Independent Study Final Presentation

Ethan Perry

Agenda

Intro

Learning

App Improvement

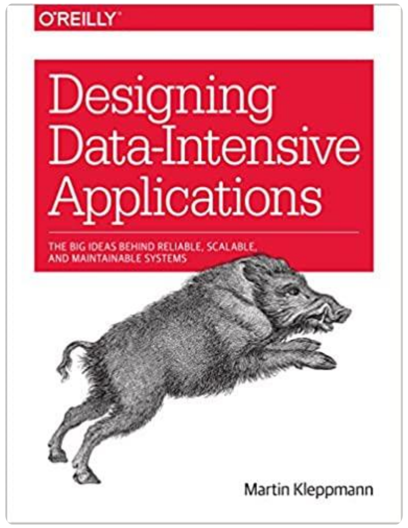
Video Demo

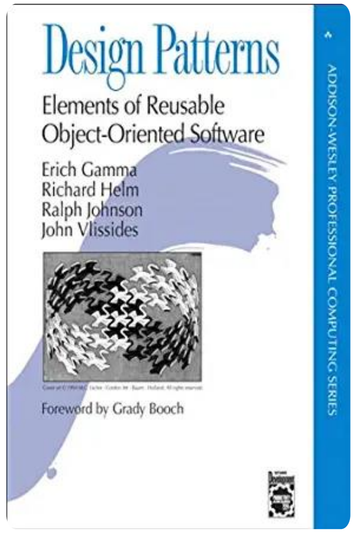
Conclusion & Future Goals

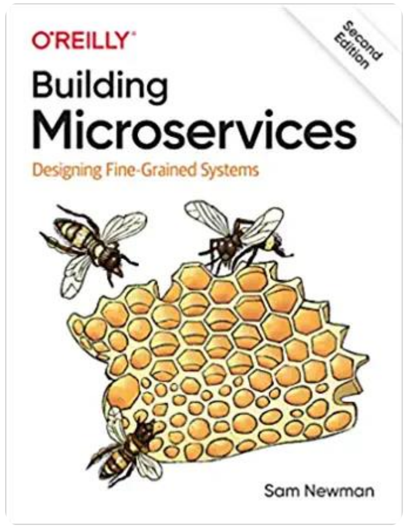
Introduction / Problem

- Three Key Project Goals:
  - Danger Dodgers app needed additional improvements to the front-end (mobile app), and back-end (cloud hosted app)
  - App needed additional documentation for developer onboarding
  - Learning new concepts!

Learning


O'REILLY
Designing Data-Intensive Applications
THE BIG IDEAS BEHIND RELIABLE, SCALABLE, AND MAINTAINABLE SYSTEMS
Martin Kleppmann


Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES


O'REILLY
Second Edition
Building Microservices
Designing Fine-Grained Systems
Sam Newman

Three Books

Importance of Books

- Designing Data Intensive Applications – Helped with rearchitecting the back-end to handle greater volumes of data
- Design Patterns – Helped with restructuring the codebase on the front-end and back-end to improve application efficiency
- Building Microservices – Helped me with redesigning the back-end in such a way that would be more scalable

Other Learning

- Became far more familiar with how to build applications on Azure
- Learned new back-end application architectures that I was unfamiliar with
- This learning has already helped me with making improvements to software as a part of my job

App Improvements

Back-end Improvements

Back-end Improvements

- Initial back-end improvements consisted of reorganizing the structure of the application
- The biggest task was the migration from the old cloud hosting platform (Heroku), over to Azure where it could be structured as a microservice backend

How does this work?

- The services which comprise the back-end are "Dockerized," meaning that they work independently of one another and can run on any environment
- Once a service is ready to be published, it is built into what is called an "image" and then pushed to Azure Container Registry (ACR)
- ACR does not run the service, but rather provides storage for its image
- Images are stored in ACR on a historical basis, meaning that if we needed to revert to an earlier version, we would have that capability

