Inscriber

by Christoffer Lundström

July 21, 2019

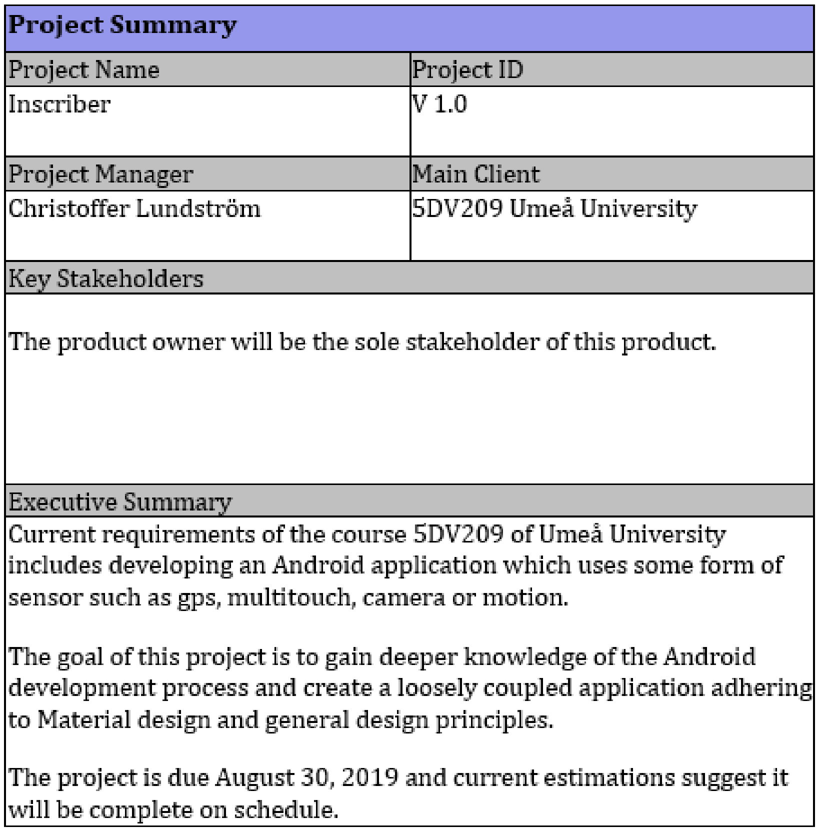
Development of Mobile Applications - 5DV209
Examinator: Johan Eliasson

UMEÅ UNIVERSITY

Contents

1 Introduction

1.1 General information



Project Summary	
Project Name	Project ID
Inscriber	V 1.0
Project Manager	Main Client
Christoffer Lundström	5DV209 Umeå University
Key Stakeholders	
The product owner will be the sole stakeholder of this product.	
Executive Summary	
Current requirements of the course 5DV209 of Umeå University includes developing an Android application which uses some form of sensor such as gps, multitouch, camera or motion. The goal of this project is to gain deeper knowledge of the Android development process and create a loosely coupled application adhering to Material design and general design principles. The project is due August 30, 2019 and current estimations suggest it will be complete on schedule.	

1.2 Vision

My vision for this project is to create an application inspired by geolocation and the phenomenon of geocaching. Similar to applications such as Pokemon GO and Geocaching^{TM}. It will be a simple but stable cloud based application where the user is able to drop short text-based notes at places they come across. The user can later look back on these locations and notes in a list together with a map and marker of the location. These notes will be called Inscriptions, like the mark you leave on a tree.

Future versions of the application could include the ability to share these Inscriptions with friends or even to the public by notifying those who come across the area you Inscribed. The ability to make time-based Inscriptions is also a worthy addition. These additions are the main reasoning behind choosing a cloud based database for the project.

1.2.1 Target audience

This app will be primarily suitable for explorers, hikers or outgoing people but usable by anyone. The application could serve as a reminder of a great mushroom spot, or to mark the start of a hiking trail. Any spot worthy of remembrance is worth an Inscription.

