

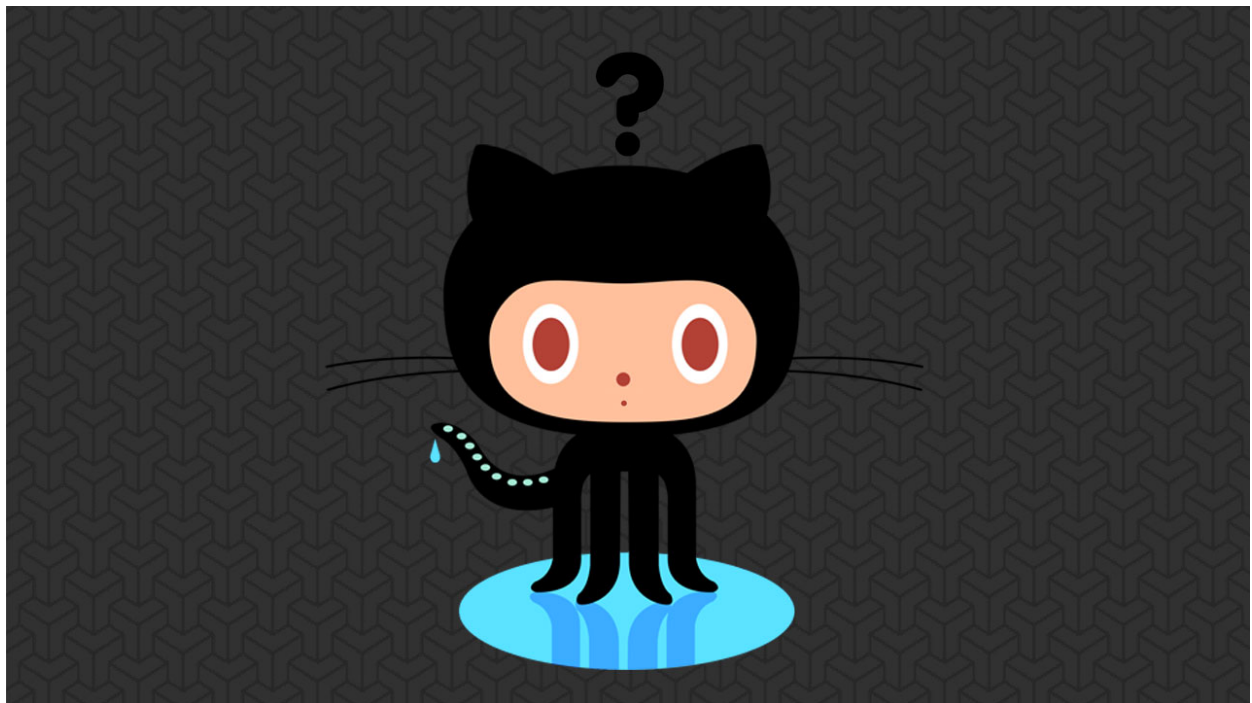
FIT3027: ANDROID IOS

Assignment 1

31/3/2017

MOBILE APPLICATION DESIGN SEPCIFICATION

ForkMe



¹

DAVID LEI

STUDENT ID: 26029391

¹ image from: <https://www.lifehacker.com.au/2013/02/ask-lh-how-the-heck-do-i-use-github/>

TABLE OF CONTENTS

1. APPLICATION CONCEPT	5
1.1. INTRODUCTION	5
1.2. APPLICATION FUNCTIONALITY	6
1.2.1. <i>GitTrends (View trending repositories)</i>	6
1.2.1.1. View the author (user/organization)	6
1.2.1.2. View a repository	6
1.2.1.3. Star a repository	6
1.2.1.4. Watch a repository	7
1.2.2. <i>MergeMe (Find nearby developers)</i>	7
1.2.2.1. Sharing information	7
1.2.2.2. Ranking Algorithm	7
1.2.2.3. Natural Language Processing Approaches	7
1.2.2.4. Geolocation Aggregation Approaches	8
1.3. TARGET AUDIENCE	8
2. IDEA INNOVATION	9
2.1. WHY IS THIS INNOVATIVE?	9
2.2. DOES IT ALREADY EXIST?	9
2.3. BUT GITHUB USES A WEB INTERFACE?	10
2.4. IS IT USEFUL?	11
3. TECHONOLOGY CONSIDERATIONS	12
3.1. ANDROID	12
3.1.1. <i>Transferring data - Protobuffers</i>	12
3.1.1.1. JSON	12
3.1.1.2. Protocol Buffers	13
3.1.1.3. HTTP Library – Volley	14
3.1.2. <i>Local Storage - SQLite</i>	14
3.1.3. <i>Markdown (.md) render – Atlassian common mark-java</i>	15
3.1.4. <i>Notifications</i>	15
3.1.5. <i>Sending Emails</i>	15
3.1.6. <i>Geolocation</i>	15
3.1.6.1. Webview	16
3.2. IOS	17
3.2.1. <i>Transferring Data – JSON</i>	17
3.2.1.1. JSON	17
3.2.1.2. Protocol buffers	17
3.2.1.3. HTTP Library - Alamofire	17
3.2.2. <i>Local Storage – CoreData</i>	18
3.2.1. <i>Markdown (.md) render – Down</i>	18

3.2.2.	<i>Notifications</i>	18
3.2.3.	<i>Sending Emails</i>	18
3.2.4.	<i>Geolocation</i>	18
3.2.5.	<i>Webview</i>	19
3.3.	MOBILE PLATFORM INDEPENDENT	19
3.3.1.	<i>GitHub API</i>	19
3.3.2.	<i>Webserver (backend)</i>	20
3.3.2.1.	Language/Framework	20
3.3.2.1.1.	Go/HTTP.....	20
3.3.2.1.2.	JavaScript/Node.js.....	20
3.3.2.1.3.	Python/Flask	21
3.3.2.2.	Deployment/Hosting – Google App Engine	21
3.3.2.2.1.	Heroku	21
3.3.2.2.2.	Google App Engine/Cloud platform	22
3.3.2.2.1.	Server Side Storage - PostgreSQL	22
3.3.2.2.2.	PostgreSQL	22
3.3.2.2.3.	Firebase.....	22
4.	INTERFACE DESIGN STORY BOARD MOCKUPS.....	23
4.1.	SPLASH SCREEN	24
4.2.	LOG IN.....	24
4.3.	TRENDING PAGE (GITTRENDS) - ANDROID.....	25
4.4.	TRENDING PAGE (GITTRENDS) - IOS	25
4.5.	STAR VIEW (SETTING A REMINDER) - ANDROID	26
4.6.	STAR VIEW (SETTING A REMINDER) - IOS	26
4.7.	REPOSITORY VIEW	27
4.8.	USER VIEW	27
4.9.	ORGANIZATION VIEW	28
4.10.	WEB VIEW	28
4.11.	STARRED REPOSITORY VIEW	29
4.12.	HAMBURGER MENU (SIDE BAR) – ANDROID SPECIFIC.....	29
4.13.	TOOLBAR – IOS SPECIFIC	30
4.14.	FIND DEVELOPERS VIEW (MERGEME).....	30
4.15.	EMAIL SCREEN (CONTACT A DEVELOPER).....	31
4.16.	NOTIFICATION	31
4.17.	SETTINGS VIEW	32
5.	SCOPE AND LIMITATIONS.....	33
5.1.	SCOPE	33
5.2.	MINIMUM VIABLE PRODUCT (MVP).....	33
5.3.	LIMITATIONS.....	34
5.4.	STRETCH GOALS (OUT OF INITIAL SCOPE).....	34

6.	PLATFORM CHOICE.....	35
7.	ESTIMATED PROJECT TIMELINE	36

1. APPLICATION CONCEPT

1.1. 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 which make use of web networking, maps and location data. Custom audio and visual processing will also be considered, although it is not part of the main functionality.

