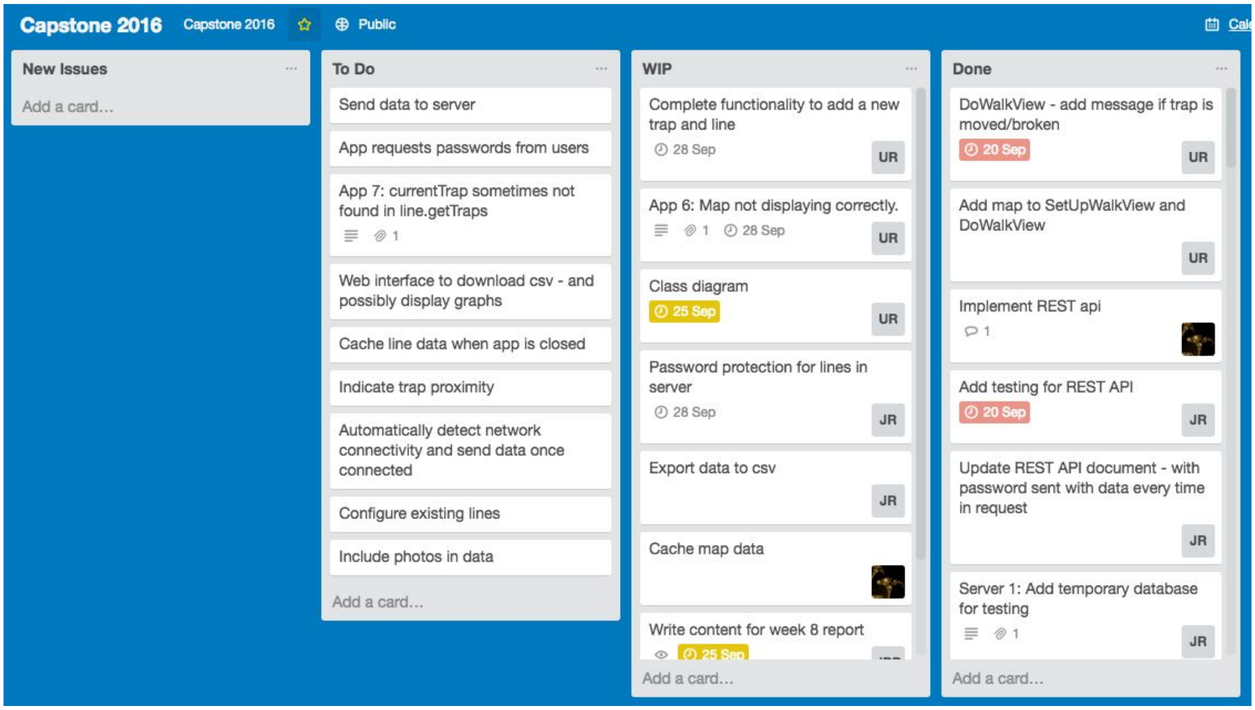
Week 8 Report

Jessica Braddon-Parsons
Cameron Dykstra
Umesh Ravji
James Rumbal

Table of Contents (Click to follow links to sections)

Project Plan

Over the semester we have been tracking our tasks on Trello, where we prioritise incoming tasks, set a due date and assign a team member(s) when work begins on the task, and move to move to “done” when it’s complete. This also helps us keep track of which tasks have been completed and by whom. Below is a screenshot of our Trello board, showing it in use. You can view the board at https://trello.com/b/mEv7nTB7.



