University of Arkansas – CSCE Department
Capstone II – Final Report – Spring 2019

Flare - Hyperlocal News Mobile App

Claire Bruscato, Jameson River Gillis, Abby Oberle, Donald Oliver

Abstract

Social media has become an integral part of daily life and, with it, the demand for information in real time. With censorship on the rise, there exists a need for an outlet where peers can anonymously share local events and posts. Flare is our solution to this need. Flare acts as a platform that stores posts for other users to interact with and comment on for a set amount of time. Posts are shared to the people in the author's vicinity, and the reach and lifespan of a given post is dependent upon the popularity of the post among those users. In hopes, this application can be used to create a stronger, safer, and more connected community.

1.0 Problem

The problem the team noticed and wanted to find a solution to fix was being able to connect with peers without the information getting monitored and manipulated. We also observed that sharing said information with people near us, quickly, is not as easy as one would think. We found that media platforms, created to deliver facts, are being increasingly monitored and censored.
Take YouTube for example. If a video uses certain words, it gets demonetized, removed, or not sent to all of the person's subscribers—if the subscriber has certain settings turned on. Therefore, decreasing the number of people who have access to those videos.
Because we all need to be aware of the news around us in order to be active members of society, it is important to fix this problem. We need all of the information there is available to form our own conclusions. Having this information stored in a single place will make it more convenient to follow current events, get more involved in the community through fun events and local businesses, all while staying safe. Not having this information stored at our fingertips on one app, can lead to uninformed members of society, an unsafe environment, and ultimately a disconnected community.

2.0 Objective

The objective of our app was to have all the information in a community stored in one single place. We planned to display all the information while keeping the more popular posts at the top of the feed; less prevalent posts would remain at the bottom. Having the popular posts at the top would make viewing important information easier while keeping all the posts visible. Our users would be able to read as much information as they chose. Our app would be designed to let users know about fun events in their community, local business openings, and dangerous situations. We also wanted a user to be able to learn about news in their community to enable them to make informed decisions.

3.0 Background

3.1 Key Concepts

The key technologies utilized in the creation of this application include: React Native, smartphone location context, Google Firebase, and Redux.

React Native is a JavaScript framework used to write real, natively rendering mobile applications for iOS and Android.This framework targets mobile platforms while using the React language Facebook's JavaScript library for building user interfaces. Developers are able to easily create applications for both iOS and Android in tandem. This framework renders using real mobile UI components, not webviews, and looks and feels like any other native mobile application. React Native applications can access platform features, such as the user's location [1].

The user's current geolocation, in coordinates, is accessed via the device's context in React Native, and is used for database queries (described further in the design section). The location is constantly updated as the user moves.

Firebase is a backend-as-a-service that began as a startup and grew into a next-generation app-development platform on the Google Cloud Platform. Firebase acts as our server, which eliminates the need for our team to write APIs. Firebase establishes connection through a WebSocket, which is much faster than HTTP. All of our data syncs automatically through that single WebSocket as our client's network can carry it. New data is sent as soon as the new data is updated. When our client saves a change to the data, all connected clients receive the updated data almost instantly, saving into binary files sent to Google Cloud Storage. Firebase also supports user authentication, which allows us to authenticate users securely [3].

Redux is a predictable state container for JavaScript applications. Redux is beneficial in writing applications that behave consistently, run in different environments (client, server, and native), and are easy to test. Redux is used to make state mutations predictable (by imposing certain restrictions on how and when updates can happen). This allows us to make the statefulness of the application predictable and easy to manipulate at all times [2].

3.2 Related Work

Tyler Droll and Brooks Buffington, while studying at Furman University in South Carolina, created an app that allowed people in close proximity, on college campuses, to interact with other user's posts. This app was called Yik Yak. Although Yik Yak allowed users to 'flag' or dislike a post to shorten the time it was available to the public, users took an unfortunate advantage of this anonymity. Debra Katz once said, "The lack of responsiveness and inability to control this devastating hate speech I think is what ultimately did this app in." [4]. Death and bomb threats also frequented the application, causing distress and fear to all of its users. Initial measures by Yik Yak to reign in unruly users included attempting to require usernames in order to hold their users accountable for their words. Unfortunately, this initiative did little to prevent instigators, and the app was sadly shut down due to its uncontrollability. By analyzing the downfalls of a similar app, we are able to use it as a blueprint to create a more successful and well-rounded application. Flare users will have to authenticate using their personal email addresses, which can provide a way to prevent anonymous threats. Posts with hate speech or swear words will be filtered out and will not shown on the app. While Yik Yak was popular among college students, Flare will have more of a wider range of audiences. We intend for the application to benefit the public in several ways, by users having the ability to alert others of local news or messages and emergencies. It is also available for businesses to use it as a marketing and advertising platform for their company, especially small or local businesses in the community. The entire purpose of the app is both forming and pursuing the concept and connectedness of a community.

