FireTweet

Android Programming Project

Alexander Zevin (ajz725)

alexzevin@utexas.edu

Links

YouTube Demo: https://youtu.be/4tWlcxNgeNI (Timeline chapters added for convenience)

Github Repository: https://github.com/ut-msco-s24/final-project-firetweet

Description

FireTweet is a social media microblogging app heavily inspired by Twitter that allows users to share text-based posts called tweets and view a feed of tweets posted by other users. FireTweet also allows users to explore trending topics using hashtags and view their own profile page. FireTweet leverages Firebase backend services including Cloud Firestore to manage the tweets, users, and authentication.

Screenshots

Finished App:

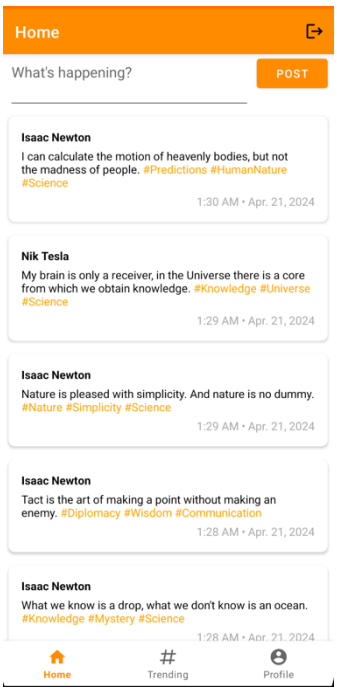

Figure 1: Tweets

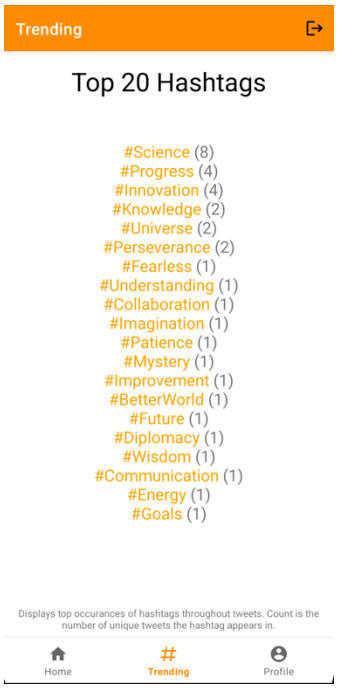

Figure 2: Trending

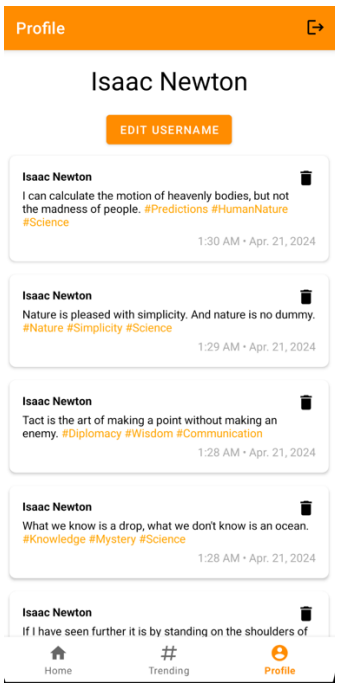

Figure 3: Profile

Proposal Mockup Comparison:

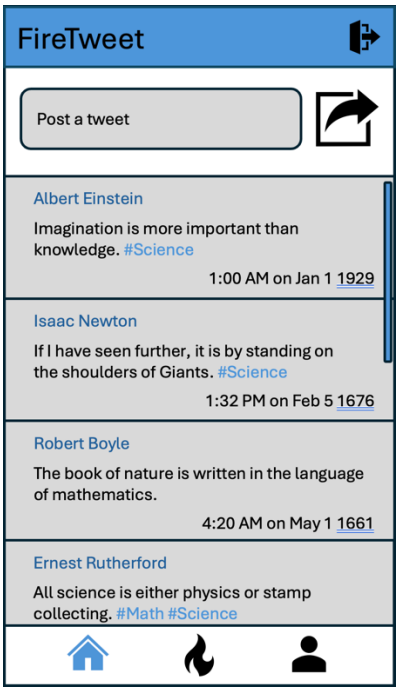

Figure 3: Tweets (Mockup)

