iOS Coding Challenge

Introduction

The Flightradar24 iOS coding challenge is designed to be executed by candidates remotely at an arranged time.

Please complete as much of the challenge as you are able to within 3 hours, but *don't sacrifice code quality (incl. tests) for speed*. Complete tasks in order and fully. We'd rather see a full solution to task 1 than incomplete solutions to more tasks. Don't worry if you don't have time to complete all tasks, we still want to review your solution.

After your tasks are completed, or your time is up, please zip your entire code folder and send your submission to your Flightradar24 contact. If you're unable to send the archive by email for whatever reason, uploading to a cloud drive and sending the link itself is also acceptable.

Good luck!

Requirements

- **Commit your progress as you would normally do. It is OK to see refactoring commits along the way.**
- **Your solution must include relevant unit tests.**

- Please submit all source code, including project files **and the git repository** (simply archive the entire project folder).
- The graphics are not important, but clean code and adherence to good practices is.
- Please do not use any 3rd party framework.
- Please do not use storyboards.
- You are free to use the Internet for any needed references, e.g. efficient algorithms.
- If you reuse somebody else's code for part of your solution, please attribute this accordingly in your code.

Coding Challenge Tasks

You are given an initial setup of an app consisting of two views organized in a tab view controller: “Flights” and “Route”. Flights is set up to fetch and then present a list of (currently dummy) entries. Route is empty. Your task is to:

1. Flights: replace the fake implementation of FlightsLoader with fetching of the actual feed data (details below). Display the results - UIKit and SwiftUI are both acceptable for UI layer.

2. Implement a simple UI for the “Route” screen. This screen displays inputs that are able to take two arguments – origin IATA code and destination IATA code. On user-action, display the origin and destination IATA codes to the user.

3. Enhance the “Routes” screen you created in step 3: on user-action, calculate if a route from origin to destination could be accomplished using one or more flights. You don’t need to show the route, a YES/NO message is enough. For calculations, use the feed data already fetched in the Flights view. Avoid using shared global state (e.g. singletons).

Example flights (not actual data):

A: LHR -> JFK

B: MUC -> JFK

C: JFK->SFO

Example UI input/output:

Origin: LHR

Destination: SFO

Answer: YES (via route LHR->JFK->SFO, but the UI doesn’t need to show this)

Feed Data

Feed URL: https://s3.amazonaws.com/ios-coding-challenge/ios-test-data.json

Sample JSON object below with short field descriptions.

```
{
    "flight_id": "edb567a", // FR24 flight ID
    "aircraft_id": "896177",
    "latitude": 30.6522,
```

```
  "longitude": -69.7767,
  "track": 246, // degrees
  "altitude": 38000, // feet
  "speed": 448, // knots
  "squawk": "4613",
  "radar_name": "F-TXKF2",
  "aircraft_model": "B77L",
  "aircraft_registration": "A6-EWE",
  "timestamp": 1505481740,
  "from_iata": "DXB", // origin IATA code
  "to_iata": "FLL", // destination IATA code
  "flight_number": "EK213",
  "on_ground": 0,
  "vertical_speed": 0, // feet / minute
  "callsign": "UAE213"
}
```

Gratitude

As a very small token of gratitude for spending time on this, we'd be happy to send you a Flightradar24 keychain and lanyard. If you're interested in this, please provide your home address when you hand in your solution.