1. GitTrends (primary functionality, high priority)
 - The ability for a user to browse through popular open source repositories currently trending on GitHub².
 - Allowing users to 'star' a repository (like a Facebook like/book mark)³.
 - Allowing users to watch a repository (you will get notifications for new pull requests/issues)⁴.
 - This will include communication from the mobile frontend to a webserver.
2. MergeMe (secondary functionality, medium priority)
 - Find developers in your area.
 - A ranking algorithm which will aggregate developers in your area based on common interest 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 geolocation/maps).
3. Events (extra functionality, stretch goal, low priority)

² GitHub trending repositories: <https://github.com/trending>

³ GitHub starring a repository: <https://help.github.com/articles/about-stars/>

⁴ GitHub watching a repository: <https://help.github.com/articles/watching-repositories/>

- Events can be used in conjunction with the secondary function of finding developers nearby, 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.

1.2. 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.

1.2.1. GitTrends (View trending repositories)

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

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).

1.2.1.1. View the author (user/organization)

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

1.2.1.2. View a repository

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

1.2.1.3. Star a repository

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

1.2.1.4. 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.

1.2.2. 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, Wi-Fi 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 nearby developers, their primary language, and if they are looking for a particular open source project, hackathon, or event etc. based on a ranking algorithm. It will only display those who enabled access to their location data and sharing their information.

1.2.2.1. 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'.

1.2.2.2. Ranking Algorithm

Approaches to the ranking algorithm include using natural language processing (NLP) to pick out keywords on what the user is sharing in terms of their skillset 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.

1.2.2.3. 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.

1.2.2.4. Geolocation Aggregation Approaches

Mobile frontends 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.

1.3. 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.

2. IDEA INNOVATION

2.1. 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 or are focused on productivity, i.e. forking, editing.

The application's focus on trending repositories and connecting people on open source projects is it's key defining difference 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 2.4.).

2.2. 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 Trends⁵
 - Cluttered UI.
 - Last update 2 years ago.
- ForkHub⁶
 - Focused on productivity i.e. forking, looking at branches, gists, less so much open source trends and connecting people.
- TopGitHub⁷
 - 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.
- OctoDroid⁸

⁵ GitHub Trends: <https://play.google.com/store/apps/details?id=me.pmpm.githubtrends>

⁶ ForkHub: <https://play.google.com/store/apps/details?id=jp.forkhub>

⁷ Top GitHub: <https://play.google.com/store/apps/details?id=com.mmazzarolo.dev.topgithub>

⁸ OctoDroid: <https://play.google.com/store/apps/details?id=com.gh4a>

- 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:

- Uncluttered UI with animations
- Focus on trending repositories and getting more awareness for open source projects rather than productivity.
- Connecting people based on geographical location and interest/skillset.

2.3. But GitHub uses a web interface?

GitHub uses a web interface which is great at browsing repositories and reading content⁹. 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 brought about products such as GitHut to turn your chrome new tab page into a viewer for trending GitHub repositories¹⁰.

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 check repositories they have started.

Furthermore, enabling connection between 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 connecting with people.

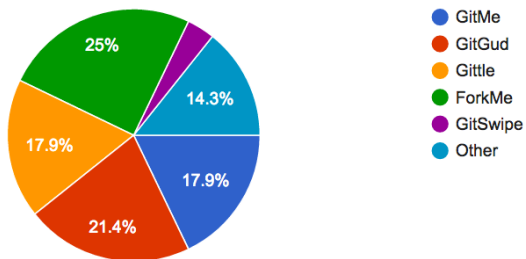
⁹ View GitHub on Mobile: <https://github.com/blog/1559-github-s-on-your-phone>

¹⁰ GitHut: <https://github.com/kamranahmedse/githunt>

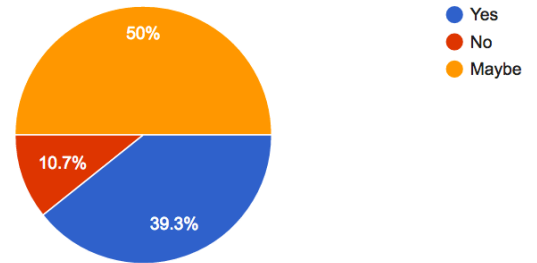
2.4. Is it useful?

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

Pls name me. (28 responses)



Would you be keen to use it? (28 responses)



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

Furthermore, groups such as Python meet-up¹¹ and Free Code Camp meet-ups¹² already existing prove that people want to be connected in working in technology, open source projects¹³ and keep up with latest trends and technologies.

¹¹ Python Meetup: <https://www.meetup.com/Melbourne-Python-Meetup-Group/>
¹² Free Code Camp Meetup: <https://www.meetup.com/FreeCodeCampMelbourne/>
¹³ Open Source Meetup: <https://www.meetup.com/csopensource/>

3. TECHNOLOGY CONSIDERATIONS

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

The functionality that needs to be supported include:

- General
 - Transferring data
 - HTTP requests to talk to backend
 - Local storage
 - Audio and visual processing
- GitTrends
 - Markdown (.md) render
 - Notifications
 - Webview for external links (e.g. to origination website)
- MergeMe
 - Sending email from application to contact developer
 - Geolocation

3.1. Android

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

3.1.1. 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.

3.1.1.1. 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.

Gson¹⁴ has the ability 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 16} with protobufs than any other JSON library, and they are both developed by Google so using Gson will give me more flexibility to choose whether to implement data transfer using JSON or Protobufs.

3.1.1.2. Protocol Buffers

Protocol buffers (protobufs) are a Google developed way to serialise structured data into binary, making it a more compressed way to transfer data when compared to JSON¹⁷.

They allow for compact typed data transfer where you create a schema for your message and don't need to parse the bitstream of serialised data into human readable strings. Protobufs have higher performance than JSON¹⁸.

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

Square's wire^{19 20} is a really nice library for lightweight protobuffs in Android. Java is also heavily supported and has good documentation for Protobuffer use²¹.

¹⁴ Gson: <https://github.com/google/gson>

¹⁵ Gson with protobuffs:

<https://github.com/google/gson/blob/master/proto/src/main/java/com/google/gson/protobuf/ProtocolAdapter.java>

¹⁶ Gson with protobuffs: <https://groups.google.com/forum/#!topic/google-gson/obCKdK18quc>

¹⁷ Gson vs protobuffs: <http://stackoverflow.com/questions/4979754/gson-vs-protocol-buffer>

¹⁸ Why choose protobuffers: <http://blog.codeclimate.com/blog/2014/06/05/choose-protocol-buffers/>

¹⁹ Wire: <https://github.com/square/wire>

²⁰ Write with protobuffs: <https://medium.com/square-corner-blog/introducing-wire-protocol-buffers-bb46f410e041>

²¹ Google Java protobuf tutorial: <https://developers.google.com/protocol-buffers/docs/javatutorial?csw=1>

3.1.1.3. HTTP Library – Volley

The two web libraries considered are Retrofit²² and Volley²³. While they are both good libraries for HTTP requests, I have chosen volley as it fits with my technology stack better since Android, Protobuffers, Gson are all Google developed.

Retrofit	Volley
<ul style="list-style-type: none">• Daily or so updates²⁴• Harder to find support for Protobuffers• You can use a converter for Protobuffers²⁵ (probably works well with wire)	<ul style="list-style-type: none">• Google developed, and since 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 updates²⁶• Seems to be easier to find examples to use volley with Protobuffers^{27 28}

3.1.2. Local Storage - SQLite

Most of the data processing will be done on the webserver side, however a SQLite²⁹ 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, repository main language, repository description, repository author/organization in the table cached_started_repositories.