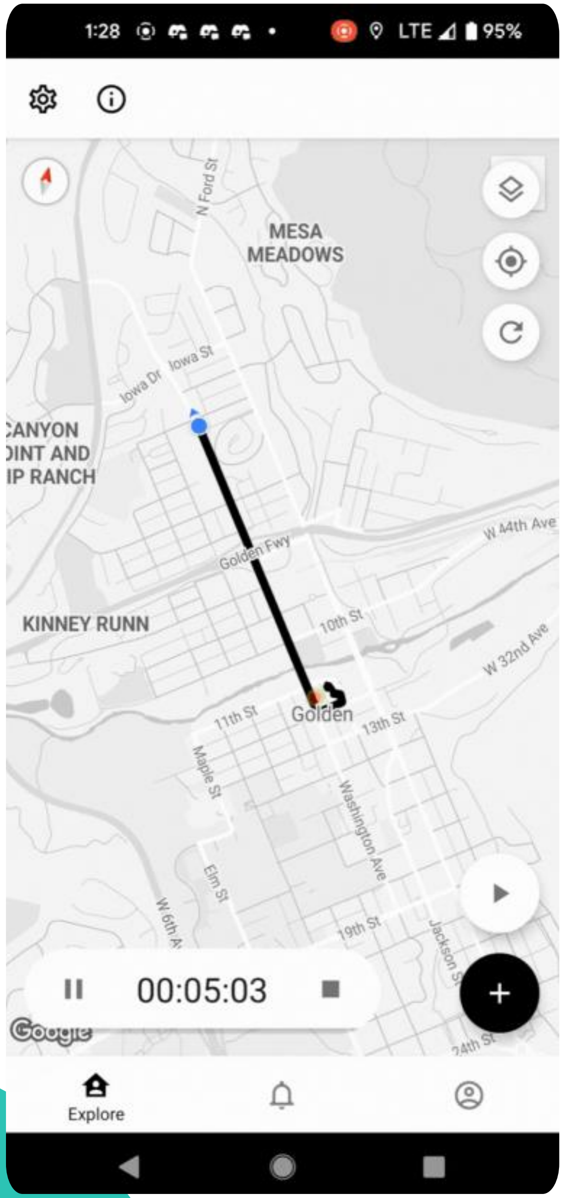
Local vs. Cloud Development

- Locally, Docker Desktop is used for development of the back-end application
- In the cloud, Azure Kubernetes Service (AKS) provides the same role, but at much larger scale and with my more processing capability

Front-end Improvements

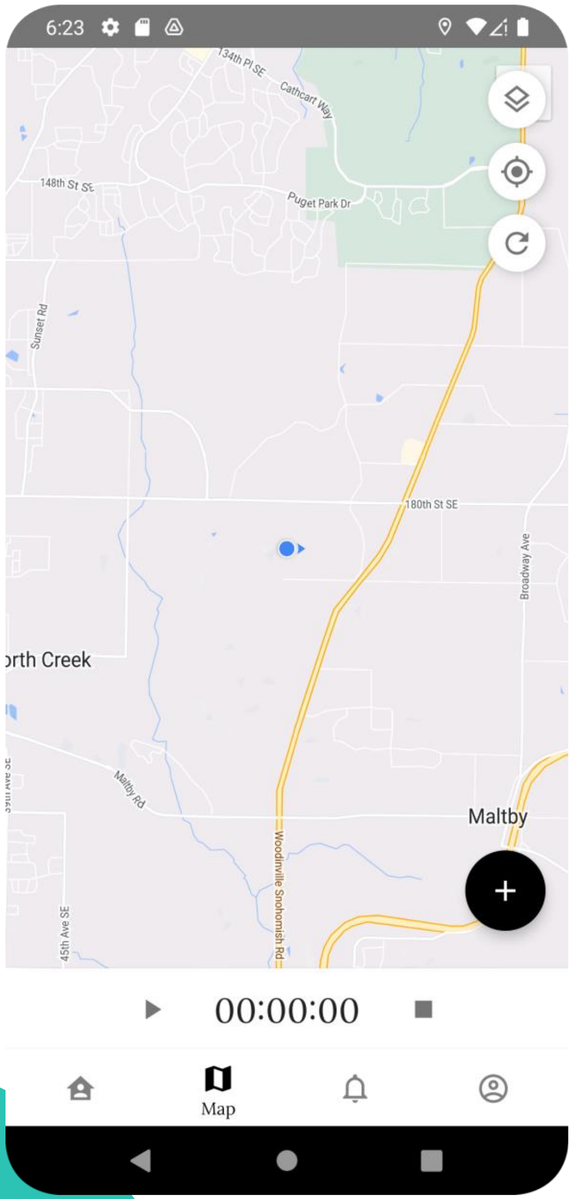
Front-End Improvements

- Initial improvements included general refactoring and organizational improvements to open the possibilities of future additions
- Better error handling was also needed throughout much of the front end



