```swift
//
//  CetgoriesList.swift
//  uniCube
//
//  Created by Aleeya Ahmad on 2/9/2025.
//

import SwiftUI

struct CategoriesList: View {
    @Environment(\.dismiss) private var dismiss

    @State private var subHeading = "WEEK 7"

    // Segmented filter
    @State private var segment: Segment = .all

    // Date filter card state
    @State private var showDateCard = false
    @State private var isAllDay = false
    @State private var tempStart = Date()
    @State private var tempEnd =
Calendar.current.date(byAdding: .hour, value: 2, to: Date())!

    // Active date range applied to the list (optional)
    @State private var activeStart: Date? = nil
    @State private var activeEnd: Date? = nil

    // Sample events (mix of free & paid)
    private let allEvents: [Event] = Event.sample

    // MARK: Filtering
    private var filteredEvents: [Event] {
        var items = allEvents

        // segment filter
        switch segment {
        case .all:
            break
        case .free:
            items = items.filter { $0.isFree }
        case .paid:
            items = items.filter { !$0.isFree }
        }

        // optional date range filter (by start time)
```

```
        if let s = activeStart, let e = activeEnd {
            items = items.filter { ($0.start >= s) && ($0.start
<= e) }
        }

        return items
    }

    // HELPERS
    private static let dmFormatter: DateFormatter = {
        let df = DateFormatter()
        df.locale = Locale(identifier: "en_AU")
        df.dateFormat = "d MMMM"       // 9 September
        return df
    }()
    private static let dmyFormatter: DateFormatter = {
        let df = DateFormatter()
        df.locale = Locale(identifier: "en_AU")
        df.dateFormat = "d MMMM yyyy"  // 16 September 2025
        return df
    }()

    private func makeRangeText(from s: Date, to e: Date) ->
String {
        let cal = Calendar.current
        if cal.component(.year, from: s) ==
cal.component(.year, from: e) {
            let left  = Self.dmFormatter.string(from:
s).uppercased()
            let right = Self.dmFormatter.string(from:
e).uppercased()
            let year  = cal.component(.year, from: e)
            return "\(left) – \(right) \(year)"
        } else {
            let left  = Self.dmyFormatter.string(from:
s).uppercased()
            let right = Self.dmyFormatter.string(from:
e).uppercased()
            return "\(left) – \(right)"
        }
    }



    var body: some View {
        NavigationStack {
```

```
            ZStack {
                Color.black.ignoresSafeArea()

                VStack(alignment: .leading, spacing: 16) {

                    // MARK: Top controls
                    HStack(spacing: 12) {
                        Picker("Filter", selection: $segment) {
                            Text("All").tag(Segment.all)
                            Text("Free").tag(Segment.free)
                            Text("$").tag(Segment.paid)
                        }
                        .pickerStyle(.segmented)
                        .tint(.white)
                        .padding(6)
                        .background(
                            RoundedRectangle(cornerRadius: 14,
style: .continuous)
                                .fill(Color.white.opacity(0.10)
)
                        )

                        Button {
                            withAnimation(.snappy) {
                                // seed temp pickers with
current active range if any
                                if let s = activeStart
{ tempStart = s }
                                if let e = activeEnd
{ tempEnd = e }
                                showDateCard = true
                            }
                        } label: {
                            Text("Choose Date")
                                .font(.body.weight(.semibold))
                                .padding(.vertical, 8)
                                .padding(.horizontal, 14)
                                .frame(minWidth: 120)
                                .foregroundStyle(.white)
                                .background(

RoundedRectangle(cornerRadius: 16, style: .continuous)
                                        .fill(Color.brandPink.o
pacity(0.70)) // #FF268A @ 70%
                                )
                        }
```

```
                        .accessibilityLabel("Choose Date")
                    }
                    .padding(.horizontal)
                    .padding(.top, 8)

                    // Section title
                    Text("WELLBEING")
                        .font(.system(size: 32, weight: .bold))
                        .foregroundColor(.brandPink)
                        .padding(.horizontal)
                        .padding(.bottom, -10)

                    Text("\(subHeading)")
                        .font(.system(size: 17, weight: .bold))
                        .foregroundColor(.brandPink)
                        .padding(.horizontal)


                    // List
                    ScrollView {
                        LazyVStack(spacing: 24) {
                            ForEach(filteredEvents) { event in
                                EventCard1(event: event)
                                    .padding(.horizontal)
                            }

                            // empty state
                            if filteredEvents.isEmpty {
                                Text("No events found.")