²² Retrofit: <http://square.github.io/retrofit/>

²³ Volley: <https://developer.android.com/training/volley/index.html>

²⁴ Retrofit commits: <https://github.com/square/retrofit/commits/master>

²⁵ Retrofit converter: <https://futurestud.io/tutorials/retrofit-replace-the-integrated-json-converter>

²⁶ Volley commits: <https://github.com/google/volley/commits/master>

²⁷ Volley + protobuffs: [https://github.com/cmcneil/volley-](https://github.com/cmcneil/volley-protobufrequest/blob/master/ProtoBufRequest.java)

[protobufrequest/blob/master/ProtoBufRequest.java](https://github.com/cmcneil/volley-protobufrequest/blob/master/ProtoBufRequest.java)

²⁸ Volley + protobuffs: <http://stackoverflow.com/questions/17057649/how-to-request-protobuf-using-volley>

²⁹ SQLite: <https://www.sqlite.org>

3.1.3. 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 HTML³⁰. Atlassian has a great Java implementation of common³¹ 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 Android³², which allows you to display markdown as formatted HTML.

3.1.4. Notifications

Notifications can use the device's 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 libraries³³. Each notification will be stored in local storage.

3.1.5. Sending Emails

Emails can easily be sent using intents³⁴. 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.

3.1.6. Geolocation

Google MAPs API³⁵ can be used to aggregate nearby developers. The application will use AndroidLocationServices³⁶ and send its location data to the webserver which performs the aggregation. This does not need to be done often (once to get the current city the user is in and once for each city change).

³⁰ Markdown: <https://code.tutsplus.com/tutorials/markdown-the-ins-and-outs--net-25482>

³¹ Atlassian commonmark-java: <https://github.com/atlassian/commonmark-java>

³² MarkdownView: <https://github.com/falnatshah/MarkdownView>

³³ Notifications: <https://developer.android.com/guide/topics/ui/notifiers/notifications.html>

³⁴ Sending emails: <http://stackoverflow.com/questions/2197741/how-can-i-send-emails-from-my-android-application>

³⁵ Maps API: <https://developers.google.com/maps/>

³⁶ Android location: <https://developer.android.com/guide/topics/location/index.html>

3.1.6.1. Webview

Webview can be implemented using WebView³⁷, allowing the webpage to be displayed inside the application, rather than having to make an intent and then prompting the user to open it in an external web browser. It allows for the viewing of the website to be done in app, resulting in a more controlled nicer user experience compared to having to return back to the application after opening it in an external app (i.e. chrome).

³⁷ Android webview: <https://developer.android.com/reference/android/webkit/WebView.html>

3.2. iOS

3.2.1. Transferring Data – JSON

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

3.2.1.1. JSON

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

SwiftJSON³⁸ nicely integrates with Alamofire³⁹ (a popular HTTP library) and is highly recommended.

3.2.1.2. 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-C⁴⁰ 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>

3.2.1.3. HTTP Library - Alamofire

Alamofire will be used as it abstracts onto of Apple’s network stack, making it easier for the developer to use⁴¹, and it also pairs really well with SwiftJSON.

³⁸ SwiftJSON: <https://github.com/SwiftyJSON/SwiftyJSON>

³⁹ Alamofire: <https://github.com/Alamofire/Alamofire>

⁴⁰ Protobuffs in Objective-C: <https://developers.google.com/protocol-buffers/docs/reference/objective-c-generated>

⁴¹ Alamofire is easy: <https://www.quora.com/How-would-using-Alamofire-benefit-me-in-Swift>

3.2.2. Local Storage – CoreData

CoreData will be used to store entities locally on an iOS device. It will store information required for notifications for stored repositories such as reminder time, repository url, repository main language, repository description, repository author/organization in the entity model `cached_started_repositories`.

3.2.1. Markdown (.md) render – Down

Apple has its own implementation of a common mark markdown render `swift-cmark`⁴². However it seems to have less traction and less updates than `Down`⁴³ which allows you to export to various formats such as HTML, XML, Web View, LaTeX etc.

3.2.2. Notifications

Notifications can use the device's internal time and notify the user 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 libraries⁴⁴.

3.2.3. Sending Emails

Emails can send using `MFMailComposeViewControllerDelegate`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.

3.2.4. Geolocation

`CLLocationManager` can be used to get the phones location⁴⁶ and then it can be sent to the webserver.

⁴² Apple markdown render: <https://github.com/apple/swift-cmark>

⁴³ Down: <https://github.com/iwasrobbed/Down>

⁴⁴ Notifications: <http://jamesonquave.com/blog/local-notifications-in-ios-8-with-swift-part-1/>

⁴⁵ Sending emails: <http://stackoverflow.com/questions/28963514/sending-email-with-swift>

⁴⁶ Location data: <http://stackoverflow.com/questions/25296691/get-users-current-location-coordinates>

3.2.5. Webview

Can be implemented using `UIWebView`⁴⁷ to embed the website content into the application. This allows for the user to have a smoother transition without needing to transfer to another application to view the webpage.

3.3. Mobile Platform Independent

3.3.1. GitHub API

GitHub current has an API⁴⁸ 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 that this is viable.

- Authentication
 - GitHub OAuth: <https://github.com/geniushkg/github-oauth>
 - GitHub 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

⁴⁷ UI Webview: <https://developer.apple.com/reference/uikit/uiwebview>

⁴⁸ GitHub API: <https://developer.github.com/v3/>

- 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>

3.3.2. 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.

3.3.2.1. 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 not so much the webserver I have done a brief overview of the languages/frameworks I am considering, along with reasons why I would choose one over the other.

3.3.2.1.1. Go/HTTP⁴⁹

Pros	Cons
<ul style="list-style-type: none"> • Want to learn Go. • Heavily used language within Google. 	<ul style="list-style-type: none"> • Little experience with Go.

3.3.2.1.2. JavaScript/Node.js⁵⁰

Pros	Cons
<ul style="list-style-type: none"> • Really popular in the industry and web development. 	<ul style="list-style-type: none"> • Not as proficient with JavaScript.

⁴⁹ Go webserver: <https://thenewstack.io/building-a-web-server-in-go/>

⁵⁰ Node Js: <https://nodejs.org/en/>

<ul style="list-style-type: none"> • Have worked with Node.js and express before. • Many useful libraries. 	
--	--

Note: Protocol buffers⁵¹ (protobufs) 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 JavaScript⁵².

3.3.2.1.3. Python/Flask⁵³

Pros	Cons
<ul style="list-style-type: none"> • Most proficient language. • Easy lightweight framework. 	<ul style="list-style-type: none"> • Want to branch out and learn new languages.

3.3.2.2. Deployment/Hosting – Google App Engine

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

3.3.2.2.1. Heroku

Can host Go, Python and Node.js applications, it can also host PostgreSQL databases⁵⁴

⁵¹ Protobufs: <https://developers.google.com/protocol-buffers/docs/reference/overview>

⁵² Protobufs in Js: <http://stackoverflow.com/questions/6912981/google-protocol-buffers-javascript>

⁵³ Flask: <http://flask.pocoo.org>

⁵⁴ Hosting PostgreSQL on Heroku: <https://www.heroku.com/postgres>

3.3.2.2.2. Google App Engine/Cloud platform

Can host a variety of languages including Go, Python and Node along with supporting Cloud storage, Mongo DB and PostgreSQL⁵⁵. Having used many Google developed products, I am inclined to go with Google App Engine.

3.3.2.2.1. Server Side Storage - PostgreSQL

A PostgreSQL database will be populated with trending GitHub repositories on an interval i.e. every 6 hours. 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 when the user asks for it.

3.3.2.2.2. PostgreSQL

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

PostgreSQL⁵⁶ 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.

3.3.2.2.3. Firebase

Another consideration is Firebase⁵⁷, which may be useful for real time chat integration for the MergeMe functionality, however this can still be done with PostgreSQL. Adding the complexity of two databases may be an overhead compared to polling PostgreSQL at intervals. Additionally, messaging can be seen as an additional feature, as part of the minimum viable product the application can just allow users to see each other's email and facilitate sending them an email.