1.2.2 Play store description

Venture into the wild and leave your mark. Leave customized messages for the next traveller. Get notified when near an Inscription.

1.2.3 Grade

I aim for the higher grade, VG or A.

2 Project plan

This will be the main project plan for the Android application. The project will be written in Java 1.8 using Android Studio v3.5.0 and GitHub for version control.
The main approach of this project is to use an agile development method to reduce the amount of documentation. It will use Google Firebase for cloud services.

2.1 Justification

This product will fill a spot as a map-based notebook which could be used for multiple purposes. The application is also a requirement of 5DV209.

2.2 Stakeholders

The project manager(Christoffer Lundström) is the sole stakeholder.

2.3 Resources

This project will require at least 80 hours of work. Some of this time will be used for research. The project manager will be the only resource available.

2.4 Hard- and Software Requirements

The application is developed for Android API 19-28 and thus requires a device running Android. Furthermore the application is projected to reserve around 60-100mb of ram on the device making it viable for most phones or tablets. The app is cloud based and requires an internet connection.

The application uses a few external libraries such as SDP(*SDP*) and SSP(*SSP*) which are extensions to the DP and SP system of Android. They scale text and ui elements automatically for most devices which reduces time spent on porting UI for different screen-sizes.

Furthermore Glide(*Glide*) and AndroidImageSlider(*Android-Image-Slider*) are used for smooth image-transitions in the app guide.

2.5 Scope, Constraints and Assumptions

The scope of this project will be limited by a strict deadline, 30th of August. This will be a limiting factor to the development of features for the application. The scope will most likely cover basic features of the application such as creating an Inscription, viewing these together with map previews of each location.

Assuming the current vision is feasible for development the features of this application should be implemented on schedule.

3 Risk analysis

3.1 List of risks

Risks	Affect	Probability	Effects
Sudden hardware failure	Project	Low	Low
Staff illness	Project	Low	Catastrophic
Changes to software requirements	Project	High	Moderate
External library compatibility issues	Project	High	Moderate
Misestimating scope and size of project	Project	Medium	Moderate
Organizational restructure	Project	Negligible	Low

3.2 Risk strategy

The first step to avoid any project risks is to use GitHub for version control. This should mitigate any sudden hardware failures. Misestimating the scale and difficulty of the features proposed in the vision is always a risk for small projects, this should be no exception.

Another strategy to avoid missing the deadline is to have the project complete by 20th of August to leave 10 days for any unforeseen events or difficulties with the project.

Changes to the software requirements is a huge risk and it is quite unavoidable due to limited experience of Android development. This is mitigated by being careful with the amount of implemented features for version 1.0. It is a safer approach to make sure a few are well functioning and thoroughly tested before release as most users only give the application one chance. This allows for a controlled rollout of updates moving forward.

3.3 Contingency and monitoring

Should the project fail to deliver on schedule there are extended deadlines to work towards, the last one being 7th of October. This project uses a few external libraries and these will be closely monitored during the development process. It is essential that no updates are pushed to the project without proper software validation and verification. Because these libraries are open source the risk of forced updates are non-existent and can be safely used.

4 Requirements Analysis

4.1 Functional

This section will cover some important use cases of the application.

Use-Case - Inscribe

Pre-condition: None.
Post-condition: POST call to cloud.

Main Scenario:
1. User wants to Inscribe a location.
2. User clicks Inscribe button.
3. Add Inscription dialog is opened.
4. User writes a memo/note.
5. User presses Add.
6. Inscription is added to *Your Inscriptions* view.
7. Marker is added to main map.

Alternative Scenario:
5. User presses back or cancel.

Use-Case - View guide

Pre-condition: Guide dialog is present.
Post-condition: Activity and DialogFragment destroyed.

Main Scenario:
1. User wants to know how the application works.
2. User clicks *SHOW ME*.
3. Guide Activity is started.
4. User scrolls through infommercial images.
5. User clicks button *GOT IT*.
6. Activity is closed.

Alternative Scenario:
2. User clicks Hide.
5. User clicks back.

