|  |  |  |
| --- | --- | --- |
|  |  | FIT3027: Android IOS  Assignment 1 |

30/3/2017

Mobile Application Design Sepcification

**ForkMe**

[[1]](#footnote-1)

David Lei

Student ID: 26029391

Table of Contents

1. Application Concept 4

1.1. Introduction 4

1.2. Application functionality 5

1.2.1. GitTrends (view trending repositories) 5

1.2.1.1. View the author (user/organization) 5

1.2.1.2. View a repository 6

1.2.1.3. Star a repository 6

1.2.1.4. Watch a repository 6

1.2.2. MergeMe (Find nearby developers) 6

1.2.2.1. Sharing information 6

1.2.2.2. Natural Language Processing Approaches 7

1.3. Target Audience 7

2. Idea Innovation 8

2.1. Why is this innovative? 8

2.2. Does it already exist? 8

2.3. But GitHub uses a web interface? 9

2.4. Is it useful? 10

3. Techonology Considerations 11

3.1. Android 11

3.1.1. Transferring data - Protobuffers 11

3.1.1.1. JSON 11

3.1.1.2. Protocol Buffers 12

3.1.1.3. HTTP Library – Volley 13

3.1.2. Local Storage - SQLite 13

3.1.3. Markdown (.md) render 14

3.1.4. Push Notifications 14

3.1.5. Sending Emails 14

3.2. iOS 14

3.2.1. Transferring Data – JSON 14

3.2.1.1. Transfer format 14

3.2.1.1.1. JSON 14

3.2.1.1.2. Protocol buffers 15

3.2.1.2. Web Library - Alamofire 15

3.2.1.3. Local Storage – CoreData 15

3.3. Mobile Platform Independent 15

3.3.1. GitHub API 15

3.3.2. Webserver (backend) 16

3.3.2.1. Language/Framework 16

3.3.2.1.1. Go/HTTP 17

3.3.2.1.2. JavaScript/Node.js 17

3.3.2.1.3. Python/Flask 17

3.3.2.2. Deployment/Hosting – Google App Engine 18

3.3.2.2.1. Heroku 18

3.3.2.2.2. Google App Engine/Cloud platform 18

3.3.2.3. Server Side Storage 18

3.3.2.3.1. PostgreSQL 18

3.3.2.3.2. Firebase 19

4. Interface Design Story Board Mockups 19

4.1. Splash Screen 21

4.2. Log in 21

4.3. Trending Page (GitTrends) 22

4.4. Repository View 22

4.5. User View 23

4.6. Organization View 23

4.7. Starred Repository View 24

4.8. Hamburger (side bar) 25

4.9. Find Developers (MergeMe) 26

4.10. Contact a developer 26

5. Scope and Limitations 27

5.1. Scope 27

5.2. Minimum Viable Product 28

5.3. Limitations 28

5.4. Stretch Goals 28

6. Estimated Project timeline 28

7. Testing 28

# Application Concept

## Introduction

ForkMe is a mobile application that aims to raise awareness of trending open source projects and help foster a community of innovation. It does this by having 3 key functional aspects.

1. GitTrends (primary functionality, high priority)

* The ability for a user to browse through popular open source repositories currently trending on Github[[2]](#footnote-2).
* Allowing users to star a repository (like a Facebook like/book mark)[[3]](#footnote-3).
* Allowing users to watch a repository (you will get notifications for new pull requests/issues)[[4]](#footnote-4).
* This will include communication from the mobile frontend to a webserver and animations making the user experience more enjoyable.

1. MergeMe (secondary functionality, medium priority)

* Find developers in your area.
* A ranking algorithm which will aggregate developers in your area based on common interested in language, project or need (advertising for a particular role etc).
* This aims to make it easier for people to find groups to work on open source projects/find like minded developers for hackathons etc.
* This will make use of geolocation to find other developers in the same city (only allowed if the user gives the application permission to access geo location/maps).

1. Events (extra functionality, stretch goal, low priority)

* Events can be used in conjunction with the secondary function of finding developers near by, for example: You could advertise a hackathon and have developers respond to it as going or not.
* This may or may not be implemented depending on time constraints.
* Note: This is deemed out of scope due to time constraints, but could be a good advertising platform.

## Application functionality

This section outlines a high level overview of the functionality in this application, further detail involving mockups can be found in the mockups section.

### GitTrends (view trending repositories)

This allows the user to browse trending repositories. The user can sort/filter the results by selecting the time frame to look at (i.e. trending repositories in the past day, the past week etc), the languages to include/exclude, the topic of the repository and the number of starts.

It will present the user with a view where they can interact with the repository by doing the following (visual representation of the view will be shown in the Storyboard section).

#### View the author (user/organization)

View the author’s/organization’s page including their bio, followers/following counts and repositories.

#### View a repository

A user can click on a repository to view it’s description, primary language, last commit, number of stars, number of forks and it’s contributors.

#### Star a repository

A user can star a repository on the mobile application which will result in the repository being started by their GitHub account (acts as a book mark). There will be an optional setting allowing the user to set a reminder to view this repository later, if set the application will give a notification to the user at the time set.

#### Watch a repository

A user can watch a repository which will result in the repository being watched by their Github account (will notify users of changes such as pull requests, commits, open issues). Notifications of a watched repository are outside the scope of this project as it may lead to the spamming of notifications.

### MergeMe (Find nearby developers)

This allows users to find developers in the same city as themselves. It will make use of the location data in a mobile device (GPS, wifi and mobile networks). It will be an optional feature for which users will need to grant the application permission to access location data and agree to share their information on their Github profile.

MergeMe will be an additional screen that shows near by developers, their primary language and if they are looking for a particular open source project/hackathon/event etc based on a ranking algorithm. It will only display those who opt into giving the application access to their location data and sharing their information.