⁵⁵ Google cloud platform hosting database:
<https://cloud.google.com/appengine/docs/flexible/nodejs/using-third-party-databases>

⁵⁶ PostgreSQL: <https://www.postgresql.org>

⁵⁷ Firebase: <https://firebase.google.com/>

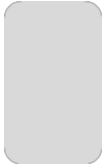
4. INTERFACE DESIGN STORY BOARD MOCKUPS

The mockups are high fidelity. The application will only support portrait view. Mockups are designed for Android and only show iOS screens when there is a large differentiation in UI.

For Android I have tried to stick with material design.

- Simplistic minimalist interface.
- Used the colour palette⁵⁸ to pick colors
 - Primary Light Blue 500 #03A9F4
 - Lighter 100 #B3E5FC
 - Darker 800 #0277BD
 - Amber 500 #FFC107
- 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 Typography⁵⁹ standard
 - Font: Roboto, 87% opacity
 - Subheading 16sp, Body 14sp

Light Blue	
500	#03A9F4
100	#B3E5FC
800	#0277BD
Amber	
500	#FFC107

Note:  means it takes you to a new screen when clicked

For iOS I have tried to stick with Flat design⁶⁰.

- Font = Helvetica Neue⁶¹
- No use of paper like/depth elements
- Flatter rectangles over cards
- No hamburger, have options to swap on the bottom instead
- Flat rounded buttons vs square with shadow

⁵⁸ Material UI color palette: <https://material.io/guidelines/style/color.html>

⁵⁹ Material UI fonts: <https://material.io/guidelines/style/typography.html>

⁶⁰ iOS design: <https://developer.apple.com/ios/human-interface-guidelines/overview/design-principles/>

⁶¹ Design Android vs iOS: <https://webdesign.tutsplus.com/articles/a-tale-of-two-platforms-designing-for-both-android-and-ios--cms-23616>

4.1. Splash 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.

ForkMe



Click to log in



4.2. Log 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.



Username/Email

darvid7

Password



4.3. Trending Page (GitTrends) - Android

- Shows a stack of cards of trending repositories, users can swipe left/right or press the buttons

Note: tensorflow/tensorflow means the author (either a user or an organization) is called tensorflow and the repository is called tensorflow (repository is after the /)

When clicked (tensorflow), takes you to organization/user view

When clicked (/tensorflow), takes you to repository view

Card flip animation, show next card

Card flip animation:

<https://developer.android.com/training/animation/cardflip.html>

Star view, will flip card and show next on return

4.4. Trending Page (GitTrends) - iOS

- flatter design
- no floating buttons
- no hamburger
- Helvetica font

iOS icons from

<https://icons8.com/web-app/for/ios7/>

Swipe effect instead of flip

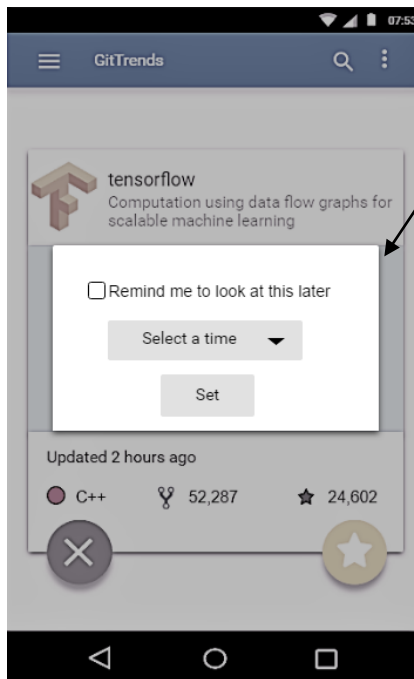
<https://github.com/reterVision/TinderSwipeCardsSwift>

Flat buttons rather than floating.

Star view, will swipe and show next card on return

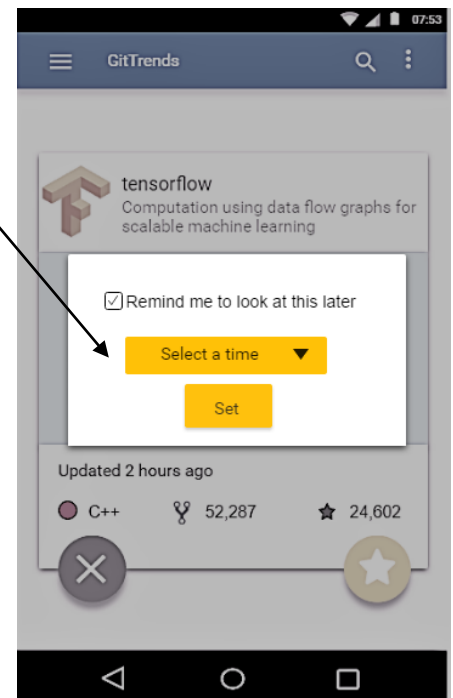
Go to other screens via toolbar rather than hamburger

4.5. Star view (setting a reminder) - Android



Pop up appears, ticking the check box colours the buttons allowing you to set the time for a reminder to look at this repository.

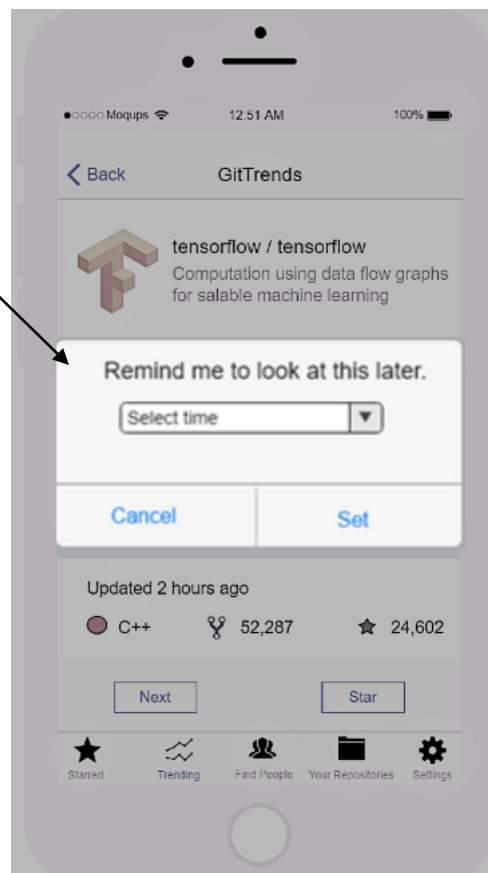
When set returns to previous screen or if canceled. Can click anywhere on the grey area to cancel.