Use-Case - Reset data

Pre-condition: User has opened the Settings view.
Post-condition: DELETE call to Cloud.

Main Scenario:
1. User wants to reset Inscriptions.
2. User clicks preference *Reset data*.
3. Confirmation dialog is opened.
4. User accepts reset.
5. User is returned to Settings view.

Alternative Scenario:
3. User clicks Cancel and aborts action.
5. User clicks back.

Use-Case - View Inscriptions

Pre-condition: User has started the application.
Post-condition: None.

Main Scenario:
1. User wants to view Inscriptions.
2. User switches to *Your Inscriptions*-tab.
3. Recycler-view is displayed with a list of Inscriptions.
4. User pinches the map preview for a closer view of location.

Use-Case - Send feedback

Pre-condition: User has opened the Settings view.
Post-condition: None.

Main Scenario:
1. User wants to send the developer feedback.
2. User clicks Send feedback preference.
3. User is offered a predefined email template with an email-app of personal choice.

Alternative Scenario:
3. User clicks back or cancel.

4.2 Non-Functional

This section will cover requirements related to performance, accessibility, stability, ethics and safety.

4.2.1 Performance

The project's current state includes only the essential features. Performance should therefore reflect this. Because the application requires an internet connection it is very important that the data-packets are light. This is one of the few ways to influence app latency other than recommending a fast connection.

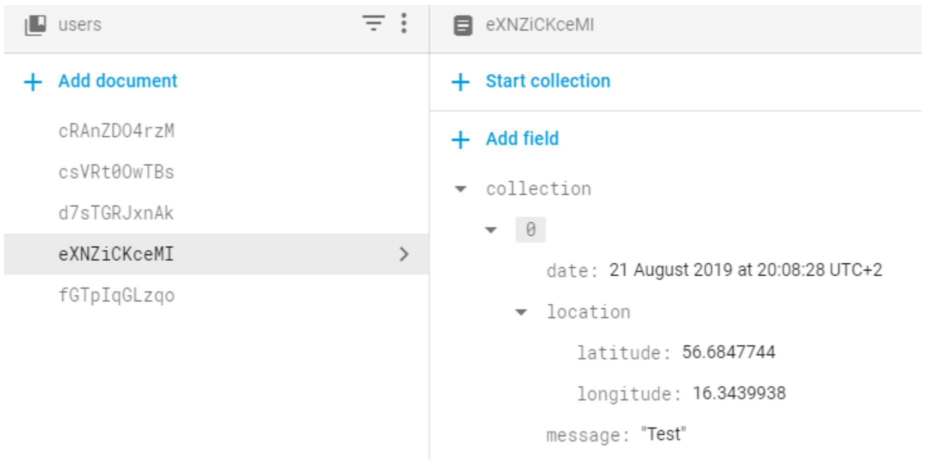
Another important feature of the application is to have private Inscriptions available even in offline conditions. In future updates a local copy of the collection could be stored on the device which is synced to the cloud when a connection is available.

To make the app light-weight and smooth it is important to use scalable solutions such as Recycler-views and frame-layouts which reduces the need of allocated memory.

4.2.2 Privacy, ethics & security

User privacy and security is an important focus of the application. Therefore only a few values which could be considered private should be collected. To identify each user a unique application id (firebase instance id) is produced. It is used for retrieving and posting new *Inscriptions* to the cloud.
By choosing this identifier the need for keeping another user reference such as email or login name is avoided. This approach requires no authentication and carries less overhead than firebase authentication but is more fragile to changes because each identifier is bound to the app installation. Reinstallation, clearing of app-cache or restoration of a backup will change the identifier and separate the user from its Inscriptions.

