Ad Auction Increment 1 - Group 38

Team Members

- Alex - tll1g19
- Carlos - cab1g19
- Jamaal - jm3g19
- Stoyan - sv1u19
- Tom - tdh1g19
- Vlad - vgh1u19

Project Overview

The application compiles and runs as expected, without errors. The application contains everything we agreed on in the first increment. We implemented the functionality to have an optimised way of reading, see all the metrics calculated from start to end and see the chart for the impressions from start to end. Also, we added the GUI that we agreed to in the first increment, which is easy to use and has a modern look. The planning was changed a little bit, which can be seen in the Actual Burndown Chart due to the changes in the deadlines, and some minor changes, but everything has been finished in time.

Key Design Choices

- Separate chart calculator and metrics calculator
- Multi-threading for GUI so it runs faster
- “Add to Compare” button in charts page instead of “Compare page” so that the graphs are computed only once
- Key GUI Decisions

Design Choices Discussion

Backend

Key Backend Design Decisions

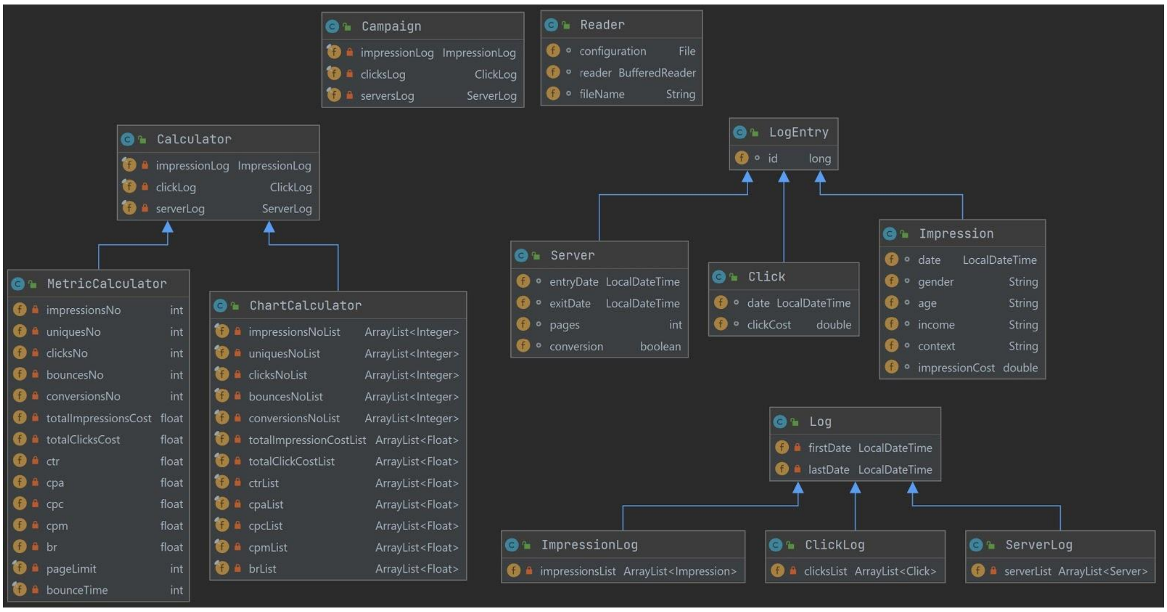
- Separate chart calculator and metrics calculator to work on different goals independently
- Multi-threading for GUI to run tasks in parallel and improve overall performance
- One-off reading of the file to reduce unnecessary computation and incredibly slow performance
- Object-orientated approach for organised code and data storage techniques for easier visualisation of project structure for development demonstrations

Separate chart calculator and metrics calculator

We have made some design choices that will facilitate the development and the maintainability of our code.

The architecture we have chosen for our backend is having separate calculators for the metrics and the chart. This will facilitate the development because sometimes we only need the metrics to be calculated, not everything for the chart and in this way it will reduce the time of processing of our program. We kept in mind the first feedback that we received and we decided to think of a good way of reading from the start, so our reading is fast and the computing of the metrics in the backend is separate from the computing of the charts which is making our programme work a lot faster because we have high cohesion and low coupling.

THIS IS THE UML for our backend, where you can see the classes and the separation of the Calculator class with 2 subclasses: MetricsCalculator and ChartCalculator.



