Software Engineering Group 3

Increment 3

Advertising Auction Dashboard

- Samuel Beresford
- Matthew Consterdine
- Emma Gadsby
- Matthew Langford
- Iovana Paylovici

Increment Planning

The final increment mostly consists of user stories related to supplementary requirements. We intended to implement the features that correspond to the first user story in the second increment but due to necessary re-prioritisation we had to defer the task. Furthermore, we decided that the next two 'MUST' tasks are crucial for the application to be received well by the user and offer enough functionality. Ideally, we would like to add each feature that corresponds to the user stories and maybe add further functionality if the time allows us.

Increment 3
As a client of the advertising agency, I want to be able to see a visual representation of the click costs, so that I can observe the distribution of click-cost.
As a client of the online marketing agency I want to be able to compare different metrics not only by looking at each separately, but placed on the same graph as this would save me a lot of time.
As a disabled user I want accessibility features such as colour blind mode or increased font size so that I can use the software as well as my well abled counterparts.
As the CEO of the marketing agency I want to be able to compare multiple campaigns so that I can quickly assess which approach is best for certain client requirements.
As a client of the advert agency I want the application to output a representation of campaign performance according to time of day so that I get a better idea of the browsing habits of our audience.
As the marketing manager I want the tool to save the graphs to an image so that I can come back to them without running the application again.
As somebody who prefers working with scatter graphs I want the tool to provide the option to change the type of graph so that I can decide which type is best in that situation.
As an employee who needs to present monthly reports to the owner of the company I want to be able to print from the application itself, so that the chart is nicely formatted and printed without any further work.

Fig. 1 Table showing the third Increment

Sprint Backlog for Next Increment

Due to a partially misleading representation of the story points in the previous increment we have assigned more priority to tasks such as creating Unit/Regressions tests in this increment. This is because we want to offer an application with no faults. Nevertheless, our main priority for the last increment is to provide a great user experience meaning that we will use the time wisely researching the needs of our users and implementing them. Although we have now had some experience with the project and could estimate the number of hours necessary for each task we have decided to continue using story points as it is both easier to answer and it conveys the actual complexity.

Task	Priority	Points
Implement histogram of click costs.	Major	5
Refine ability to overlay graphs with different metrics.		5
Create Unit/Regression tests.		6
Create data for multiple campaigns.		2
Load multiple campaigns.		2
Compare graphs from different campaigns.		4
Implement features such as increased font size and other colour schemes.		3
Implement option for the user to choose any time of the day or week.	Minor	4
Implement time scale.		4
Add a save as image option.		2
Add more types of graphs.		4
Print from File tab.		3
Add a save as PDF option.		3
Add content to help section.		3

Fig. 2 Table showing the tasks to be done in the next sprint

Scenarios and storyboards

Scenarios proved to be a useful tool in the development of the product as it gives insight into how a user would work with the application. The first scenario we included describes the user interaction with a Splash screen which has been helpful in creating the final design. The second scenario corresponds to a task from the next increment. In order to provide each of our users the same satisfaction we think that the tool should provide accessibility features for disabled users.

Scenario 1 - Increment 2

1. Elena just got employed at a small company and she's responsible for the marketing of the company. Luckily, the company has just received a tool from the online advertising agency which analyses the performance of their campaigns. Elena opens the tool and she gets presented with a Splash screen.
2. The screen asks for a campaign to be imported so Elena browses the file directories until she finds the latest campaign and then clicks the Import button.
3. The tool shows an error as she did not name the campaign, she then names the campaign and proceeds to import it.
4. After waiting for it to load the tool opens successfully and she goes on analysing the campaign.

Scenario 2 - Increment 3

1. Andy is a software engineer who is currently working on a tool which analyses the performance of an online marketing campaign. He and his group have implemented the required functionality and they're now working on improving the user experience. In order to check this, he proceeds testing the tool by using it.
2. He imports a campaign and proceeds to filter the results by demographics and other key metrics.
3. Although he is happy with the overall design of the application, he is aware that the user experience is extremely important and so he decides to research into this.
4. He then consults his team about adding features for colour-blind and other disabled users.
5. The team starts by using appropriate colour schemes, placing the legend in the charts and adding other features for those visually impaired.

