Software Engineering Group 3

Increment 1

Advertising Auction Dashboard

* Samuel Beresford
* Matthew Consterdine
* Emma Gadsby
* Matthew Langford

Iovana Pavlovici

Increment Planning

The feedback we were given points to a few issues with the user stories. That is the size of them, and their relation to the requirements. We tried to fix these issues in the next increment by rewriting the user stories, which were split between multiple increments, and by adding a higher dependency between these and the requirements. Furthermore, we separated the non-functional requirements from the rest as it seemed to be a better fit for the MoSCoW prioritisation style.

|  |  |  |
| --- | --- | --- |
| **Increment 2** |  | **Non-functional Requirements** |
| As an employee in a small business I want to filter by demographics and other factors so that we can tailor future campaigns towards these users. |  | A software developer who is part of the testing team I want the code to be separated into logical segments by functionality so that the testing goes smooth and I can report any issues as soon as possible. |
| As the owner of a travel agency I want to be able to filter by time unit so that I can check if people were more interested in our deals during specific months. |  | As an experienced software developer I want the whole coding process to be documented by the team so that any of our team members can work on the same classes if necessary. |
| As the head of the marketing agency I would like to be able to view more complex key metrics such as the bounce rate so that I can look at a more detailed analysis. |  | As a software developer I want to write code that is easily extensible so that if more functionality is needed the process will be quick and manageable. |
| As a website developer I want to be able to define the criteria for the bounce rate provided so that I can monitor how engaging the website is. |  | A software developer I want to use many external libraries in the project so that we can reduce unnecessary work. |
| As the CEO of the company I want to be able to look at multiple campaigns at a time so that we can compare the revenue quickly. |  | *Fig. 2 Table of non-functional requirements* |
| As the head of advertising for a company I want to be able to control the date range of the graphs so that I focus on specific data between two points in time. |  |  |
| *Fig. 1 Table showing the second Increment* |  |  |

# Sprint Backlog for Next Increment

A few of the ‘Must’ tasks were categorised as having ‘Minor’ priority, this is not due to the fact that we do not intend implementing them but because these tasks can be done towards the end of the increment as they are fairly easy to implement, consisting only of 1 Story Point.

|  |  |  |
| --- | --- | --- |
| **Task** | **Priority** | **Points** |
| Filter by demographics: age, income, and gender. | Major | 5 |
| Filter by date range. | 4 |
| User defined time unit | 2 |
| Implement the option to choose how a bounce is registered. | 4 |
| Show more than one graph so user can compare advertising campaigns. | 3 |
| Create Unit/Regression tests | 3 |
| Calculate a graph of CTR against time. | Minor | 2 |
| Calculate a graph of CPA against time. | 2 |
| Calculate a graph of CPC against time. | 2 |
| Calculate a graph of CPM against time. | 2 |
| Calculate a graph of the bounce rate against time. | 2 |
| Calculate a graph of total cost against time. | 2 |

*Fig. 3 Table showing the tasks to be done in the next sprint*

# Scenarios and storyboards

In order to include the key features in the UI we wrote a few scenarios for the first two increments. This was helpful for the next stage, which mostly consisted of creating storyboards. Although the scenarios do not cover all the features required in the UI, it has been helpful to have some description to relate to in the later stages of the implementation.

## Scenario 1 - Increment 1

1. Julian, the owner of an online marketing agency, has been very busy travelling lately. He wants to know if his employees did a good job with the latest client without having to call a meeting so he opens the tool he commissioned for, which analyses the advertising campaigns.
2. This is the first time he is using it so he hopes the interface will be intuitive enough not to ask for any indications from the employees. He clicks on the file tab and sees the ‘Import’ field. He then proceeds to click on the field.
3. This opens a dialog box and he further selects the folder containing the csv files provided to him by the advert network.
4. He looks at the features provided on the left side of the application and then selects the gender, age, income and context he’s interested in. He also chooses a key metric such as impressions. This generates a graph in the right pane.
5. He is mostly happy with the results and other key metrics such as unique clicks and conversions.