4.6. Star view (setting a reminder) - iOS

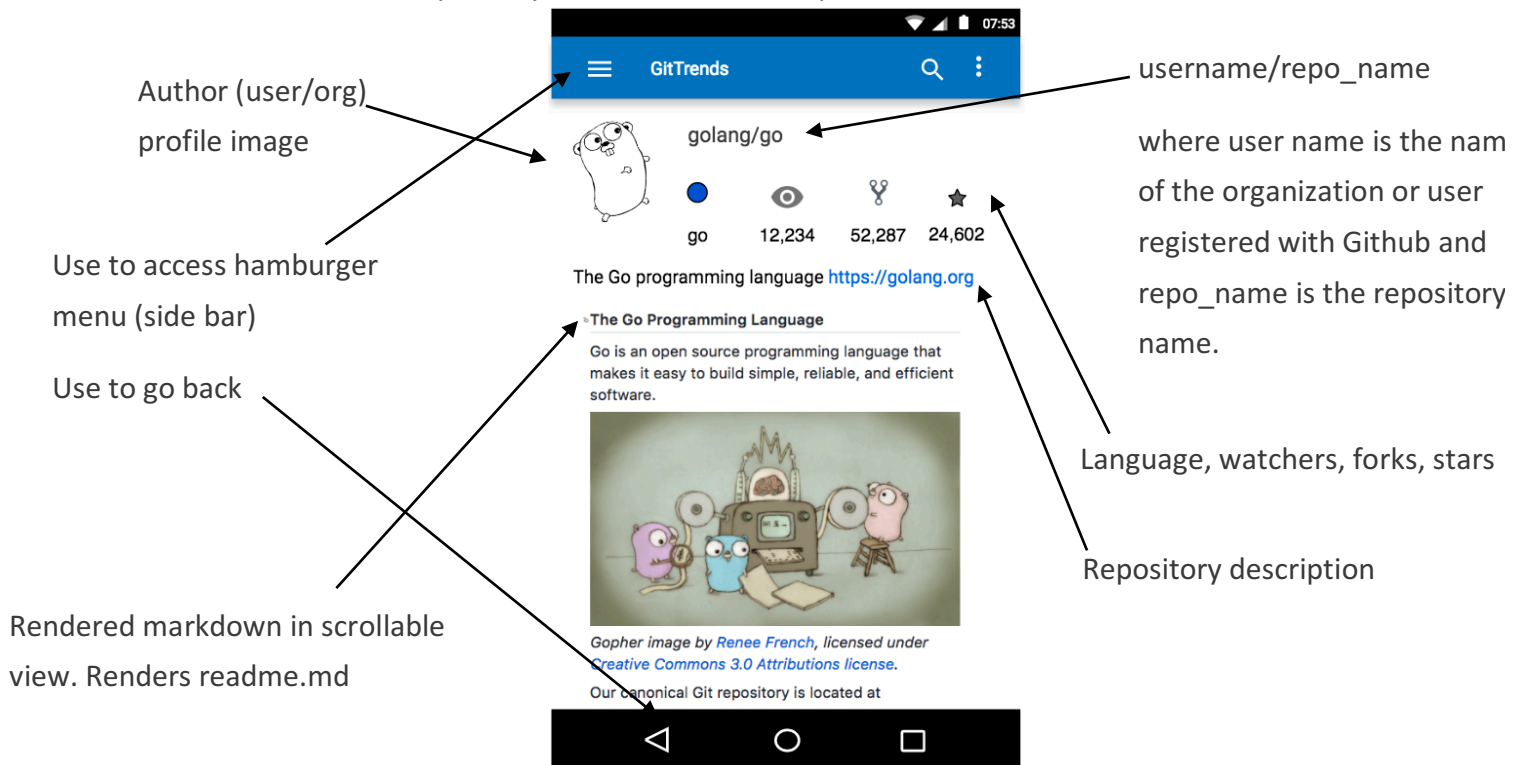
Pop up appears, allowing you to set the time for a reminder to look at this repository.

When set returns to previous screen or if canceled.