Multi-threading for GUI

We decided to have the GUI on a different thread than the coding of the backend, so it won't interfere with the reading and calculations. In this way, because the GUI runs on its thread the initial loading time for our app is hugely reduced, because the GUI is created on a different thread at the same time as the reading is done, so as soon as the app is run, the GUI is loaded.

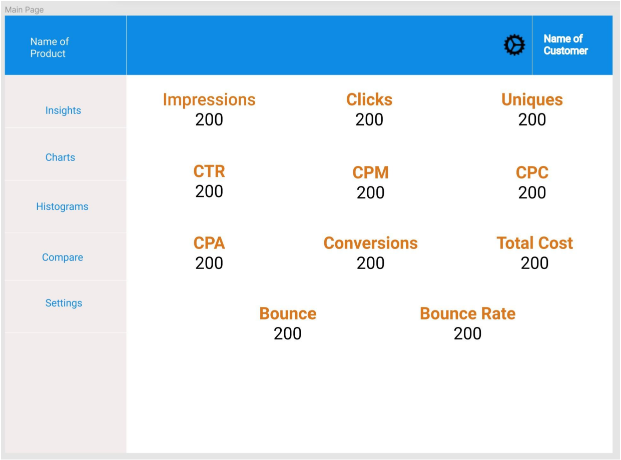
Backend future improvements

- Improved file reading due to potential problems with anomalous records, format changes, CSV file selection and file reading performance
- Controller code to improve Model and View integration and improve user interaction performance

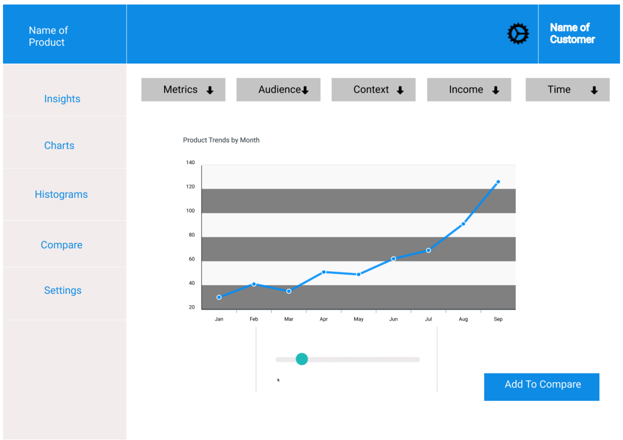
Key GUI Design Decisions

- "Add to Compare" button in charts page instead of "Compare page" so that the graphs are computed only once
- The vertical menu on the left side always visible for ease of access
- "Setting" was also added on the vertical menu instead of the small icon on the top of the page for a more streamlined design easier to understand

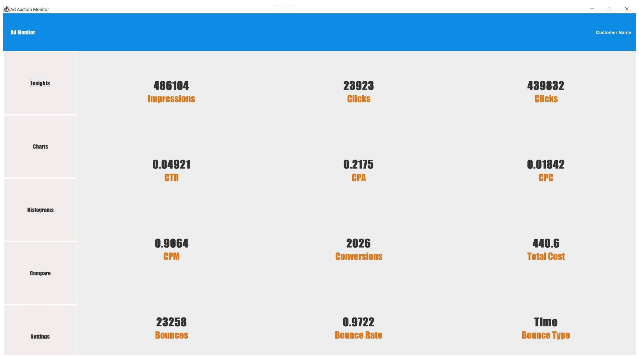
Initial GUI Prototype



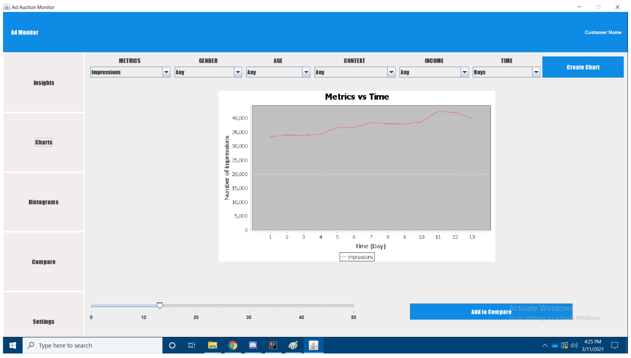
Impressions 200
Clicks 200
Uniques 200
CTR 200
CPM 200
CPC 200
CPA 200
Conversions 200
Total Cost 200
Bounce 200
Bounce Rate 200