Capstone 2016 · Capstone 2016 · ☆ · Public · Cal

New Issues

- Add a card…

To Do

- Send data to server
- App requests passwords from users
- App 7: currentTrap sometimes not found in line.getTraps · 1
- Web interface to download csv - and possibly display graphs
- Cache line data when app is closed
- Indicate trap proximity
- Automatically detect network connectivity and send data once connected
- Configure existing lines
- Include photos in data
- Add a card…

WIP

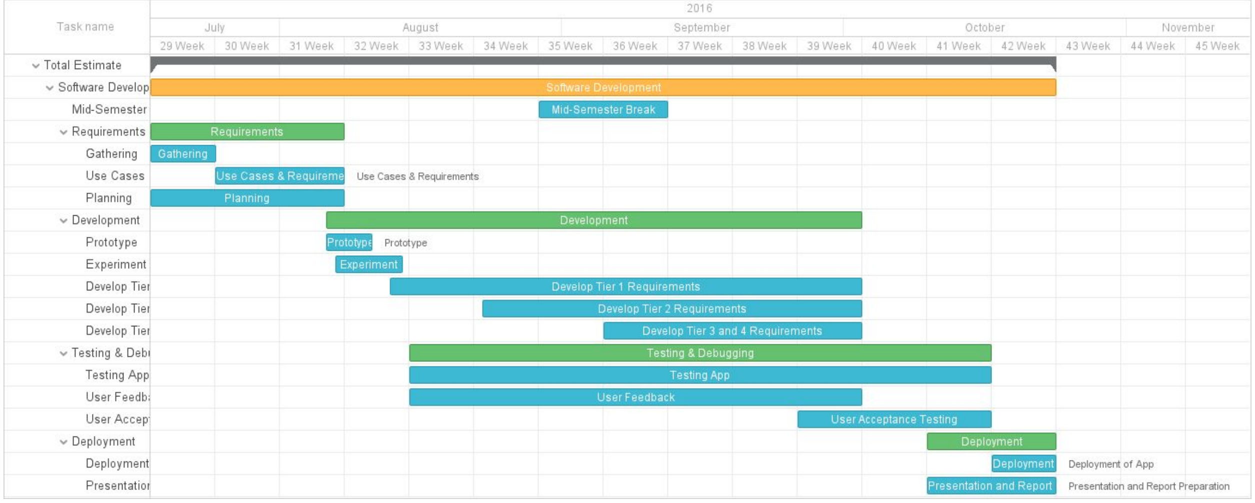
- Complete functionality to add a new trap and line · 28 Sep · UR
- App 6: Map not displaying correctly. · 1 · 28 Sep · UR
- Class diagram · 25 Sep · UR
- Password protection for lines in server · 28 Sep · JR
- Export data to csv · JR
- Cache map data
- Write content for week 8 report · 25 Sep
- Add a card…

Done

- DoWalkView - add message if trap is moved/broken · 20 Sep · UR
- Add map to SetUpWalkView and DoWalkView · UR
- Implement REST api · 1
- Add testing for REST API · 20 Sep · JR
- Update REST API document - with password sent with data every time in request · JR
- Server 1: Add temporary database for testing · 1 · JR
- Add a card…

We have also updated our Gantt chart for the semester, extending development time as needed.

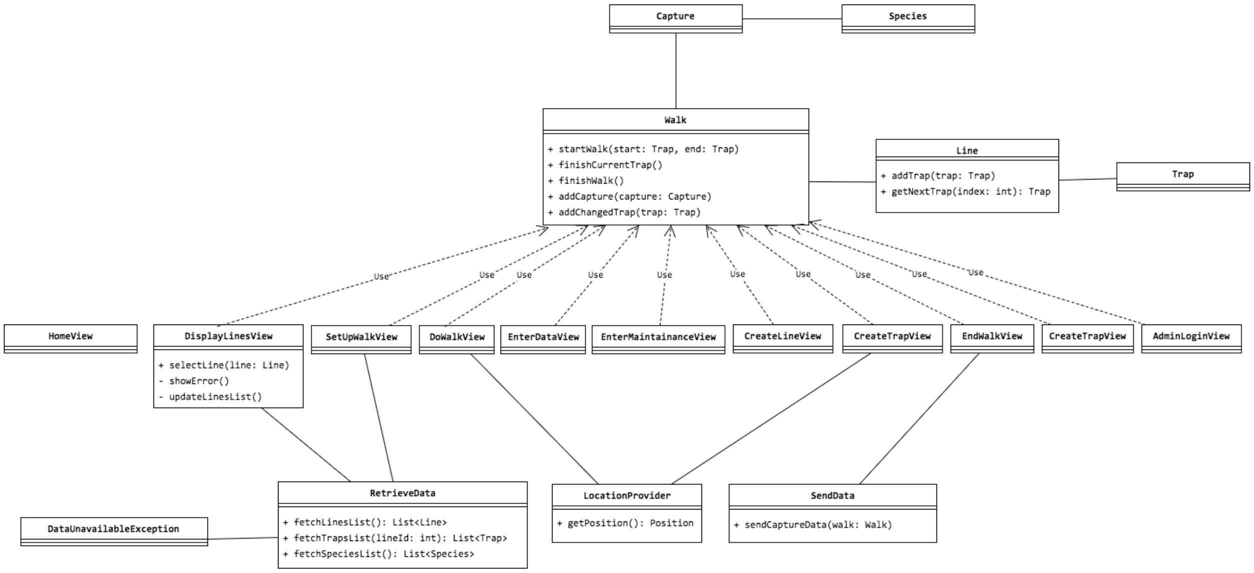
Generated with online service GanttPro.com



