Gururaj Shriram

CSC 411: Project Implementation

Advisor: Dr. Geoff Sutcliffe

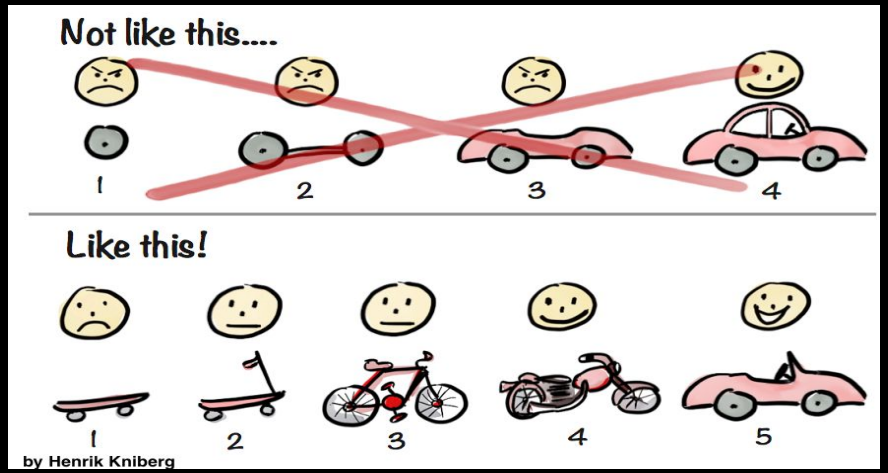
Project Overview

Motivation

- Knowledge is easily accessible and available via the Internet...
- ... But not ALL knowledge exists online
- Local knowledge can differ from region to region
- How can we share local knowledge?

Goals

- Share local knowledge with an Android app, Q&A!
  - Allow users to post questions and receive answers from users in the same location in real time
- Go from idea -> design -> implementation -> Minimum Viable Product!

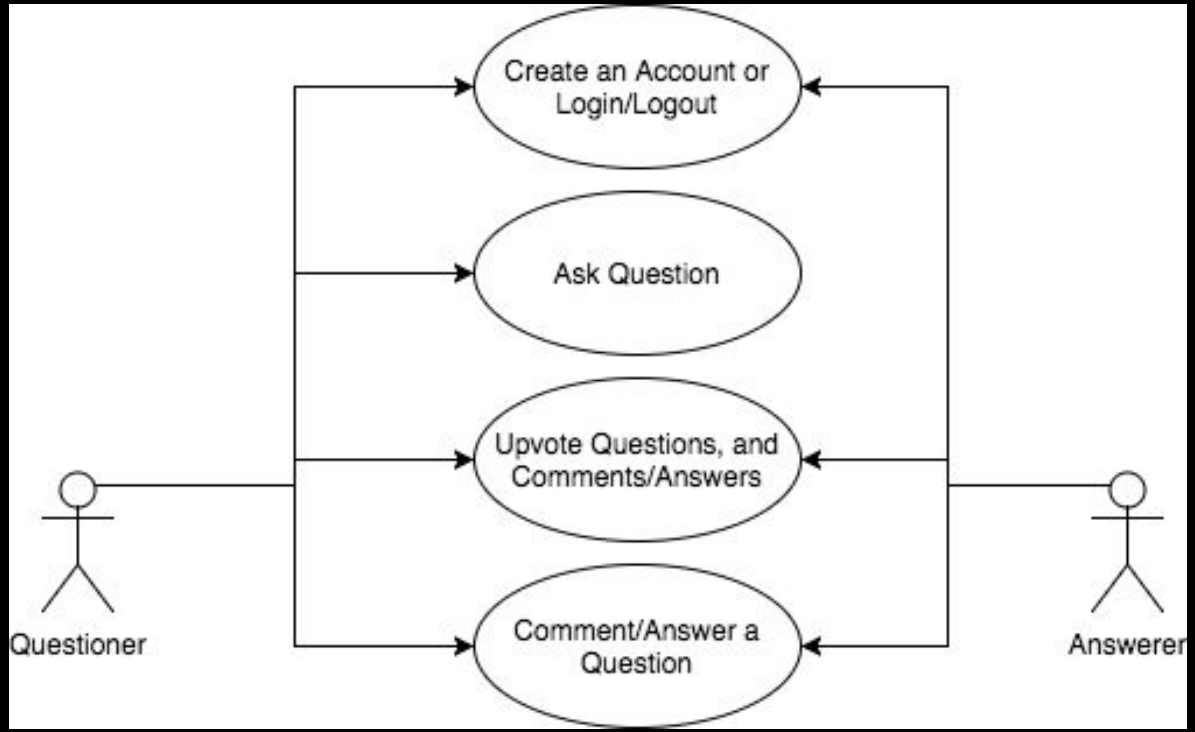


Technologies

- Android libraries
- Java for app logic
- XML for UI
- Firebase for Mobile Backend as a Service (MBaaS)
- GeoFire for Realtime Location queries (no logo :( )
- Node.js for push notifications