#### Sharing information

Users will be able to share what they are looking for i.e. ‘Looking for a team member for GovHack Melbourne 2017’ or ‘Looking for a backend developer for my application’.

#### Ranking Algorithm

Approaches to the ranking algorithm, include using natural language processing (NLP) to pick out key words on what the user is sharing in terms of their skill set and what they are looking for and then using a classifier to group people based on geographical location and key words identified by the NLP algorithm, but this may be outside the scope of the mobile application for 1 semester.

#### Natural Language Processing Approaches

These are several approaches to Natural Language Processing investigated, the appropriate library and language will be selected after the rest of the tech stack (webserver, database, language on server side, deployment/hosting).

* Google’s cloud platform, would work nicely with hosting the webserver and database on Google cloud.
  + <https://cloud.google.com/natural-language/>
* Stanford CoreNLP suite, only supports Java.
  + <http://stanfordnlp.github.io/CoreNLP/>
* Natural Language Toolkit, only supports Python.
  + <http://www.nltk.org/>

Note: I have not covered Events as they are out of scope for the timeframe and are low priority.

#### Geoloation Aggregation Approaches

Mobile front ends will send location data to a webserver which can sift through all of it and keep buckets of developers in the same city. A table/bucket for each city could store all entries for developers which can be easily accessed and updated to get all developers in a city. This can be further broken down into City and Language tables/buckets.

## Target Audience

Due to the nature of the application requiring a user to have a GitHub account to be authenticated, the application is aimed at GitHub users which includes but is not limited to:

* Professional developers/software engineers.
* Students.
* Hobbyists.
* Amature/novice programmers.
* Anyone who is interested in open source projects.

# Idea Innovation

## Why is this innovative?

As far as I know is application doesn’t already exist, there are similar implementations but they act more as just GitHub viewers/are focused on productivity i.e. forking, editing etc.

I intend the focus on trending repositories and connecting people on open source projects to be the key defining differentiating with existing products.

Furthermore, there is already a market for users wanting to collaborate, share information, work on open source projects together and keep up to date with latest trends so there should be a market for this (see 2.1.4.).

## Does it already exist?

Several GitHub view applications already exist, however they each have their drawbacks and focus more on just viewing trending repositories/forking on mobile etc rather than on gathering people to work on projects.

* GitHub Trends[[5]](#footnote-5)
  + Cluttered UI.
  + Last update 2 years ago.
* ForkHub[[6]](#footnote-6)
  + Focused on productivity i.e. forking, looking at branches, gists, less so much open source trends and connecting people.
* TopGitHub[[7]](#footnote-7)
  + Last update 2 years ago.
  + Doesn’t look like you can select date range.
  + Nice UI but I believe it can be made more fun with animations.
* OctoDroid[[8]](#footnote-8)
  + Does a lot and kept updated, however I feel all of the features it has may over complicate the application and make users feel overwhelmed.

What my application will do differently to stand out will be:

* Crisp fun and interactive UI/UX
* Focus on trending repositories and getting more awareness for open source projects rather than productivity.
* Connecting people based on geographical location and interest/skill set.

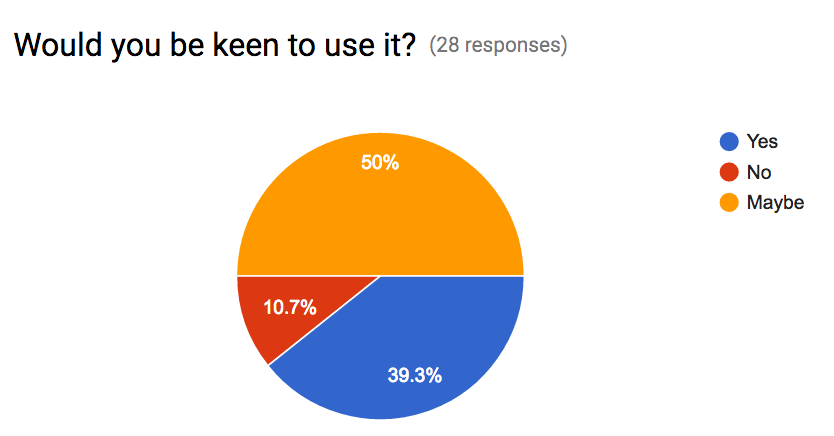
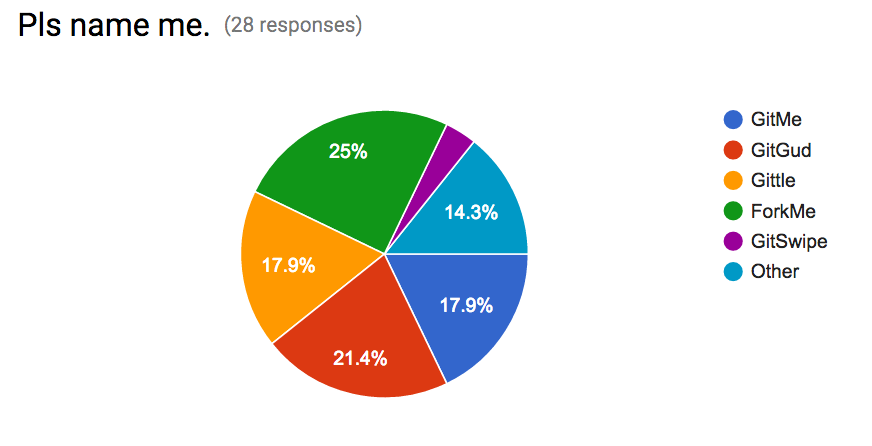
## But GitHub uses a web interface?

GitHub uses a web interface which is great at browsing repositories and reading content[[9]](#footnote-9) however after talking to fellow developers/students, not many people check the trending repositories page on a regular basis (it isn’t even on your GitHub home page). This bought about products such as GitHut to turn your chrome new tab page into a viewer for trending GitHub repositories[[10]](#footnote-10).