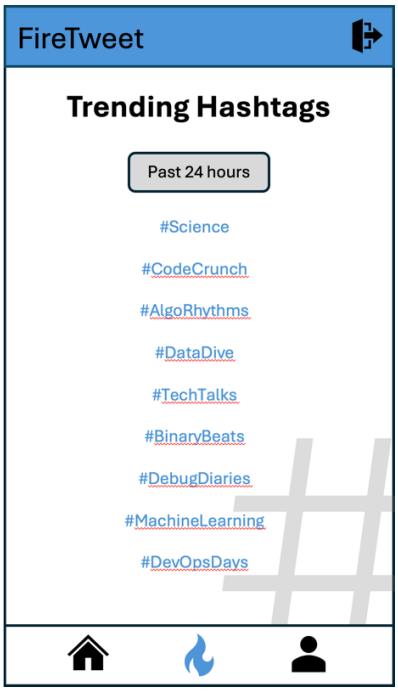

Figure 4: Trending (Mockup)

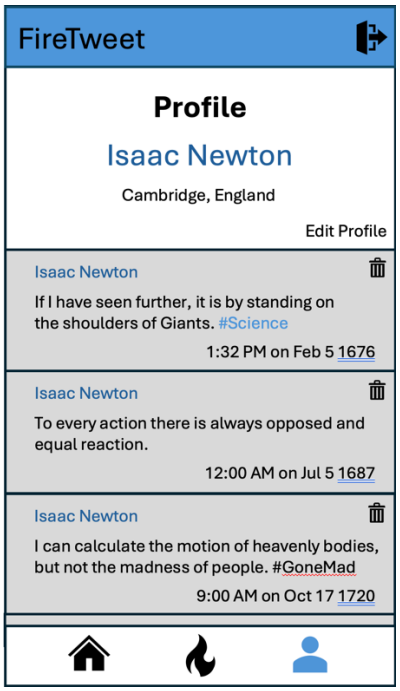

Figure 5: Profile (Mockup)

The project proposal mockups show that nearly all planned functionality was built.

Android Features

FireTweet was built using the "Bottom Navigation Views Activity" template in Android Studio which was the foundation of the navigation experience. The template included a bottom navigation bar that allowed users to easily switch between the Home feed, Trending fragment, and the Profile fragment. Using Android's navigation bar and action bar offered a consistent Android experience throughout the app.

Services and APIs

Cloud Firestore: To manage the tweets, FireTweet uses Cloud Firestore. Tweets are stored and managed within a collection called "tweets", with each tweet document containing the fields "id", "userName", "text", and "timestamp". This setup allows real-time updates and retrieval of tweets. Its real-time capabilities and seamless integration with the app made this the ideal database solution for FireTweet.

Firebase Authentication: FireTweet uses Firebase Authentication for user management. This allows users to create and sign into accounts using a username and password. Firebase Authentication provides built-in support for managing authentication and handling data security. The app uses authentication intents to facilitate the design of the login and sign-up experiences. These intents manage the flow of user authentication and guides the user through the process of registering or logging in.

Artificial Intelligence Use

Test Data Generation: ChatGPT was helpful to generate test data including usernames and tweets which were used for demos and testing. Having this test data in a streamlined format made it easy to copy and paste it into the app. Prompt: *Create test data. Give me 1) A famous scientist's name, 2) An email for them that concatenates their first and last initial and then adds it to @test.com (For example, Albert Einstein is 'ae@test.com'), 3) A quote from that scientist followed by 1-3 hashtags (#ExampleHashtag) related to the quote. The length of the quote + all the hashtags should be no longer than 180 characters. Do this for 4 scientists and give 5 quotes + hashtags for each one. Make some of the hashtags repeat several times throughout the quotes.*

Highlighting Hashtags in Text: I consulted ChatGPT to design the logic of highlighting the hashtags within text. It was challenging to fine-tune the function to handle edge cases, including punctuation attached to hashtags. *Prompt: "Create a function in Apex to highlight hashtags in a given string. The hashtags should be highlighted in orange, and it should ignore trailing punctation. Provide a complete function."*