User locations stored together with private messages in Inscriptions can be very intrusive for privacy. Therefore the data is to be separated from any details which could identify a user. The collected data should also be transferred to firebase over a secure connection (HTTPS)(*Developer Policy Center - Privacy*).


users

Add document

cRAnZDO4rzM

csVRt0OwTBs

d7sTGRJxnAk

eXNZiCKceMI

fGTpIqGLzqo

eXNZiCKceMI

Start collection

Add field

```
collection
  0
    date: 21 August 2019 at 20:08:28 UTC+2
    location
      latitude: 56.6847744
      longitude: 16.3439938
    message: "Test"
```

Figure 1: Anonymous Inscription database entries.

In case of a complete data-leak no user should have to worry about being identified. From a security perspective end-users are the final variable to rely on. It should be clear in the application policy that no sensitive information should be posted in the message field of the Inscription and the users should be well-informed. This is likely the greatest privacy and security risk of all.

The ethical dilemma of collecting user data is an incongruous topic. The risks of keeping anonymity and privacy for users is of course that the service is used illegally or maliciously. Developers have an obligation to maintain confidentiality not only to their subject platform, but to the states in which the app is used.

In version 1.0 Inscriptions are private by default and does not include the ability to share Inscriptions publicly which avoids some of the risks previously mentioned, which also is recommended by Googles policy center(*Developer Policy Center - Enforcement*). However, in future updates there should be robust continuous monitoring of public user generated content (Inscriptions) and sampling of data to discover malicious activity and to make sure no policies are breached. One such monitoring system could include manual or automatic scanning of content and a report feature with which the community self-moderates(*Developer Policy Center - User generated content*).

Future plans include encryption of private Inscriptions for another layer of privacy for the user, but also to keep public Inscriptions unencrypted for monitoring purposes.

5 Design

The app is designed to have instantaneous access to its core features. A simple swipe or press of a tab will be enough to switch between the main views. Two of those main features are the map view and the inscription view.

The first design choice was to create a main activity which sets up a toolbar, fragment-container and a tab-manager.

This section will cover some of the fragments used in the application. All these views are also designed with landscape compatibility.

5.1 Map Fragment

This fragment is created in the main container at start. It implements a Google map which is used to find a user's location and look around the general area for Inscribed locations. It also handles permission requests, creation of a guide dialog and an *Add Inscription* button which follows a general green theme of the app and is compliant with *material design*(*Android Material Design - Buttons*). It is placed at an easily reached location of the screen.

The guide dialog is either dismissed or used to start a new guide activity intended to guide the user through the application.

The add inscription dialog opens a simple *EditText* dialog in which the user can add notes to the current location. This dialog creates a new instance of an Inscription object which is returned to the map fragment, added to the users collection and a *POST* request is sent to the cloud via use of the *InscriptionService*.

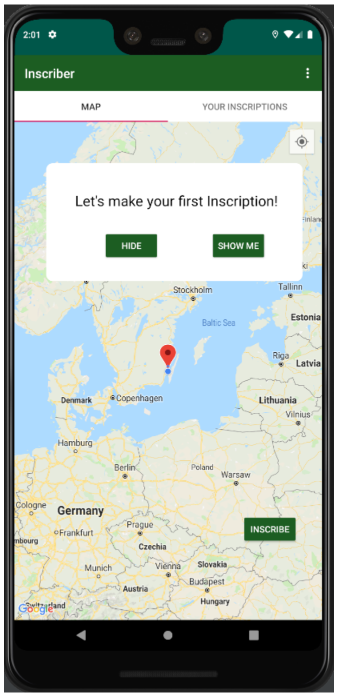


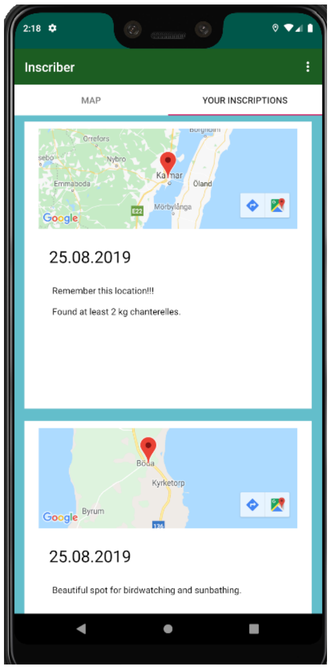
Figure 2: The fragment contains a map, button and dialog.