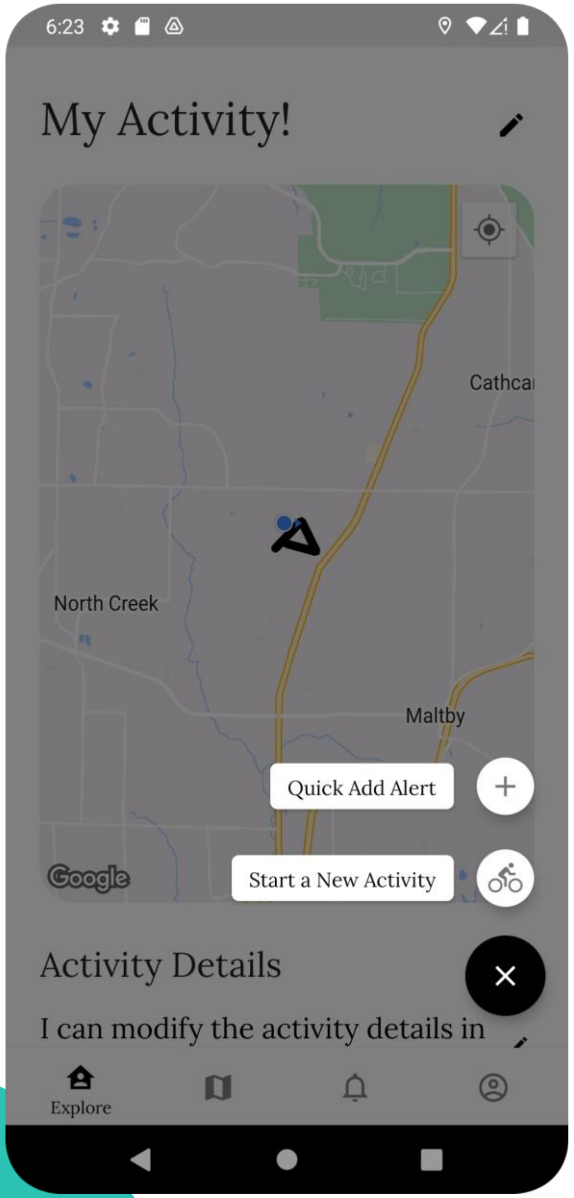
Old Map View

- Map was black and white
- Pill view looked a bit strange due to some UI incompatibilities
- Two fabrication buttons in bottom right corner, without a description of their purpose
- Settings and info bar at top of page, reducing overall view of the map



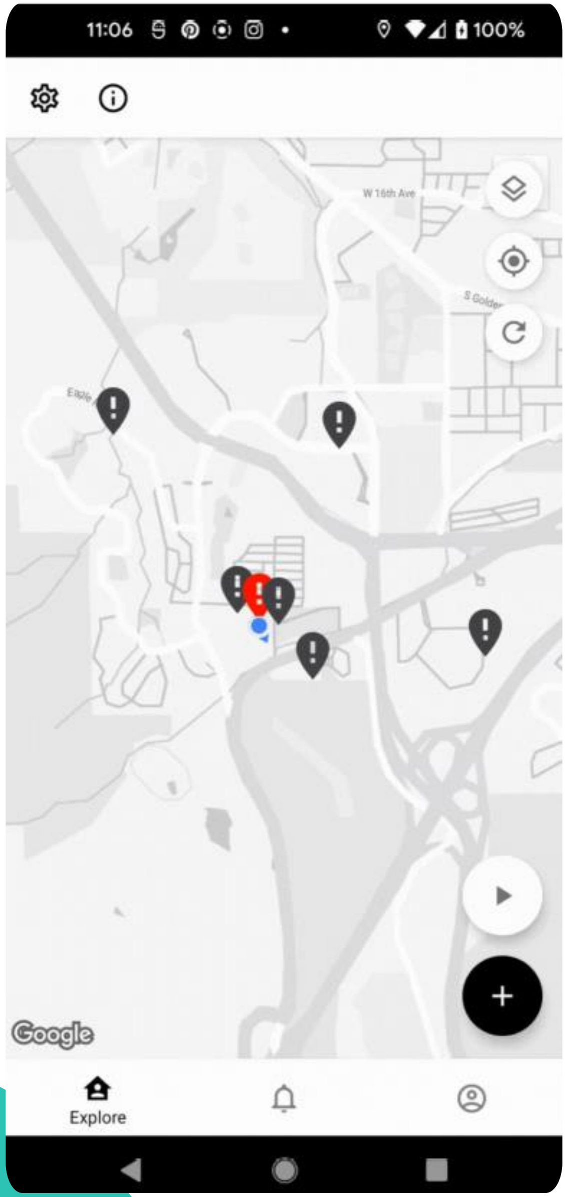
New Activity Recording UI

- Pill became a flat menu
- Went from two fab icons down to one menu
- Moved settings and info page into the account settings tab, so no more bar on the top of the screen