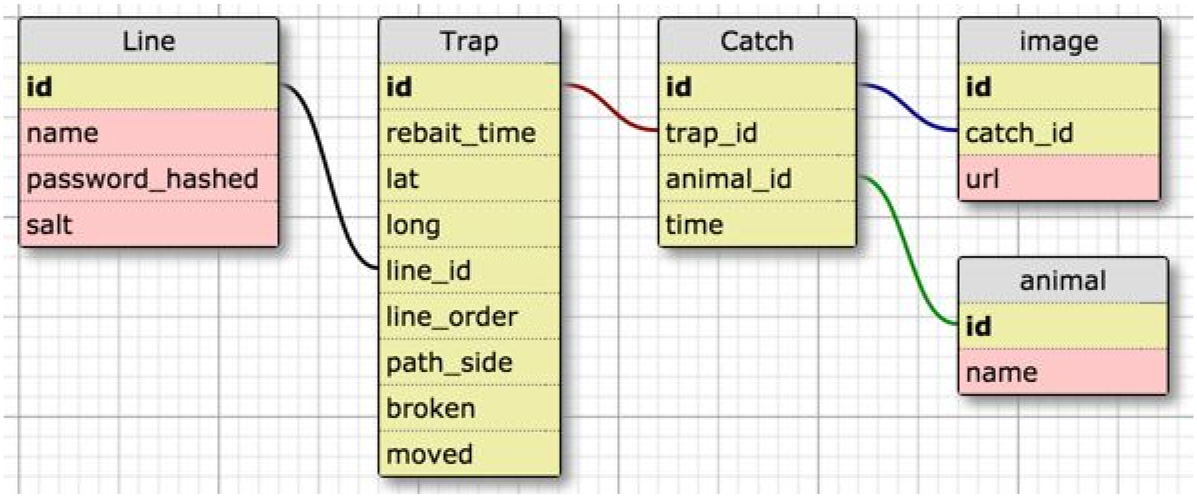
Architecture

Class diagram

We have a basic class diagram, which is currently missing some detail as we continue to develop the app and refine our design. You can view the image here.



Updated database diagram



Server REST API

The documentation for the server's REST API is in a PDF in the repository, but a copy being kept up to date can also be found here.

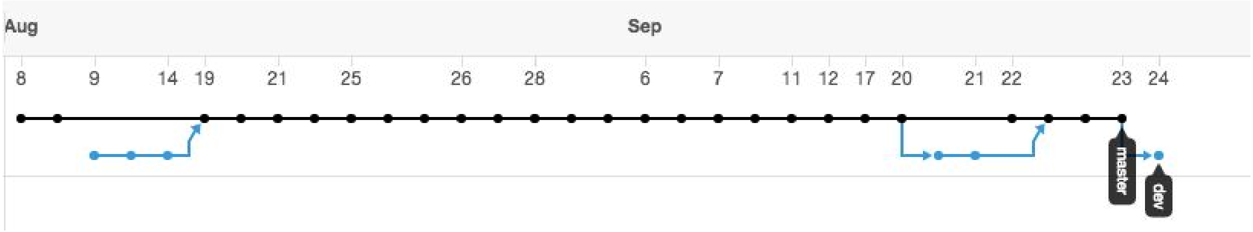
Repository and Issue Tracking

Server

The server repository is worked on by James and Cameron.