Design Choices

Due to the feedback we received in the last increment, we decided that a Splash screen would be the best solution for importing/opening the CSV files. In order to import the data the user has to choose all three files (i.e. server, click and impression tables) and then name the campaign. We chose to show an error if the user does not name the campaign, as we thought that having a default name would be frustrating for the user when trying to re-open it from the saved files. Although initially we were going to have the option to either import in the folder without opening it or to open it and select the files, we could not accomplish that as JavaFX does not support having both DirectoryChooser and FileChooser at the same time. Thus, we chose to have the user pick the files, as it allows them to confirm that they are present. Moreover, the Splash screen has a loading bar. We think this is a plus for the user as the wait for the files to load can be seen on the bar and will probably be less stressful.

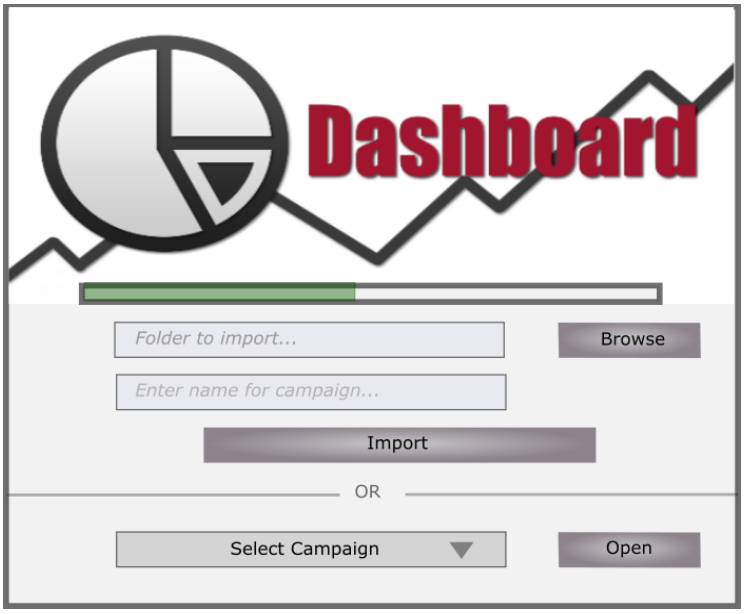

Fig. 3 A design for the Splash Screen

Fig. 4 The design in use for the Splash Screen

As the calculations for the metrics took a while, we used a background thread for the calculations to enable the user to continue using the auction tool.

To maximise space for the graph we moved the table that displays the metric results to the left pane. You can use the table to add different series to the graph (shift + select or ctrl + click). The overlay function also works with different filters by selecting (from drop downs) and adding them (add button).

Burndown chart

Although we tried to follow the ideal line of the burndown chart, some of the tasks proved to take far more time than we initially thought they would. As an example, calculating the graphs for CTR, CPA, CPC, CPM took us a few solid hours of work. We also had to refactor some of the SQL queries as they would fail the JUnit tests added later on. Further on, tasks such as comparing graphs by overlaying them took a long time to be completed as unfortunately we did not decide on the design approach beforehand. In the end we decided to use the JavaFX overlay functionality which proved to be a great solution although we consider this task to be only partially done and will be completed for the next increment.