4.0 Design

4.1 Requirements and Use Cases

The requirements and use cases of this project are as follows (**items in bold are updates made since the project began**):

List of requirements:

- The application should be able to display a list of text posts from users near the current user.
  - Posts have no username attached - each post is anonymous (though the user id is tracked in the database).
  - **Posts do have a username attached, this username is created when the account is made, and can be changed at any time.**
- The application should be able to sort the list of posts in descending "importance".
  - Posts become more important as they become re-posted more and as the user moves closer to their origin point.
  - **Importance levels are 'new' (sorted by date) and 'hot' (sorted by reposts).**
- The application should allow the user to create and send a text post.
  - This post will initially be set to a low (500 meter) radius and short (24 hour) lifespan.
    - **200 meter radius and 24 hours lifespan.**
  - This post will be seen by users (via the post list) within the radius and during the post's lifespan.
- The application should allow users to repost messages from the post list.
  - Reposting a post will increase its radius by a small (200 meter) amount and increase its lifespan by a small (2 hour) amount.
    - **200 meter increase and 24 hour increase.**
  - Reposting a post will not change the post's original origin point (center).
  - Reposting a post will not create a duplicate item on the list of posts.
- The application should frequently update the list of posts while the user is viewing it.
  - **Updates are instantaneous and automatic.**
- **Users are able to comment on posts.**

List of use cases:

- A user is interested in seeing local messages. They open the app to view a list of relevant posts near them.
- A user is interested in spreading a certain message. They open the app to create a message that is sent to nearby users. If the message is popular, it will get reposted and spread to more users.
- A user is interested in reporting an emergency. They open the app to create a message that is sent to nearby people who may be affected.
- A local business is interested in advertising an event. They open the app to create the message. If users like it, they repost it to to spread details to more users.

4.2 High Level Architecture

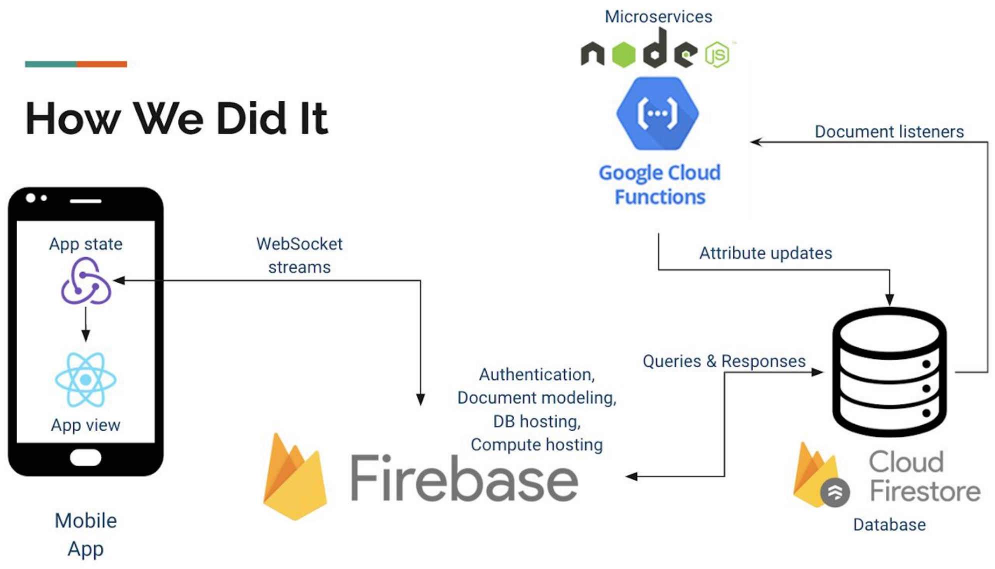
The project is based entirely in software. The main portion of the project is a mobile app for Android and iOS smartphones. In order to easily create applications for both Android and iOS in tandem, our team utilized React Native to build the app using Javascript [1]. While this has caused the app to use more memory and the be slower overall, it sped up the development time and allowed the team to implement extra features. React Native also allows access to the smartphone's context, namely the current geolocation, which is constantly streamed to the app as it updates.

Since the majority of the functions for this app take place using data from a database, our team has utilized Google Firebase to act as a backend-as-a-service to accomplish backend and database tasks without needing to set up a server [3]. We were able to use the Javascript API to continuously query for data using the format described in the graph below.

Our app needed to perform more than just reading and writing to a database on the backend, as we needed to perform atomic operations on the number of reposts in order to accomplish real-time geofencing. To accomplish this, our team utilized Google Cloud Functions to set up a small server that listens for events on Firebase and updates documents accordingly using atomic operations.

In order to handle authorization, our team utilized Firebase, which has features that make authorization easy and secure. We created sign in and sign up forms in the app, and connected those to the Firebase authorization features.

In order to make it easier to transfer data (both to the web and throughout the app), our team utilized Redux, a Javascript library that allowed us to make the statefulness of the app predictable and easy to manipulate at all times [2]. Redux also handles communications to the backend.