By using a mobile application, I aim to increase the viewer base for GitHub’s trending repositories along by maintaining the user base via the use of notifications to remind users to look at repositories they have started.

Furthermore, enabling finding other like minded developers is a completely new feature to any existing product and will work great on mobile devices as they are primarily used for messaging and contacting people.

## Is it useful?

As part of intending this application to be open source and to gauge how useful it will be I conducted a poll help me name this application and to see if there a need for it. Fellow IT students from various universities (Monash, Uni Melb, RMIT, UNSW, USYD) filled it out.

The results show that it has had a relativity positive reaction and people are inclined to use it or at least give it a try.

Furthermore, groups such as Python meet-up[[11]](#footnote-11) and Free Code Camp meet-ups[[12]](#footnote-12) already existing proving people want to be connected in working in technology, open source projects[[13]](#footnote-13) and keep up with latest trends and technologies.

# Techonology Considerations

This section outlines the technology considered for the overall functionality of the application for the Android platform, iOS platform and webserver and database backend.

The functionality that needs to be supported include:

* General
  + Transferring data
  + HTTP requests to talk to backend
  + Local storage
* GitTrends
  + Markdown (.md) render
  + Notifications
* MergeMe
  + Sending email from application to contact developer
  + Geo location

## Android

I will develop using android 7.1 Nougat API 25 and I will use android support library to versions 5.0 – 4.0 depending on time constraints.

### Transferring data - Protobuffers

Data will need to be transferred from GitHub’s API to the webserver backend and from there to the mobile front end. Below are the considerations in transferring data for Android front end clients.

I have chosen to try to user Protobuffers and will have JSON as my backup plan.

#### JSON

JSON is a commonly used data interchange format for web and mobile devices. There are many libraries to support the use of JSON and it is the most widely used.

Gson[[14]](#footnote-14) has the ablity to convert Java Objects to JSON and vice versa with very easy to use toString(), toJson() and fromJson() methods.

Furthermore, Gson seems to be more compatible[[15]](#footnote-15) [[16]](#footnote-16) with protobuffers than any other JSON library, they are both also Google developed so using Gson will give me more flexibility to choose whether to implement data transfer using JSON or Protobuffers.

#### Protocol Buffers

Protocol buffers (protobuffers) are a Google developed way to serialize structured data to a binary making it a more compressed way to transfer data when compared to JSON[[17]](#footnote-17).

They allow for compact typed data transfer where you create a schema for your message and don’t need to parse the bitstream of seralised data into human readable strings. Protobuffers are higher performance than JSON[[18]](#footnote-18).

While they are more complex than just using JSON, it is a technology I want to learn to use and are more efficient.

Square’s wire[[19]](#footnote-19) [[20]](#footnote-20)is a really nice library to for lightweight protobuffs in Android. Java is also heavily supported and has good documentation for Protobuffer use[[21]](#footnote-21).

#### HTTP Library – Volley

The two web libraries considered are Retrofit[[22]](#footnote-22) and Volley[[23]](#footnote-23), while they are both good libraries for HTTP requests, I have chosen volley as it fits with my technology stack better as Android, Protobuffers, Gson are all Google developed.

|  |  |
| --- | --- |
| **Retrofit** | **Volley** |
| * Daily or so updates[[24]](#footnote-24) * Harder to find support for Protobuffers * You can use a converter for Protobuffers[[25]](#footnote-25) (probably works well with wire) | * Google developed, because I am using Gson or Protobuffers which are also Google developed, I am more inclined to use a Google developed library for HTTP as it will probably be more compatible * Monthly or so updates[[26]](#footnote-26) * Seems to be easier to find examples to use volley with Protobuffers[[27]](#footnote-27) [[28]](#footnote-28) |

### Local Storage - SQLite

Most of the data processing will be done on the webserver side, however a SQLite[[29]](#footnote-29) database to cache data locally which can be used for notifications will be used it is natively supported and easy to work with.

The SQLite instance will store information for a notification such as reminder time, repository url, repostiroy main language, repository description, repository author/organization in the table cached\_started\_repositories.

### Markdown (.md) render – Atlassian common mark-java

Readme.md files are popular on GitHub to explain a project/repository. Markdown is an extremely simple markup language which converts into HTML[[30]](#footnote-30). Atlassian has a great Java implementation of common[[31]](#footnote-31) mark to render markdown text that is compatible with android. It also seems to have regular commits.

A good backup would be Markdown View for Android[[32]](#footnote-32) which allows you to display markdown as formatted HTML.

### Notifications

Notifications can use the phones internal time and notify users based on the time they set (i.e. remind me in 10 mins, 1 hour, 5 hours, 24 hours). Notifications can be easily built using the standard libraries[[33]](#footnote-33). Each notification will be stored in local storage.

### Sending Emails

Emails can easily be sent using intents[[34]](#footnote-34). This app will only allow sending of emails to contact developers. Responding to emails will be done on the email client of choice of the user. Instant messaging is out of scope.

### Geo location

Google MAPs API[[35]](#footnote-35) can be used to aggregate near by developers. Ideally the android application will send it’s location data to the webserver which will do the aggregation. This can be done at irregularly as the granularity of location data needed is the city of the user. Sending location data to a server[[36]](#footnote-36) will use AndroidLocationServices[[37]](#footnote-37).

## iOS

### Transferring Data – JSON

Data will need to be transferred from GitHub’s API to the webserver backend and from there to the mobile front end. Below are the considerations in transferring data for iOS front end clients.

For iOS it will be easier to use JSON as there is more support for JSON and no official support for using Protobuffers in iOS.

#### JSON

JSON seems to be the easier approach when dealing with iOS as Protobuffers are not officially supported stable.

SwiftyJSON[[38]](#footnote-38) nicely integrates with Alamofire[[39]](#footnote-39) (a popular HTTP library) and is highly recommended.

