Friendr

CS 407 Design Document

Apoorva Parmar, Josh Radochonski, Dennis Chia, Brandon Xia, Srishti Gupta

Purpose

When we join a new course we are always looking for someone to study with and get help with homework. TAs are not always there to help us. Also, there is no efficient and user-friendly system in place for Purdue students to connect with their classmates and form study groups and get help on homework. Our app will help students to find other students by offering them a search mechanism, random pairing, and an intuitive personal messaging system. We will provide a system through the use of an Hybrid application which will work on cross platforms making it user friendly on all mobile and web applications.

Design Outline

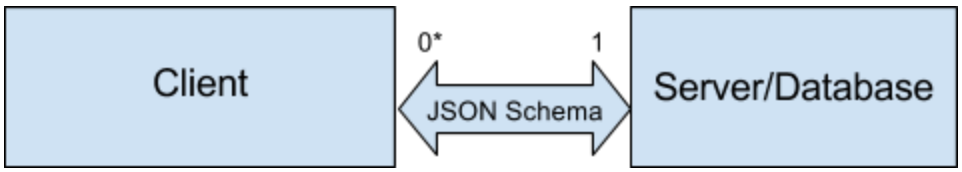
Our application matches users together with similar activity goals in mind. Our implementation will utilize the client-database model. The database will keep track of all the user information and the currently searching users. The client will fetch the database information as the user uses the hybrid app.

1. Client
    a. We will use AppGyver as the system to create and deploy the application to iOS, Android, and as a Web page.
    b. The client will be running on the user's mobile device or computer web page and will contain the main interface for the app.
    c. The information the client retrieves from the database will be of JSON format.

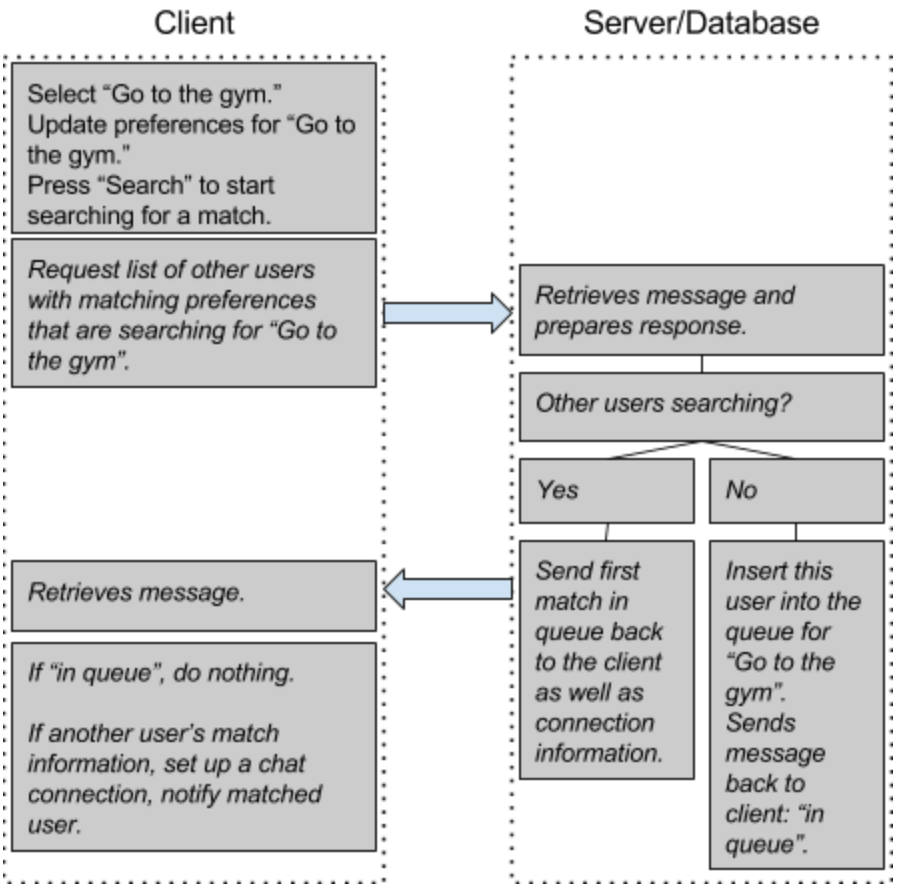
2. Server/Database
    a. We will use Firebase as the server and database for this application.
    b. The database will contain all of the user's information and preferences.

c. The database notifies the client when it gets modified and the client reflects the new modifications.