Actual GUI with all the decisions applied



486104 Impressions
23923 Clicks
439832 Clicks
0.04921 CTR
0.2175 CPA
0.01842 CPC
0.9064 CPM
2026 Conversions
440.6 Total Cost
23258 Bounces
0.9722 Bounce Rate
Time Bounce Type



METRICS: Impressions
GENDER: Any
AGE: Any
CONTEXT: Any
INCOME: Any
TIME: Days

GUI Design Changes

- Added "Create Chart" Button on the charts page
- Added "Bounce type" in the insights page grid for easier understanding of bounce metrics
- Split "Audience" into "Gender" and "Age" for improved filtering

"Add to Compare" button on the charts page instead of the "Compare" page

We decided to have an "Add to compare button" on the "Chart page", where the charts are created so that after the graph is created and the user wants to compare it, the graph is already created and just added to the compare page. Doing this will improve the efficiency of our program. If we had the buttons to create the charts on the compare page, then it would've recreated the chart, which is useless, because we already have it computed. So this is a GUI design choice we made to enhance the efficiency of our application and to improve the user experience. Having an Add to Compare button on each chart is easier to be used than having to rechoose all the filters on another page.

Prioritisation

We used some prioritisation to ensure that the best value is delivered in time because the supervisor told us that the code should run fast and that the reading part should be efficient: so we decided to prioritise having the reading and backend part done optimised, so our code runs faster and we created a well-designed modern-look GUI, which is easy to use by the user and it is really simple to be understood. We have decided to do so in our first increment because we thought this gives the most value to the user and can be delivered in time.

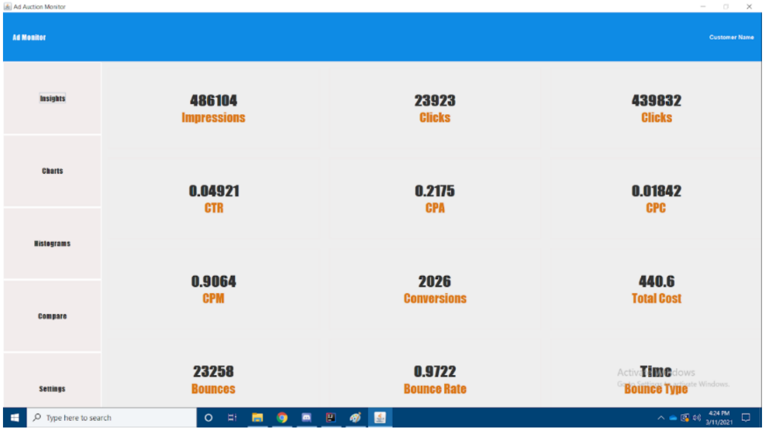
Key test outputs

We have designed some tests to see the correctness of our application and verify if it is optimised”

1) Metrics page and charts page created using the provided excel sheet (2 weeks)

This file test has: ~400.000 inputs

- Runs in 3.30s
- Goes well with our code architecture and (the classes we used and model we chose – see the UML)
- Gives the correct output
- Is optimised