Design

Use Cases

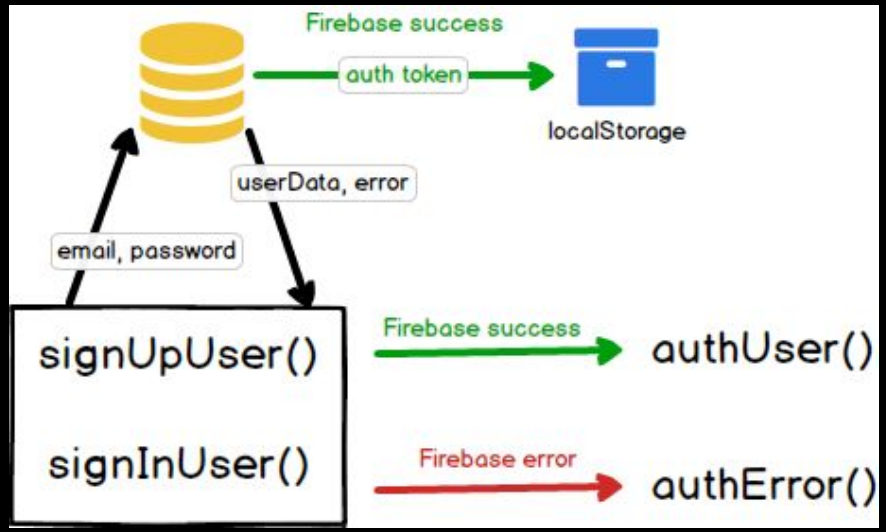

Create an Account or Login/Logout
Ask Question
Upvote Questions, and Comments/Answers
Comment/Answer a Question
Questioner
Answerer

Authentication

- Firebase Authentication provides the backend services and UI libraries (Firebase UI)
  - OAuth 2.0 which uses transport layer security for added confidentiality
  - OpenID Connect which is an identity layer on top of the OAuth 2.0 protocol
  - Auth with Email-Password or Google single sign on

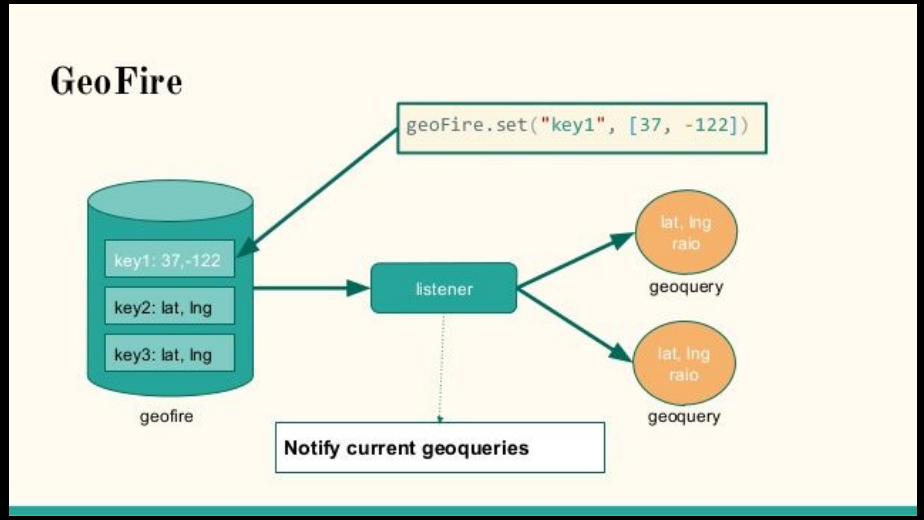
Authentication

- Sign in
  - Email-Password or OAuth token sent to Firebase Auth SDK
  - If the user does not have an account, Firebase UI prompts to create one
  - If login is successful, Q&A can access basic user info
  - If login is unsuccessful, Firebase UI redirects back to the login screen