#### Protocol buffers

There doesn’t seem to be an official Google Protobuffer API for Swift. However, there are several 3rd party projects working on it. Google supports Objective-C[[40]](#footnote-40) with Protobuffers, so integrating Swift with 3rd party libraries with an iOS app seems doable but a lot of extra work.

* Apple (prerelease) <https://github.com/apple/swift-protobuf/>
* Independent (seems to be stable) <https://github.com/alexeyxo/protobuf-swift>

#### HTTP Library - Alamofire

Alamofire will be used as it abstracts onto of Apple’s network stack making it easier for the developer to use[[41]](#footnote-41), it also pairs really well with SwifyJSON.

### Local Storage – CoreData

CoreData will be used to store entities locally on an iOS device. It will store information required for notifications for stared repositories such as reminder time, repository url, repository main language, repository description, repository author/organization in the entity modle cached\_started\_repositories.

### Markdown (.md) render – Down

Apple has its own implementation of a common mark markdown render swift-cmark[[42]](#footnote-42) however it seems to have less traction and less updates than Down[[43]](#footnote-43) which allows you to export to various formats such as HTML, XML, Web View, LaTex etc.

### Notifications

Notifications can use the phones internal time and notify users based on the time they set (i.e. remind me in 10 mins, 1 hour, 5 hours, 24 hours). Notifications can be easily built using the standard libraries[[44]](#footnote-44).

### Sending Emails

Emails can send using MFMAILComposeViewControllerDelegate[[45]](#footnote-45). This app will only allow sending of emails to contact developers. Responding to emails will be done on the email client of choice of the user. Instant messaging is out of scope.

### Geo location

CLLocationManager can be used to get the phones location[[46]](#footnote-46) and then it can be sent to the webserver.

## Mobile Platform Independent

### GitHub API

GitHub current has an API[[47]](#footnote-47) out that can be used to get information on users, repositories, trends and authentication.

Ideally I want to set up a REST API on my own webserver that interfaces with GitHub’s API and pushes information to the mobile frontends.

This section outlines the ability to get the relevant information needed to prove this is viable.

* Authentication
  + GitHub OAuth for android: <https://github.com/geniushkg/github-oauth>
  + GitHut OAuth documentation: <https://developer.github.com/guides/basics-of-authentication/>
* Trending Repositories
  + GitHub search API: <https://developer.github.com/v3/search/>
  + Examples:
    - <http://stackoverflow.com/questions/30525330/how-to-get-list-of-trending-github-repositories-by-github-api>
    - https://gist.github.com/jasonrudolph/6065289
* Getting repository data
  + GitHub user documentation: <https://developer.github.com/v3/users/>
  + $ curl <https://api.github.com/users/darvid7/repos>
* Getting user data
  + GitHub repository documentation: <https://developer.github.com/v3/repos/>
  + $ curl <https://api.github.com/users/darvid7>
* Staring/Watching a repository
  + <https://developer.github.com/v3/activity/>
* Other useful links
  + <https://gist.github.com/caspyin/2288960>

### Webserver (backend)

The focus is on the application, however it will require a webserver to interact with. This section outlines the pros and cons of various languages/frameworks and deployment/hosting strategies that may be employed.

#### Language/Framework

Depending on time and workload I will choose the most appropriate language/framework later on. As this unit is focused on the Mobile Application aspect and no so much the webserver I have done a brief overview of the languages/frameworks I am consideration along with reasons why I would choose one over the other.

##### Go/HTTP[[48]](#footnote-48)

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| * Want to learn Go. * Heavily used language within Google. | * Little experience with Go. |

##### JavaScript/Node.js[[49]](#footnote-49)

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| * Really popular in industry and web development. * Have worked with Node.js and express before. * Many useful libraries. | * Not as proficient with JavaScript. |

Note: Protocol buffers[[50]](#footnote-50) (protobuffers) can be used with

* Go
  + <https://github.com/golang/protobuf>
* Python
  + <https://developers.google.com/protocol-buffers/docs/pythontutorial>
* JavaScript
  + <https://github.com/dcodeIO/ProtoBuf.js/>

However there seems to be more support for Python and Go and less for JavaScript[[51]](#footnote-51).

##### Python/Flask[[52]](#footnote-52)

|  |  |
| --- | --- |
| **Pros** | **Cons** |
| * Most proficient language. * Easy lightweight framework. | * Want to branch out and learn new languages. |

#### Deployment/Hosting – Google App Engine

I’m considering these two to host/deploy the backend based on personal preference.

##### Heroku

Can host Go, Python and Node.Js applications, it can also host PostgresSQL databases[[53]](#footnote-53)

##### Google App Engine/Cloud platform

Can host a variety of languages include Go, Python and Node along with supporting Cloud storage, Mongo DB and PostgresSQL[[54]](#footnote-54). Having use so many Google developed products I am inclined to go with google App Engine.

#### Server Side Storage

Will be populated with trending GitHub repositories on an internval i.e. every hour. Most of the data will be processed on the server side before being sent to mobile client i.e. sort all repos by language first before sending to the mobile client if the user specifies it.

Additionally the server side storage will store messages (todo)?

##### PostgreSQL

Since SQLite will be used for local storage, using a SQL database for server storage makes the most sense.

PostgresSQL[[55]](#footnote-55) is compatible with:

* Go <https://github.com/go-pg/pg>
* Python <https://wiki.postgresql.org/wiki/Python>
* JavaScript <https://github.com/brianc/node-postgres>

So it will be able to work with which ever web framework is chosen for the backend.

##### Firebase

Is another consideration is Firebase[[56]](#footnote-56) which may be useful for real time chat integration for the MergeMe functionality, however this can still be done with PostgressSQL. Adding the complexity of two databases may be more overhead compared to polling PostgressSQL at intervals. Additionally messaging can be seen as additional feature, as part of the minimum viable product the application an just allow users to see each other’s email and facilitate sending them an email.