4.3 Risks

Risk	Risk Reduction
A user could spam the app.	The app will feature a timer so that an individual user can only make a certain number of posts per day. **Left as future work.**

The app could put personal data at risk.	The app utilizes Firebase, so that we are not personally storing any data and do not have to worry about security. Additionally, all of the data in Firebase is encrypted.
A user could post hate messages.	The app will filter out curse words. **Left as future work.**
A user could post illegal messages.	Since the user gave the service their email address and location, authorities would be able to take action against the individual.

4.4 Tasks

Bolded text indicates updates after the project began.

Many of these tasks were performed in tandem.

1. Gain a comprehensive understanding of the how the app will work at a high level (1 wk)
   - Each member of the team should be able to explain how the app will function at a user interface level.
   - Each member of the team should be able to explain how the app communicates with Firebase.
   - Each member of the team should understand the Firebase "schema"
2. Design of app user interface (**2** wks)
   - The app user interface was created using sketching tools, and distributed to the team.
3. Implementation of app user interface (**4** wks)
   - The user interface was developed in code, and updated as updates to the design were made.
   - The initial user interface used sample data.
4. Implementation of app authorization (**1 day**)
   - Authorization was added using Firebase Authorization.
5. Design and implementation of the Firebase "schema" (**1** wk)
   - Sample data was added to Firebase as the UI was developed.
6. Connect Firebase backend to user interface (**5** wks)
   - The UI was connected to Firebase using Redux.
   - The Firebase code was updated to use subscriptions - continuous queries.
   - Google Cloud Functions were added to extend the Firebase Geofencing capabilities.
7. Testing (**1** wk)
   - The app was tested on iOS and Android using a variety of inputs.
   - The app was tested on simulators and real devices.

8. Document everything (**1** wk)
    - The final project report, presentation, poster, and video was created. Additional documentation for how to build the app will be bundled with the source code.

4.5 Schedule

Tasks	Dates
1. Comprehensive app understanding	1/14-1/21
2. Design user interface	1/21-2/04
3. Implement app user interface	1/21-2/18
4. Implement app authorization	2/18
5. Design Firebase "schema"	2/18-2/25
6. Connect Firebase to UI	2/25-4/01
7. Test the app	4/01-4/08
8. Document everything	4/08-4/22

4.6 Deliverables

- Design Document: An overview of each software component of the project. Includes timelines for the expected time to complete each task.
- Sample data and data "schema" for Firebase: Firebase does not have Schemas like a traditional database, but sample data will be included (which will look similar to a schema) and it will be ensured that all future data look like the sample data.
    - Along with a diagram of the Firebase data.
- Javascript code for the entire app front-end.
    - React Native uses JSX, which is a subset of Javascript. All of the code we write should be EcmaScript 6 compliant JavaScript.
        - No native Android or iOS code was used.
    - Any Cloud Functions written will be submitted as well, no matter what language is used.
    - The source code will be seperated in a model/view/controller setup (for React and Redux, we would use action/component/reducer).
    - The source code will be managed using Git.
- Final Report for the project (this report).
- Final deliverable

- This will be a zip file containing everything related to the project (all of the above), and will be sent to the instructor.

5.0 Key Personnel

Claire Bruscato – Bruscato is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. She has completed relevant courses (Computer Science required courses as well as a graduate level database class). She was responsible for the design and implementation of the Firebase "schema" as well as connecting the Firebase backend to user interface.

River Gillis – Gillis is a senior Computer Engineering major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses (Cmp. Engineering required courses, Database Management Systems). He has also completed two software engineering internships at Google and is working on a React app for the Department of Education Reform. He was responsible for ensuring the team stays on track, as well as helping to develop the app architecture and implementation of Firebase and Redux code.

Abigail Oberle – Oberle is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. She has completed relevant courses (Computer Science required courses, including Software Engineering). She spent the past summer interning with Karpel Solutions in St. Louis, Missouri. She was be responsible for the implementation of OAuth and Google Maps' APIs.

Donald Oliver – Donald is a senior Computer Science major in the Computer Science and Computer Engineering Department at the University of Arkansas. He has completed relevant courses (Computer Science required courses, including Software Engineering). He spent the past summer as an undergraduate research assistant for Dr. Qinghua Li in the Data Security and Privacy lab, and is currently an Applications Development intern for JB Hunt in Lowell, AR. He was be responsible for the design and implementation of the application's front-end/UI in JSX and assisted in development concerning Cloud functions and connecting the UI to the tech stack

6.0 Facilities and Equipment

No facilities or equipment were utilized for this project.

7.0 References

[1] React Native Framework, https://facebook.github.io/react-native/

[2] Redux Library, https://redux.js.org/

[3] Google Firebase, https://firebase.google.com/

[4] New York Times, https://www.nytimes.com/2017/05/27/style/yik-yak-bullying-mary-washington.html/