Ad Auction Monitor

Ad Monitor

Customer Name

486104 Impressions

23923 Clicks

439832 Clicks

0.04921 CTR

0.2175 CPA

0.01842 CPC

0.9064 CPM

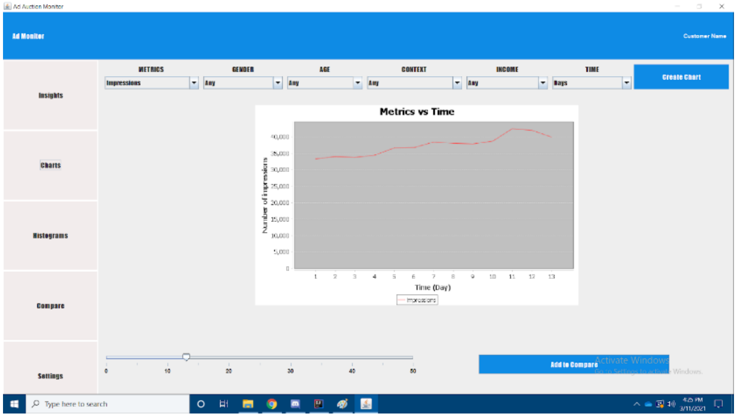
2026 Conversions

440.6 Total Cost

23258 Bounces

0.9722 Bounce Rate

Time Bounce Type


Ad Auction Monitor

Ad Monitor

Customer Name

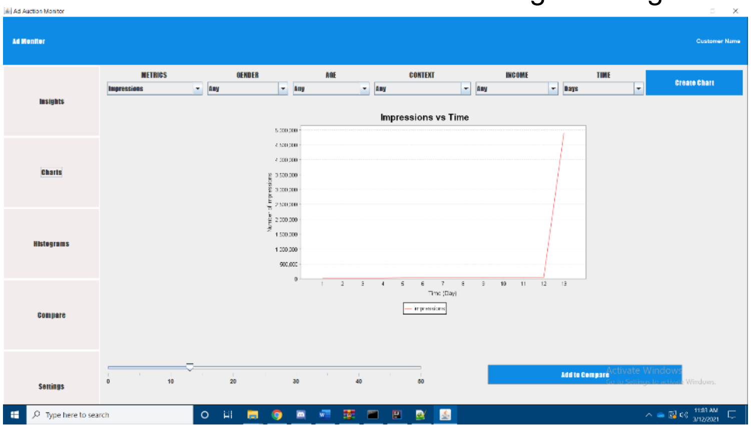
METRICS: Impressions; GENDER: Any; AGE: Any; CONTEXT: Any; INCOME: Any; TIME: Days

Metrics vs Time

2) We created another test with a larger input

The file test has: ~5,4 million inputs

- Runs in 11.28s
- When using larger inputs, it still goes well with the coding design we chose (the classes we chose and the code architecture – see the UML)
- Gives the right output
- Is efficient when working with larger files

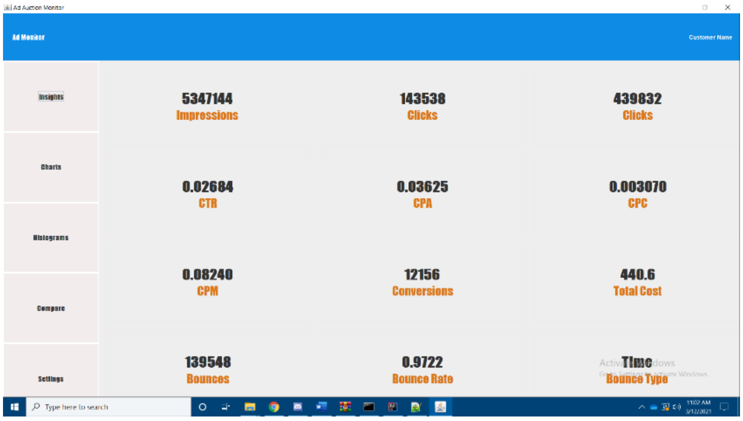

Ad Auction Monitor

Ad Monitor

Customer Name

METRICS: Impressions; GENDER: Any; AGE: Any; CONTEXT: Any; INCOME: Any; TIME: Days