Finding Trending Hashtags: Identifying the trending hashtags was another task ChatGPT was used for. It took a while to define precisely what a trending hashtag should be. The first versions of code I wrote simply counted the number of times the hashtags occurred, regardless of if it was unique to a tweet. It was helpful to consult with ChatGPT when I started figuring out edge cases to cover. *Prompt: "Write a Kotlin function for an Android app that fetches tweets from a Firestore collection, extracts hashtags from the text, and counts the occurrences of each unique hashtag to identify the top 20 trending hashtags."*

Designing Edit and Save Username Buttons: Designing the "Edit Username" and "Save Username" button on the Profile page to edit the usernames was tedious, since it involved changing the text on the buttons depending on whether the user was editing or not. I also was not certain what the UI for this feature should look like, so consulting ChatGPT gave some ideas. *Prompt: "I'm developing a user profile page for an Android app and need help designing the logic for 'Edit Username' and 'Save Username' buttons. The 'Edit' button should allow the user to modify their username, and the 'Save' button should update the user information in Firebase. Provide a possible implementation using Kotlin and XML"*

UI/UX

It was important for me to design an app that looked nice. To do so, I tried to adhere to Google's design principles and the material design color palette. By using Android's default navigation bar and action bar, the app maintains a familiar user experience across screens.

Backend Logic

The backed logic was designed for real-time interaction and data handling. Two demonstrations of this are the trending hashtags calculation and the hashtag highlighting. The trending hashtags feature fetches tweets, extracts hashtags, and counts their occurrences in real-time using Firestore. In retrospect, this is not an ideal solution for maintaining the hashtags since it involves heavy processing each time the tweet list is updated. If FireTweet were truly scalable, I would need to find another algorithm or process to determine popularity. FireTweet uses SpannableStrings to distinguish hashtags within tweets by coloring them orange. This is done in real time as the text is displayed on the screen. This solution is more efficient, since it applies the cosmetic change as the tweet is displayed on the screen.

Takeaways

This project was the first time I had the opportunity to implement a database solution in a mobile app. Although I was familiar with databases before the project, I did not have experience with solutions like Firebase (outside of the small demos/projects we had in class). It was impressive to see just how quickly I could use Firebase to create a semi-scalable app, and how most functionality was available to me by drag and dropping the google-services.json and making some small adjustments.

Challenges

Although Firebase made some things easier, it was far from perfect. It took some configuration, including updating my Gradle scripts, to get it to work properly. This was frustrating because on the surface, what seemed to be easy tasks became time consuming. There were several times when building the app that I didn't know if my Firebase configuration was wrong, if the project needed to be re-built, or if the code I wrote was simply not right. Using loggers to display error messages helped to overcome these challenges.

Launching the App

When launching the app, you will be prompted to create an account or log-in. Here are some test accounts for your convenience:

Email	Password
ae@test.com	Alex12345
in@test.com	Alex12345
mc@test.com	Alex12345
nt@test.com	Alex12345

