Group 4 - Report

Advanced Software Engineering, 2016

Dave Lewis, Lucas Rijllart, Francesco Soave, Stan Lewry

1. Introduction

With the introduction of the agile software development process within the last 10 years, development has had to accommodate for the fast iterative deployment of a constantly changing codebase. Having multiple developers working on the same project simultaneously, necessitates outstanding communication skills, and a strict process for managing assets.

2. What we wanted to learn

With this in mind, the team (consisting of four people, each with varying backgrounds and experiences in Software engineering and development) aimed to utilise an automated process to expedite development at every availability. Technologies such as automated testing, automated deployment, Kanban, and integration with communication technologies were all at the forefront of our expected learning outcomes. As no members of the team had any prior experience with the Android platform this was also seen as a great opportunity to gain experience with it.

3. Overview of approach

The team approached the task by deciding to minimise the amount of processing that would occur locally on the user's phone. This would allow for more testing to occur on a standardized set of server equipment, to enable the application to run at very similar speeds across all mobile hardware and to avoid excessive usage of network data.

The first stage of development began with implementing a server side database containing all of the house price data from the last 30 years. With much of the data being superfluous to our needs, the data was pre processed (stripping out columns such as street name etc) using a powershell script and stored in the database in its streamlined form (scripts and sample of queries used are in the REPORT/data_processing folder). Once within the database, this data was merged with data containing the latitude and longitude of every postcode.

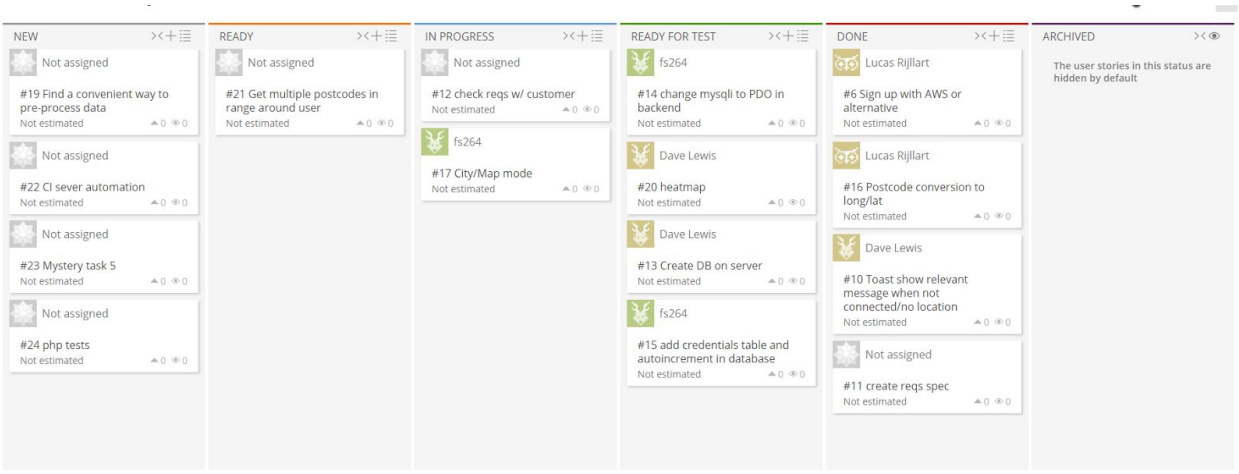
A server side php Page would then contact the database, querying the relevant data, this is where the majority of real time computation would occur.

Finally the application itself, with its minimal local codebase was developed in Android, implementing location based functionalities as well as the possibility to search for a specific postcode.

4. Project Organisation and Tools

In accordance with our original project plan, one team member was assigned the role of Team Leader. This member would be our point of communication with the customer and help manage the workload among team members.

Taiga Kanban (an Agile software) was used to schedule and break up the project into smaller more manageable chunks. These were then be given a status (ready for development, in progress etc.) and this allowed the team to easily see what work needed to be completed and what was finished.



Taiga Kanban

The integration between Kanban and the Slack communication client enabled all team members to stay up to date with the status of every task, and to see who was currently working on which aspect of the project.

The project could not have been completed without an online Git repository and this system proved the most vital part of the entire project, allowing team members to work on the same codebase simultaneously, accessible from any location. In addition this allowed the team to back up code regularly and create and merge separate branches of the application as well as produce deliverables.
Git/Github was then linked with Slack, CircleCI, Infer and Travis-ci.