The relationship between the client and database will be many-to-one, meaning that many users will have the ability to connect to one and only one database.
Firebase, our database, has built-in authentication and user storage. On create of an account, the client will call the Firebase API to create a new user. On subsequent login attempts the client will also call the API to validate the user.



Example Flow of Event: Matching a user with another person.



Design Issues

Functional

1. Do users need to login to use our service?
    a. Option 1: Allow the creation of login id using a Purdue email account.
    b. Option 2: User will be given a unique password

    Decision: We chose to create accounts for first time users using Purdue email accounts for security measures. Users are not being provided with unique username so that their identities does not remain secretive and they can communicate with others effectively. For compatibility, we also decided the user to be logged in the app until he/she decides to logout of the app instead of entering password every time to use the app.

2. How will the users be notified if they are matched?
    a. Option 1: Each matched user will receive the notification
    b. Option 2: Each can view each other's profile

    Decision: We chose to send notifications to each user whose preferences matches. We also provide them the flexibility to view their respective profiles and choose to whether they want to talk to them or not after being matched.

3. How will the users communicate through this app?

    Option 1: Have the user's phone number stored on profile and exchange numbers when people are matched.

    Option 2: Have a chat within the app that users can message each other with.

    Decision: Chat within the app. Having the user's phone numbers exchanged poses a security and privacy risk to the users. We decided that having a chat within the app would be more convenient for the user and add additional functionality to our app. In the case where users become inappropriate or threatening, users will be able to block each other on chat.

4. How will users meet up once matched with the app?

Option 1: The users will choose a time they are available and the app will tell them where and when they will meet.

Option 2: The users will be connected via chat and decide on their own where and when to meet.

Option 3: Assume that the users are available when they start searching and meet up at an appropriate location based on the activity.

Decision: Users will be connected via chat and decide on their own. This gives the most freedom to the users and also covers potential problems like users changing their minds or suddenly being not free. The chat will also allow for users to communicate in the future through our app.

5. How the user will know about his notified friends?

Option 1: Through recent chats

Option 2: Through recent matches

Decision: We decided to create a list of recent conversations, so that the user can contact the person again if he wants to work with him/her again. There will also be a list of recent matches so if person by mistake deletes the notification from the notification bar, it will be still in the app.

Non-functional

1. What will we do for our backend database/server?

Option 1: Google Firebase

Option 2: Create our own server and database.

Decision: Google Firebase. We decided that it would be easier and more reliable to use firebase as a backend service.

2. What platform will we write our app for?

Option 1: Android only

Option 2: iOS only

Option 3: Hybrid application

Decision: Hybrid application. Our very first intent was to just make one mobile application, however we decided that a hybrid application that works on all platforms would enable us to write and maintain less code and at the same time give users across different mobile platforms the same feel and experience. Which is why we started using AppGyver, an application built on top of PhoneGap, making using of our JavaScript and AngularJS skills to build a fluid UI.

