Chat: Slack - **https://dontdinealone.slack.com/messages/CA6JHDWP6/ (join here)**
Board: Trello - https://trello.com/b/8FILZJCb (join here)
Github: https://github.com/tylerhu3/DontDineAlone
Meeting Location:
BE 340 - 11am-11:30am Mondays
SE 308/9 - 2:00-3:00pm Tuesdays and Thursdays

Product Owner: Tyler
Scrum Master S1: Jean
Scrum Master S2: Tarun
Scrum Master S3: Mac
Scrum Master S4: Cody

Team Responsibilities

Product Owner: Tyler Hu
qhu27@uscsc.edu
tylerhu@yahoo.com
510-701-7525
https://github.com/tylerhu3

Sprint 1 Scrum Master: Jean Park
jejupark@ucsc.edu
kestry@gmail.com
415-378-1642
415-203-1321 (boyfriend's phone, text for emergencies)
www.github.com/kestry

Sprint 2 Scrum Master: Tarun Salh
tsalh@ucsc.edu
tsalh2019@gmail.com
510-945-8873
www.github.com/tsalh

Sprint 3 Scrum Master: Mackenzie Huynh
mhuynh2@ucsc.edu
408-207-7249
www.github.com/machuynh408

Sprint 4 Scrum Master: Cody Cunningham
cohcunni@ucsc.edu

codyhc520@gmail.com
650-296-3506
www.github.com/codyscode

Definition Of Done

- Code is completed
    - Code is written to complete all tasks or subtasks
    - Author of code reviews their own work to see it completes all tasks successfully
        - If the author is satisfied then push the code to github
    - Check if to see if code follows the coding standards and style guidelines set by the team
- Peer Review
    - Further review by other team members look over the code that the author has pushed to github
    - Review coding standards and style guidelines of their code
    - Manage version control and merge conflicts
- Q/A
    - Code passes unit tests
    - Code walkthrough does not have any bugs
    - Code does not have any bugs

Development

- Android App
    - Almost all programming was done in Android using Android Studio
        - Login activity, register activity, editing user profile was all done in Android
    - All Q/A such as SUTs/CUTs using mockito objects, JUnit, Expresso were done using android
- Cloud Firestore
    - Backend services were done using Firebase
    - Profile, email, and password information was all stored in the Firebase database
    - Programming was done using Android
- Local Server
    - A local server was written and used to handle the matching activity for the app
        - Enqueue users who wanted to be matched

        - Matched users in their preference groups and created a chat room for them to connect with each other
    - Server code was written in C#

Architecture:

The architectures used are Model-View-Presenter (MVP) and Clean Architecture.
It is a simplified adaptation of code found at:

https://github.com/googlesamples/android-architecture

which shows how a codebase gradually transitions into more sophisticated structures.

MVP is as follows:

[physical data] -- [model of data] -- [presenter that uses the model and views to create our presentation for the user] -- [view which are dumb UI services like widgets/buttons/etc]

Clean Architecture inserts another layer to MVP in between model and presenter. This layer is called the domain and consists of use cases and business logic. In our project, we call use cases "services" to also make this more intuitive.

Presenters and views are both within the Activity for now as well. In order for the Activity to be more organized, the code is separated into five sections:

1. Variables/Constructos/Initializers
2. Lifecycle overrides, in order of lifecycle.
3. Presenter functions
4. Navigation functions
5. Callbacks

At the moment, the layout has heavy dependency usage because of the direct calls to Entity. But since we are mostly new to the technologies and concepts, it is a simpler start.

Codebase Layout:

Folders:
  main
    - data
        - entity: Different models of our user. (non-traditional use of name)
        - model: Model of data
    - domain: Use cases
    - net: Match server
    - res: Constants
    - util: Utility classes
    - (no folder): activities
  androidTest
    - general: general tests and helpers
    - SUT
    - (no folder): our activity tests
  test
    - domain: Use case tests

Glossary:
    - "documentId" == uid

Coding Style Guidelines

General

In general, we want to follow the Google Style guidelines. The following are just some things of note.

Comments

- // Comments have a space in between the // and the first letter to improve readability.

Comments

- Brackets should be used for if and else statements, unless the statement is written on the same line. This is important to prevent some difficult bugs.
    - E.g.

        - if (foo == null) return;
        - if (foo == null) {
            return;
          } else {
            // do something
          }

Naming

- Acronyms
    - Acronyms like USA or SSN should only have their first letter capitalized if it is an inner word.
        - E.g. userSsn, getUid, fromUsaToChina
- Variables
    - In general, variable names be descriptive, unless there are generally known conventional variables.

- Objects
    - Xml objects and Android widgets should be prefixed by their type so that we know what they are and are also fast to search and type.
        - E.g. buttonRegister, textViewTitle, editTextName
- Functions
    - All functions should be camel-cased for consistency.
        - E.g. getDisplayName()
- Test Functions
    - test<What>_<With/On>_<Should/Behavior>
    - Example:
        - testButton_OnClick_ThrowException()
        - testButtonClick_WithValidUserInfo_SaveProfile()
- Classes:
    - Data objects and models are named plainly.
    - Business logic functions/Use Case/ Helpers are appended with the word “Service”
    - Tests of the class are appended with the word “Test”

Loops

- Favor for-each loops for performance whenever possible.
    - E.g.

```
int sum = 0;
```

```
for (Foo a : mArray) {
     sum += a.mThing;
}
```

Layout

- Activities
    - Order Lifecycle methods in order of Lifecycle:
        - onCreate
        - onStart
        - onResume
        - onPause
        - onStop
        - onDestroy
    - Have Lifecycle methods before your methods
    - Have your navigation methods in separate section below your other methods above the callbacks
        - They should be called “goTo_____Activity”
    - Have callbacks at the bottom of the activity source code