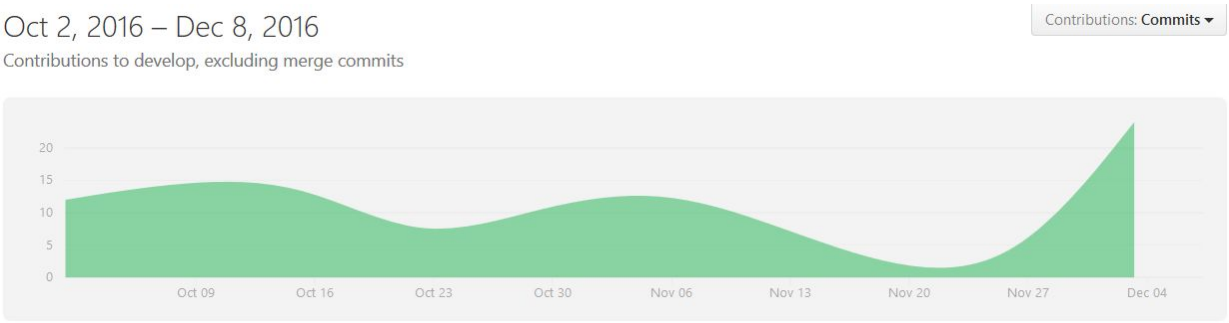
We met as a team regularly to carry out semi-formal code reviews. Because of the rapid nature of our development we would agree as a group what needed to be reviewed and go over that part of the code base together. These sessions were instrumental in the dissemination of knowledge of the code base throughout the team and ensure there where no single point of failure.

We have analysed the two extremes of version control.

Master exclusively: Keeping all changes and all people working on the master branch.
Pros: simple, no difficult operations in Git
Cons: not optimal code can be in master, hard to follow
Effectivity: good for small teams

Excessive branching: Making a develop/features/hotfixes/bugs branches.
Pros: structured, allows more functionality
Cons: complicated to use, accidents will happen
Effectivity: good for big teams

We have decided to use the best of both worlds, but follow more closely a simple structure as we are in a rather small team. We have kept a master branch for deliverables and big updates and a develop branch for all of the side work. We will also kept the git structure flexible, to allow branching for unexpected events or unforeseen difficulties.



Git overview of commits

5. What went well

5.1 Github - Online repository and githhooks

For many team members this was the first time interacting with an online repository for managing the codebase. However, after a few early hiccups, the team got to grips with the process. Team members were being able to work on code at any time, from any location without worrying about causing errors in other people's code that may not be noticed. The Github service even made the code review process easier as the team could instantly look at a log of who pushed which code, and when, meaning that if any team member didn't understand something, they could easily see who they ought to sit down with to talk through any issue.

To separate working, tested code from code that was still in development and may contain instabilities, we created a separate branch for team members to work on continuously, introducing experimental features that might not even end up in the final release. This ensured there was always a stable release at the head of the “master” branch that the customer could use without worry of crashes.

In the unfortunate circumstance where an error had been introduced, and the exact cause could not be determined, Git allowed us to fall back to an earlier version right before the introduction of the bug, so catastrophic amounts of work were never at stake of being lost.