3. What app language and framework we will be using?

Option 1: Java

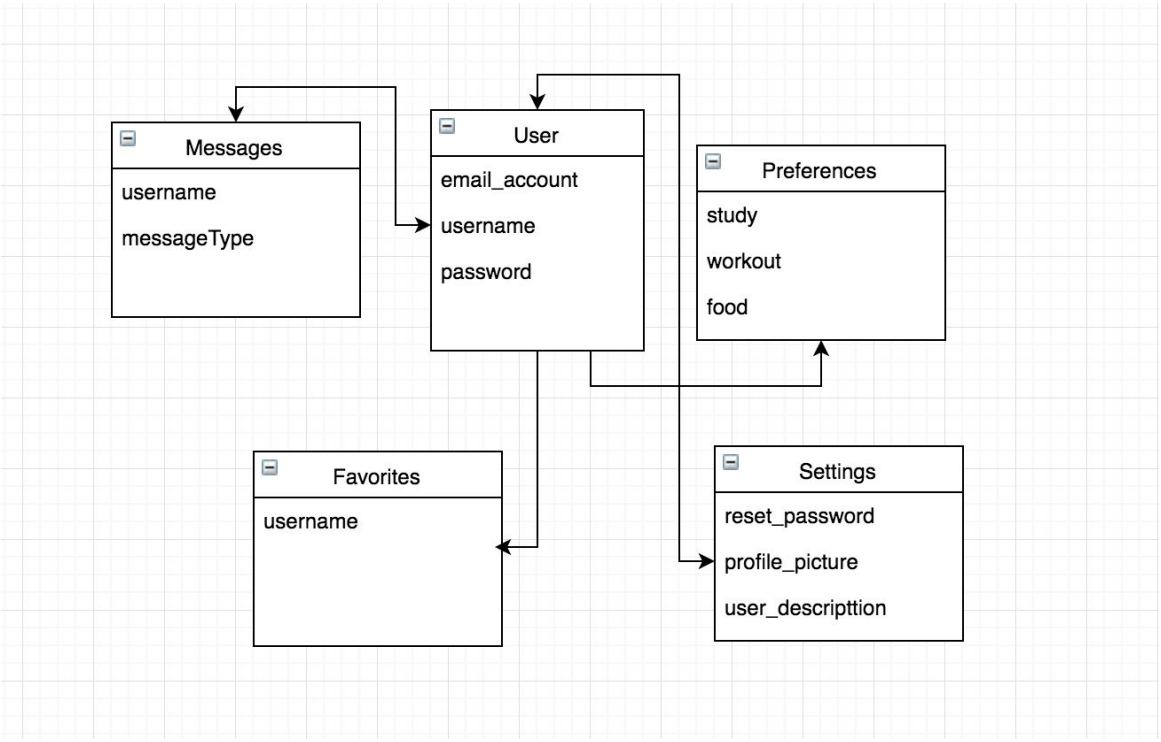
Option 2: C++

Option 3: Javascript, HTML(webapp)

Decision: We decided to use java as it is the most convenient and most common, and also standardized by Google.

Design Details

Outline: **Class Diagram**



Description and Interaction of Classes:

The **front end** will display a list of activities a user is interested to do. Based on user's preferences a call is made to the **back end** to match the user with another user having similar preferences. The **database** will run a query to retrieve user's information and then do a search analysis with other user's for match. Once matched the data is processed, it is passed to the front end which will display a modal view of the user's most recent activity. The search on the front end can either be random or deliberate giving the user the choice to find someone from the list of the users he has been matched to already.

Main components :

- LoginScreen
    - This class would be the starting activity of the app.
    - It would provide the functionality to a user who wants to login or register.

- Register Screen
    - This class provides the options to register an account.
    - Fields such as email, name, password, phone would be required by the user in order to create an account.

- HomeScreen:
    - This class would control the main display of the app.
    - It would be the main page for the user to search for another user to do an activity with (i.e. studying, eating, working out, etc.)
    - It would display a side drawer which will consist of User Settings, Favourites , Preferences , Messages and Logout.
    - It will show a modal view of a user's most recent activities.

