Single Parent Community

02-Mar-2021

AFRIN Rukaya(A0113802W)
CHAN Jian Liu (A0226741H)
LIU Lei (A0214899L)
RAMAKRISHNAN Niveditha (A0214867W)
XIAO Changwei (A0226757U)
ZHANG Hongduo (A0226744B)
ZHOU Yanjun (A0226701N)
ZHU Haokun (A0226723H)

Product Introduction

The Single Parent Community (SPC) project consists of an Android app (for users) and a JavaWeb app (for admin). Technologies used in the project includes Android (Java), Spring Boot (Java), Python (Flask), Firebase Cloud Firestore and Firebase Cloud Messaging.

The Android app allows users (specifically made for single parents) to exchange parenting experiences, make friends or even develop new relationships! Although originally proposed as a dating application for single parents, our app has evolved into a social platform for single parents to discuss parenting tips, share their troubles and seek advice from other single parents who truly understand their situations. We have amended the dating feature into a friend feature by using a machine learning algorithm to help users find other users who have similar interests and backgrounds.

As for the JavaWeb app built for admins, we strongly believe that there is a need to have admins to oversee the interactions in the social community in order to provide a safe and comfortable environment to the single parents using our Android app.

Background

Taking care of children as a single parent can be challenging. Meeting new people and building relationships are equally challenging for single parents.

There are many social media platforms like Facebook, and Reddit built with the general users in mind, and dating apps like Tinder trying to help people find love partners. However, these social platforms are not catering towards the single parent community and the dating apps are often designed for younger users who are not ready to settle down.

On the general social platforms, single parents may feel uncomfortable sharing their marital status as well as their struggles as a single parent. They may be self-conscious of how people would look at them through their profiles and sharing. Thus, we decided to create a social platform catering towards the single parent community, providing them with a platform where they can openly and comfortably share their profiles and sharing among other single parents.

Our app is catering to:

- Single parents who are looking for suggestions on the difficulties they are facing.
- Single parents who are looking for friends who have similar interests/lifestyles and kids in the same age group in order to share thoughts, parenting advice, tips etc.
- Single parents who are looking for genuine and serious partners to build warm and happy families together

System Features Developed

1. Android--Authentication: register, login, logout, facebook login, change password, textbox validation. These features were developed using the Firebase Authentication library and Facebook SDK.

2. Android--Forum: A forum for single parents to discuss issues on dating, parenting, childcare etc. Users can post new forum topics to raise a question, and other users could then contribute to the topic by commenting to the original post. They may also reply to comments under the topic.

3. Android--Forum Category Subscription: Allows users to subscribe to a forum category and display it in descending order by date and time on the "personalised" tab.

4. Android--Forum Latest Tab: Allows users to view the latest forum topics by date and time posted by other users.

5. Android--Create New Forum Topics: This allows users to create a new topic and start a discussion under any one of our 6 given categories. When users click submit, the topic is saved in the Cloud Firestore database as a document.

6. Android--Friends: Allows users to add friends through a list of potential matches (see the algorithm behind the calculation of potential matches below), as well as through a search by user's username. Added friends will appear in the "friends" tab.

7. Android--Push Notification: Uses Firebase Cloud Messaging to host and send notifications to users when a user replies to a Forum topic or comment.

8. Android--Setting: Users can change their full name, password and profile description. They can also refer to external resources to get more help, send our team feedback about the application and logout. These features have been implemented via the Preference Library from Android.

9. JavaWeb--View Users: Admin can view the full list of users on the JavaWeb and search users by username. Admin can also view a copy of the agreement agreed by users when they registered an account on the android app (user's full name and registration/agreement date is displayed).

10. ML(Python+JavaWeb)--Sentimental Analysis: Admin can calculate users' sentimental scores based on the comments they posted in the forum, and then display the scores for admin to achieve further operations (eg. suspend a user account if his negative score is fairly high)

11. ML(Python+JavaWeb)--Recommend list of potential matches for users. Using KMeans algorithm, we find clusters within all users exist in our database based on their profiles (self-description, interest, profession, number of kids, kids description, etc), and for each user we recommend users within the same cluster with him/her as potential matches. Note that here we made an assumption that the higher similarity between two people the higher possibility that they can get along well.

12. Android and JavaWeb--Suspend Users: Admin can suspend users by a click of a button. Users are prohibited to log into the android app after being suspended. Users would need to send an email to the admin to discuss the possibility of unsuspension.

Recommendations

In the future, our application can implement a chat feature which allows friends to have personal chats with each other. This will allow for deeper bonds to form and provide users the convenience to communicate with each other directly through the app.

Editing and deleting of forums and comments by the users should also be implemented in the future to allow changes to writing mistakes.

Viewing of forums and comments on the JavaWeb for admin should also be implemented to allow a smoother checking of the user's writing. Especially when an admin notices the specific user has a very high negative score on his/her sentimental analysis.

Additionally, we could implement a full text search feature using the third-party search service "Algolia" as recommended by the official firebase documentation. This will allow users to search for any topics or discussions they are interested in based on keywords. The reference: https://firebase.google.com/docs/firestore/solutions/search

What's more, we can implement more login features such as Google login and get more user data from their social account. It will make the authentication process more convenient.

Lesson Learnt

Android:

1) We learnt to connect Firebase Cloud Firestore to our Android app to write and read data from the database, and also learnt to use the Cloud Firestore query to query data from the database.
2) We learnt to create recycler views as well as Firestore recycler views to display rows of data dynamically.
3) We learnt to use the navigation bar and tabs (switching tabs by swiping).
4) We learnt to implement the Android push notification by using the retrofit2 library as the messaging API and using Firebase Cloud Messaging to host and send notification.
5) We enhanced our skill in beautifying the Android app UI.

Flask:

1) We learnt to use Flask to host two servers at different ports to achieve the following tasks:
    a) Retrieve user data from Firebase Cloud Firestore

    b) Calculate the sentimental scores of users' comments
    c) Find potential matches for each user
    d) Persist results into Firebase Cloud Firestore

ML:

1) We learnt that feature engineering (data cleaning, transformation techniques, NLP) is time consuming but crucial for analysis at later stage
2) Figuring out a suitable ML algorithm for catering our business purpose requires a fair amount of researching and experimenting. We did quite a lot of trial and error and finally managed to figure out a reasonable model.

Java Web:

1) We learnt to retrieve data from Firebase Cloud Firestore, and enhanced our skills in Spring security (password hashing) and beautifying the web UI with Thymeleaf.
2) We learnt to invoke call to python Flask from the JavaWeb app to start the ML calculations.

General thoughts:

- Communication across different technological platforms is crucial. We spent a long time figuring out an effective way to communicate among Android, Java Spring Boot, python (Flask), Firebase Cloud Firestore and Firebase Cloud Messaging.
- We learnt that apart from the technical work we did for the project, the communication among our group members is also equally important as it allows the workload to be handed over to one another smoothly. For example, the structure and coding style are different, when we merged our codes, there were problems surfaced. After an effective communication with our members, we were able to solve the problem.
- We learnt that it is very efficient to subdivide features and assign them to team members as tasks. Each person just needs to focus on completing their own tasks, greatly reducing the frequency of errors.
- Peer coding and debugging are very helpful and effective and can help team members build strong and trustful connections!