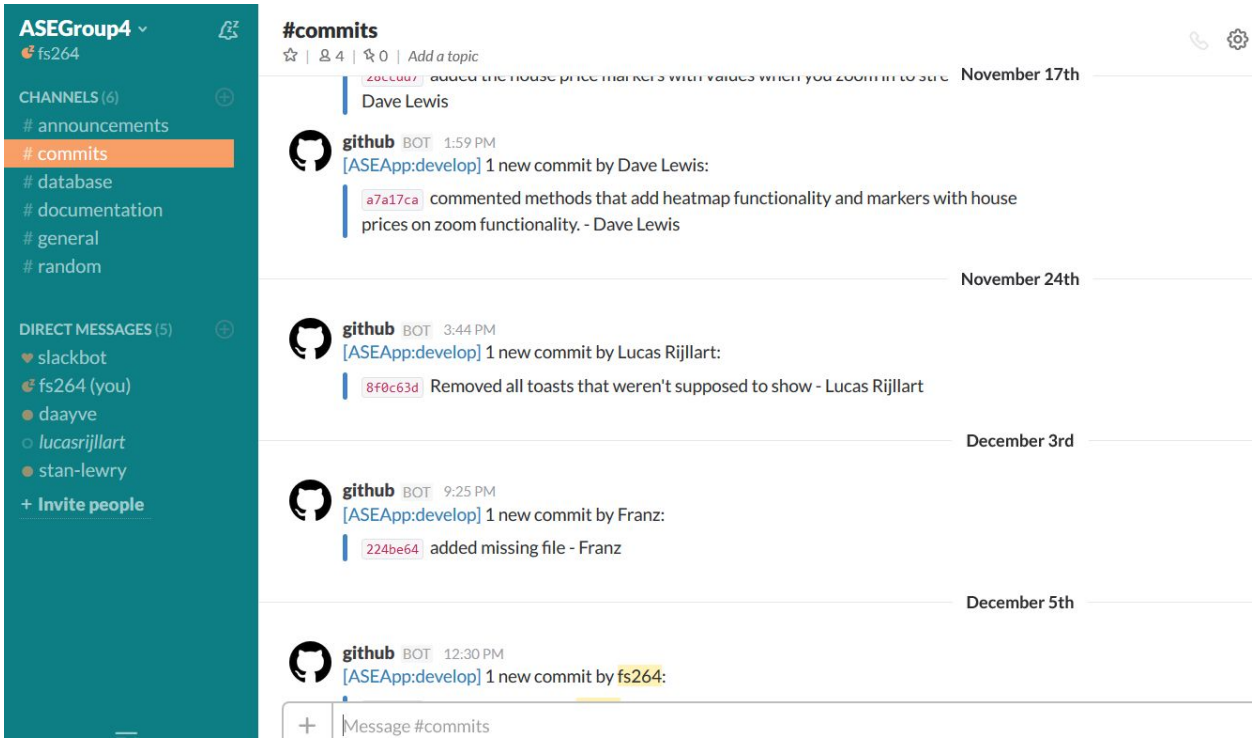
User side githooks were also implemented that automatically ran all existing written tests on the project, whenever the user attempted to push. If the tests all passed and a successful build could be created the push would be successful and an automated code coverage report would be displayed for the user (this was done with Jacoco). This helped to minimise errors distributed throughout the codebase and also keep the team regularly informed of areas in which the the project was lacking in written tests.

5.2 Communication

With efficient communication between team members being the main struggle with high speed iterative development, it was very pleasing to find that this seemed to be one area where the team shone. Slack was chosen as the platform of choice for its ability to be integrated with so many external agile systems, in addition to its own Chatbot help system, groups and notifications.

We created a web based hook to integrate Github within the communication tool. this informed the team with instant notifications whenever a new update had been pushed to the server. Ensuring the team knew when they should pull the codebase themselves and check for merge conflicts.

We also integrated the Kanban project management system within Slack. This informed the team whenever a task had been completed by a team mate. This ensured that team members didn't waste time thinking about how to implement a certain task then to check the Kanban software later and find it had already been completed. It also proved to be a decent motivator early on development. With team members taking pride in accomplishing as many tasks as possible.



Slack

6. What didn't go well

6.1 Project management

With a team leader in place, and as our only source of communication with the customer, the team needed guidance. However the team leader often acted as a regular member of the team and did very little to coordinate the workflow of different team members. With no tasks being assigned, the workflow quickly descended into each team member just grabbing tasks from Kanban and completing them as they pleased.

After a while however, the use of Kanban became less and less effective. With nobody creating new tasks, and existing tasks not being assigned Kanban became abandoned in favour of team members independently completing tasks that they believed to be the most important use of their time. With this being the case, the amount of work produced began to decline dramatically.