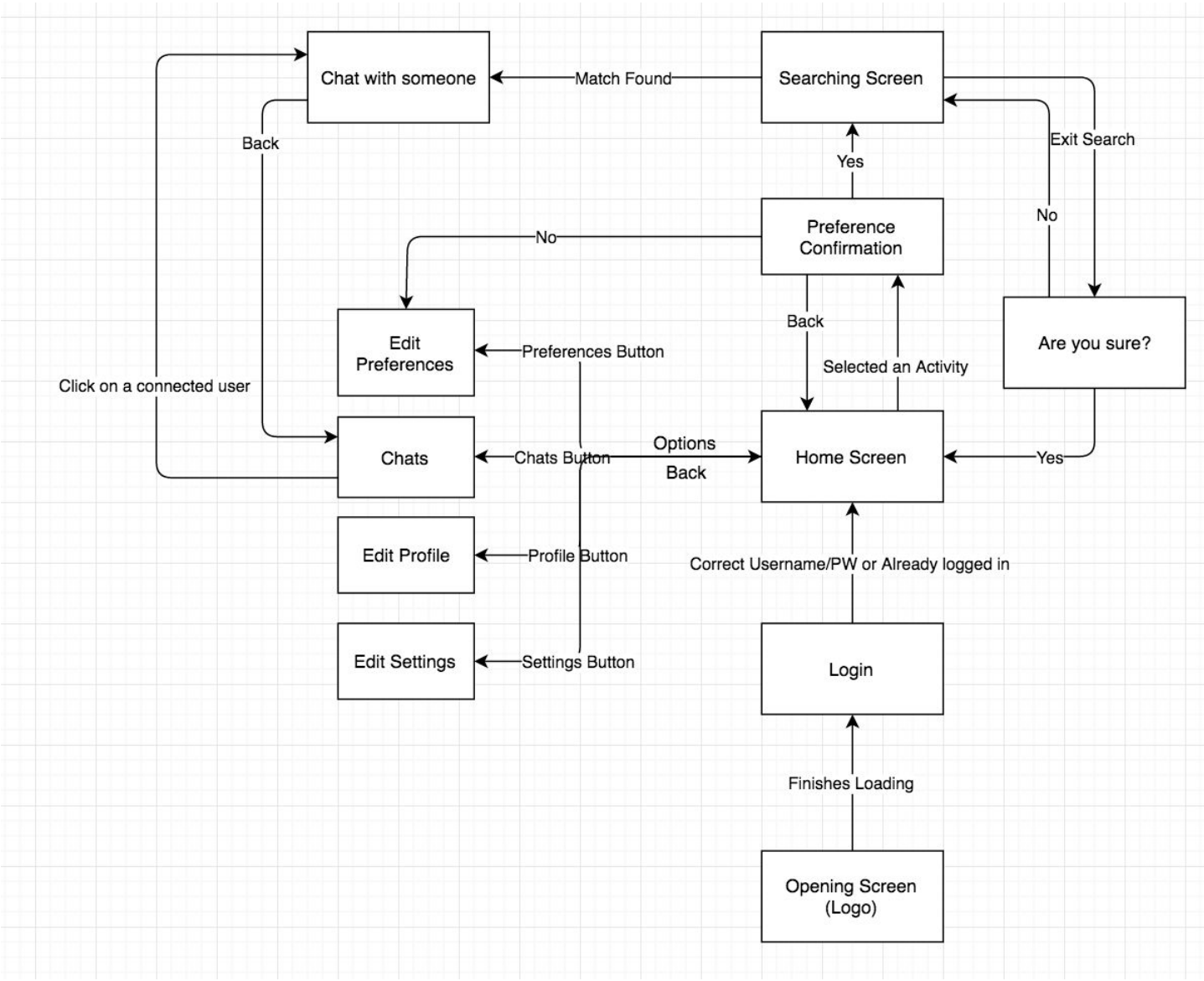
- Drawer View:
    - This class manages the properties and contents of the drawer that can be opened up by a left slide on the home screen.
    - It contains a List of tabs : Messages, Settings, Preferences. Further,it has options such as Feedback, Help, Logout and displays the profile image of the user.
    - Objects of the specific classes and call to the appropriate class is made based upon the user selection.

- Settings View:
    - This class manages the settings of the app such as change of user's name, email and password.

- Messages View:
    - This class would have a list of conversations between the user and the person he is randomly matched with.
    - It contains a list of options such a delete a conversation or view a single conversation.