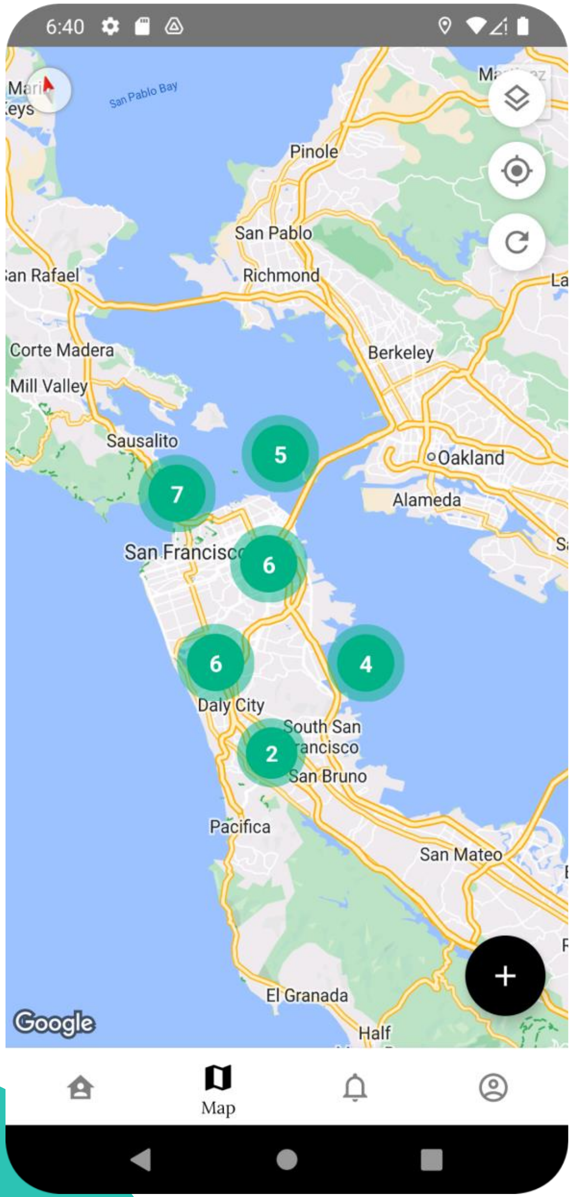
New Fab Button Interaction

- Fab button can now be clicked from any page, instead of only on the map page
- The fab menu also has “helpers” which explain what the action is for

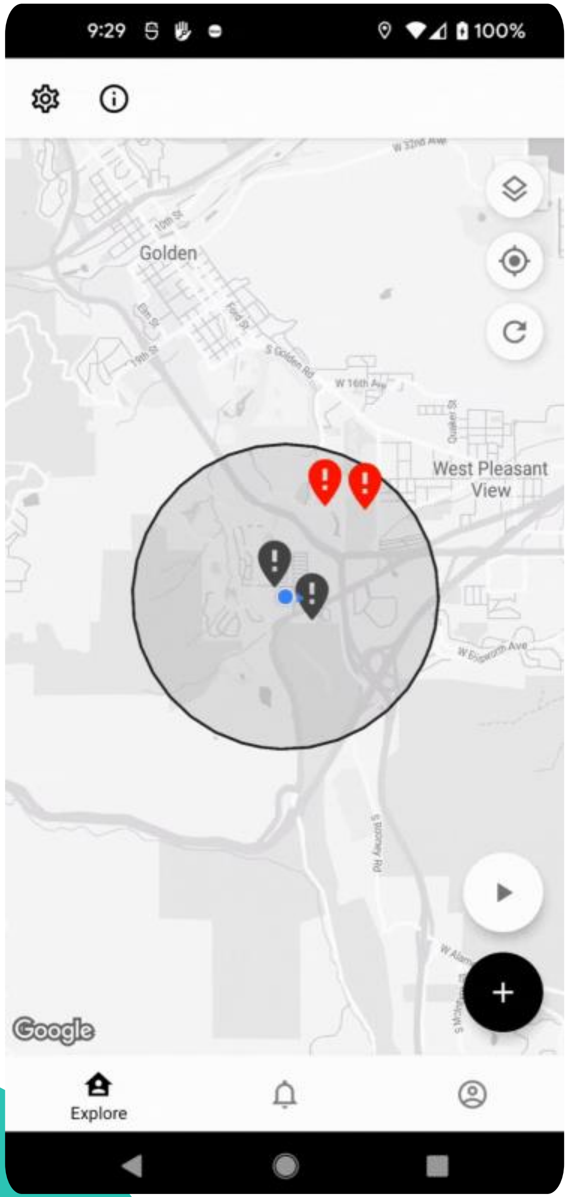


Old Alerts UI

New Clustering for Map Alerts

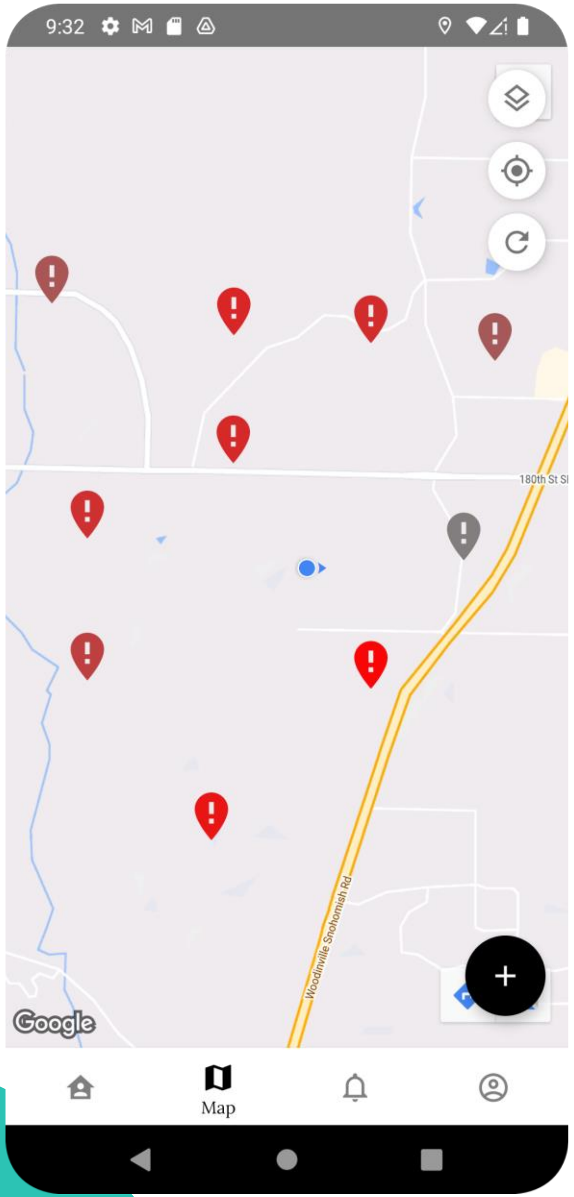


- Clusters alerts which are close in distance into a bubble displaying the count for that region
- Video demo later in this presentation