                                    .foregroundStyle(.white.opa
city(0.7))
                                    .padding(.top, 32)
                            }
                        }
                        .padding(.bottom, 24)
                    }
                }

                // MARK: Date filter card â no opaque dimmer
(clear overlay for tap-to-dismiss)
                if showDateCard {
                    // Invisible, tappable backdrop to dismiss
                    Rectangle()
                        .fill(.clear)
                        .contentShape(Rectangle())
                        .ignoresSafeArea()
```

```
                        .onTapGesture {
                            withAnimation(.snappy)
{ showDateCard = false }
                        }

                    DateFilterCard(
                        isPresented: $showDateCard,
                        isAllDay: $isAllDay,
                        startDate: $tempStart,
                        endDate: $tempEnd,

                    ) { start, end in
                        // Apply the chosen range to the list
filter
                        activeStart = start
                        activeEnd = end

                        // update the banner text
                        if start == .distantPast || end
== .distantFuture {
                            subHeading = "WEEK 7"
                        } else {
                            let s = min(start, end)
                            let e = max(start, end)
                            subHeading = makeRangeText(from: s,
to: e)
                        }
                    }
                    .transition(.move(edge: .bottom).combined(w
ith: .opacity))
                    .zIndex(1)
                }
            }
            .navigationBarTitleDisplayMode(.inline)
            .toolbar {
                ToolbarItem(placement: .navigationBarLeading) {
                    Button { dismiss() } label: {
                        HStack(spacing: 6) {
                            Image(systemName: "chevron.left")
                                .font(.system(size: 16,
weight: .medium))
                            Text("Home")
                                .font(.system(size: 17,
weight: .medium))
                        }
                        .foregroundColor(.white)
```

```
                    }
                }
            }
            .tint(.white)
        }
        .navigationBarBackButtonHidden(true)
        .preferredColorScheme(.dark)
    }
}

// MARK: - Sliding Card

private struct DateFilterCard: View {
    @Binding var isPresented: Bool
    @Binding var isAllDay: Bool
    @Binding var startDate: Date
    @Binding var endDate: Date
    //@Binding var subHeading: String

    var onApply: (_ start: Date, _ end: Date) -> Void

    var body: some View {
        VStack(spacing: 0) {
            Capsule()
                .fill(Color.secondary.opacity(0.5))
                .frame(width: 44, height: 5)
                .padding(.top, 8)
                .padding(.bottom, 4)

            VStack(spacing: 0) {
                // All-day
                HStack {
                    Text("All-day")
                        .foregroundStyle(.primary)
                    Spacer()
                    Toggle("", isOn: $isAllDay)
                        .labelsHidden()
                        .tint(.brandPink)
                }
                .padding(.horizontal, 16)
                .padding(.vertical, 14)

                Divider()

                // Starts
                LabeledDateRow(label: "Starts", date:
```

```
$startDate, disabledTime: isAllDay)

                Divider().overlay(Color.white.opacity(0.25))

                // Ends
                LabeledDateRow(label: "Ends", date: $endDate,
disabledTime: isAllDay)

                // Actions
                HStack(spacing: 12) {
                    Button("Clear") {
                        onApply(Date.distantPast,
Date.distantFuture) // clears filter
                        withAnimation(.snappy) { isPresented =
false }
                    }
                    .buttonStyle(.bordered)
                    .tint(.white.opacity(0.6))

                    Button("Apply") {
                        // keep ordering sane
                        var apply_start = min(startDate,
endDate)
                        var apply_end = max(startDate, endDate)

                        onApply(apply_start, apply_end)
                        //assign variable to subHeading

                        withAnimation(.snappy) { isPresented =
false }
                    }
                    .buttonStyle(.borderedProminent)
                    .tint(.brandPink)
                }
                .padding(16)
            }
            .foregroundStyle(.white)
            .background(
                // Card style without heavy opacity to respect
accessibility contrast
                RoundedRectangle(cornerRadius: 22,
style: .continuous)
                    .fill(Color(uiColor: .systemBackground))
                    .overlay(
                        RoundedRectangle(cornerRadius: 22,
style: .continuous)
```

```
                        .stroke(Color.secondary.opacity(0.15),
lineWidth: 1)
                )
            )
        }
        .padding(.horizontal, 12)
        .padding(.bottom, 8)
        .frame(maxWidth: .infinity, alignment: .bottom)
        .ignoresSafeArea(edges: .bottom)
        .accessibilityAddTraits(.isModal)
    }
}

private struct LabeledDateRow: View {
    let label: String
    @Binding var date: Date
    var disabledTime: Bool

    var body: some View {
        HStack(spacing: 12) {
            Text(label)
                .frame(width: 70, alignment: .leading)

            Spacer(minLength: 0)

            DatePicker("", selection: $date,
displayedComponents: .date)
                .labelsHidden()
                .datePickerStyle(.compact)
                .padding(10)
                .background(RoundedRectangle(cornerRadius:
12).fill(Color(uiColor: .secondarySystemBackground)))

            DatePicker("", selection: $date,
displayedComponents: .hourAndMinute)
                .labelsHidden()
                .datePickerStyle(.compact)
                .padding(10)
                .background(RoundedRectangle(cornerRadius:
12).fill(Color.white.opacity(0.12)))
                .disabled(disabledTime)
                .opacity(disabledTime ? 0.45 : 1)
        }
        .padding(.horizontal, 16)
        .padding(.vertical, 14)
    }
```

```
}

// MARK: - Event model & card

private enum Segment { case all, free, paid }

private struct Event: Identifiable {
    let id = UUID()
    let isFree: Bool
    let title: String
    let start: Date
    let end: Date
    let location: String
    let imageName: String?