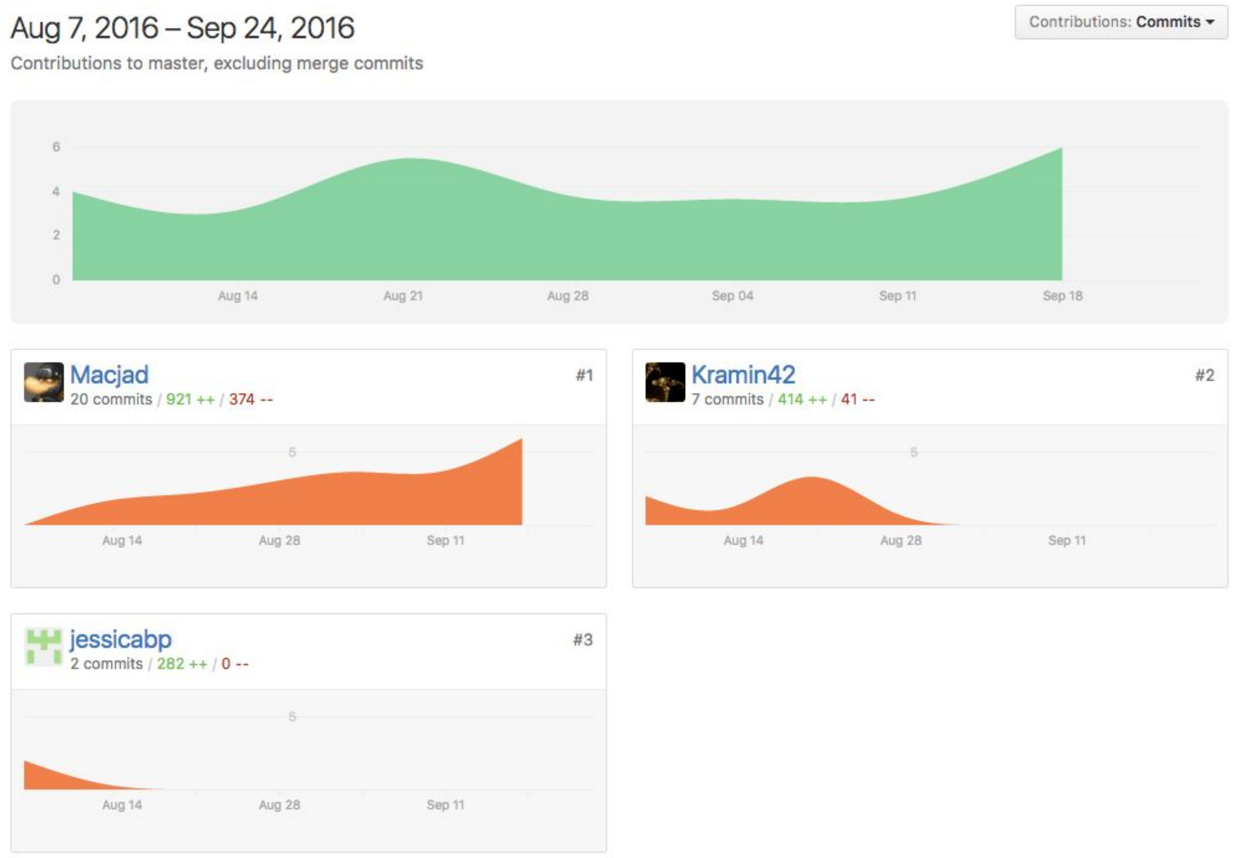
Branching

There is very little branching as there was very little parallel development. This is due to team members committing and pushing their work very regularly and working closely together.