Old Alert Color Coding

- Alerts used to be red if they were created by the user, or grey if they were not

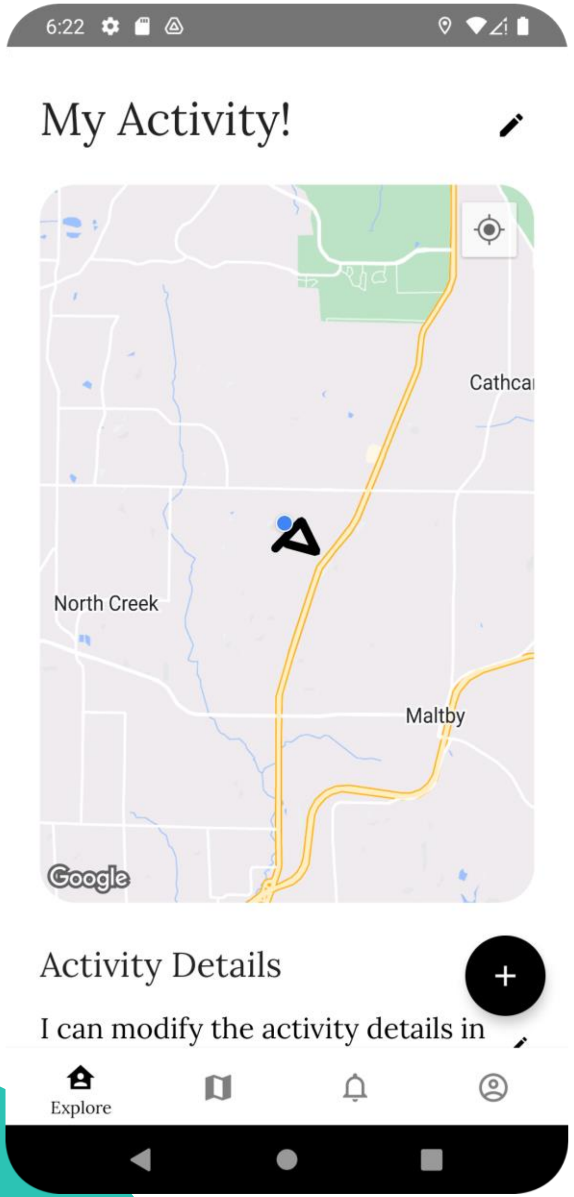


New Color-Coded Alerts

- Alerts are now color coded based on the user given danger score
- A score of zero will produce a nearly grey marker
- A score of ten will produce a bright red marker
- Everything in between follows the color gradient

New Features

New Concept – Activities

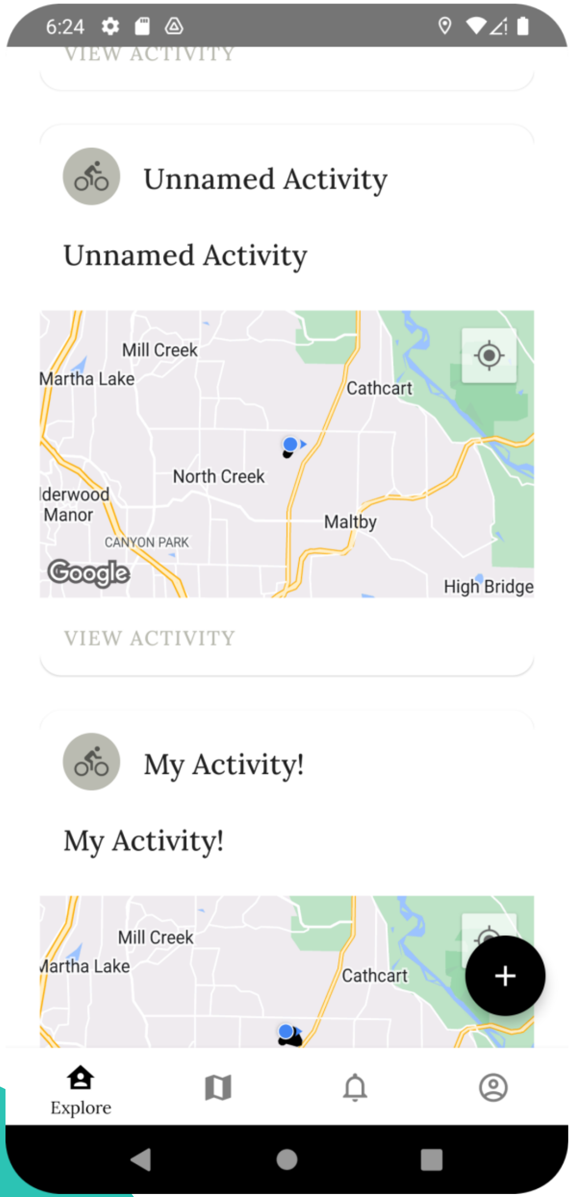


Activity Details

I can modify the activity details in

- Activities are created when a user records their ride
- It captures the user's ride, as well as the alerts which they report along their way