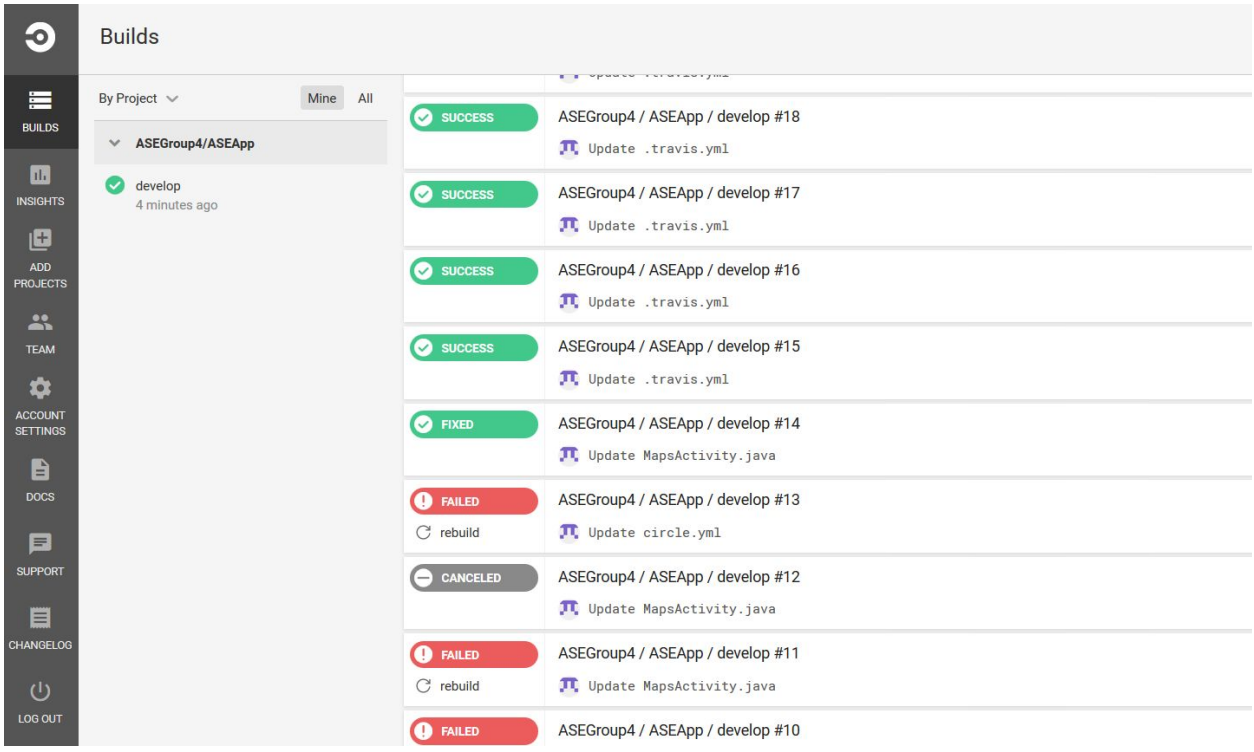
This is partly due to the fact that we didn’t really use an Agile approach (intended as an iterative process) during the development. In fact, the structure of the project almost never required us to go back and change or update our old code (except for minor fixes). For almost every task, we built it, it was working and it didn’t need to be modified at later stage. This resulted in a soft Agile approach (at a general level, based upon weekly meeting) while instead the proper development process was done in a more Waterfall approach, with each member of the team working on his own task. The agile approach was in fact mostly adopted to do code reviews within the team, where each member would go through the code in order to have every member aware of the new changes.

Also, the tasks weren’t in such a high number to actually actively require a tool like Kanban, which in fact wasn’t really used after the first few weeks. Based on the tasks required by the documentation, and features that the team agreed to implement, the tasks were assigned to the members.

6.2 Continuous integration

With respect to our project plan, we intended to automate as many development processes as we could, to streamline the development process and ensure time was not wasted manually completing uneccesary tasks.
However, continuous integration was not employed within the development process until very late in the day. This meant that part of deployment of the project (particularly the backend PHP files hosted on the server) had to be handled manually throughout the development process. This slowed down our workflow dramatically and hindered us fare more than spending a little extra time setting it up at the start would have.

Our server side php code was modified often, but with no automated deployment, this code needed to be re-uploaded to the server manually. This was often neglected, and bugs began to arise because of this. These bugs were not spotted easily as the code that was pulled from the git repo, and that team members were using to debug, was not the same as the code that was actually being used at runtime.
In hindsight, the team should have set up this automation at the very beginning of the project, possibly even before the first line of code was written. This would have saved a lot of wasted debug time and even the time it took team members to manually drag and drop files from one location to another.



CircleCI

6.3 Testing

Another hurdle upon which the team stumbled, was testing. Here again we decided to write our own tests. Thinking that in the short term, this would be faster than setting up automated testing. With such a small codebase it seem futile to waste time automating a process that we believed could be completed in a minimal amount of time. Unfortunately.

As a result of manual testing the team found that the testing suite was nowhere near thorough enough; providing far too little code coverage and letting far too many bugs slip through to the master branch when creating a new public release. Bugs in the very original codebase (task 2) were not discovered until near the very end of development, although this is also related to the fact that we didn't have tools and knowledge about testing during the first tasks.

In addition, the team found implementing their own tests very difficult due to the nature of the online application. To compensate for this, several classes needed to be implemented using "Mockito".