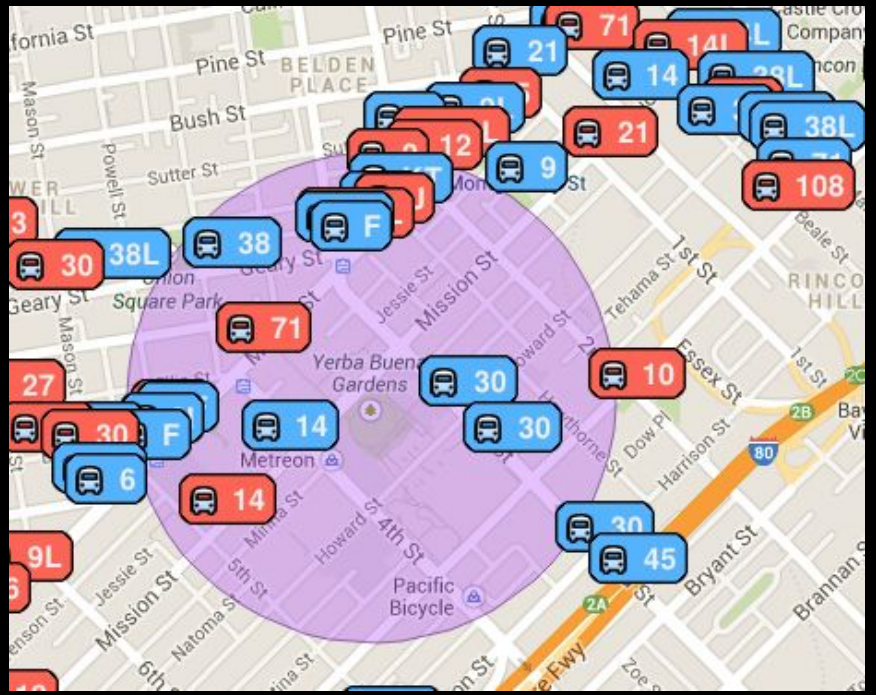
GeoFire: Recording Locations

- GeoFire stores string keys with locations and a GeoHash
- Keys = IDs of question threads
- Locations = Lat-Long pairs
- GeoHash = Hash of Lat-Long pair used to quickly check if one key is near another



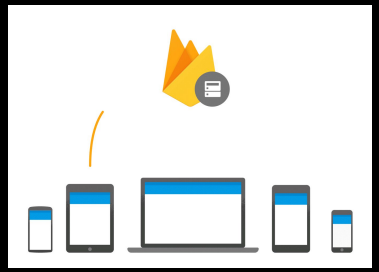
GeoFire: Querying Locations

- GeoFire can give all keys near center, c, within a radius, r.
- GeoFire creates a GeoQuery which listens asynchronously and returns all keys within the radius and when a key enters or exits the radius



Data Model and Storage

- All data stored in Firebase Realtime NoSQL Cloud Database
  - Realtime synchronization, no networking code :)
  - Data is also persisted locally, so previously loaded data will display if the app is offline
- NoSQL vs Relational Model
  - NoSQL Pros: speed, performance, scalability
  - Relational Pros: no redundancies, cheap storage, consistency (not eventual)