New Home Page – My Activities


6:24

VIEW ACTIVITY

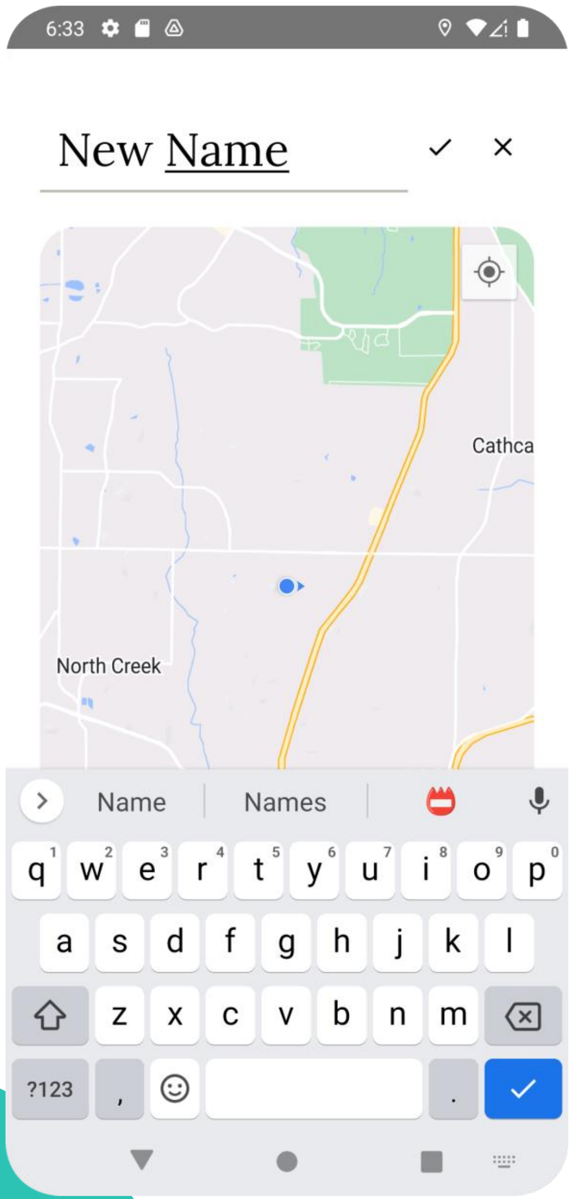
Unnamed Activity

Unnamed Activity

VIEW ACTIVITY

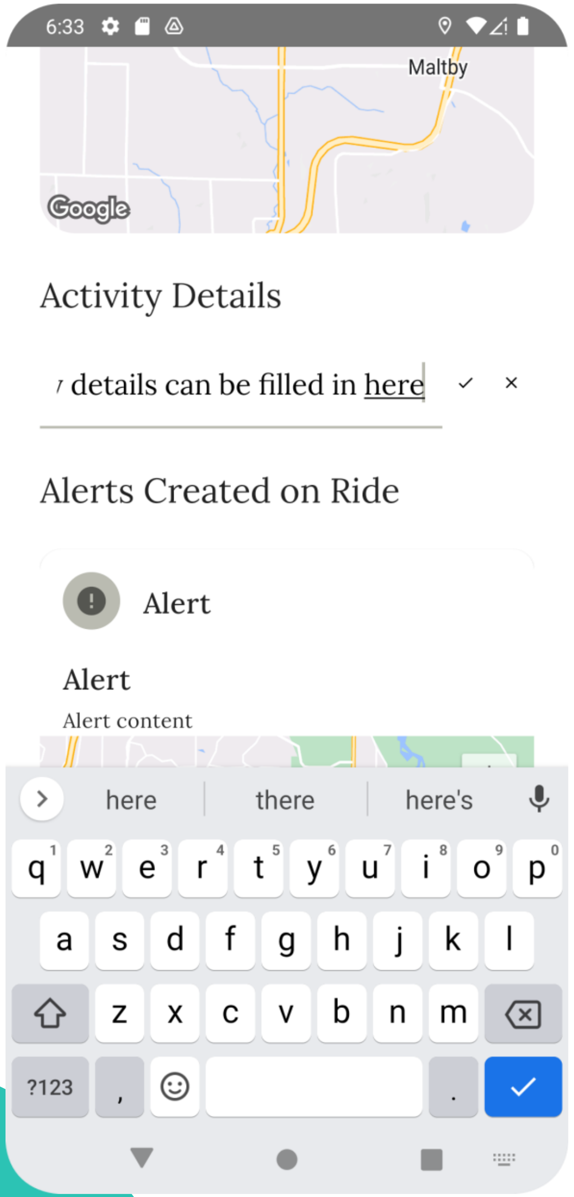
My Activity!