Automated testing was however implemented in the final stage of development. This was enlightening as it showed how our code had fallen very short of meeting the original specification.
The team quickly became aware of how important and useful proper code coverage and thorough testing would have been much earlier in development.

In the final stage continuous integration was implemented with different tools like Facebook Infer (as pre-commit hook), CircleCI (for every push to the repository) and Travis-ci (which didn't completely work due to its beta-release for Android).

7. Conclusions

While the above might paint a picture of a team that did not work well together, or of the produced work being poor, quite the opposite is true.

Overall the team cooperated well to produce a terrific piece of functional software including the following functionalities:

- A working UK house price heatmap
- house price information based on the user's current location (both with heat map and icons displaying the value for each street)
- A postcode search option, displaying house prices for any uk postcode
- A city averaged house price heat map.
- A GPS functionality that shows you the quickest way to get to the postcode you entered in the search option
- Details about Latitude/Longitude of displayed postcode

Using the following technologies:

- Kanban (Taiga)
- Github (with hooks)
- Slack
- JUnit
- Mockito
- Jacoco (code coverage)
- CircleCI
- Travis-ci
- Facebook Infer (pre commit)

If we had to start over a new similar project (in terms of development approach) we would like to focus more on the automation tools (as it's something that was lacking in the initial stage of this project) and to understand exactly how to get the best out of testing processes as well as Version Control systems.

8. Further work

8.1 Technologies

However, in future projects, the team will build upon the lessons learned and, depending on the project requirements, ensure the following:

- Automation first
- Use scrum meetings in conjunction with Kanban
- Better team structure
- Look further into data logging in order to easily track bugs and errors
- Explore tracking tools like Twitter Fabric
- To implement integrations with the communication tool since the very beginning
- Analyse how users interact with our application (https://fullstory.com/features)
- Research about the User Experience to understand better what functionalities might be implemented

8.2 Features

In regards of features that might be implemented in the app, and for which we didn't have time to implement during the project, some idea that we considered are:

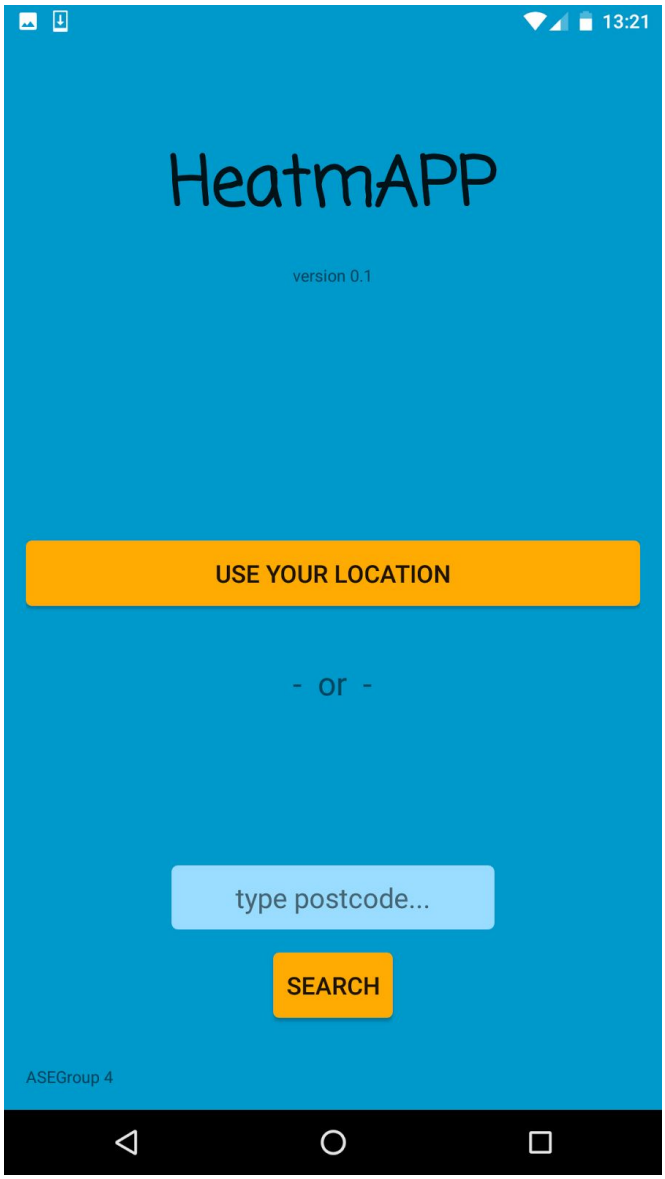
- Displaying graphs and charts. Google provides a nice API to implement graphs via Javascript. This would require to have a new activity as a webView and basically display a webpage with the graphs.