Commits

There has been a fairly consistent rate of commits since the repository was created. However not all team members have been committing consistently over this time. This is because Cameron stopped work on the server and started helping out with the app, such as looking into map APIs and creating spikes. All commits are commented.



Aug 7, 2016 – Sep 24, 2016

Contributions to master, excluding merge commits

Contributions: Commits

Macjad #1
20 commits / 921 ++ / 374 --

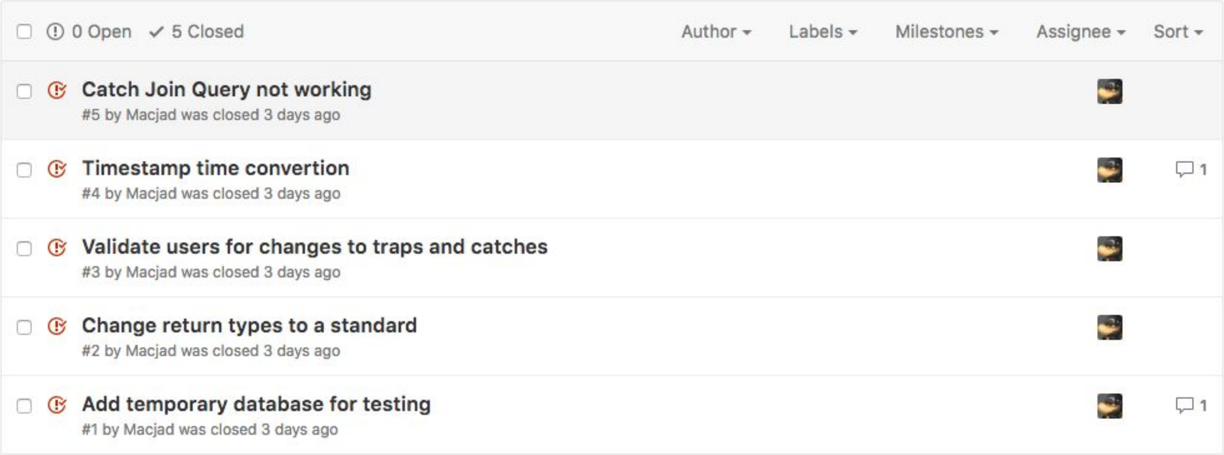
Kramin42 #2
7 commits / 414 ++ / 41 --

jessicabp #3
2 commits / 282 ++ / 0 --

Issues

There are not very many issues in the repository's issue tracking system. This is due to the way we have used it in parallel with Trello. If we encounter a new task which needs to be completed, we have added it to Trello as a new card. This has meant less issues are created in GitHub as we only add an issue to the issue tracking system if is it a bug or problem we have found in our code. This also creates a new task card in Trello for us to deal with through our usual workflow process. However this combination of tools has meant we are not always good at ensuring we close our GitHub issues.

Generally the person who has created an issue in GitHub has also assigned themselves to the issue. However once the issue is in Trello it is reconsidered and may then be assigned to any team member as appropriate. This explains why the issues are all assigned to one member in the issue tracker, as this is not the primary system we use and so it is not always kept up to date.



□ Catch Join Query not working
#5 by Macjad was closed 3 days ago

□ Timestamp time convertion
#4 by Macjad was closed 3 days ago
1

□ Validate users for changes to traps and catches
#3 by Macjad was closed 3 days ago