    // Display text using Swift's FormatStyle (Apple docs)
    var dayLine: String {

start.formatted(.dateTime.weekday(.wide).day().month(.wide).yea
r())
    }
    var timeLine: String {
        "\(start.formatted(date: .omitted, time: .shortened)) â
\(end.formatted(date: .omitted, time: .shortened))"
    }
}

private struct EventCard1: View {
    let event: Event

    var body: some View {
        VStack(alignment: .leading, spacing: 10) {
            ZStack(alignment: .topLeading) {
                Group {
                    if let name = event.imageName, !
name.isEmpty {
                        // Check if this is the therapy dogs
event and wrap with NavigationLink
                        if name == "doggy" {
                            NavigationLink(destination:
EventViewPage()) {
                                Image(name)
                                    .resizable()
                                    .scaledToFill()
                            }
                        } else {
```

```
                            Image(name)
                                .resizable()
                                .scaledToFill()
                        }
                    } else {
                        ZStack {

Rectangle().fill(Color.white.opacity(0.08))
                            Image(systemName: "photo")
                                .font(.system(size: 40))
                                .foregroundStyle(.white.opacity
(0.7))
                        }
                    }
                }
                .frame(width: 392, height: 126) // Figma size
                .clipShape(RoundedRectangle(cornerRadius: 12,
style: .continuous))
                .opacity(0.65)                    // 65% opacity
                .clipped()

                // Price/Free pill
                Text(event.isFree ? "FREE" : "$")
                    .foregroundColor(.white)
                    .font(.system(size: 14, weight: .regular))
                    .frame(width: 83, height: 30) // Figma size
                    .background(
                        RoundedRectangle(cornerRadius: 7,
style: .continuous) // radius 7
                            .fill(Color.brandPink.opacity(0.70))
            // 70% opacity
                    )
                    .padding(10) // keep for outer spacing if
you want

            }

            Text(event.title.uppercased())
                .font(.headline)
                .foregroundColor(.white)

            HStack(spacing: 8) {
                Image(systemName: "calendar")
                Text(event.dayLine)
            }
            .font(.subheadline)
```

```
            .foregroundColor(.white)

            HStack(spacing: 8) {
                Image(systemName: "mappin.and.ellipse")
                Text(event.location)
            }
            .font(.subheadline)
            .foregroundColor(.white)

            HStack(spacing: 8) {
                Image(systemName: "clock")
                Text(event.timeLine)
            }
            .font(.subheadline)
            .foregroundColor(.white)

            Rectangle()
                .fill(Color.brandPink.opacity(0.8))
                .frame(height: 1)
                .padding(.top, 8)
        }
        .accessibilityElement(children: .combine)
    }
}

// MARK: - Sample data

private extension Event {
    static var sample: [Event] {
        var cal = Calendar.current
        cal.timeZone = .current

        func make(_ y:Int, _ m:Int, _ d:Int, _ h:Int, _
min:Int) -> Date {
            cal.date(from: DateComponents(year: y, month: m,
day: d, hour: h, minute: min)) ?? .now
        }

        let y = 2025

        return [
            Event(isFree: true,
                  title: "Therapy Dogs",
                  start: make(y, 9, 9, 11, 0),
                  end:   make(y, 9, 9, 13, 0),
                  location: "RMIT Building 10, level 4",
```

```
                imageName: "doggy"),

            Event(isFree: false,
                title: "Salsa Class",
                start: make(y, 9, 9, 12, 0),
                end:   make(y, 9, 9, 13, 0),
                location: "RMIT Active Hub, Building 8, Level
3",
                imageName: "salsa"),

            Event(isFree: true,
                title: "Meditation Session",
                start: make(y, 9, 11, 12, 0),
                end:   make(y, 9, 11, 13, 0),
                location: "RMIT Building 47, Level 1, Room
8",
                imageName: "meditation"),

            Event(isFree: true,
                title: "Walk the Labyrinth - Quiet Your
Mind",
                start: make(y, 9, 12, 11, 0),
                end:   make(y, 9, 12, 13, 0),
                location: "RMIT Building 47, Level 3",
                imageName: "lab"),

            Event(isFree: true,
                title: "Managing Stress & Anxiety 101",
                start: make(y, 9, 13, 12, 0),
                end:   make(y, 9, 13, 13, 0),
                location: "Online via MS Teams",
                imageName: "Stress"),

            Event(isFree: true,
                title: "Mindfulness with LEGOÂ®",
                start: make(y, 9, 13, 12, 0),
                end:   make(y, 9, 13, 14, 0),
                location: "RMIT Building 12, Level 4, Rooms
115-116",
                imageName: "lego"),

            Event(isFree: false,
                title: "10 Minute Massage",
                start: make(y, 9, 13, 12, 0),
                end:   make(y, 9, 13, 14, 0),
                location: "RMIT Building 12, Level 4, Room
```

```
108",
                    imageName: "massage")
        ]

    }
}

// MARK: - Brand color

private extension Color {
    /// #FF268A
    static let brandPink = Color(red: 1.0, green: 38/255.0,
blue: 138/255.0)
}




#Preview {
    CategoriesList()
        .preferredColorScheme(.dark)
}
```