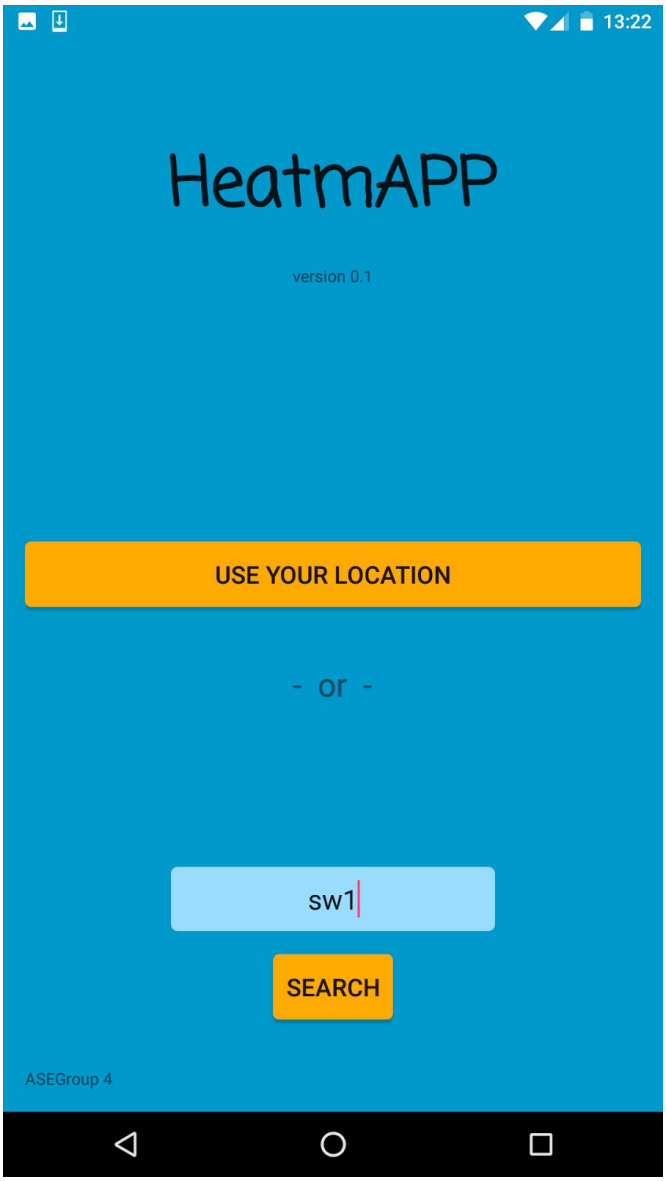
- Adding historical data. As we are currently only looking at data from the last three years, we have lots of unused data in the land registry file. We could implement a slider to look at data from previous years and this would change the colour of the heatmap.

