Team Go Software Design Principles Test

The following is a vague description of our vision for Go that is meant to both confound and inspire you:

> Imagine an app that helps you get off your device. No more likes, no more FOMO, no more isolation and endless content holes. No more loneliness!
>
> Imagine an app you can use to watch Netflix with a buddy, or have a potluck with your close friends. Go bouldering or check out the best concerts and events happening in your city--or even go meet a group of strangers for a bike ride! It's all possible.
>
> Go is the world's first Social Life app that helps you spend more time with people and actually do things.

Hard problem, right? What might it look like? How might it work? That's up to you.

Your mission is to use Flutter and Dart to build an app with the following primary functions:

1. Imagine **a scrollable "feed" of activities** inspired by the description above. Any activity is fine--please feel free to get bizarre and use your own real-life hobbies as inspiration for mock data. The feed list of activities shown should scroll infinitely, but contain at least 50 unique cells/items upon first launch (define and handle the data source however you like). Each cell should each convey at a minimum the "who, what, where, and when" of the activity (laid out however you see fit).

2. In addition, allow for a button to **add a New Activity**. There should be enough information gathering done during this flow such that we can fully-form a new item in the feed based on the New Activity you just created (again: "who, what, where, and when"). After the New Activity is created, insert it into the feed and re-render the feed.

3. When tapping on an existing activity in the feed, show a focused **detail view of the tapped activity** (e.g. a full-screen modal, or an expanded inline view, etc). If the activity was created by the current user, allow the item's properties/meta data to be edited and saved and re-rendered in the main feed.

General guidance

- Treat this as a module or screen that you would build on as part of production code. While reviewing your code, we will be primarily concerned with your used of abstractions, interfaces, exception handling, comments, variable/module naming, code linting, etc.
- On unit tests: we believe in them. You should demonstrate that you can use them, and how to run them. Or else convince us we're wrong. ;)
- On visual design: You are not expected to be a UI/UX expert, but your on-screen elements should be well-balanced and your user flow should be sensible. Images and even colors are totally optional; for this exercise, clean is better than pretty, and messy is worse than ugly.
- You may extend the parameters of the test and build additional functionality in your project if you feel it is required to properly show off your skills (or just for fun!), but remember that quality is better than quantity.
- Use third-party, open source modules as you see fit--just be sure to *also* show enough of your own code that we can get a sense of your code and abilities.
- Do not worry about building onto a device; simulators are fine.
- Ensure everything builds properly on the latest macOS.

Submitting your project:
When you're all done, upload your project to GitHub and include a `README.md` that details your build environment, any instructions to build and test the project (from scratch!), and any other context that is important to understand while using or evaluating your project. We will use the project to discuss the technical decisions you make.

When you're done, email us the project link and any additional musings you'd like to share. :)

#go/recruiting