My Activity!



Activity Details Page

- Activity title and description can be edited without any additional menus
- A map view shows the route taken for the activity


6:33
Maltby
Google

Activity Details

details can be filled in here

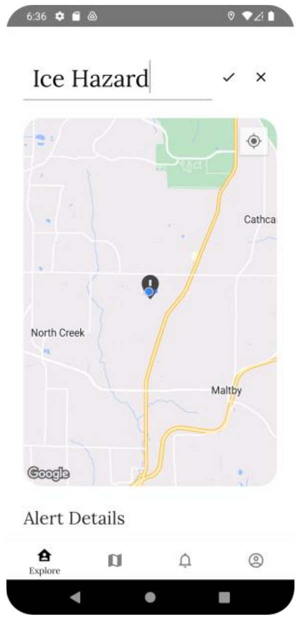
Alerts Created on Ride

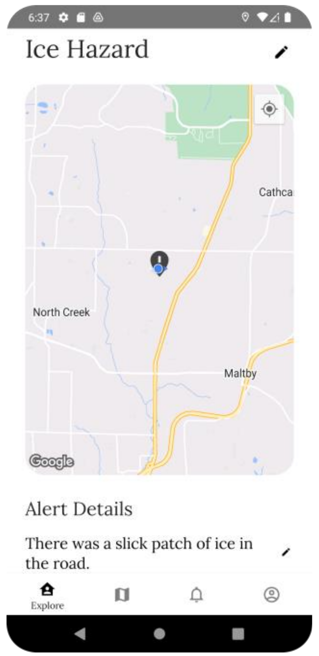
Alert

Alert

Alert content

Easily Edit Activities and Alerts





There was a slick patch of ice in the road.

Alert Details Page

New Back-End Features

- Addition of the activity feature API support
- This new feature also resulted in a change of the database storage model

Recording / Ride Storage

- Rides were not being stored prior to the new activity feature
- Rides are stored as a list of latitude and longitude points with a timestamp
- These lists are first serialized and then compressed before being stored in the database

Other General Improvements

- Combination of app settings and user settings
- New font for the app
- New styling of buttons, text, headings, and map views

Algorithm Improvements

The Old Algorithm

- The old algorithm would take the newest alerts from an area around the user's GPS location
  - The number of alerts, as well as the area of alerts displayed to the user was based on user preferences entered into the app
- The issue with the old algorithm:
  - All alerts looked to be the same in terms of their level of danger

New Algorithm

- Based on multiple metrics:
  - Time
  - Upvote/Popularity
  - Proximity to other alerts, affecting time decay
  - User assigned hazard score
    - Doesn’t affect timeout, but does change color of marker

Function of New Algorithm

- If a user is currently performing an activity, they will be able to see the 10 closest hazards to them, within a radius set in user preferences
- These hazards are color coded from red, meaning the most significant hazard which was reported the most recently, to grey, which will indicate the least significant hazards

Algorithm

- Parameters: Base Timeout (T), Maximum Distance (M), Number of Nearby Alerts (N), Distance Multiplier (D), Popularity Multiplier (P), Alert Upvotes (F)
- Search for the N closest alerts within distance M
  - If there are less than N alerts within distance M, this area is sparse, and Final Timeout = Base Timeout
- Compute the average distance from the current alert to N closest alerts
- Final Timeout = T * (Avg Distance * D) * (F * P)

