CS 290 Final Project: Friender

Programmer Documentation

By David Haotian Zheng (dhz), Wilson Zhang (bz43), Liane Yanglian (xy48) and Henry Yu (ay64)

Gitlab link: https://gitlab.oit.duke.edu/bz43/friender

Website: https://davidhzheng1996.github.io/friender/

Overview

Friender is a location based interactive app that allows users to view other users' current activity/status and interacts with other users.

- Users can add their current statuses alongside pictures or any audio they want to record.
- Users can also look at other users on the map, click on them and view other users' statuses.
- Users can also interact with other users by using the poke button.

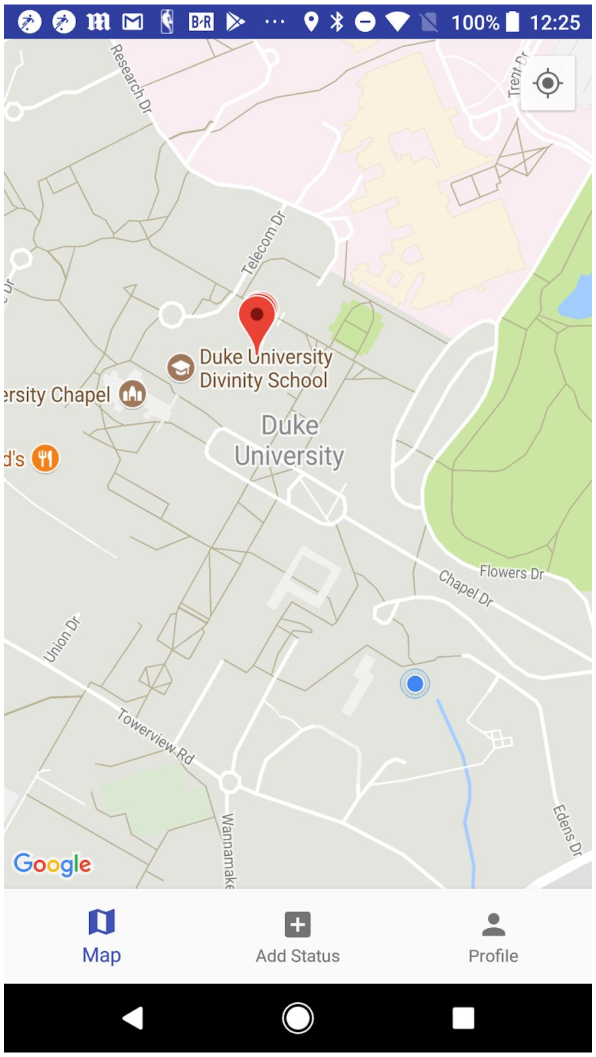


Figure 1: Front page of friender.

In the following sections, we will be going through the different functionalities of our app in depth.

FrontEnd

Login and Sign up

When a user first uses the app, he or she will encounter a sign up page. The user can then choose to sign up if the user has not done so by inputting his or her email, name, phone number and password. All these information would be passed to the backend (golang) and an authentication email will be sent out. Errorcodes will then be sent from the backend to the frontend and based on these errorcodes, different scenarios can occur, including the following:

- Email already exists.
- The 2 passwords do not correspond.
- Server down. (if the backend is not running)

Furthermore, at the backend, the passwords will be hashed so anyone operating the backend will not be able to see passwords.

The user will then go to his or her email and authenticate, after which the user can attempt to login. Different scenarios based on errorcodes from golang can occur:

- Email invalid.
- Password incorrect.
- Account not authenticated.
- Server down.