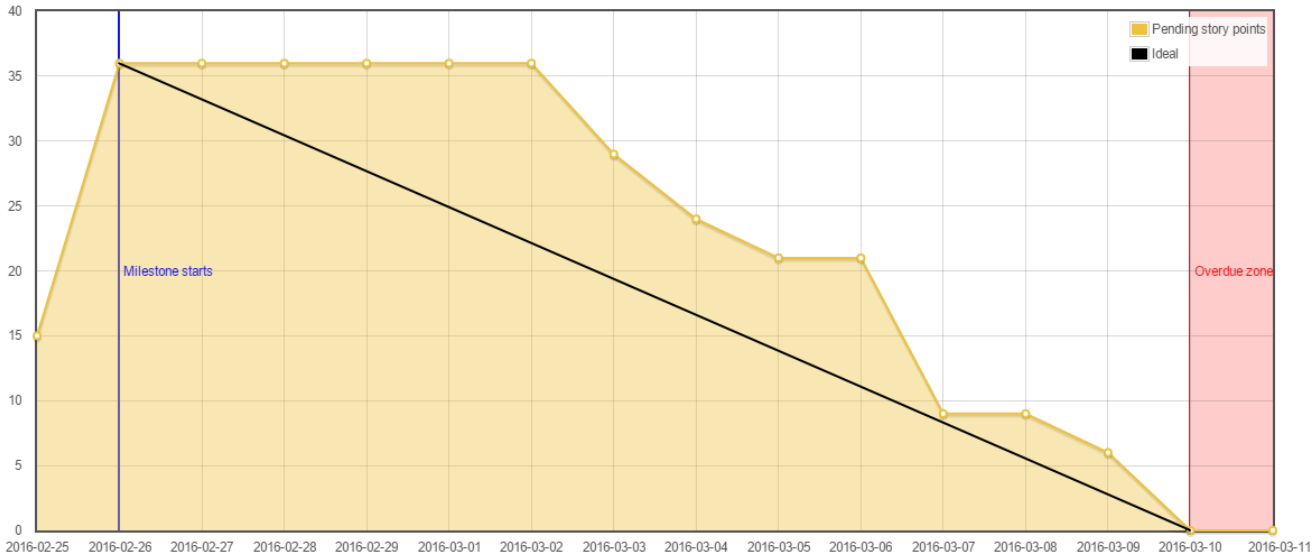

Fig. 5 Illustration of the implementation progress over time for the second increment

Testing

Unit testing

In order to check the correctness of the graphs constructed by the application, we created a suite of JUnit tests, to compare our results to what we have manually calculated.

Each set of tests is associated with one type of graph, for example: ImpressionsDataTest checks the output of ImpressionsGraphConstructor. There is one test class for each constructor class. The tests compare the Y-value of the first data point generated for all of the filters that can be applied to the graph constructor, so there are tests for each option in:

Time granularity, Gender, Age, Income, and Context.

With the provided test data, the only practical way to find out the true values of these tests would be to conduct queries on the database. Obviously, if there are flaws in the query, the false positives will appear when it is reused in the actual application. To solve this problem, we created a quick Java application that automatically generates three CSV files, from which a database with known values can

be constructed. We know that the code that constructs a database from CSVs is correct thanks to tests conducted in the previous increment.

The Java application constructs a list of all possible combinations of user details (180 unique users, given by |Gender x Age x Income x Context| = 2 x 5 x 3 x 6 = 180 - see an extract of this list in Appendix A). Then, as long as this group of users always appears as a whole in the data, we can be certain that the proportion of users that fit a criteria is the same as the proportion of users in the list. For example: exactly half of all impressions will fit the filter 'Gender = Female', and one fifth of clicks will satisfy 'Age = <25'.

This test suite allows us to perform regression tests, as the tests themselves can be constructed before all of the functionality has been implemented.