Impressions vs Time


Ad Auction Monitor

Ad Monitor

Customer Name

5347144 Impressions

143538 Clicks

439832 Clicks

0.02684 CTR

0.03625 CPA

0.003070 CPC

0.08240 CPM

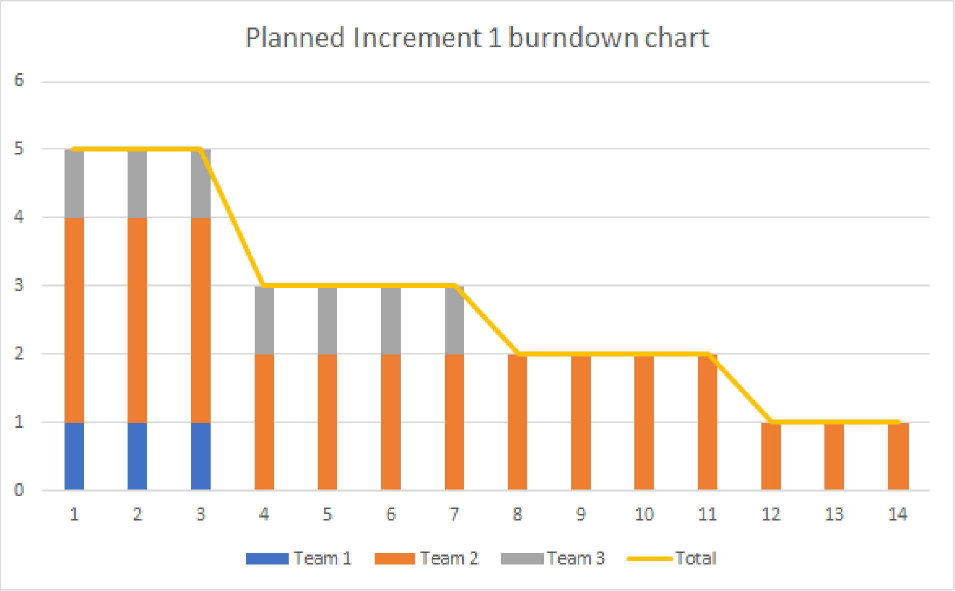
12156 Conversions

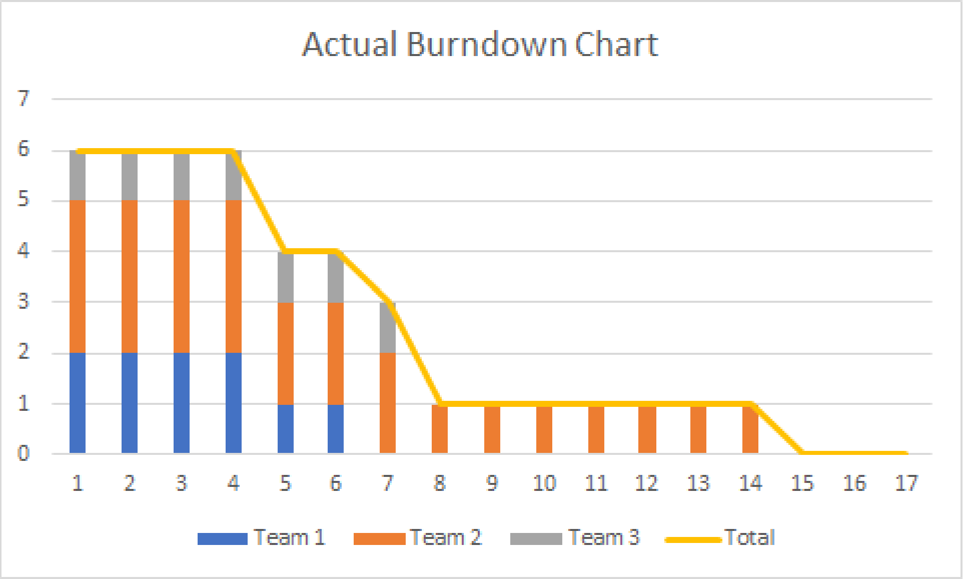
440.6 Total Cost

139548 Bounces

0.9722 Bounce Rate

Time Bounce Type





Actual Burndown Chart

Day 1-4

Early code for reading metrics
Designed GUI

Day 5-7

Code to calculate chart data
Performance optimisation
Basic GUI coded (Menu and metrics page)
Code to display metrics as charts

Day 8-14

GUI can now display charts

Day 15-17

Documentation

Increment 2 Sprint Plan

- Give user time granularity control over charts (Medium)
- Display metrics as a histogram (Large)
- Give user control to filter metrics and charts by date range etc. (Large)
- *GUI customisation (font, colours, graph)* (Small)
- *Display metrics over time* (Small)
- Coding design of pages of “compare metrics”, “histogram” and “settings” (Large)
- Adding functionality to the “compare metrics”, “histogram” and “settings”(Medium)
- Compare graphs (two lines) (Medium)
- Testing Functionality of Program and making adjustments (Medium)

The tasks are coded to t-shirt sizes to represent the size of the task and the work it will likely take to complete.

All Teams:

- Testing Functionality of Program and making adjustments

Team 1:

- Metrics over time
- Time granularity calculations
- Calculate filters

Team 2:

- Coding pages of “compare metrics”, “histogram” and “settings”
- Adding functionality to the “compare metrics”, “histogram” and “settings”
- GUI customisation

Team 3:

- Histograms
- Compare graphs (two lines)