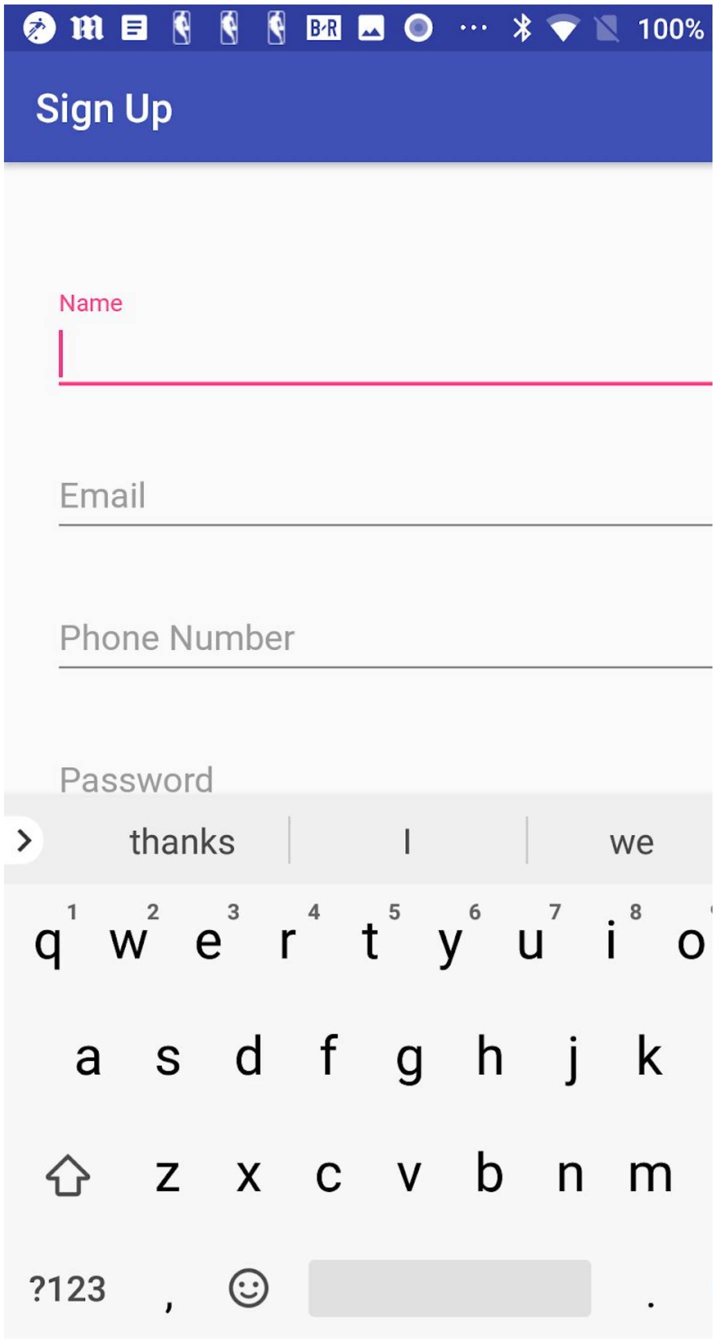


Figure 2: Sign up page.