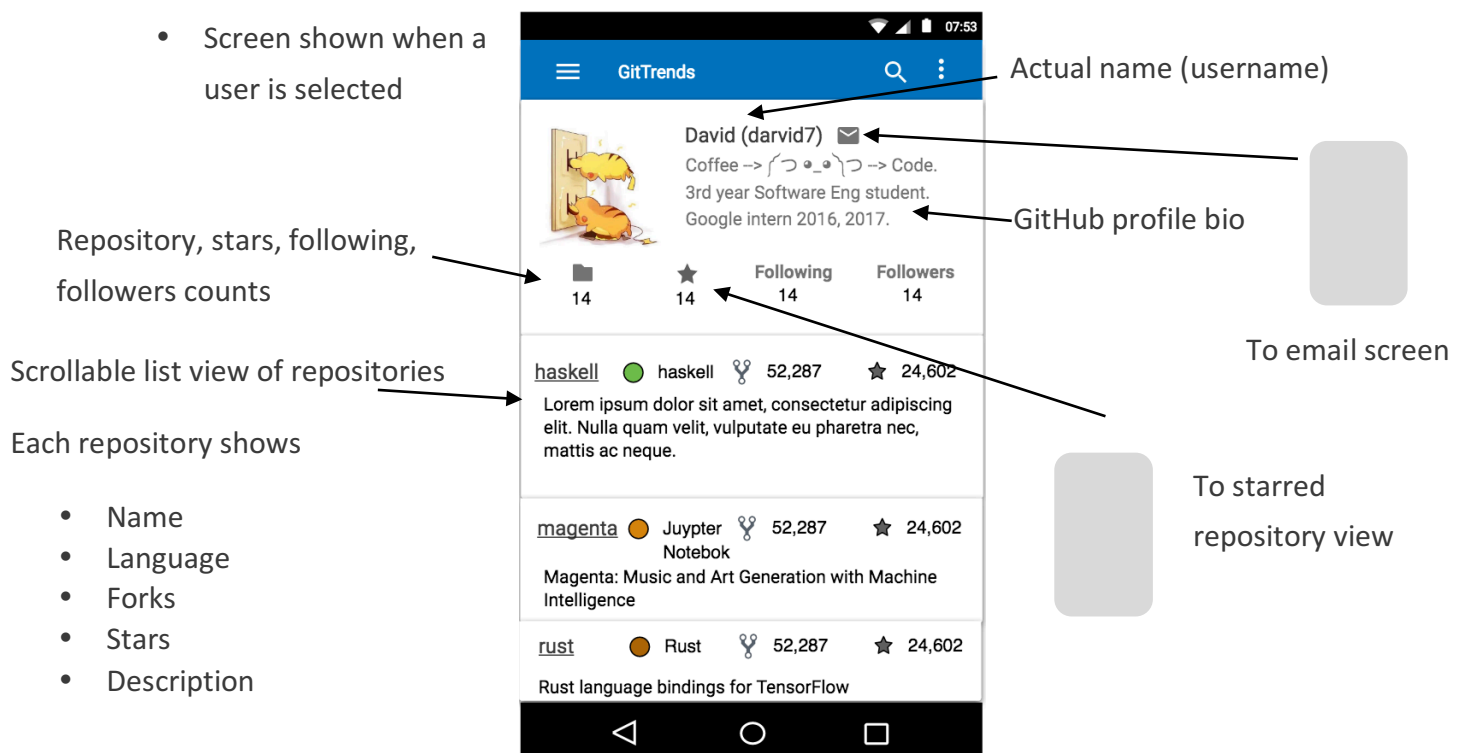
4.7. Repository View

- screen when a repository is selected, shows repo info & renders readme.md



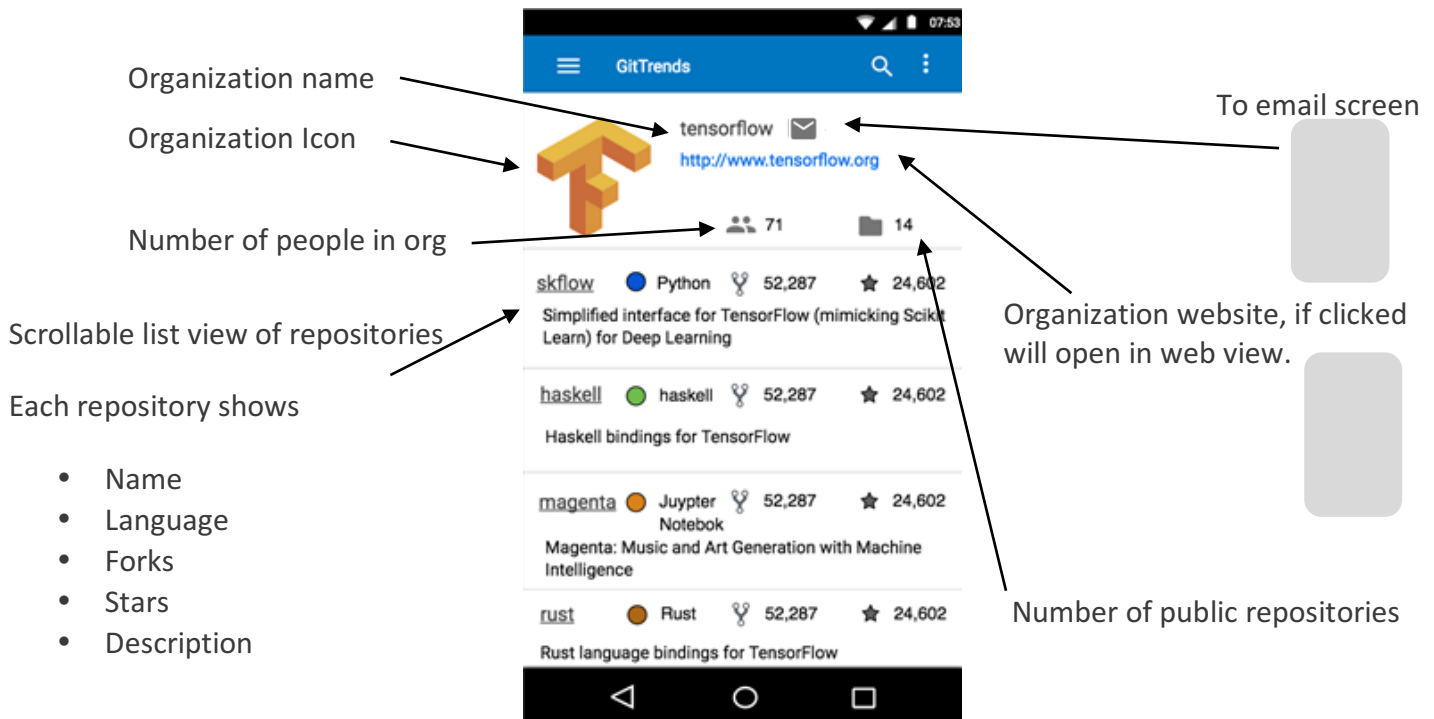
4.8. User View

- Screen shown when a user is selected



4.9. Organization View

- Screen shown when an organization is selected



4.10. Web View

- Screen shown when an external website URL is clicked on.
- Allows for smoother transition between application and web page



4.11.Starred Repository View

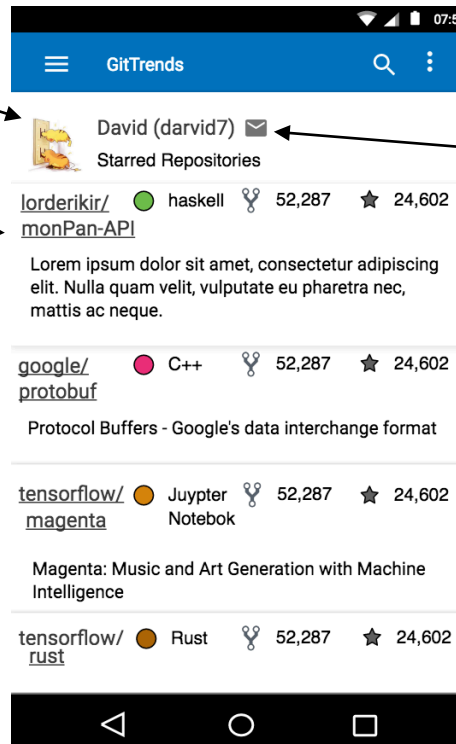
- Shows the repositories a user has starred

User's name and picture

Scrollable list view of starred repositories

Each repository shows

- Name
- Language
- Forks
- Stars



To email screen

4.12. Hamburger Menu (side bar) – Android specific

accessible from anywhere in the application by clicking on

Note: can exit hamburger menu by clicking any part of the app that isn't part of the menu i.e. the grey area.

To starred repository view for logged in user

To Find Developers View (MergeMe)



icons from

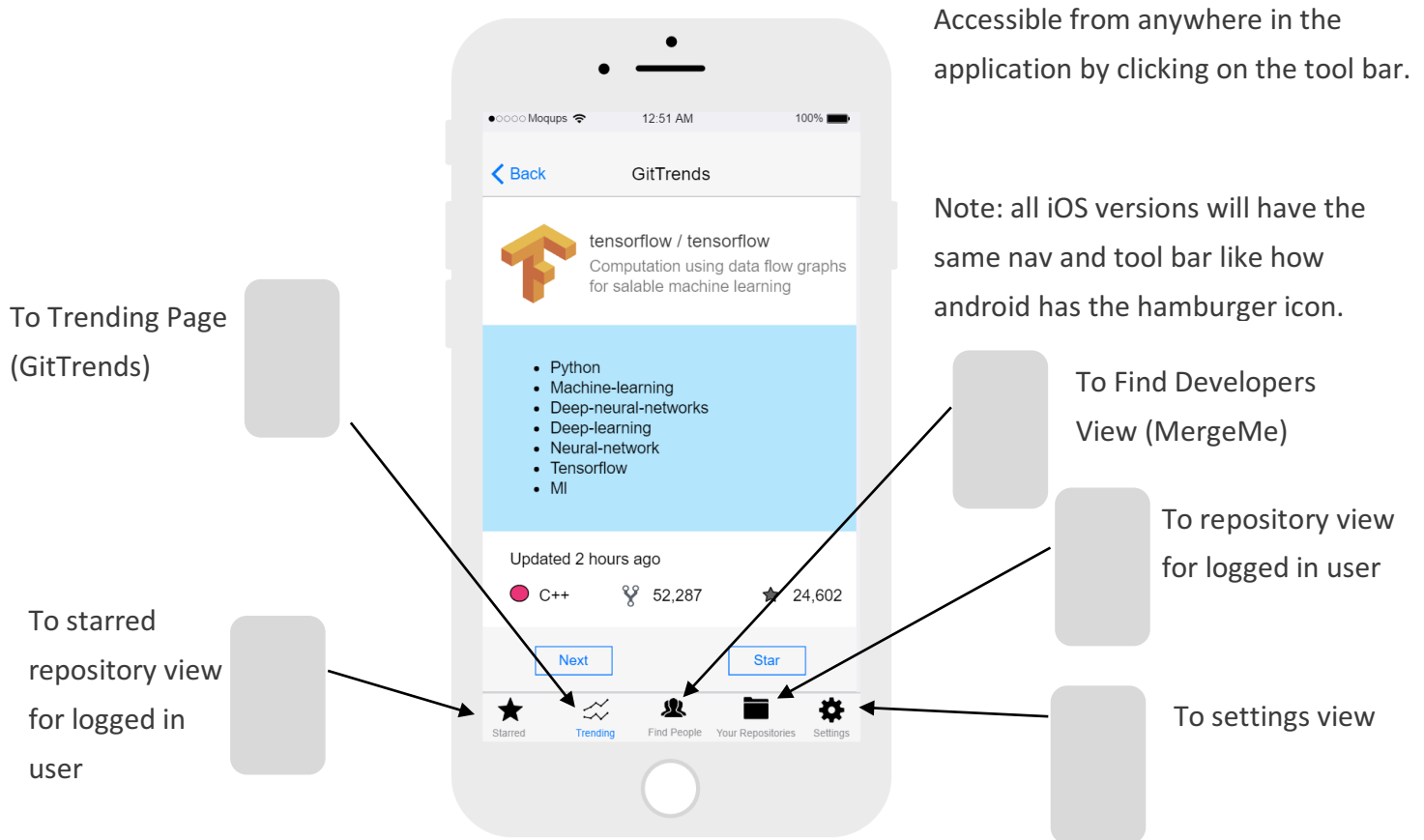
<https://www.materialui.co/icons>

To Trending Page (GitTrends)