Data Model and Storage

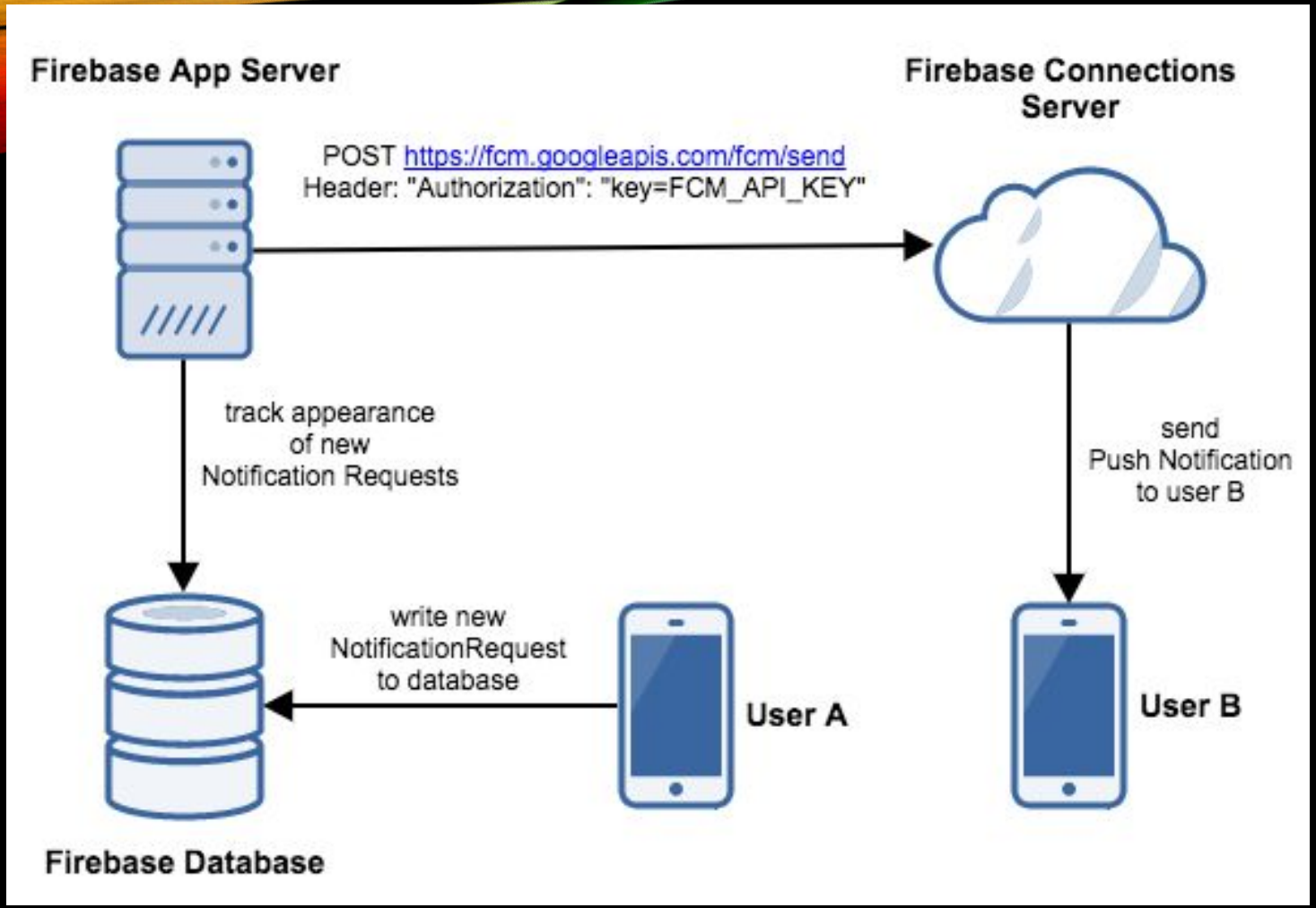
- The data is stored as JSON with 5 objects:
  - Threads: contain the metadata about each question thread including a unique ID, question title, number of likes, and timestamp
  - Posts: contain the posts of the question thread including the main question and all of the comments and answers in the thread
  - Geofire: contains the location data of a question thread based on where the user was located when he or she posted the question
  - Users: basic user data
  - FcmTokens: contain a list of Firebase Cloud Messaging tokens which are generated to send notifications to devices