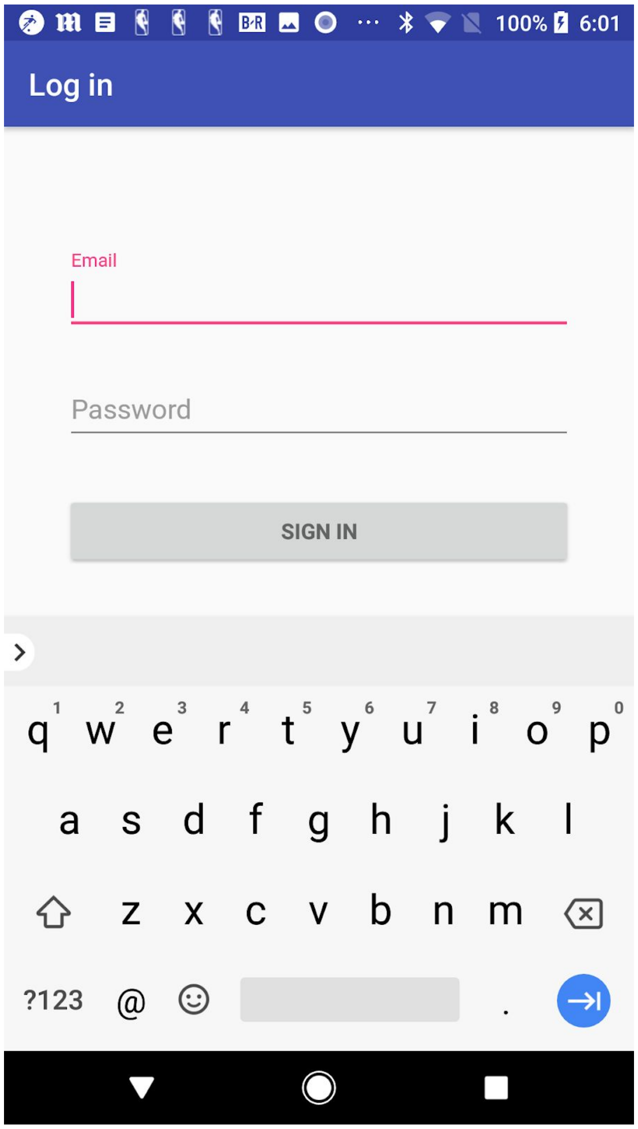


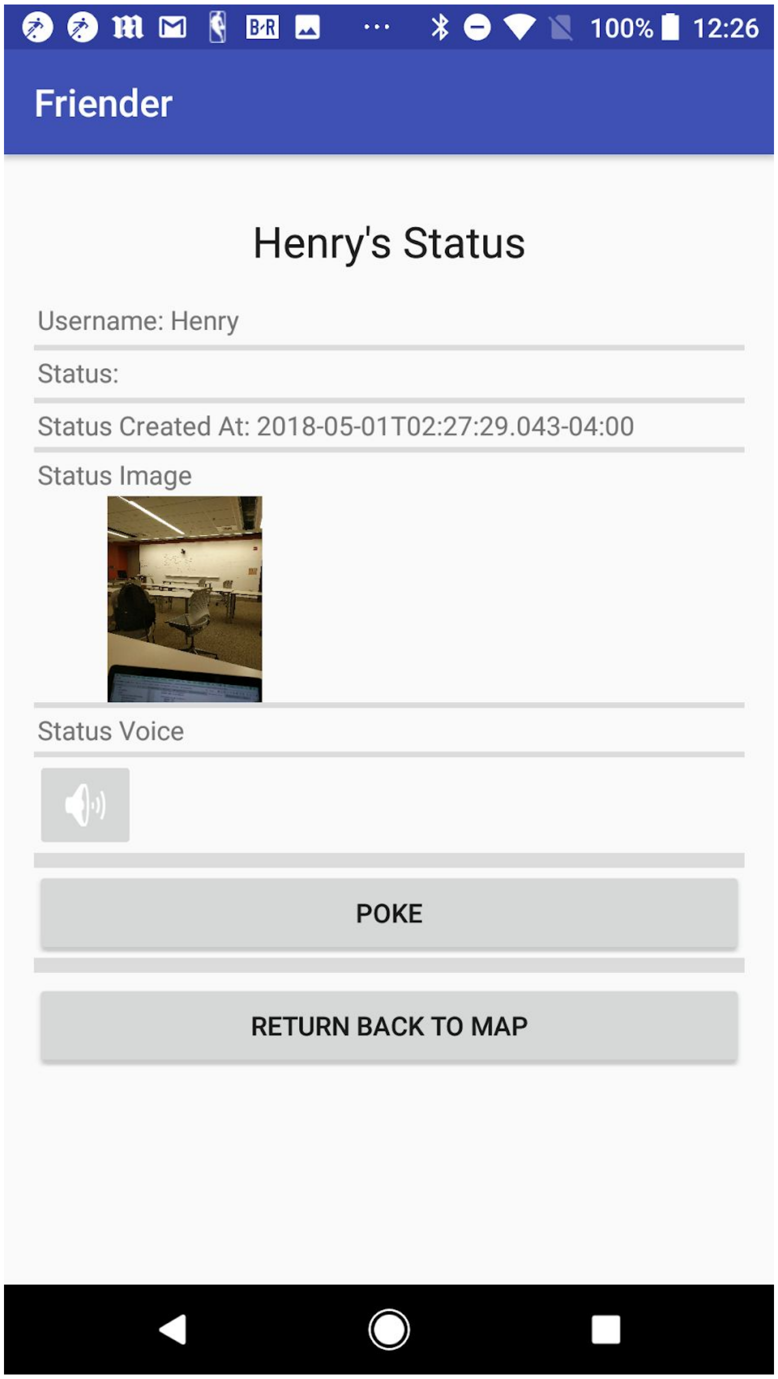
Figure 3: Login page.

Showing users on Map

Since friender is a location-based app, we incorporated the google map api into the app in MapsActivity.java and made this activity our main activity. For each user, we used .getLatitude() and .getLongitude to get their current location, send it to the back end and lastly, set a marker there. The marker is clickable through OnMarkerClickListener and clicking on the marker of one user would allow the current user (say user a) to view the other user 's (say user b) status page.

Viewing Status Page

After user a clicks on user b’s marker, the MapsActivity java class would go into StatusActivity java class. On the status page, user a would be able to view user b’s current status along with accompanying images and audio recordings. If user a wants to interact with user b, he or she would be able to click the “poke” button, which would send a notification (through fcm) to user b, notifying user b that he or she has been poked by user a. Thus friender allows user to engage in user interaction.



Username: Henry

Status:

Status Created At: 2018-05-01T02:27:29.043-04:00

Status Image

Status Voice

Figure 4: Status page.