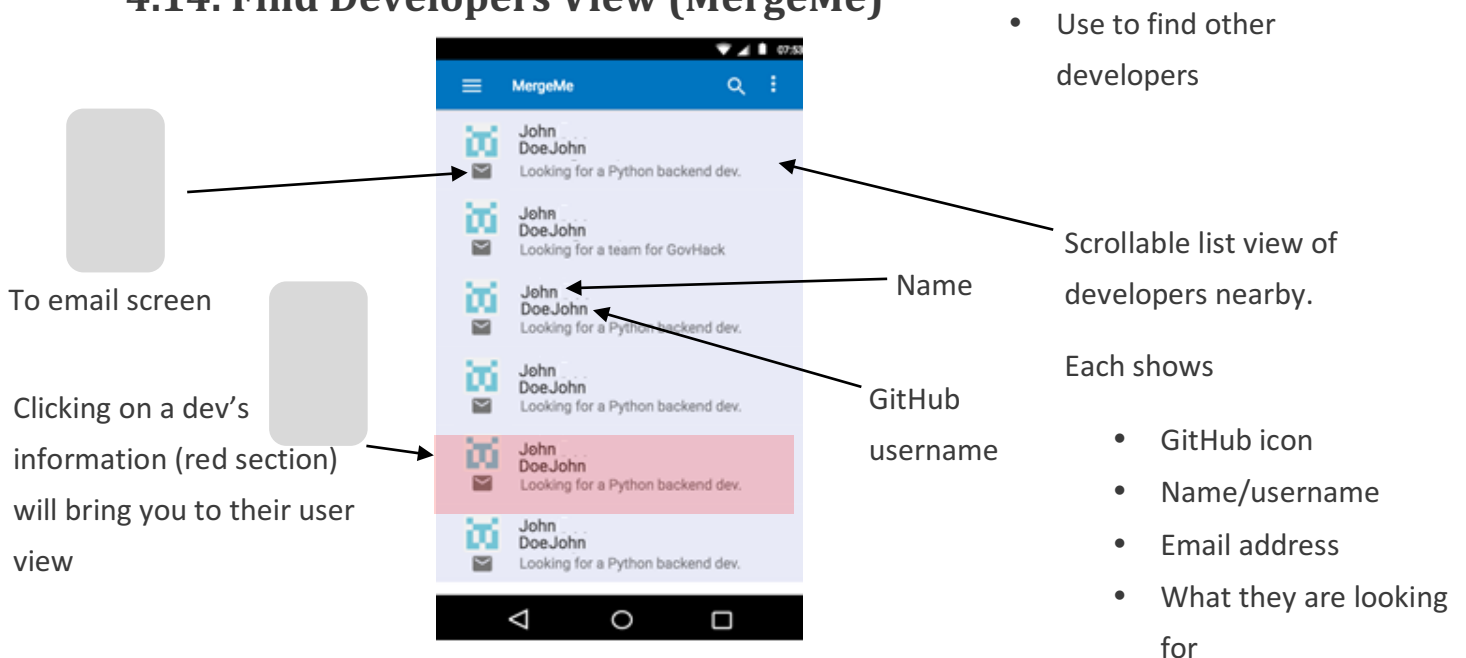
To repository view for logged in user

To setting view

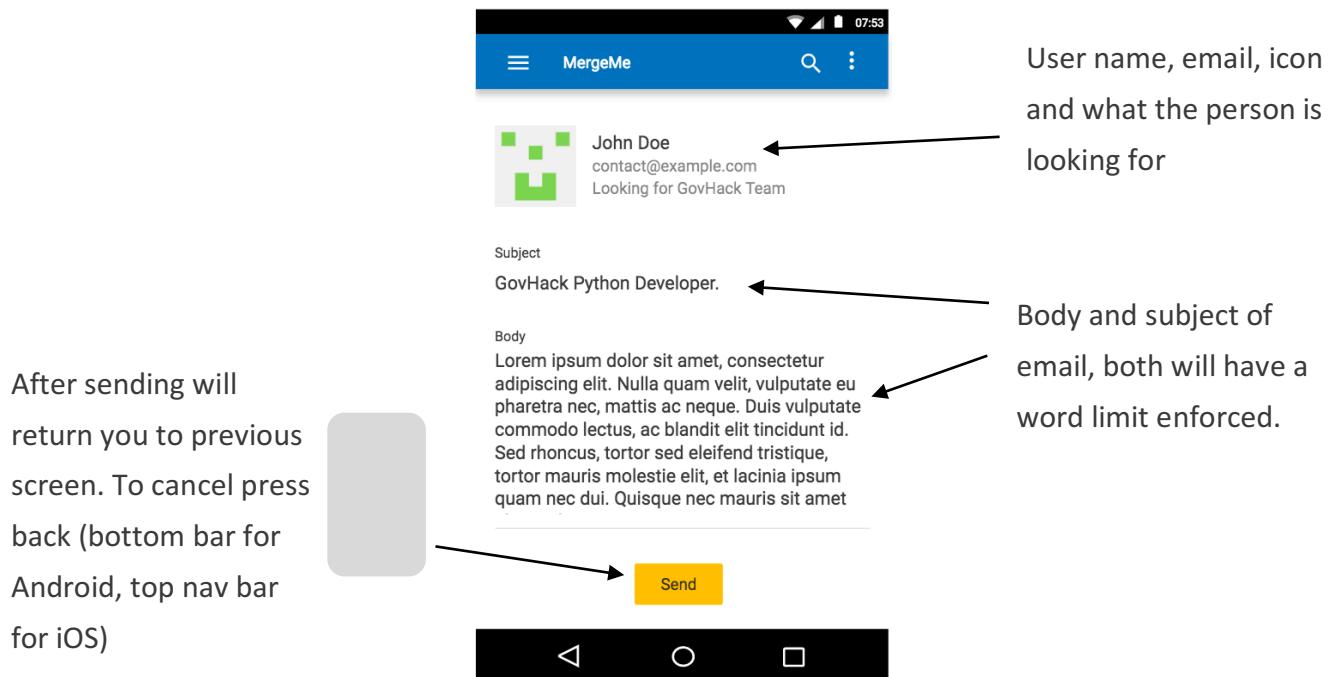
4.13. Toolbar - iOS Specific



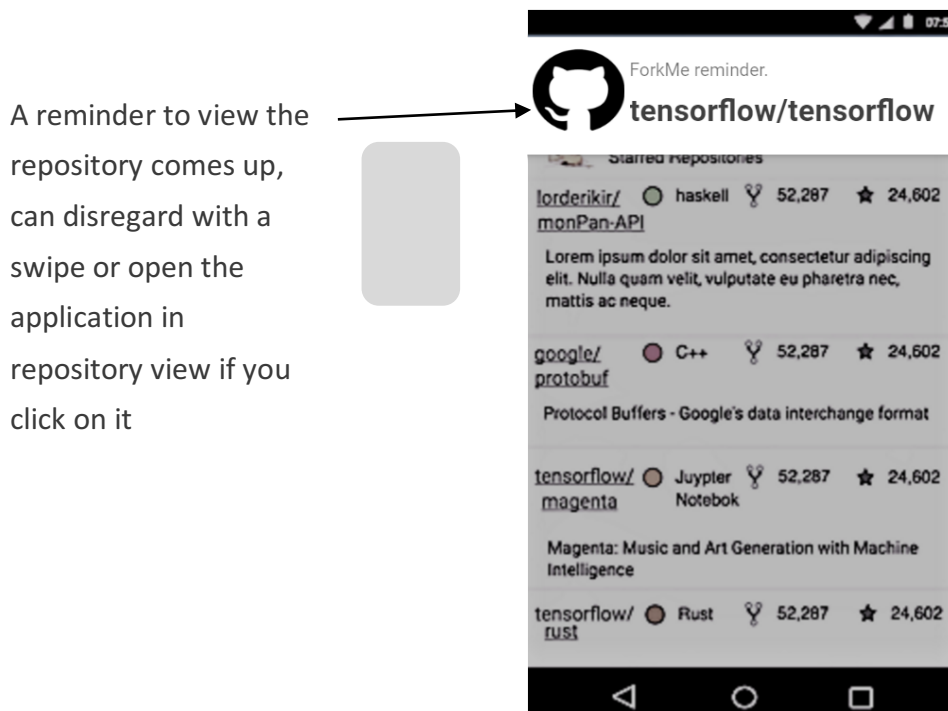
4.14. Find Developers View (MergeMe)