Data Model and Storage

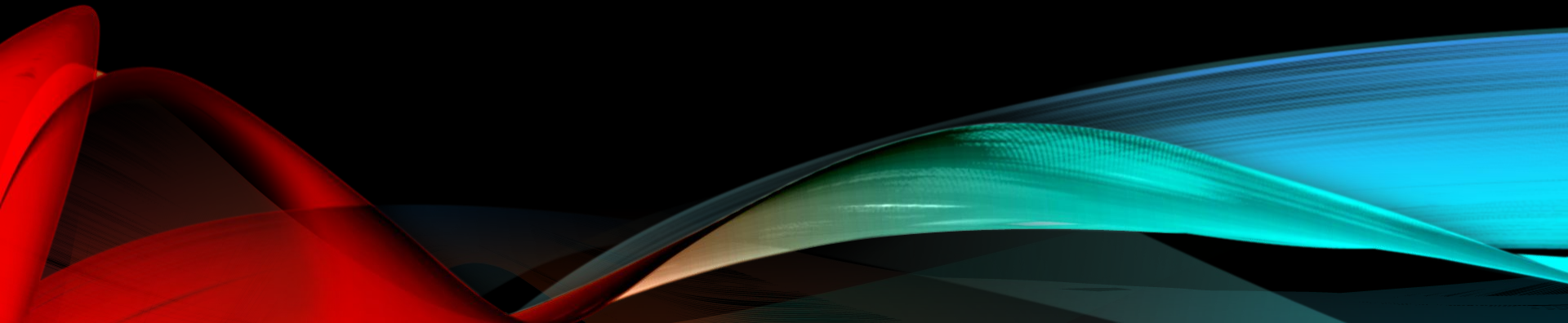
Data Model and Storage

Notifications

- Notifications are sent when a user participates in a question thread and there is some new activity in the thread.
- A Firebase Cloud Function will trigger on a write to the DB
- The function will use the Firebase Admin Node.js SDK to send a message request to the Firebase Cloud Messaging (FCM) servers
- The FCM servers send messages/notifications to the users' devices using a pre-generated FCM token


Firebase App Server
Firebase Connections Server
POST https://fcm.googleapis.com/fcm/send
Header: "Authorization": "key=FCM_API_KEY"
track appearance of new Notification Requests
send Push Notification to user B
write new NotificationRequest to database
User A
User B
Firebase Database

Demo and Retrospective



Demo

Retrospective

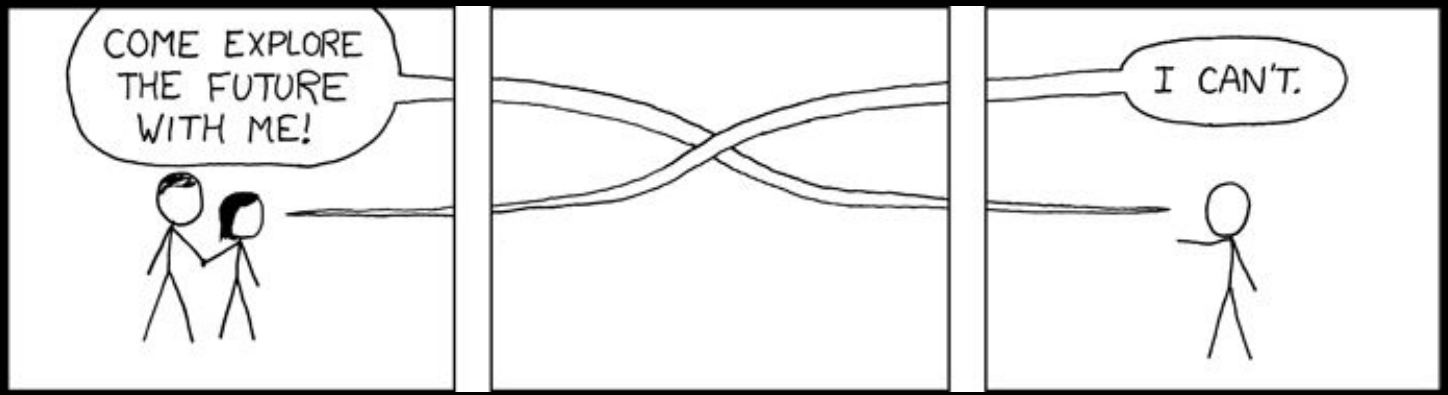
- The Good
  - Designed and built an app from scratch that works!
  - With tweaking, could be put on play store :D
- The Bad
  - Because of time complexity, several features were not implemented
  - Infrastructure for some features were created but the implementation was not done in time

Future Ideas

- UI Redesign
- Customization options
  - Search radius flexibility
  - Favorite posts and created posts
  - Sort by popularity
  - Add pictures/media to threads
- More cloud functions for optimizations
- Detect whether a question can be answerable in a search query
- User analytics and studies
- So much more!

Challenges

- Asynchronous problems
- Event handling
- Some lack of support/examples for certain libraries



References

- [1] Google.: Firebase Documentation. firebase.google.com/docs/. (2017)
- [2] Google.: GeoFire for Java - Realtime location queries with Firebase. github.com/firebase/geofire-java/. (2017)
- [3] Udacity.: Firebase in a Weekend: Android by Google. www.udacity.com/course/firebase-in-a-weekend-by-google-android--ud0352/. (2016)
- [4] Caio Ariede.:React Native: Developing an app similar to Uber in JavaScript.
- https://www.slideshare.net/caio.ariede/react-native-developing-an-app-similar-to-uber-in-javascript/. (2017)
- [5] Ihor Vitruk.: Firebase Cloud Messaging for Push Notifications. TechMagic, http://blog.techmagic.co/firebase-cloud-messaging-for-push-notifications/. (2016)


Q&A