5.2 Inscription-list Fragment

This fragment contains a recycler-view responsible for displaying Inscriptions and keeping the list of Inscriptions synced with firebase. It does this by using the custom class *InscriptionService* to attach a firestore *snapshot-listener* on the users database-entry which acts as a webhook and immediately sends any updates to its listeners.

A Google *lite-map* is also used for each location. It is a tiny interactive map with a marker of the Inscription location(*Android Maps SDK - Lite mode*).

Currently there are no limitations on the amount of Inscriptions in the list, or on the frequency of Inscriptions saved. This must be addressed prior to a public release to avoid spam or expensive traffic fees.



25.08.2019

Remember this location!!!

Found at least 2 kg chanterelles.

25.08.2019

Beautiful spot for birdwatching and sunbathing.

Figure 3: Two inscriptions are shown in the recycler-view.

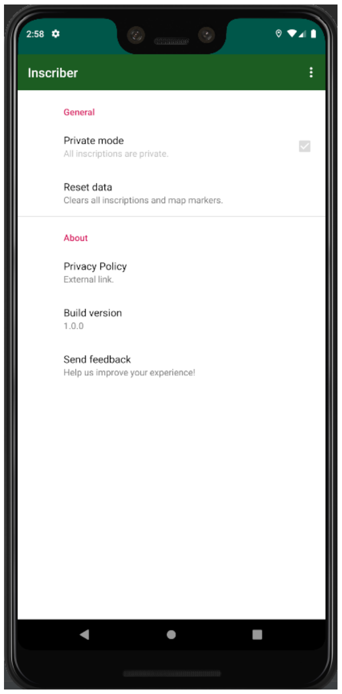
5.3 Settings Fragment

This fragment is opened from the toolbar button and displayed in the main fragment container. It is important that the fragment is a singleton to prevent duplicate actions, therefore each press of the settings button checks for duplicates of the instance in the fragment-manager.

As of version 1.0 all Inscriptions are private and the the only actions available in the fragment are *Reset Data* and *Send feedback*.

Reset Data uses *InscriptionService* to communicate with firebase and reset the users collection. This does not reset the firebase instance id to avoid unnecessary document creation in firestore cloud.

Send feedback is designed to start an email activity with pre-filled subject and email sender fields for easy communications with the developers.



General

Private mode ✓
All inscriptions are private.

Reset data
Clears all inscriptions and map markers.

About

Privacy Policy
External link.

Build version
1.0.0

Send feedback
Help us improve your experience!

Figure 4: The settings view.

5.4 Diagrams

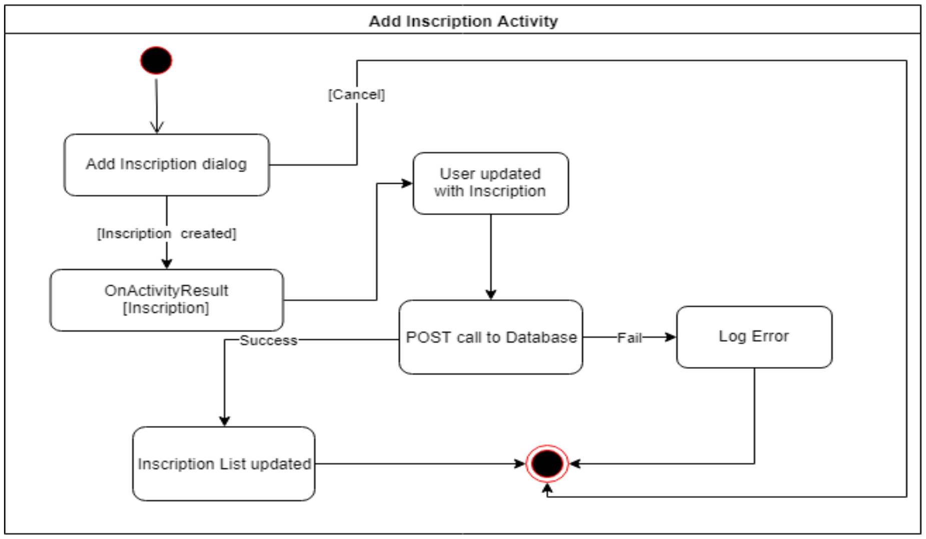


Figure 5: Activity flow of adding an Inscription.

