Homework 3
Samman Tyata

Question 1: The various states that an app can enter on iOS.

1. Not Running
   - State: The app has not been launched.
   - Transition: Occurs when the app is launched from the home screen or a push notification.

2. Inactive
   - State: The app is in the foreground but not receiving events. This can happen during temporary interruptions (like an incoming phone call or SMS).
   - Transition: The app will move to this state just before entering the active state.

3. Active
   - State: The app is in the foreground and receiving events. This is the normal state of operation for the app.
   - Transition: The app enters this state from the inactive state and remains active until interrupted or sent to the background.

4. Background
   - State: The app is no longer visible to the user but continues to run in the background. The app can complete tasks or perform specific actions while in this state.
   - Transition: The app transitions to this state when it goes from active to inactive, typically when the user presses the home button or switches to another app.

5. Suspended
   - State: The app is in the background but not executing any code. It remains in memory and can be quickly returned to the active state.
   - Transition: This state occurs automatically after a period in the background, without requiring any action from the developer.

For managing app state, we can rely on:

- **@Environment(\.scenePhase)** to respond to app lifecycle events.
- **@State** for simple, view-specific state management that doesn't persist beyond the lifecycle of the view.
- **@AppStorage** and **@SceneStorage** for persistent storage that works well across lifecycle changes.
- **Task** and async functions to handle background work efficiently.
- **NotificationCenter** if you need more specific lifecycle notifications.

Question 2: The various states that you must consider for your app, why you must consider it, and what must happen in each state.

Here's a breakdown of the key states, why they are important, and what actions should occur in each state:

1. Launch State
Why: This is the initial state when the app starts. Proper initialization is necessary for a good user experience.

What Must Happen:
- Load essential resources, such as user preferences or previously saved data.
- Initialize any required services (e.g., location services, network connectivity).
- Display a splash screen while setting up the app.

2. Active State
Why: This is when the user interacts with the app.

What Must Happen:
- Allow users to browse attractions, view recommendations, and add or rate locations.
- Update the user interface dynamically based on user interactions (e.g., new attractions appearing based on their preferences).
- Fetch and display live data, such as nearby attractions and local events.

3. Inactive State
Why: This state occurs briefly when the app transitions between states, such as when receiving a phone call or an alert.

What Must Happen:
- Pause ongoing activities (like animations or data fetching) to prevent unnecessary resource usage.
- Save any temporary data or user inputs that might be lost.

4. Background State
Why: Users may switch to other apps, and handling background tasks is important.

What Must Happen:
- Save the app state, including user inputs and unsaved changes (e.g., lists of favorites or notes).
- If applicable, continue fetching data or processing requests (such as user location).

5. Suspended State
Why: This state indicates that the app is in memory but not running code.

What Must Happen:

- No specific actions are required but ensure that any unsaved state is handled before the app goes into this state.
- Consider potential data loss if the system decides to terminate the app due to memory constraints.

6. Terminated State
Why: This state occurs when the user forcefully closes the app or the system terminates it to free up resources.

What Must Happen:

- Save all essential data and user preferences to persistent storage (e.g., User Defaults, Core Data, or database).
- Ensure that the app can restore the user state on the next launch (e.g., showing the last viewed attractions or previously searched preferences).

7. Error State
Why: Network issues or other errors can occur at any time during app use.

What Must Happen:

- Display user-friendly error messages or alerts when an error occurs (e.g., network disconnection, failed data fetch).
- Provide options for users to retry the action or check their connection settings.