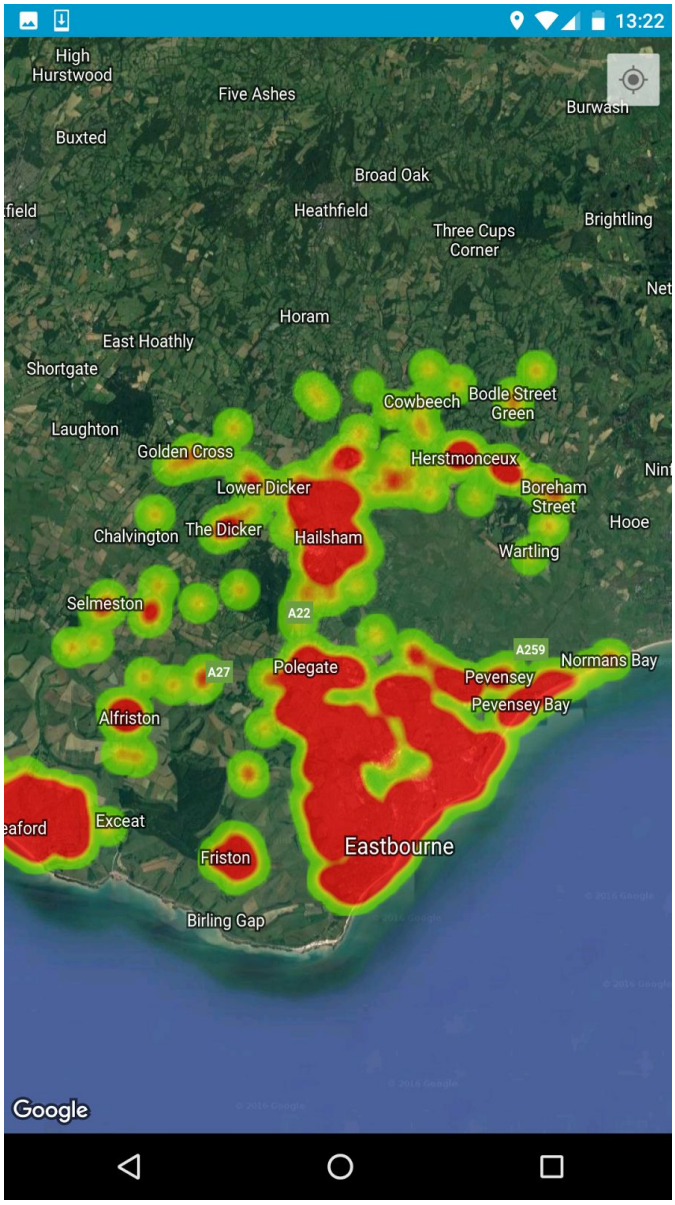
- Adding social media integration (Facebook, Twitter, Snapchat ect.) to improve the User Experience of the app

Learning from these lessons, and the key ideas from this course, the team should be able to set up the workflow much better, since the first steps, and this would lead for sure to a good outcome.

