GeofenceReminders

GeofenceReminders is an iOS app that allows users to set location-based reminders using geofencing. Users can select locations on a map, define a geofence radius, add notes, and receive notifications when entering or exiting the geofenced area. The app features a tabbed interface with a map view for location selection and a list view for managing reminders, built with SwiftUI, CoreLocation, MapKit, Core Data, and UserNotifications.

Features

- **Map-Based Location Selection**: Browse and select points of interest (POIs) on a map, fetched from the Overpass API.
- **Geofence Monitoring**: Set geofences with customizable radii and receive notifications on entry/exit.
- **Reminder Management**: View, edit, and delete reminders in a list view, with persistent storage using Core Data.
- **Local Notifications**: Alerts for geofence events and test notifications for debugging.
- **Offline Support**: Displays an offline mode indicator when network requests fail.
- **Debugging Tools**: In-app logs for authorization status, geofence events, and notifications, with buttons to request permissions, test notifications, and clear geofences.

Approach

I break the application into smaller parts for simplicity. First I created a simple app in which I gave a hardcoded location , wrote functions to check whether they work fine if the user enters or exits the saved radius and coordinates. I added a simple button to test the notifications in a simpler manner. After testing notifications I tried to integrate the functions with my notifications such that notifications trigger when the user enters or exits the saved location. After doing that I then moved towards developing the application as a separate project which uses **CoreLocation , MapKit** and **CoreData** along with views that are required by our required application. I then tried to integrate my previous notifications triggering code and location manager functions into the main application.

The app is designed to provide a seamless user experience for location-based reminders, leveraging Apple's native frameworks for robust functionality:

- **Architecture**:
  - **SwiftUI**: Used for the UI, with a tabbed interface (`ContentView`) containing `MapView` (for location selection) and `ReminderListView` (for reminder management).
  - **MVVM Pattern**: View models (`MapViewModel`, `ReminderListViewModel`) manage business logic, separating UI from data handling.
  - **Core Data**: Persists reminders (`GeofenceReminder` entities) for reliable storage and retrieval.
  - **CoreLocation**: Handles geofence monitoring and location authorization, with robust logging for debugging.
  - **MapKit**: Displays an interactive map for selecting POIs and visualizing geofences.
  - **UserNotifications**: Schedules local notifications for geofence entry/exit events and provides a test notification feature.
  - **Network Service**: Fetches POIs from the Overpass API using Combine for asynchronous data handling.
- **Key Components**:
  - **`MapViewModel`**: Manages location fetching, geofence monitoring, and notifications. Integrates authorization handling, logging, and test features inspired by a simplified test app (`LocationViewModel`).
  - **`ReminderListViewModel`**: Handles Core Data operations for reminders, ensuring real-time updates across views.
  - **`ContentView`**: Coordinates the tabbed UI, sharing view models for consistent state management.
  - **Debugging Features**: Added buttons to request location authorization, test notifications, and clear geofences, with a log display in both tabs for real-time feedback.
- **Integration Strategy**:
  - Enhanced `MapViewModel` to include robust geofence monitoring and logging from a test app, ensuring compatibility with existing map and reminder features.
  - Added UI controls (buttons and logs) to `MapView` and `ReminderListView` for debugging without disrupting core functionality.

    - Ensured the `Info.plist` supports location permissions and background modes for geofencing.

Third-Party Libraries

No third-party libraries are used. The app relies entirely on Apple’s native frameworks:

- **SwiftUI**: For the user interface.
- **CoreLocation**: For geofencing and location services.
- **MapKit**: For map display and interaction.
- **Core Data**: For persistent storage of reminders.
- **UserNotifications**: For local notifications.
- **Combine**: For handling asynchronous network requests.

This approach minimizes dependencies, ensuring compatibility and reducing maintenance overhead.

Setup Steps

Follow these steps to set up and run the GeofenceReminders app on macOS using Xcode.

Prerequisites

- **macOS**: Ventura 13.0 or later.
- **Xcode**: Version 16 or later.
- **iOS Device or Simulator**: iOS 18 or later recommended.
- **Apple Developer Account**: Required for signing the app (optional for simulator testing).

1. **Verify Info.plist**:

Ensure `Info.plist` includes:

```
<key>NSLocationAlwaysAndWhenInUseUsageDescription</key>
<string>We need your location to monitor geofences in the background.</string>
<key>NSLocationWhenInUseUsageDescription</key>
<string>We need your location to set up geofences.</string>
<key>UIBackgroundModes</key>
<array>
   <string>location</string>
</array>
```

        - These keys are already configured in the provided `Info.plist`.
2. **Build and Run**:

    - Select a simulator (e.g., iPhone 16 Pro, iOS 18) or connect an iOS device.
    - Build and run the app (Cmd+R).
    - Grant location ("Allow Always") and notification permissions when prompted.
3. **Test Geofencing in Simulator**:

In the Simulator, go to Features > Location > Custom Location to simulate geofence events by entering longitude and latitude coordinates. Follow these steps to test the app's geofence functionality, ensuring logs and notifications are triggered correctly. If the user enters coordinates inside or outside the geofence, the app will respond accordingly

Prepare the App:

- Run the GeofenceReminders app in the simulator (e.g., iPhone 16 Pro, iOS 18) using Xcode (Cmd+R).
- Grant location permissions by selecting "Allow Always" when prompted.
- Grant notification permissions by selecting "Allow".
- Create a test geofence:
    - In the MapView, tap a pin (e.g., Central Park at (40.785091, -73.968285)).
    - In the ReminderDetailView, set a radius (e.g., 100m), add a note, and tap "Save".
    - Verify logs in the MapView or ReminderListView show:

Started monitoring geofence: ID=[id], Name=[name], Center=(40.785091, -73.968285), Radius=100m

Notification scheduled for [id]: [name]

Test with Custom Location:

- In the Simulator, go to **Features > Location > Custom Location**.
- **Enter Coordinates Inside the Geofence**:
    - Latitude: 40.785091
    - Longitude: -73.968285
    - Click OK.
    - **Expected Outcome**:
        - The app logs: Entered geofence: [id] in both MapView and ReminderListView.
        - A notification appears with title "Geofence Alert" and body "You have entered/exited [name]".
        - If no log or notification appears, wait 5–10 seconds or proceed to step 3 (GPX file).
- **Enter Coordinates Outside the Geofence**:

    - Calculate a point outside the 100m radius (100m ≈ 0.0009 degrees):
        - Latitude: 40.785991 (40.785091 + 0.0009)
        - Longitude: -73.969185 (-73.968285 - 0.0009)
    - In Features > Location > Custom Location, enter these coordinates and click OK.
    - **Expected Outcome**:
        - The app logs: Exited geofence: [id].
        - A notification appears for the exit event.
- **Toggle Between Coordinates**:
    - Switch back to the inside coordinates (40.785091, -73.968285) and then to the outside coordinates (40.785991, -73.969185) multiple times, waiting 5–10 seconds between changes.
    - **Expected Outcome**:
        - Logs and notifications for Entered geofence and Exited geofence each time.
- **Handle Invalid Coordinates**:
    - If the user enters invalid coordinates (e.g., latitude outside -90 to 90, longitude outside -180 to 180), the simulator ignores them, and no geofence events are triggered.
    - If the coordinates are valid but far from the geofence (e.g., (0, 0)), no events are logged, as they are outside the monitored region.

4. Usage

- **Map View**:
    - Browse POIs fetched from the Overpass API.
    - Tap a pin to set a reminder, specifying radius and note.
    - Use buttons to request authorization, test notifications, or clear geofences.
    - View logs for authorization and geofence events.
- **Reminder List View**:
    - View all reminders with name, note, and radius.
    - Swipe to delete reminders.
    - Use buttons to trigger test notifications or clear geofences.
    - Monitor logs for debugging.
- **Debugging**:
    - Check logs in both tabs for authorization status, geofence monitoring, and notification scheduling.
    - Use the "Test Notification" button to verify notification functionality.

Trade-offs

- **Simulator Limitations**:
    - Geofence monitoring in the Xcode simulator is less reliable than on a physical device. The GPX file mitigates this, but physical device testing is recommended for accurate results.
    - Trade-off: Simplified testing with GPX files sacrifices real-world accuracy but enables development without hardware.
- **Network Dependency**:
    - POIs are fetched from the Overpass API, requiring an internet connection. Offline mode displays a warning but limits functionality.
    - Trade-off: Real-time POI data enhances user experience but introduces a network dependency. Local caching could be added in future iterations.
- **Single Geofence per Location**:
    - Each reminder corresponds to one geofence. Multiple geofences per location are not supported to simplify the UI and Core Data model.
    - Trade-off: Simpler implementation reduces complexity but limits advanced use cases.
- **Log Display**:
    - Logs are shown in both tabs for debugging, which may clutter the UI in production.
    - Trade-off: Enhanced debugging capabilities improve development but may require UI refinements for end users.

Assumptions

- **iOS 18 Compatibility**: The app is developed for iOS 18, leveraging SwiftUI and MapKit features. Older iOS versions may require additional testing.
- **User Location**: Default coordinates `(40.785091, -73.968285)` (Central Park, NYC) are used for testing. Users can set custom locations via the UI or modify code for local coordinates (e.g., `(31.5204, 74.3587)` for Lahore).
- **Overpass API Availability**: The app assumes the Overpass API is accessible for POI data. No fallback data source is implemented.
- **Core Data Model**: Assumes a single `GeofenceReminder` entity is sufficient for storing reminders. Schema changes may be needed for additional features.
- **Simulator Testing**: In the Simulator, go to Features > Location > Custom Location to simulate geofence events by entering longitude and latitude coordinates.