Add Status Page

If a user wants to add status, the user can simply click on the center of the navigation bar and go into AddStatusActivity java class. On this page, the user can write his or her current status, add any accompanying images or audio, both of which will be stored in firebase. The user can then press the update button, allowing the new status to be updated and displayed onto his or her status page, then the user can press the back to map button to go back and browse the map.

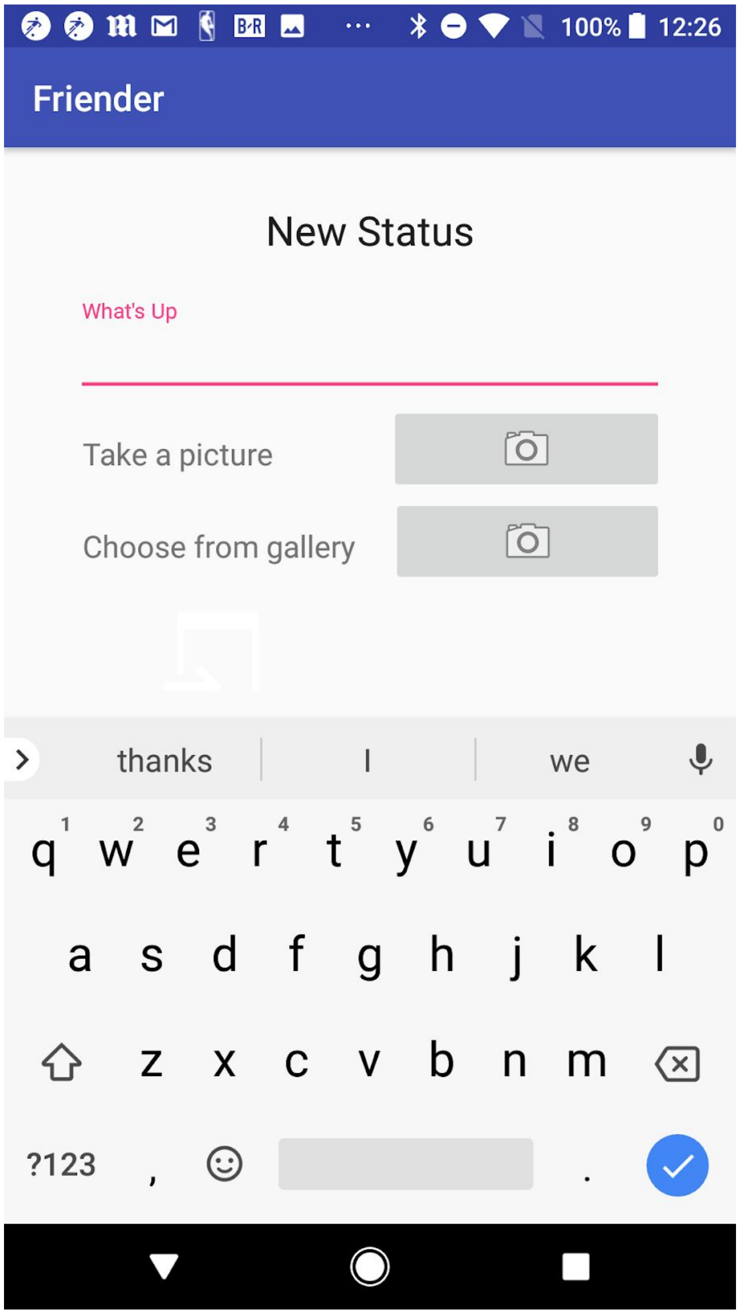
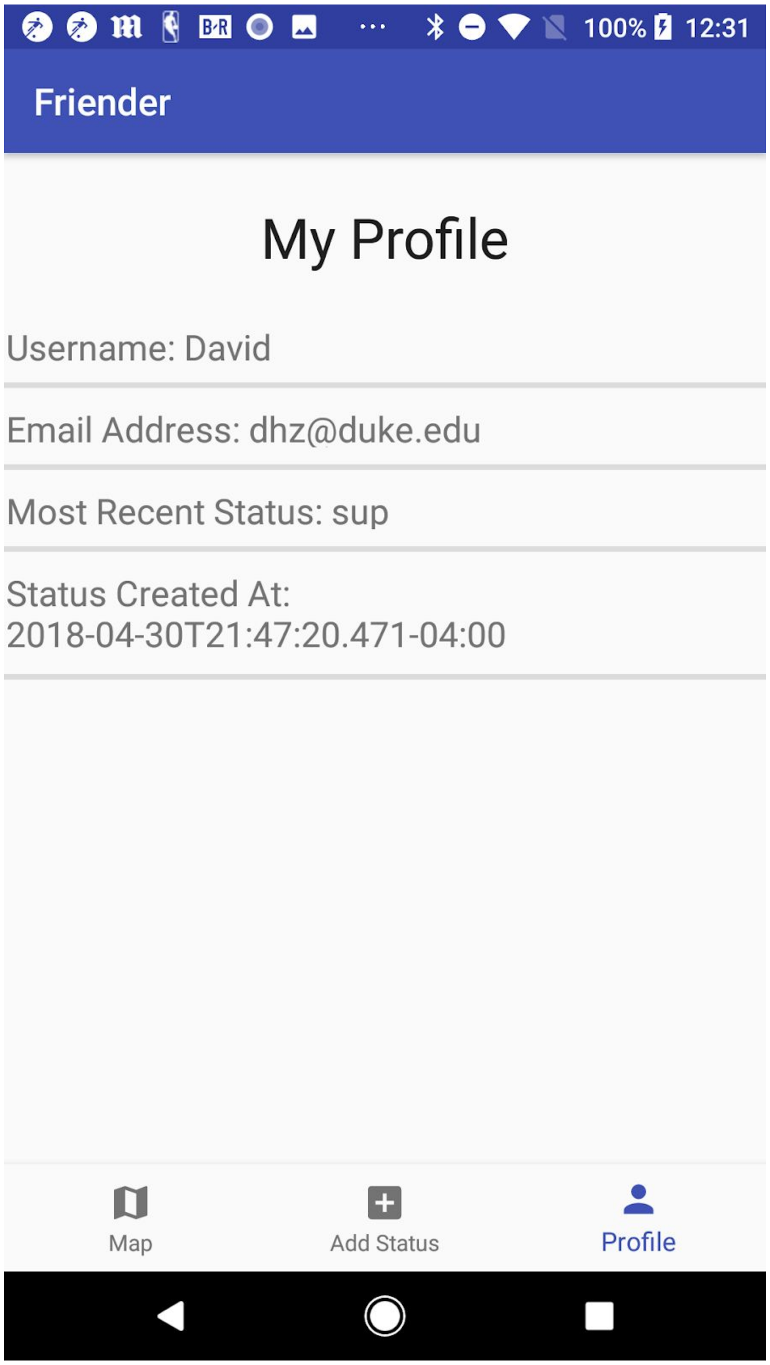