□ Change return types to a standard
#2 by Macjad was closed 3 days ago

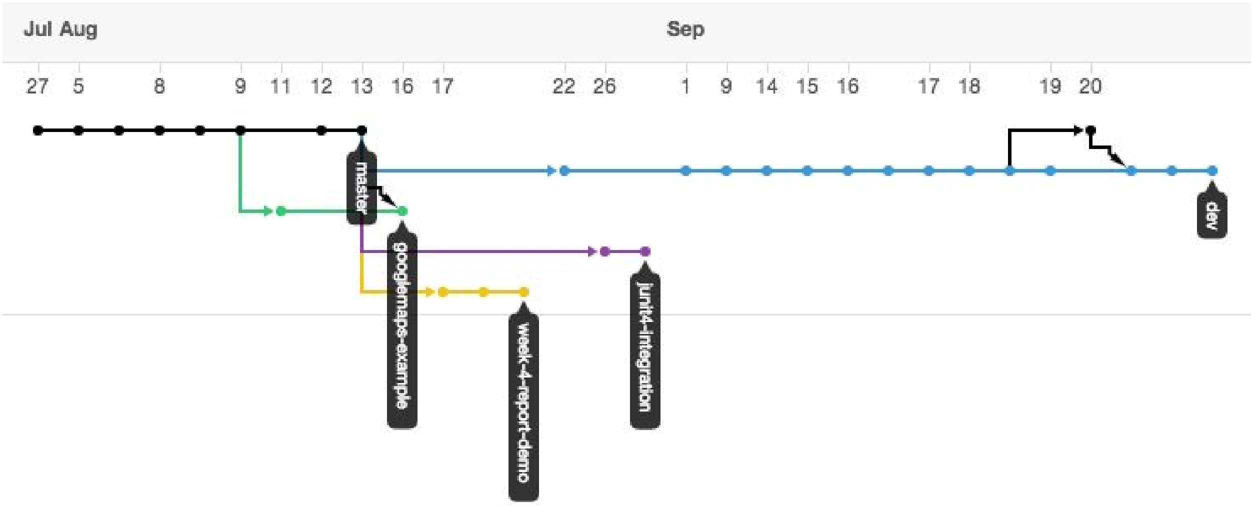
□ Add temporary database for testing
#1 by Macjad was closed 3 days ago
1

App

The app repository is worked on by Jessica and Umesh.