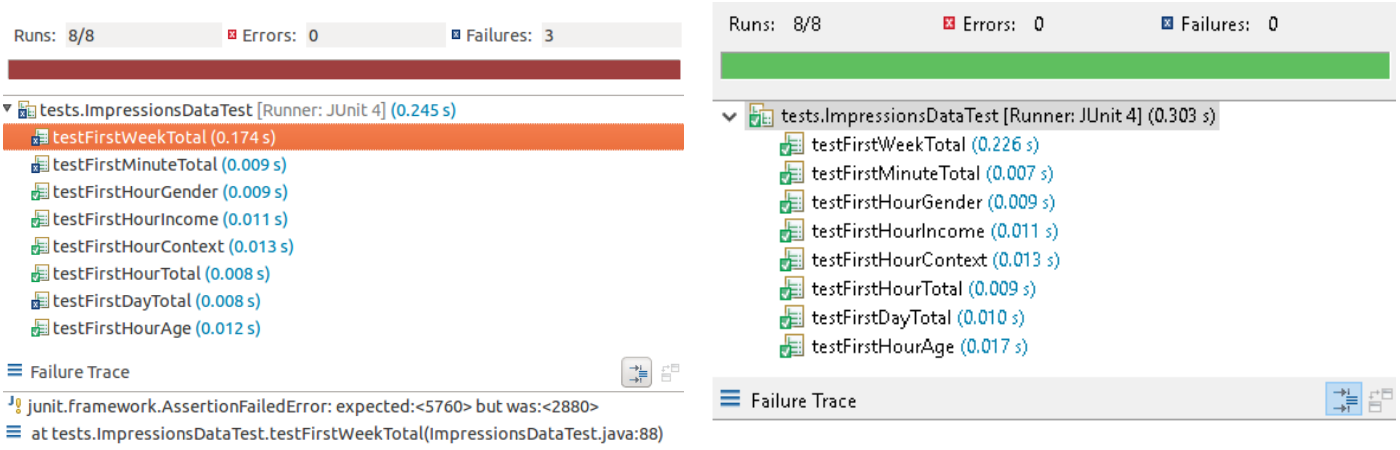

Fig.6 Unit test results during and at the end of the sprint, testing as we add features.

A/B Testing

In order to test the current design of the UI we have performed a series of tests, one being a form of A/B testing. That is, we compared the control version (the design we used for the first increment) and the new interface design. The new application design uses an accordion layout in order to encapsulate the large blocks of text, resulting in the view being less cluttered - but possibly harder to use. We also added further functionality to the application and then presented each version to a group of users. Our goal was to check both the design and the functionality. In order to do this we asked each group to rate the interfaces on a scale from one to five, where five means great, based on the two criteria. Based on the results, the control version had a much better impression on the design but it almost equalled the B version on functionality. As the group sizes were small we tried not to look too much into the results as they could be misleading. However, due to lack of time we decided to continue using our first version. Nevertheless, our purpose has been achieved as this way we had both the application tested and the UI.

