Dynamic Profile Ordering

Product Spec

Our Product team wants to experiment with the order of information in a user's profile, to see if first impressions are important. They are looking to answer questions like: Should the profile photo be first? What if we placed the about me text higher? Is the school important for a College aged demographic?

Your assignment is to build an app that displays user profiles one at a time, with the order of the profile fields defined by a configuration object. There should also be a button that allows you to navigate to the next user.

Matching the profile layout exactly is not required for this assignment; just use your best judgment to get close to what's shown in the demo video.

Implementation Details

You will need to access two API endpoints for your project. The base URL is:

`http://hinge-ue1-dev-cli-android-homework.s3-website-us-east-1.amazonaws.com/`

`GET /users`

This endpoint returns a list of users and their profile information. Each user will have an ID, name, and gender. Other fields will be optional.


```
{
  "users": [
    {
      "id": 1,
      "name": "Jim",
      "gender": "m",
      "about": "*Looks at Camera*",
      "photo": "https://tinyurl.com/mpckmdss"
    },
    {
      "id": 2,
      "name": "Pam",
      "gender": "f",
      "photo": "https://tinyurl.com/3vpr35zj",
      "school": "Pratt Institute"
    },
```

```
    {
      "id": 3,
      "name": "Michael",
      "gender": "m",
      "about": "Scarn, Michael Scarn. Shaken not stirred.",
      "photo": "https://tinyurl.com/2bsthyda",
      "school": "School of Life"
    },

    …

  ]
}
```

`GET /config`

This endpoint returns the display order of the profile fields.

```
{
    "profile": ["name", "photo", "gender", "about", "school", "hobbies"]
}
```

Expectations

1. A consistent and well defined app architecture. You may choose whatever architecture you want; focus on applying it consistently throughout your project.
2. Production-quality code.
    - No commented out code, no crashes, broken UI, etc.
    - Edge cases covered
    - At least some test coverage
2. Written in Swift (for iOS candidates) or Kotlin (for Android candidates).
3. Provide a README describing architectural decisions you made, any libraries used, and anything else you would like to highlight as part of your solution.