4.15. Email Screen (contact a developer)



4.16. Notification

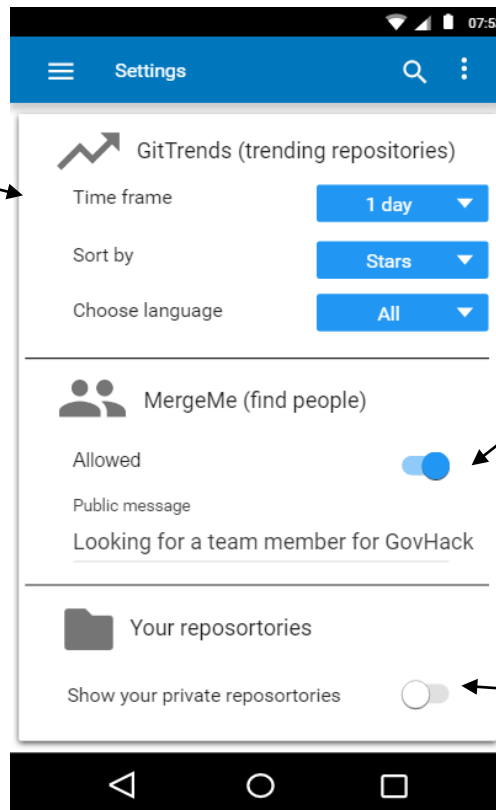


4.17. Settings view

- use to change settings for GitTrends, MergeMe and your repositories.

Use to specify results of trending repositories by

- Selecting time frame
 - last day, last week, last month etc
- Sort results by
 - number of stars, number of forks etc
- Choose language
 - Choose programming language to search for



Allow MergeMe to find people near you. Set public message here. Click to type, when you click outside it will be set. Disabled with not allowed.

Setting to show your own private repositories (only to yourself)

5. SCOPE AND LIMITATIONS

5.1. 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

5.2. Minimum Viable Product (MVP)

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

- GitTrends
 - a. Users able to view trending repositories
 - b. Users able to view other users' profiles
 - c. Users able to view other organizations' profiles
 - d. Users able to star a repository
 - e. Users able to watch a repository
 - f. Users able to set notifications to remind them to look at a repository later
- MergeMe
 - g. Users able to share what they are looking for
 - h. Users able to see other developers
 - i. Users able to send emails to other developers
 - j. Users able to send location data to webserver
- General
 - Web networking to communicate with server
 - GitHub OAuth
 - Geolocation information sent from mobile clients to server, aggregated by server and sent back

5.3. Limitations

Limitations include

- Time constraints (only have around 10 weeks)
- Programming ability in Java/Swift (not my most proficient languages)
- 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 nearby developers view. Pretty complex for a 10 week mobile app.
- Instant chat
 - Not needed for this application, but would be a cool feature.

5.4. Stretch Goals (out of initial scope)

Implementation of 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 developers based on
 - Geolocation, interests, skillset, public message
- Natural language processing to interpret what users share regarding
 - Skillset, public message, 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.

6. PLATFORM CHOICE

I have chosen to develop this application for the Android OS for the following reasons:

- I am more confident in Java than Swift
- The technology stack for Android fits together well as many of them are Google developed/owned which will mean they will be easier to integrate and find support for.
 - Protobuffers/Gson
 - Volley
 - SQLite
 - PostgreSQL
 - Atlassian common-mark
 - Webserver (Go/Python/JavaScript)
 - Deploy/host on Google cloud platform
- I want to learn how to use more of Googles technology as I'll be interning there next year 😊

7. ESTIMATED PROJECT TIMELINE

Week	Tasks	Milestone	Potential Setbacks
5	<ul style="list-style-type: none"> Define scope Write specification Storyboard UI Decide on Mobile operating system to develop for 	Assignment 1	
6	<ul style="list-style-type: none"> 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 <ul style="list-style-type: none"> Authenticates via GitHub's OAuth Can send and accept geolocation data from a webserver Can populate views based off local information 	Dealing with API keys, should be good tutorials out there though. Accessing Geolocation data may be hard, lots of support for this though.
7	<ul style="list-style-type: none"> Server and database send good mock information (information that is expected and not just random data) Set up navigability between screens Mock web API sending back expected data for all calls Implement GitTrends trending screen 	A working mobile app that <ul style="list-style-type: none"> Gets expected data back from web API Has a working prototype of GitTrends 	
8	<ul style="list-style-type: none"> Scope Audio Visual processing to see if it will fit into project Extend GitTrends by add viewing a user/organization's repo, starred repos 	A working mobile app that <ul style="list-style-type: none"> Has all of GitTrends functionality 	Spending too much time focused on audio and visual aspect of application/researching too much in to it. To mitigate this I will set a limit of 5 hours to look into Audio and Visual processing and to see if it will fit in the application.

9	<ul style="list-style-type: none"> Implement MergeMe based off mock web data In app emails In app webview 	A working mobile app that <ul style="list-style-type: none"> Has MergeMe functionality Webview Able to send emails from app 	Lots of support for all these, shouldn't be too hard.
10	<ul style="list-style-type: none"> Prototype with functioning web backend with basic ranking/classifying algorithm 	A backend that <ul style="list-style-type: none"> Gets data from GitHub's API Posts data allowing repositories to be started, watched etc from mobile front ends. Able to group users based on location and language and send that to front ends <p>Prototype 1 ready</p> <ul style="list-style-type: none"> Prototype with GitTrends, MergeMe and backend that works. 	<p>Ranking/classifying algorithm will be hard. Maybe use something out of the box such as a decision tree to output a binary value determining if a dev should be displayed on another dev's MergeMe list.</p> <p>Python scikit has lots of classifiers which may be of use.</p> <p>Worst case have users set their specified language and base it off that and the city they are in.</p> <p>Time limit to explore a more complex solution rather than worse case will be set to 10 hours.</p>
11	<ul style="list-style-type: none"> In app notifications and reminders Review scope, functionality and limitations and adjust User acceptance testing for prototype 2 Start Firebase Labs⁶² testing 	A working app that <ul style="list-style-type: none"> Will notify users after they set a timer to look at a starred repository 	Getting people to test the app, can bribe with food.

⁶² Firebase test lab: <https://firebase.google.com/docs/test-lab/>

12	<ul style="list-style-type: none"> • If there is time add extra animations/sounds, work on stretch goals • Else clean up code. • Firebase labs testing 	<ul style="list-style-type: none"> • Run app by a UX designer • Ensure JavaDoc is used and codebase is readable for others (get 2 people to through codebase) <p>Prototype 2 ready</p> <ul style="list-style-type: none"> • Key functionality implemented (GitTrends, MergeMe) along with notifications and polish animations/sounds 	<p>Getting people to look at the code base, I can buy them coffee to convince them.</p> <p>Getting a UX designer, I know a few but they might be busy, but can be bribed with coffee.</p>
13	<ul style="list-style-type: none"> • Polish application • Work on stretch goals • User acceptance testing for final application 	<ul style="list-style-type: none"> • Ask 10 other IT students to try prototype & give feedback • Test on firebase labs 	Getting people to test the app, can bribe with food.
14	<ul style="list-style-type: none"> • User acceptance testing for final application • Polish application • Work on stretch goals 	<p>Fully working application as specified by the scope of this design doc.</p> <p>Ready to launch and make this project open source.</p> <p>Final application ready</p> <ul style="list-style-type: none"> • Ready to launch • Code base isn't spaghetti • All functionality in scope implemented • Passed user acceptance testing and UX designer approves. 	If the application is not ready to release, don't release it but ensure it is good enough to meet the spec for assignment 3.

Note: Unit testing will be used throughout, along with Firebase labs to test for application on multiple devices later on.