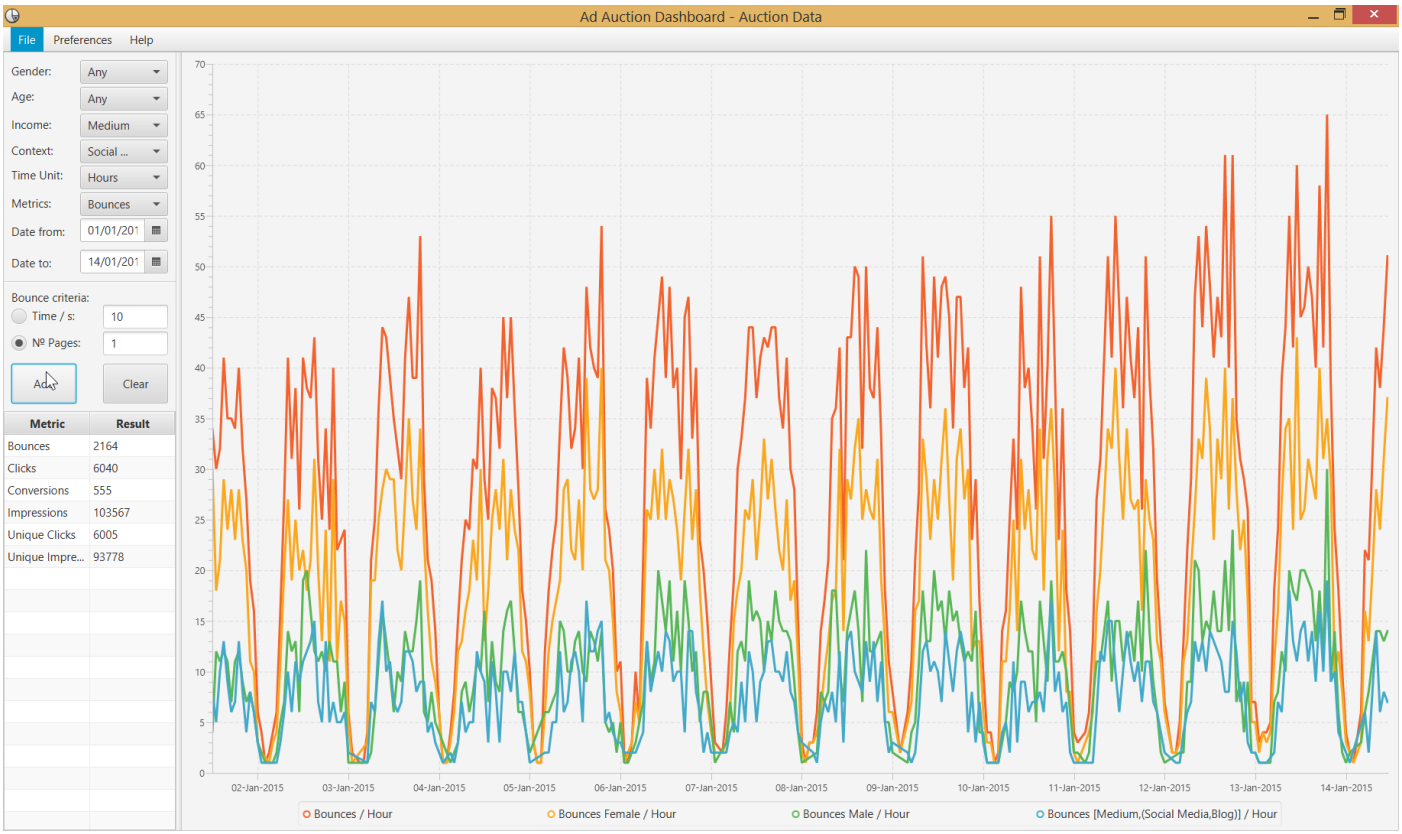

Fig. 7 Control version or version A

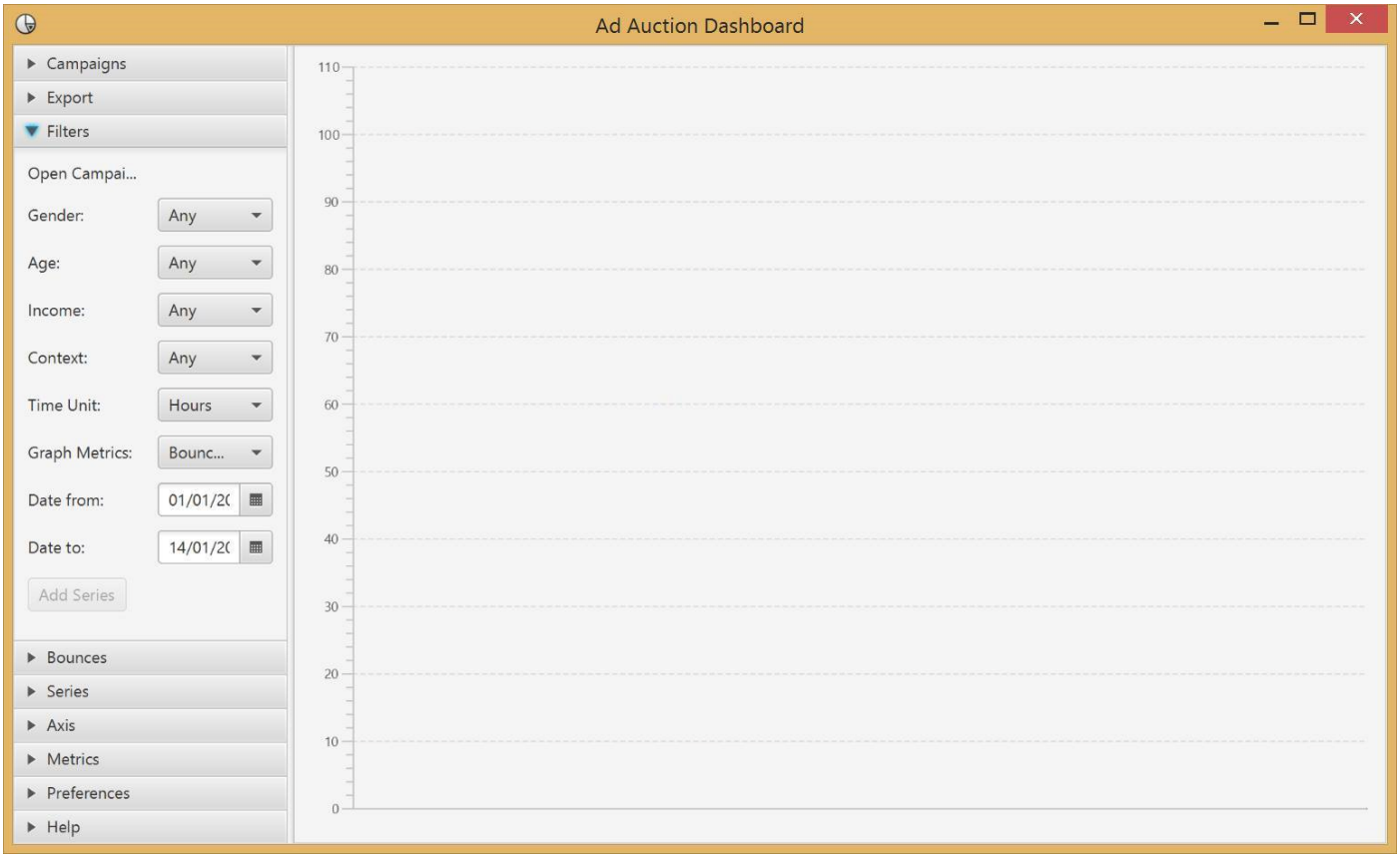

Fig. 8 Image of the accordion layout interface - version B

	Version A		Version B	
	Design	Functionality	Design	Functionality
Group A	3.7	3		
Group B			2.9	3.2

	Design		Functionality	
	Version A	Version B	Version A	Version B
	3	3	4	5
	5	3	3	4
	4	2	3	2
	4	3	2	2
	4	3	4	4
	3	2	3	3
	2	2	4	3
	4	4	3	2
	4	5	2	4
	4	2	2	3

Fig. 9 Test results for each version

Code Review

As in the previous increment we followed an MVC approach in order for each team member to work on what they are best at. This has allowed to review each other's code in isolation, as soon as it was pushed on sourcekettle thus enforcing good programming practices.

Appendix A: Extract of users used to construct test database

User ID	Gender	Age	Income	Context
1	Male	<25	Low	News
2	Female	<25	Low	News
3	Male	25-34	Low	News
4	Female	25-34	Low	News