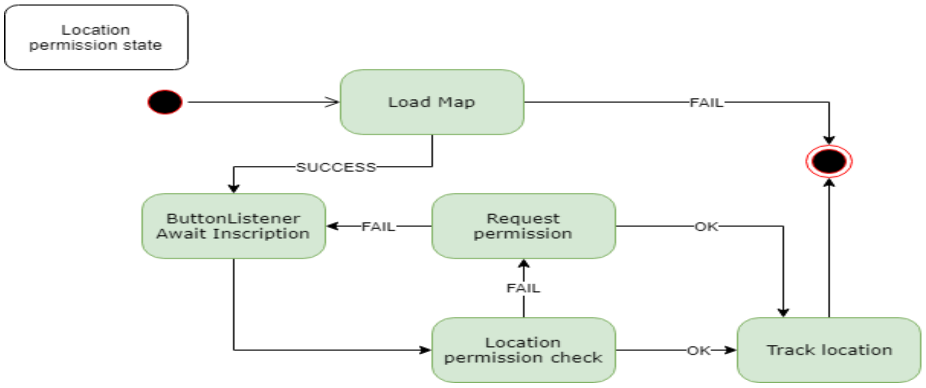

Location permission state
Load Map
FAIL
SUCCESS
ButtonListener Await Inscription
FAIL
Request permission
OK
FAIL
Location permission check
OK
Track location

Figure 6: Location state machine.

5.5 Reflection

The design of this app is fairly straightforward with a main fragment-container which is used as a placeholder for fragments added, removed or replaced by the fragment-manager. This approach is more dynamic and flexible rather than using different activities or hard-coded fragments. In that regard the app is designed with scalability in mind, and in preparation for future updates.

In early development each component implemented its own CRUD operations to the database. This turned out to be a bad approach which tightly coupled the components and made it hard to keep track of which object was most recently updated. This was later centralized to a single service managing all database-operations which facilitated communications.

During development the UI colour schemes and design templates changed at least a few times. The colour scheme is inspired by the green contrast of nature and the blue sky surrounding it. The scheme seemed fitting for an app targeted to explorers and hikers.

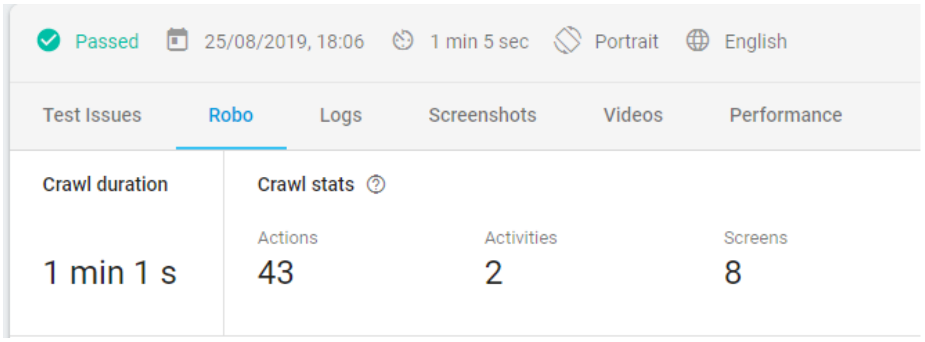
The choice of a cloud based backend for the app seemed like a necessity from the start. Had I begun using a local database it would most likely be difficult to migrate existing user data to the cloud in future updates.

I feel that the app almost is ready for a first release, however some additions would have to be made prior. Some of these are limiting of rate for Inscriptions, writing an app policy, unit testing with edge cases and saving of state when hiding the guide dialog.