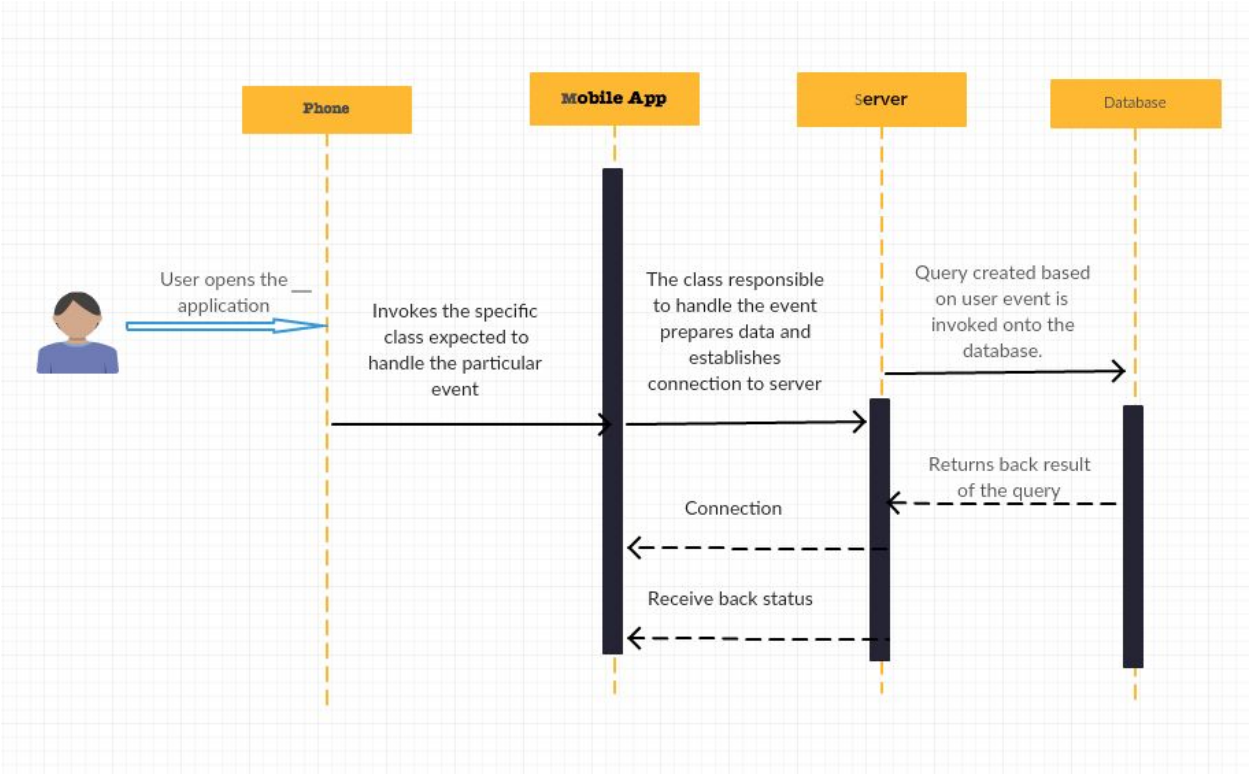
- Preferences View:
    - This class would have a list of user's preferences for specific activities. For examples for studying a user would have the option to add his classes, for food a user could set his preferences to vegetarian/non-vegetarian and for workout a user could set his preferences to cardio workout or core workout etc.

State Diagram



Sequence Diagram:

The sequence diagram showing potential sequence when a user opens the mobile application.