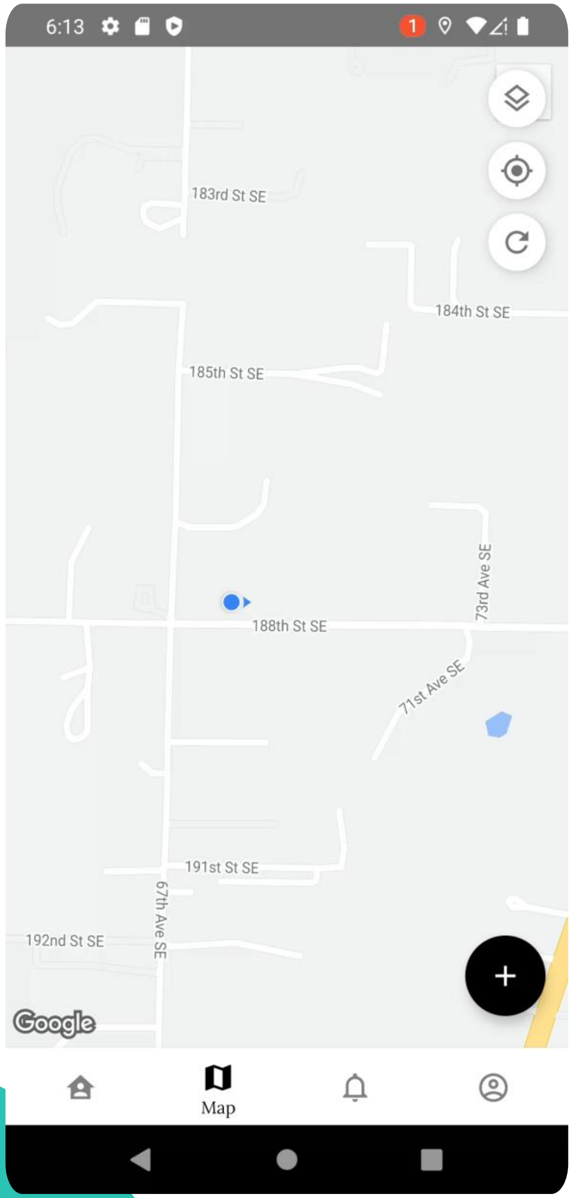
Algorithm Implications

- Greater average distance means a longer lifespan, and so does greater popularity / upvotes
- A popular alert in a sparse area will last the longest, where an unpopular alert in a crowded area will expire relatively quickly

Video Demo

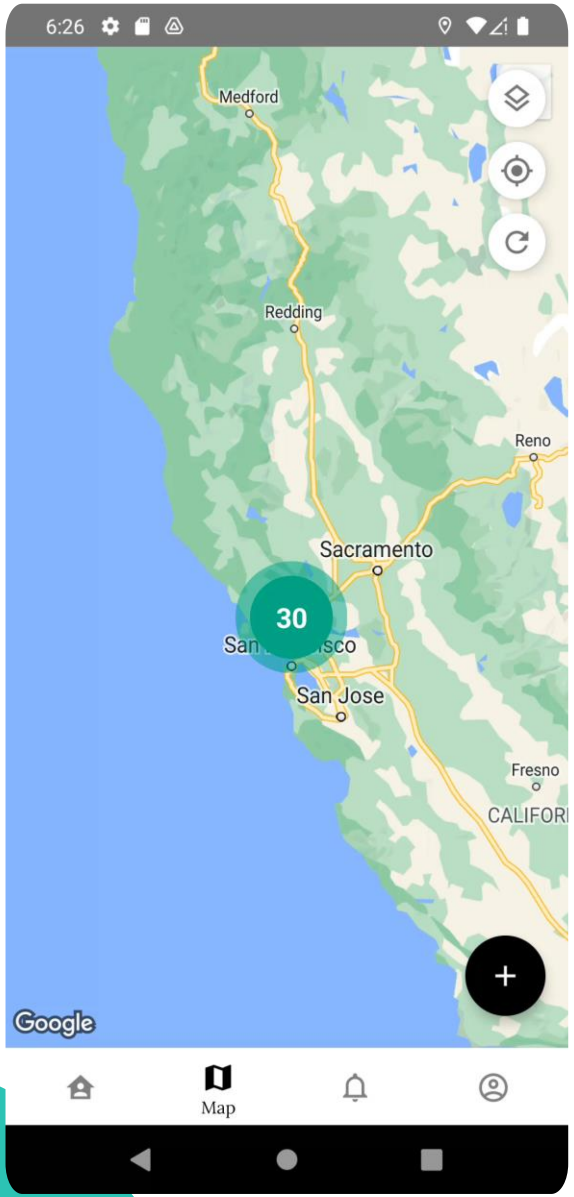
Video Demo

- First video demo features the new activity capture functionality, the core feature of the application
- Second video demo shows the map clustering functionality when the user is not currently inside



Demo #1

- First shows new ride record UI
- Next I jump to the activity page and refresh
- I can see my most recent activities in this view
- I click into an activity to see the route I took
- I can click into an activity where I made an alert to see where I created that alert



Demo #2

- This demo shows the clustering
- Random markers were generated for this example in a metropolitan area
- The clustering helps with user clarity when looking at alerts

Developer Onboarding Docs

Developer Doc Improvements

- Developer docs originally lived within each repository; they were somewhat comprehensive but needed improvement
- New developer docs live in the project website

Conclusion

Conclusion

- Wholistically, the improvements promoted the following areas of the app:
  - Usability – how the user interacts with the application and their general experience with the app
  - Scalability – our ability to replicate instances of the back-end application to serve more users in the future
  - Extensibility – any developers ability to make future improvements to the application in the future

Usability

- The application behaves more like a travel or biking app that many people are already familiar with
- More actions are labeled with text descriptions or icons, making it easier for people to understand how to use the app

Scalability

- Since our services are now containerized, we can scale “horizontally” as more users join our app
- This implies that we can adjust resources as the app grows, using only what we need and always keeping ahead of the user curve

Extensibility

- We have better documentation and organization of code, implying that future additions to the app should be easier than before

Future of the Project

Where do we go from here?

- Immediate next step is onboarding the app to the beta testing environment
- Possible suggestion:
  - Kevin, Lois, and I set up bi-weekly or monthly meetings for the fall to see the beta testing through and get this launched
  - Or if you plan to have someone else working on this project, I can spend some time getting them on-boarded with the codebase

Thank you