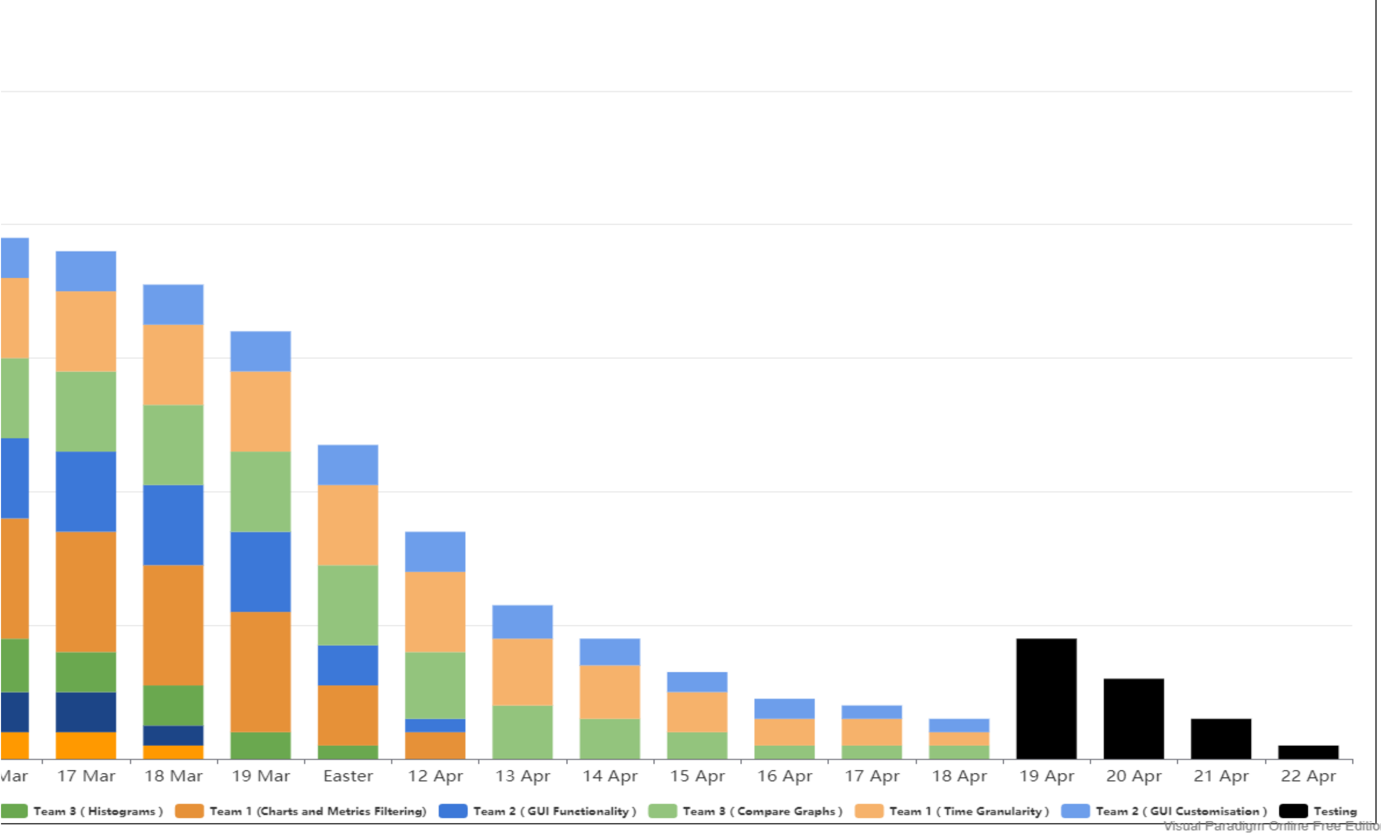
Day	Team 1	Team 2	Team 3
Increment 1 Deliverable Day			
12/3	Metrics over time	Coding pages of “compare metrics”, “histogram” and “settings”	Histograms
13/3	^		^
14/3	^		^
15/3	^		^

16/3	^		^
17/3	^		^
18/3	^		^
19/3	Calculate filters	Adding functionality to the “compare metrics”, “histogram” and “settings”	^
Easter			
12/4	Calculate filters	Adding functionality to the “compare metrics”, “histogram” and “settings”	Compare graphs
13/4	Time granularity calculations	GUI customisation	^
14/4	^		^
15/4	^		^
16/4	^		^
17/4	^		^
18/4	^		^
19/4	Testing Functionality of Program and making adjustments		
20/3			
21/3			
22/3			
Increment 2 Deliverable Day			

Days highlighted in red will represent quick SCRUM meetings where we address what we are currently working on and determine if we are still on schedule.

See the burndown chart below.

Increment 2 Burndown Chart



For the burndown chart, we tried to add more detail by splitting the tasks into three main sections. The colours indicate our teams (so long as we have 2 people working together then we won't enforce specific people to certain roles.) Blue for team 1 (backend) red for team 2 (GUI) and green for team 3 (new features.) The final stage of the sprint plan will focus on the testing of functionality, a mistake we learnt from the end of increment 1.