Figure 5: Add status page.

Profile Page

The profile page clearly displays the user's various information, including name, email, most recent status at the timestamp of the most recent status and this page is can be called by clicking the right profile icon of the navigation bar on map through using an intent to pass information from MapActivity to ProfileActivity.



My Profile

Username: David

Email Address: dhz@duke.edu

Most Recent Status: sup

Status Created At: 2018-04-30T21:47:20.471-04:00

Figure 6: Profile Page.

Notification

One user pokes another, say user a pokes user b. If friender is running on the background, user b will simply receive a notification, informing user b that he or she was poked and poked by user a. If the app is running, user b would receive a pop-up window containing the same information. The notification is created and sent by fcm (firebase cloud messaging). Whenever user a presses the poke button, user a's email and user b's email will be sent to the backend (golang), grabbing each user's fcm registration token (this is unique and gets updated on an hourly basis), allowing a connection to be established and sending a notification based on user b's token. Receiving a notification is handled in the MyFirebaseMessagingService service class, allowing a received notification to be read and sent to the device's notification manager.

BackEnd

For backend, we used firebase for storing images, audio files and fcm-based notification as well as Golang on an Iris framework for authentication and storing user information.

1. **Golang**

   We used Golang's Iris framework as a web server to server API requests such as getting other users' locations and sending out poke notifications. The backend store uses mongodb's Atlas service for a replicated key-value store.

2. **Firebase**

We mainly used the Firebase Storage and database in order to store images and audio files whenever a user create those in the add status activity. Whenever another user accesses that user's status, the corresponding images and audio files will be automatically downloaded from firebase and displayed. Furthermore, we used fcm to store and send messages in between users whenever one pokes another.

Difficulties and Solutions

We encountered several difficulties when creating the app:

1. We were unable to figure out how to implement notifications when the app is running hence we designed a popup window to show notification when the app is running.
2. We also encountered difficulties when storing images and audio with golang hence we used firebase to store non-text information.