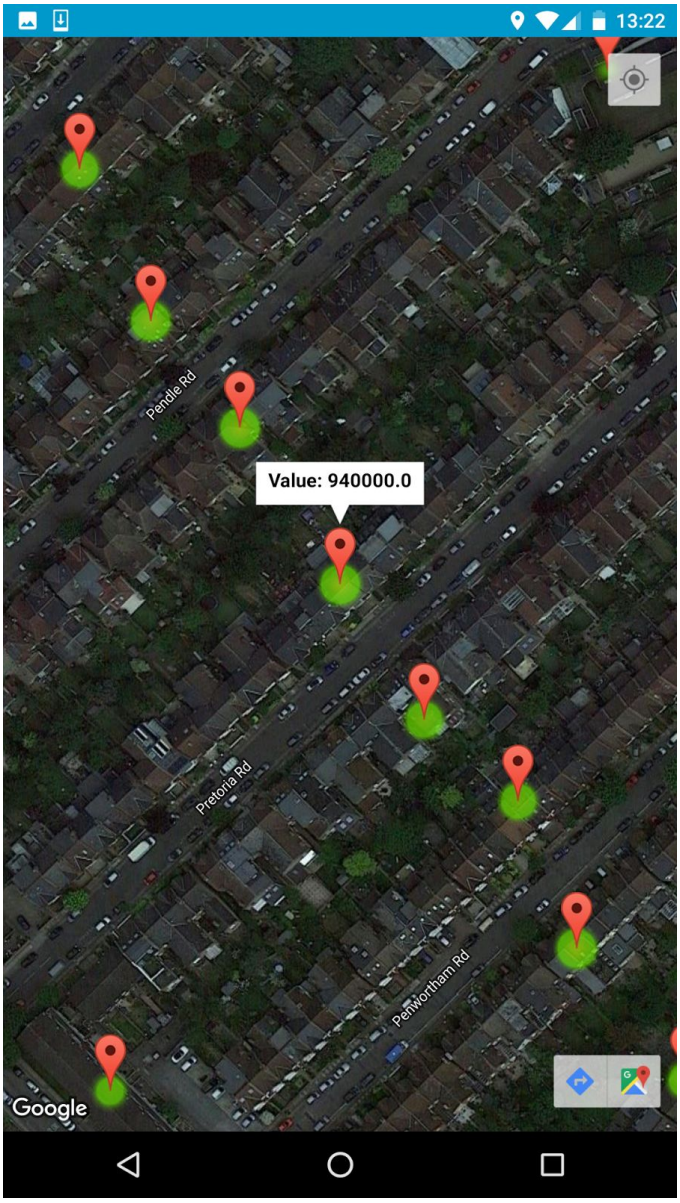
9. Screenshots

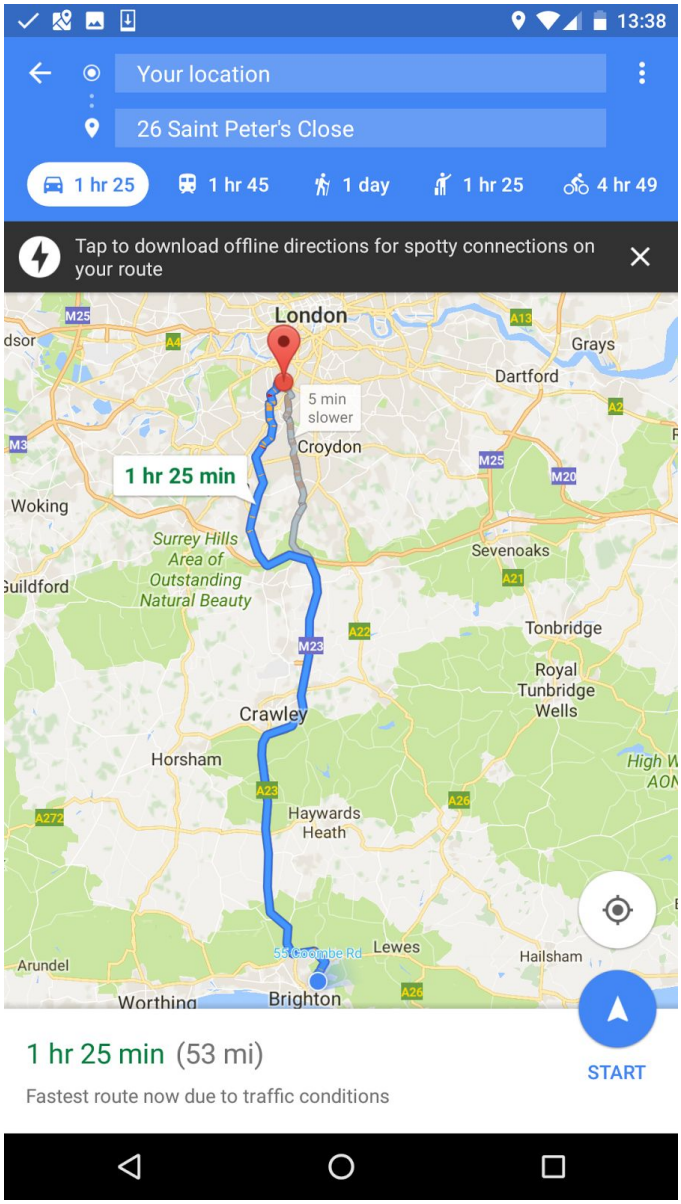





13:22
High Hurstwood
Five Ashes
Burwash
Buxted
Broad Oak
Heathfield
Brightling
Three Cups Corner
Horam
East Hoathly
Shortgate
Cowbeech
Bodle Street Green
Laughton
Golden Cross
Herstmonceux
Lower Dicker
Boreham Street
Chalvington
The Dicker
Hailsham
Hooe
Wartling
Selmeston
A22
A259
Normans Bay
A27
Polegate
Pevensey
Pevensey Bay
Alfriston
Exceat
Eastbourne
Friston
Birling Gap
Google


13:22
Clairview Rd
Parklands Rd
Woodhook Rd
Brookview Rd
Thrale Rd
Abbotsleigh Rd
Gracedale Rd
Nimrod Rd
Fayland Ave
Ribblesdale Rd
Blagborough Rd
Penrith St
Moyser Rd
Pretoria Rd
Penwortham Rd
Welham Rd
Mitcham Ln
Dahomey Rd
Fernthorpe Rd
Mitcham Lane Baptist Church
A216
Corsehill St
Greyswood St
Google


13:22
Pendle Rd
Value: 940000.0
Pretoria Rd
Penwortham Rd
Google


13:38
Your location
26 Saint Peter's Close
1 hr 25
1 hr 45
1 day
1 hr 25
4 hr 49
Tap to download offline directions for spotty connections on your route
London
Grays
Dartford
5 min slower
Croydon
1 hr 25 min
Woking
Surrey Hills Area of Outstanding Natural Beauty
Sevenoaks
Tonbridge
Royal Tunbridge Wells
Crawley
Horsham
Haywards Heath
Lewes
Hailsham
Arundel
Brighton
1 hr 25 min (53 mi)
Fastest route now due to traffic conditions
START