# Interface Design Story Board Mockups

The mockups are high fidelity. The application will only support portrait view.

For Android I have tried to stick with material design.

* Simplistic minimalists interface.
* Used the colour pallet[[57]](#footnote-57) to pick colors
  + Primary Light Blue 500 #03A9F4
  + Lighter 100 #B3E5FC
  + Darker 800 #0277BD
  + Amber 500 #FFC10
* Floating action buttons (however, I use two because there are 2 main functionalities).
* Use of animations, cards and list views.
* Input directly under text label
* Text following Typography[[58]](#footnote-58) standard
  + Font: Roboto
  + Subheading 16sp
  + Body 14sp

## ANDROID%20IOS/Screen%20Shot%202017-03-29%20at%2011.03.15%20PM.pngSplash Screen

* user clicks on the image to log in
* same for Android and iOS

When clicked, GitHub icon slides up (animation), checks if the user access token, send to log in screen if needed. Else to GitTrends.

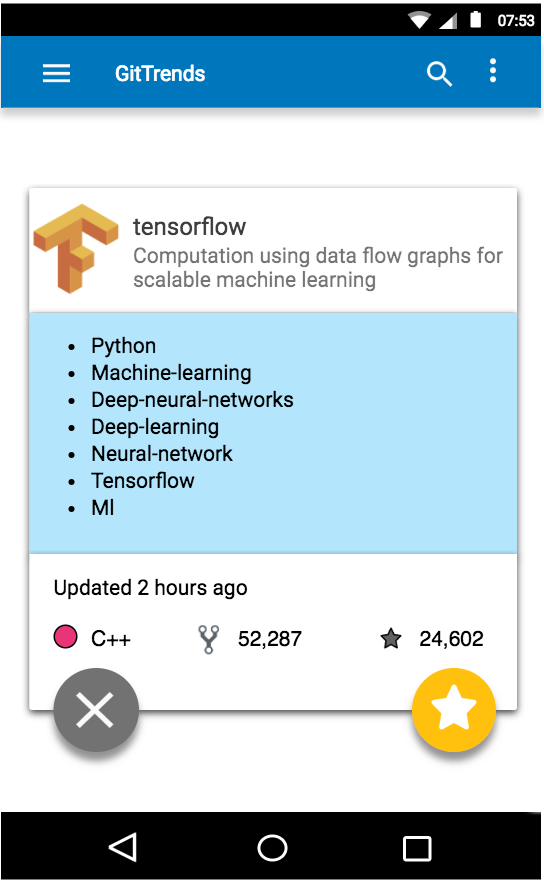
## ANDROID%20IOS/Screen%20Shot%202017-03-29%20at%2011.03.05%20PM.pngLog in

* user logs in with GitHub details
* attempts log in when user clicks out of password field

When password is correct send to Trending Page Screen.

## Trending Page (GitTrends)

* Shows a stack of cards of trending repositories



../SrcnShots/Screen%20Shot%202017-03-30%20at%2011.02.09%20PM.png

When clicked, takes you to organization view

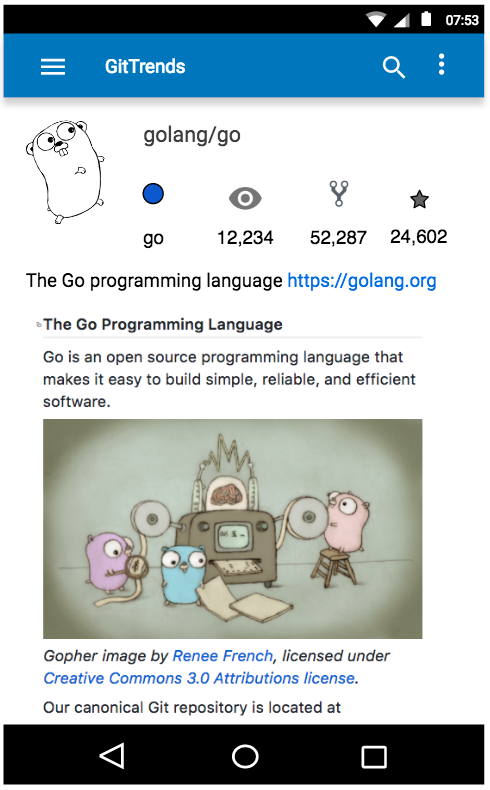
/

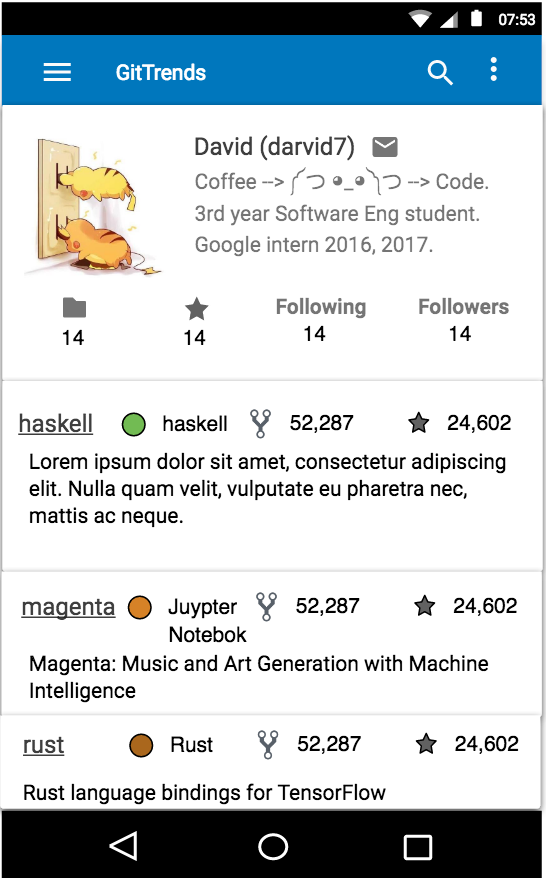
When clicked, takes you to repository view