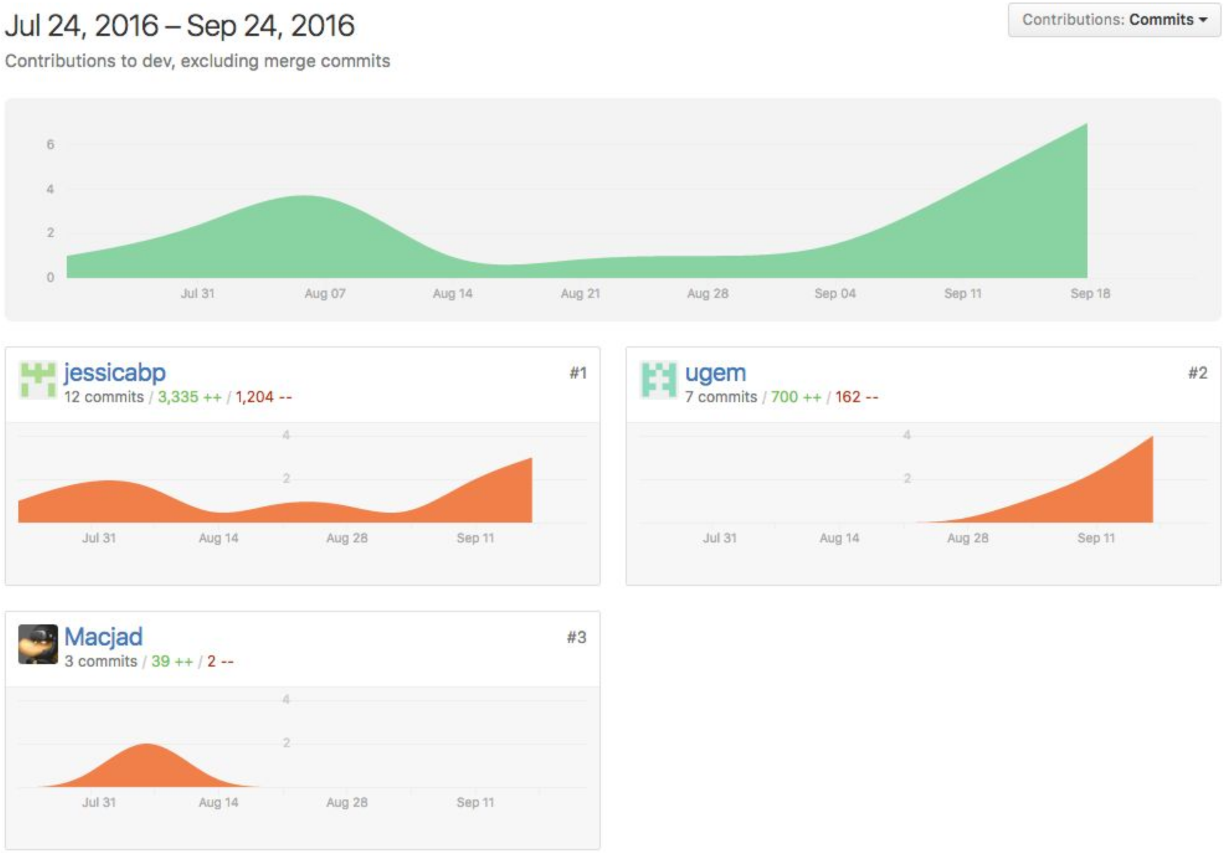
Branching

Unlike the server, the app has multiple branches. Earlier in the project we created a variety or branches as we worked on different proof of concept prototypes. Since that time all app development has occurred in the dev branch, with the only branching from there occurring when multiple team members worked in parallel and had to then merge their changes back together. Once we are happy with the level of functionality provided and stability of the app we will merge it back into master and continue development on the dev branch.



Commits

There was a clear period over mid-semester break where there were very few commits. This is due to not much development happening over that time as team members were busy and we found it much harder to communicate which further hindered work. Early on there were commits from James as he set up the TravisCI integration, but since then the commits have been shared fairly equally between Jessica and Umesh. All commits are commented.



Jul 24, 2016 – Sep 24, 2016

Contributions to dev, excluding merge commits

Contributions: **Commits** ▾

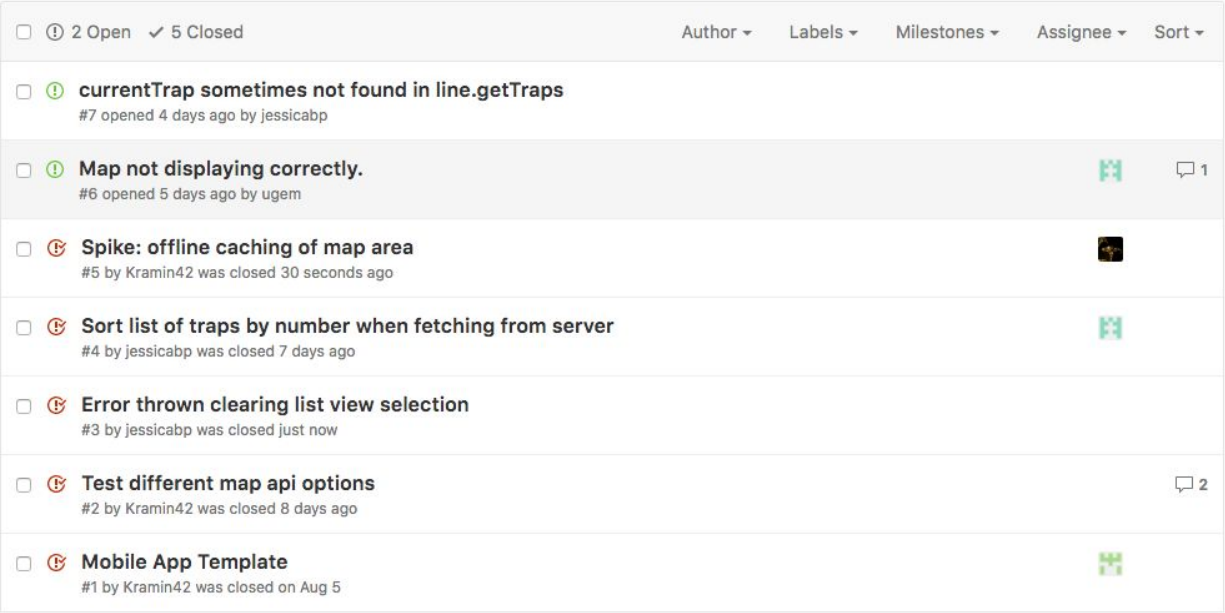
jessicabp #1
12 commits / 3,335 ++ / 1,204 --