To summarize I feel that the app development has been a tough, but worthwhile learning experience, and I feel very satisfied with the end results.

6 Test plan

During development the app has undergone a bare minimum of testing. Most of the tests have been manual, which are prone to errors. Because of this, and lack of time for writing unit tests an automated apk analyser from firebase has been used.



Passed 25/08/2019, 18:06 1 min 5 sec Portrait English

Test Issues | Robo | Logs | Screenshots | Videos | Performance

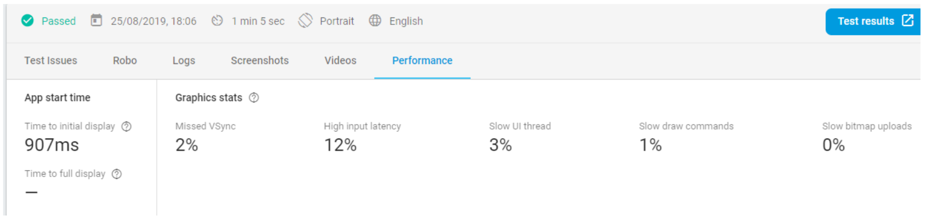
Crawl duration

1 min 1 s

Crawl stats

Actions	Activities	Screens
43	2	8

Figure 7: Firebase test lab results.



Passed 25/08/2019, 18:06 1 min 5 sec Portrait English Test results

Test Issues | Robo | Logs | Screenshots | Videos | Performance

App start time

Time to initial display
907ms

Time to full display
—

Graphics stats

Missed VSync	High input latency	Slow UI thread	Slow draw commands	Slow bitmap uploads
2%	12%	3%	1%	0%

Figure 8: Startup times and latency.

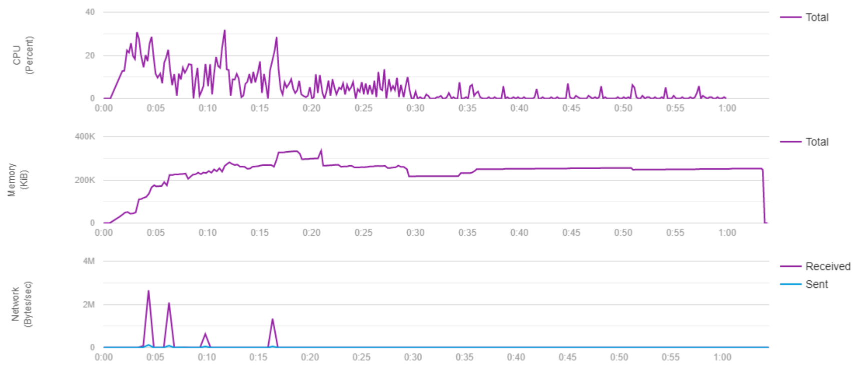


Figure 9: Performance graph.

7 References

Online sources

Bumptech. *Glide*. URL: https://github.com/bumptech/glide. (accessed: 25.08.2019).

Google. *Android Maps SDK - Lite mode*. URL: https://developers.google.com/maps/documentation/android-sdk/lite. (accessed: 25.08.2019).

— *Android Material Design - Buttons*. URL: https://material.io/components/buttons/. (accessed: 25.08.2019).

— *Developer Policy Center - Enforcement*. URL: https://play.google.com/about/enforcement/. (accessed: 25.08.2019).

— *Developer Policy Center - Privacy*. URL: https://play.google.com/about/privacy-security-deception/. (accessed: 25.08.2019).

— *Developer Policy Center - User generated content*. URL: https://play.google.com/about/restricted-content/user-generated-content/. (accessed: 25.08.2019).

intuit. *SDP*. URL: https://github.com/intuit/sdp. (accessed: 25.08.2019).

— *SSP*. URL: https://github.com/intuit/ssp. (accessed: 25.08.2019).

Smarteist. *Android-Image-Slider*. URL: https://github.com/smarteist/Android-Image-Slider. (accessed: 25.08.2019).