Card spin off animation, show next card

Card spin off animation, show next card

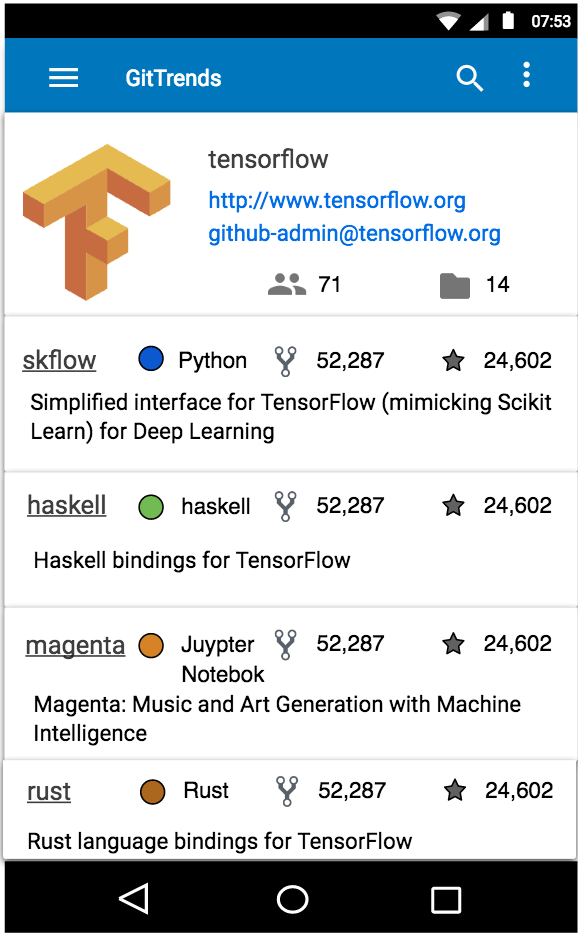
## Repository View



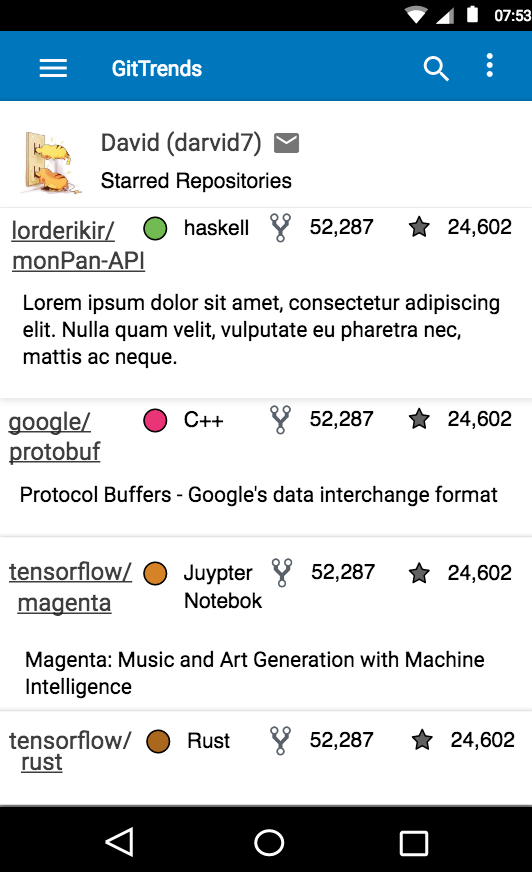


## User View

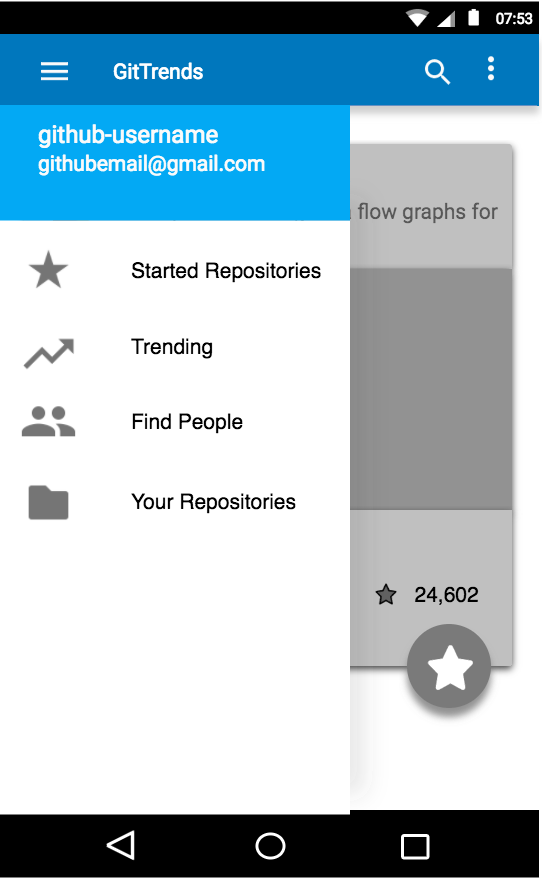
## Organization View

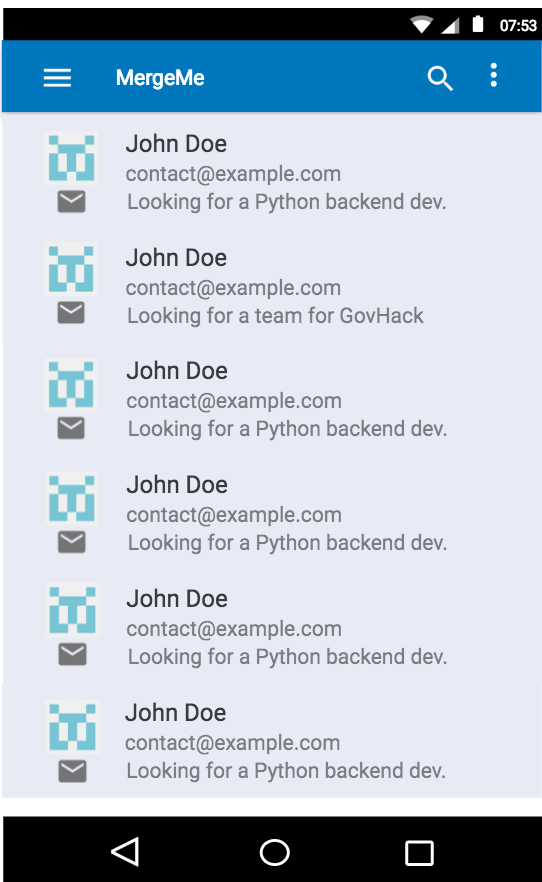


## Starred Repository View



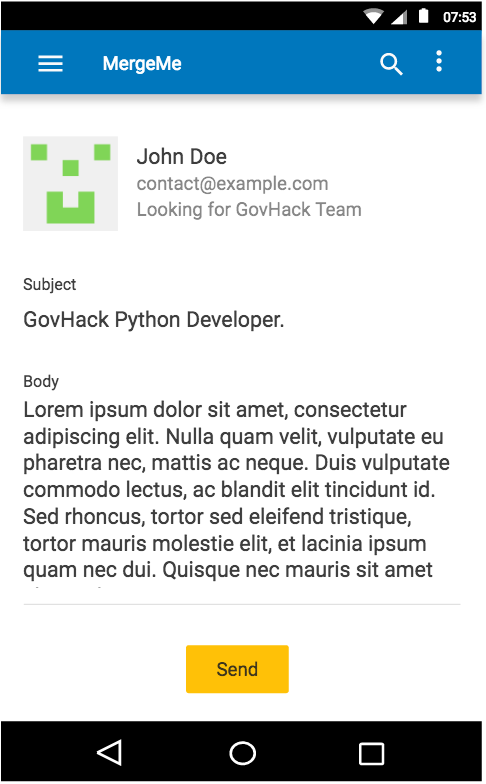
## Hamburger (side bar)





## Find Developers (MergeMe)

## Contact a developer



# Scope and Limitations

## Scope

As mentioned in the introduction, the application consists of 3 main aspects of functionality. However, due to time constraints the scope for this is limited to the primary and secondary key functionality aspects.

1. GitTrends (primary functionality)
2. MergeMe (secondary functionality)

The MVP outlines the minimum requirements for this app to be functional the limitations will discuss

## Minimum Viable Product (MVP)

The MVP consist of just GitTrends and MergeMe along with GitHub OAuth to log in.

1. GitTrends
   1. Users able to view trending repositories
   2. Users able to view other users profiles
   3. Users able to view other organizations profiles
   4. Users able to star a repository
   5. Users able to watch a respository
   6. Users able to set notifications to remind them to look at a repository later
2. MergeMe
   1. Users able to share what they are looking for
   2. Users able to see other developers
   3. Users able to send emails to other developers
   4. Users able to send location data to webserver

## Limitations

Limitations include

* Time constraints (only have around 10 weeks)
* Programming ability in Java/Swift (not my most proficient langauges)
* Natural Language Processing
  + I have no have experience with NLP, so this may be slightly out of scope for this unit as it is focused on Mobile Applications.
* Classifier
  + Out of scope for a mobile application unit and relatively hard to as I have only built binary classifiers so far.