ugem #2
7 commits / 700 ++ / 162 --

Macjad #3
3 commits / 39 ++ / 2 --

Issues

As with the server, there are not many issues in the GitHub issue tracking system as we primarily use Trello for new tasks, and only use GitHub for bugs and problems in the code. Also, as we manage the issue as a task in Trello, GitHub is not kept updated, which explains why they are not always closed in a timely manner or assigned to a user in GitHub.



□ 2 Open ✓ 5 Closed

□ **currentTrap sometimes not found in line.getTraps**
#7 opened 4 days ago by jessicabp

□ **Map not displaying correctly.** 1
#6 opened 5 days ago by ugem

□ **Spike: offline caching of map area**
#5 by Kramin42 was closed 30 seconds ago

□ **Sort list of traps by number when fetching from server**
#4 by jessicabp was closed 7 days ago

□ **Error thrown clearing list view selection**
#3 by jessicabp was closed just now

□ **Test different map api options** 2
#2 by Kramin42 was closed 8 days ago

□ **Mobile App Template**
#1 by Kramin42 was closed on Aug 5

Automated Regression Testing

Server

On the server, Flask allows us to create a test client which lets us to send requests to a wrapped application through the URL mapping in the server. The server tests using the Python unittest module, currently test the functionality of the API by analysing the responses returned from the client and checking the data stored in the database to check for integrity of data stored and the state of the database. All tests are set up by creating the test client and inserting test data into the database. Tests run include checking responses from the API and making sure that the intended functionality of the API is correct, and checking that the correct exceptions are thrown depending on the input to the API. Through the use of the test client, whenever commits are pushed to the repository, Travis CI is able to set up the correct environment that the server operates under (creating database and logging files), and automatically run the tests. The test module is able to be run as a normal module.

App

The app currently contains whitebox unit tests. These are used to test that functions within classes perform their expected tasks. Currently there are unit tests for four of the classes, but we are currently working on increasing our class coverage. These tests can be run by running the test classes they belong to, or by running gradle test. They also run by TravisCI each time a new commit is pushed to the repository.

Other Quality Assurance

Spike

A spike written up while deciding our maps API can be found here.

End User Testing

We have shown our concepts and progressing app to people who might use the app and have received feedback about what they like and dislike about the user interface. For instance, they commented that they are likely wearing gloves or have dirty hands while walking, so want to touch their phone as little as possible. This led us to try and figure out how we can tell them where the traps are even if GPS is unreliable and they aren't always looking at their phone. It also led us to create a simple interface with the most used options easily available. It was also noted that the four options of species did not cover everything, so we need to be sure to allow users to select less common options.

Platform Compatibility Testing

We have had some difficulty getting the app to run on actual mobile devices, so the majority of our testing has been performed on computers running the app of their desktop. We have however also run the app in mobile simulators to test the UI and compatibility with mobile platforms. This includes running an iPhone simulator to ensure the buttons appear as expected.

Code Walk-Throughs

We have performed limited code walk-throughs within our team, and this has resulted in some refactoring. For instance when going back over the Walk class, it because apparent that some properties were set but never really used, such as startTrap and nextTrap. After spotting this, we removed the properties and refactored the code to reflect this (as seem in the diff here). Another was the going through the functionality of the app, changing the location of where the data is formatted as a response, and also checking to make sure that we pick up on any exceptions that the server would be able to generate.

Product

Executing

Our code can be found on GitHub, with one repository for the app, and another for the server. The server is running at traptracker.pythonanywhere.com. The app can be run on desktop using the “gradle run” command, or (with the right computer and set up) in an iPhone or Android simulator using “gradle launchIPhoneSimulator” or “gradle android”. It will connect to the server and fetch information from there.

Demonstration

During week 9 we plan to demonstrate our product running, connecting to the server, selecting and completing a line, and sending data back to the server.