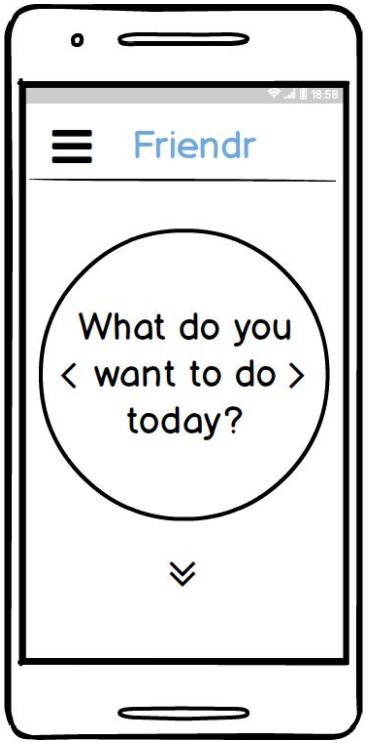


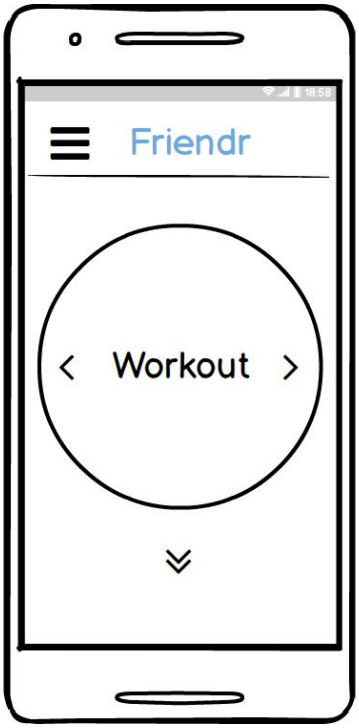
UI Mockup

- The Login page of the application

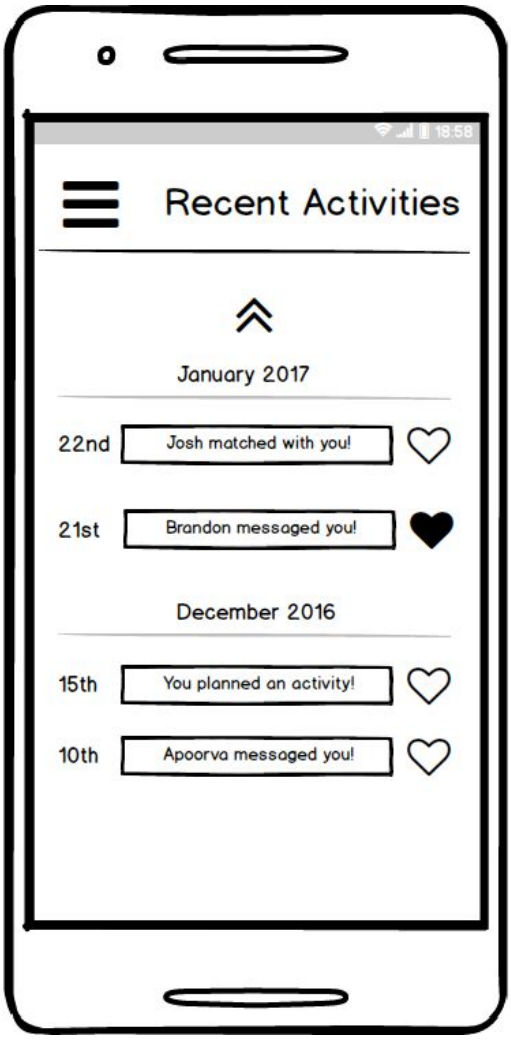
- This is the registrations page where users create a new account

- This is the home page where users choose an activity to do. Users slide right / left to choose from the categories of activities.

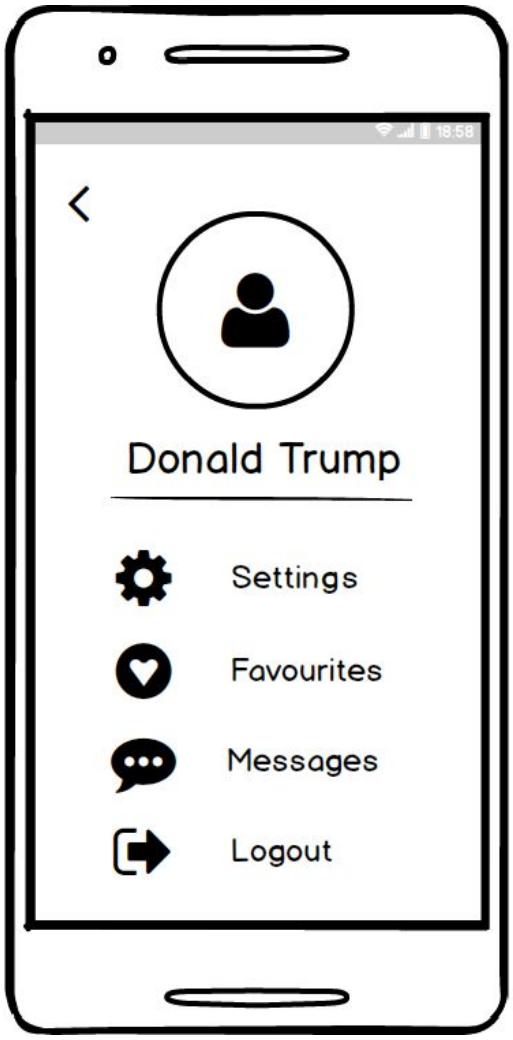




- From the home page. The user can swipe up to go to the recent activities page
- In this page, users can also favorite the activity or click on the activity to jump to the respective page



- This page shows up when the navigation button is clicked



Team 9