5	Male	35-44	Low	News
6	Female	35-44	Low	News
7	Male	45-54	Low	News
8	Female	45-54	Low	News
9	Male	>54	Low	News
10	Female	>54	Medium	News
11	Male	<25	Medium	News
12	Female	<25	Medium	News
13	Male	25-34	Medium	News
14	Female	25-34	Medium	News
15	Male	35-44	Medium	News
16	Female	35-44	Medium	News
17	Male	45-54	Medium	News
18	Female	45-54	Medium	News
19	Male	>54	Medium	News
20	Female	>54	Medium	News
21	Male	<25	High	News
22	Female	<25	High	News
23	Male	25-34	High	News
24	Female	25-34	High	News
25	Male	35-44	High	News
26	Female	35-44	High	News
27	Male	45-54	High	News
28	Female	45-54	High	News
29	Male	>54	High	News
30	Female	>54	High	News
31	Male	<25	Low	Shopping
32	Female	<25	Low	Shopping
33	Male	25-34	Low	Shopping
34	Female	25-34	Low	Shopping
35	Male	35-44	Low	Shopping
36	Female	35-44	Low	Shopping
37	Male	45-54	Low	Shopping
38	Female	45-54	Low	Shopping
39	Male	>54	Low	Shopping
40	Female	>54	Medium	Shopping
...				