* Ranking algorithm
  + Will need to take input from NLP algorithm, classifier and geo location and then rank developers based off that for a user’s near by developers view. Pretty complex for a 10 week mobile app.
* Instant chat
  + Not needed for this application, but would be a cool feature.

## Stretch Goals (out of initial scope)

Implementing the following are out of initial scope due to time constraints.

* Event functionality where events can be created, shared and advertised on the mobile app
* Classifier to group like minded devs by based on
  + Geolocation
  + Interests
  + Skill set
  + What they are looking for
* Natural language processing to interpret what users share regarding
  + Skill set
  + What they are looking for
  + Interests
* Chat application
* Ranking algorithm, takes in
  + output of NLP algorithm
  + output of classifier
  + geolocation

to create a list of developers who will match well with the current user.

# Estimated Project timeline

|  |  |  |
| --- | --- | --- |
| **Week** | **Tasks** | **Milestone** |
| 5 | * Define scope * Write specification * Storyboard UI * Decide on Mobile operating system to develop for | Assignment 1 |
| 6 | * Set up GitHub repositories (private until after unit). * API keys for database and hosting services. * Mobile front end populates based off local data * Navigation between screens * OAuth * Mock webserver that sends mock data. * Mobile frontend that sends and accepts geolocation data from webserver. | A working mobile app that   * Authenticates via GitHub’s OAuth * Can send and accept geolocation data from a webserver * Can populate views based off local information |
| 7 | * Server and database send good mock information (information that is expected and not just random data) |  |
| 8 | * Audio Visual processing |  |
| 9 |  |  |
| 10 |  |  |
| 11 |  |  |
| 12 |  |  |
| 13 | * User acceptance testing * Firebase Labs[[59]](#footnote-59) testing |  |
| 14 | * User acceptance testing |  |