Firebase Schema

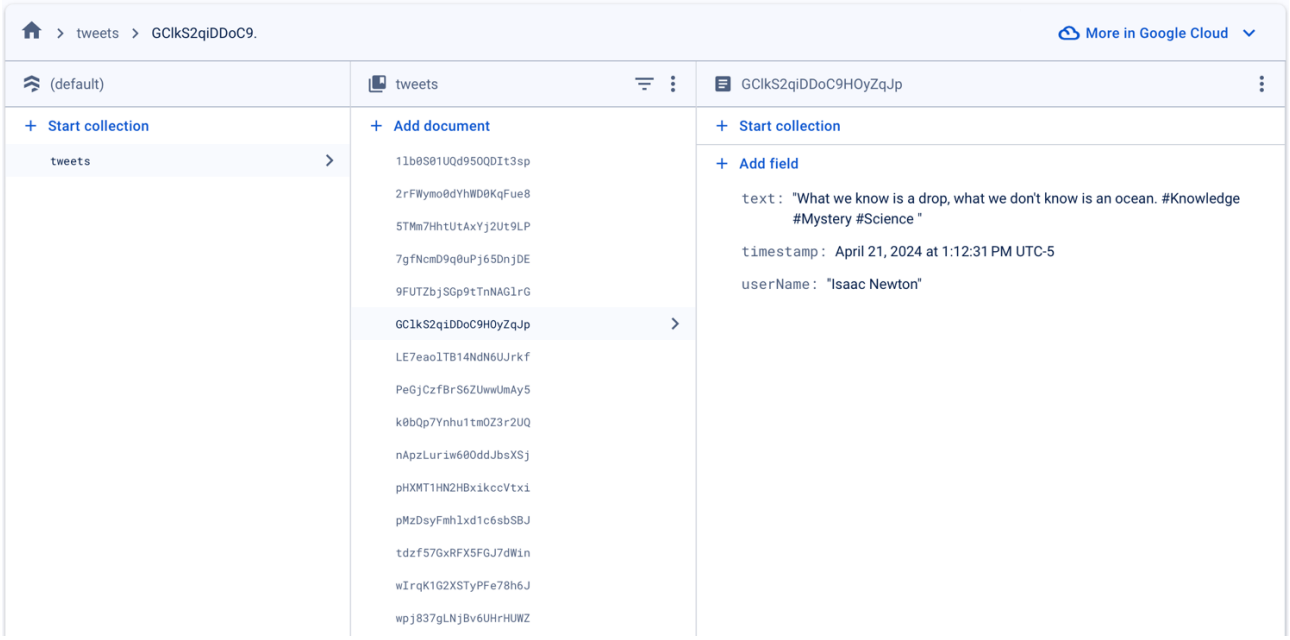

Figure 6: Cloud Firestore

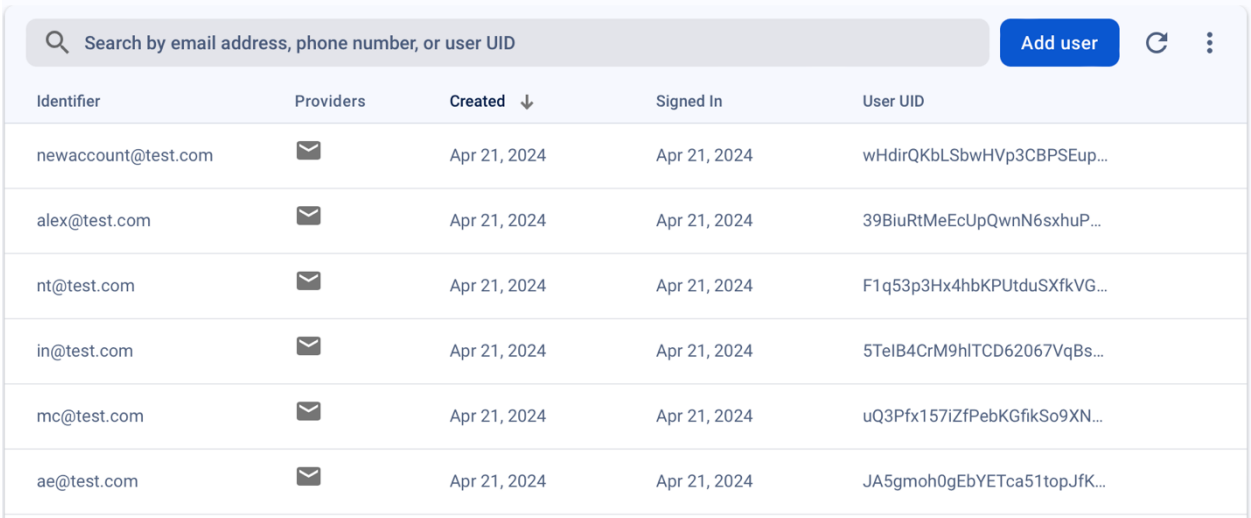

Figure 7: Firebase Authentication

Lines of Code

```
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
XML                             26             41             35            605
Kotlin                           9             63              2            563
-------------------------------------------------------------------------------
SUM:                            35            104             37           1168
-------------------------------------------------------------------------------
```

Figure 8: CLOC Output (`cloc -by-file-by-lang app/src/main/`)