for comparisonIndex in (currentIndex+1)..<group.count {
                            if group[comparisonIndex].monthlyCost > group[maxIndex].monthlyCost {
                                maxIndex = comparisonIndex
                            }
                        }
                        if maxIndex != currentIndex {
                            let temp = group[currentIndex]
                            group[currentIndex] = group[maxIndex]
                            group[maxIndex] = temp
                        }
                    }
                    groupedSubscriptions[type] = group
                }
            }
            subscriptions = []
            for type in typeOrder {
                if let group = groupedSubscriptions[type] {
                    subscriptions.append(contentsOf: group)
                }
            }
        }
    }

    -------

    // Output View

    import SwiftUI

    struct OutputView: View {
        var subscriptions: [Subscription]
        var totalCost: Double
        var totalBudget: String
        var displayText: String
        var displayColor: Color

        func getColor(for type: String) -> Color {
            switch type {
            case "Streaming":
                return Color.red.opacity(0.2)
            case "Music":
                return Color.green.opacity(0.2)
            case "Gaming":
                return Color.blue.opacity(0.2)
            case "Software":
                return Color.yellow.opacity(0.2)
            case "News":
                return Color.purple.opacity(0.2)
            case "Fitness":
                return Color.orange.opacity(0.2)
            default:
                return Color.gray.opacity(0.2)
            }
```

```
    }

    var body: some View {
        VStack(spacing: 10) {
            if !subscriptions.isEmpty {
                Text("Total Cost: $\(totalCost, specifier: "%.2f")")
                    .font(.headline)
                Text("Budget: $\(totalBudget)")
                    .font(.headline)
                    .padding()
                Text(displayText)
                    .foregroundColor(displayColor)
                    .font(.subheadline)
            }
            List(subscriptions) { subscription in
                VStack(alignment: .leading) {
                    Text(subscription.name)
                        .font(.headline)
                    Text("Cost: $\(subscription.monthlyCost, specifier: "%.2f")")
                    Text("Type: \(subscription.type)")
                        .font(.subheadline)
                        .foregroundColor(.gray)
                }
                .padding()
                .frame(maxWidth: .infinity, alignment: .leading)
                .background(getColor(for: subscription.type))
                .cornerRadius(8)
                .listRowInsets(EdgeInsets())
            }
        } .navigationTitle("Results!")
    }
}

-------

//  Subscription Tracker

import SwiftUI

@main
struct Subscription_TrackerApp: App {
    var body: some Scene {
        WindowGroup {
            ContentView()
        }
    }
}
```