1. image from: <https://www.lifehacker.com.au/2013/02/ask-lh-how-the-heck-do-i-use-github/> [↑](#footnote-ref-1)
2. GitHub trending repositories: <https://github.com/trending> [↑](#footnote-ref-2)
3. GitHub staring a repository: <https://help.github.com/articles/about-stars/> [↑](#footnote-ref-3)
4. GitHub watching a repository: <https://help.github.com/articles/watching-repositories/> [↑](#footnote-ref-4)
5. <https://play.google.com/store/apps/details?id=me.pmpm.githubtrends> [↑](#footnote-ref-5)
6. <https://play.google.com/store/apps/details?id=jp.forkhub> [↑](#footnote-ref-6)
7. <https://play.google.com/store/apps/details?id=com.mmazzarolo.dev.topgithub> [↑](#footnote-ref-7)
8. <https://play.google.com/store/apps/details?id=com.gh4a> [↑](#footnote-ref-8)
9. <https://github.com/blog/1559-github-s-on-your-phone> [↑](#footnote-ref-9)
10. <https://github.com/kamranahmedse/githunt> [↑](#footnote-ref-10)
11. <https://www.meetup.com/Melbourne-Python-Meetup-Group/> [↑](#footnote-ref-11)
12. <https://www.meetup.com/FreeCodeCampMelbourne/> [↑](#footnote-ref-12)
13. <https://www.meetup.com/csopensource/> [↑](#footnote-ref-13)
14. <https://github.com/google/gson> [↑](#footnote-ref-14)
15. <https://github.com/google/gson/blob/master/proto/src/main/java/com/google/gson/protobuf/ProtoTypeAdapter.java> [↑](#footnote-ref-15)
16. <https://groups.google.com/forum/#!topic/google-gson/obCKdK18quc> [↑](#footnote-ref-16)
17. <http://stackoverflow.com/questions/4979754/gson-vs-protocol-buffer> [↑](#footnote-ref-17)
18. <http://blog.codeclimate.com/blog/2014/06/05/choose-protocol-buffers/> [↑](#footnote-ref-18)
19. <https://github.com/square/wire> [↑](#footnote-ref-19)
20. <https://medium.com/square-corner-blog/introducing-wire-protocol-buffers-bb46f410e041> [↑](#footnote-ref-20)
21. <https://developers.google.com/protocol-buffers/docs/javatutorial?csw=1> [↑](#footnote-ref-21)
22. <http://square.github.io/retrofit/> [↑](#footnote-ref-22)
23. <https://developer.android.com/training/volley/index.html> [↑](#footnote-ref-23)
24. <https://github.com/square/retrofit/commits/master> [↑](#footnote-ref-24)
25. <https://futurestud.io/tutorials/retrofit-replace-the-integrated-json-converter> [↑](#footnote-ref-25)
26. <https://github.com/google/volley/commits/master> [↑](#footnote-ref-26)
27. <https://github.com/cmcneil/volley-protobufrequest/blob/master/ProtoBufRequest.java> [↑](#footnote-ref-27)
28. <http://stackoverflow.com/questions/17057649/how-to-request-protobuf-using-volley> [↑](#footnote-ref-28)
29. <https://www.sqlite.org> [↑](#footnote-ref-29)
30. <https://code.tutsplus.com/tutorials/markdown-the-ins-and-outs--net-25482> [↑](#footnote-ref-30)
31. <https://github.com/atlassian/commonmark-java> [↑](#footnote-ref-31)
32. <https://github.com/falnatsheh/MarkdownView> [↑](#footnote-ref-32)
33. <https://developer.android.com/guide/topics/ui/notifiers/notifications.html> [↑](#footnote-ref-33)
34. <http://stackoverflow.com/questions/2197741/how-can-i-send-emails-from-my-android-application> [↑](#footnote-ref-34)
35. <https://developers.google.com/maps/> [↑](#footnote-ref-35)
36. <http://stackoverflow.com/questions/21762293/run-gps-as-background-service-and-send-coordinates-to-web-server-php> [↑](#footnote-ref-36)
37. <https://developer.android.com/guide/topics/location/index.html> [↑](#footnote-ref-37)
38. <https://github.com/SwiftyJSON/SwiftyJSON> [↑](#footnote-ref-38)
39. <https://github.com/Alamofire/Alamofire> [↑](#footnote-ref-39)
40. <https://developers.google.com/protocol-buffers/docs/reference/objective-c-generated> [↑](#footnote-ref-40)
41. <https://www.quora.com/How-would-using-Alamofire-benefit-me-in-Swift> [↑](#footnote-ref-41)
42. <https://github.com/apple/swift-cmark> [↑](#footnote-ref-42)
43. <https://github.com/iwasrobbed/Down> [↑](#footnote-ref-43)
44. <http://jamesonquave.com/blog/local-notifications-in-ios-8-with-swift-part-1/> [↑](#footnote-ref-44)
45. <http://stackoverflow.com/questions/28963514/sending-email-with-swift> [↑](#footnote-ref-45)
46. <http://stackoverflow.com/questions/25296691/get-users-current-location-coordinates> [↑](#footnote-ref-46)
47. <https://developer.github.com/v3/> [↑](#footnote-ref-47)
48. <https://thenewstack.io/building-a-web-server-in-go/> [↑](#footnote-ref-48)
49. <https://nodejs.org/en/> [↑](#footnote-ref-49)
50. <https://developers.google.com/protocol-buffers/docs/reference/overview> [↑](#footnote-ref-50)
51. <http://stackoverflow.com/questions/6912981/google-protocol-buffers-javascript> [↑](#footnote-ref-51)
52. <http://flask.pocoo.org> [↑](#footnote-ref-52)
53. <https://www.heroku.com/postgres> [↑](#footnote-ref-53)
54. <https://cloud.google.com/appengine/docs/flexible/nodejs/using-third-party-databases> [↑](#footnote-ref-54)
55. <https://www.postgresql.org> [↑](#footnote-ref-55)
56. <https://firebase.google.com/> [↑](#footnote-ref-56)
57. <https://material.io/guidelines/style/color.html> [↑](#footnote-ref-57)
58. <https://material.io/guidelines/style/typography.html> [↑](#footnote-ref-58)
59. <https://firebase.google.com/docs/test-lab/> [↑](#footnote-ref-59)