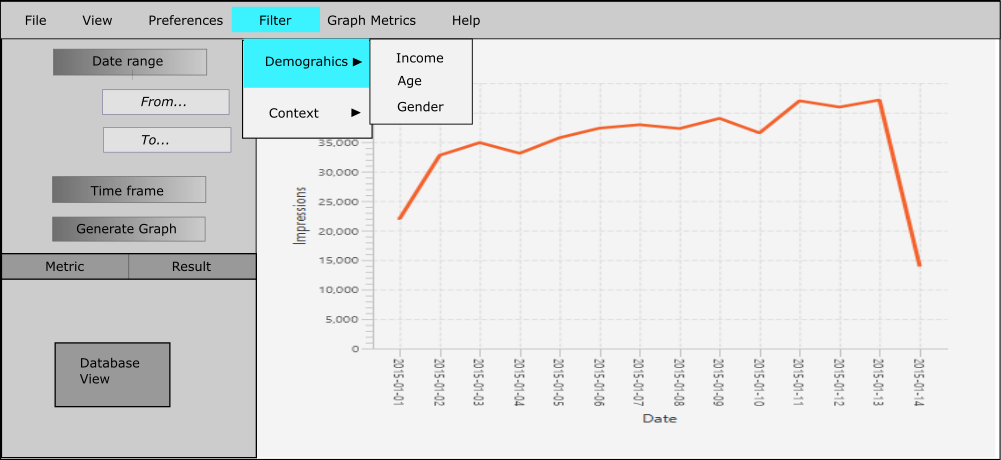
## Scenario 2 - Increment 2

1. Klaus is a very successful photographer who has just received a tool from the marketing agency that analyses the ad campaigns he commissioned for. He opens the tool for the first time and imports the files he was given from the Import field, under the File tab.
2. He proceeds to filter the results by demographics (gender, age and income), context, and date range. He also filters it by time units by choosing to look at specific hours.
3. He wants to know how many ad viewers have actually bought any of his photographs or commissioned him. He doesn’t know which key metric to choose so he heads to the Help tab.
4. He looks through the Help tab and sees that the key terms are explained in there. He realises he needs to look at the conversion rate.

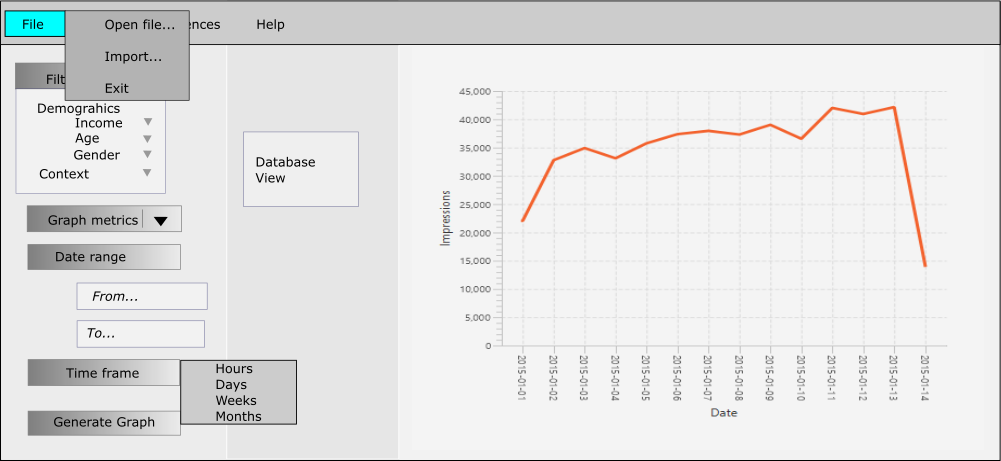
This generates a graph which demonstrates that the ad campaign has been quite successful.

# Creating the GUI

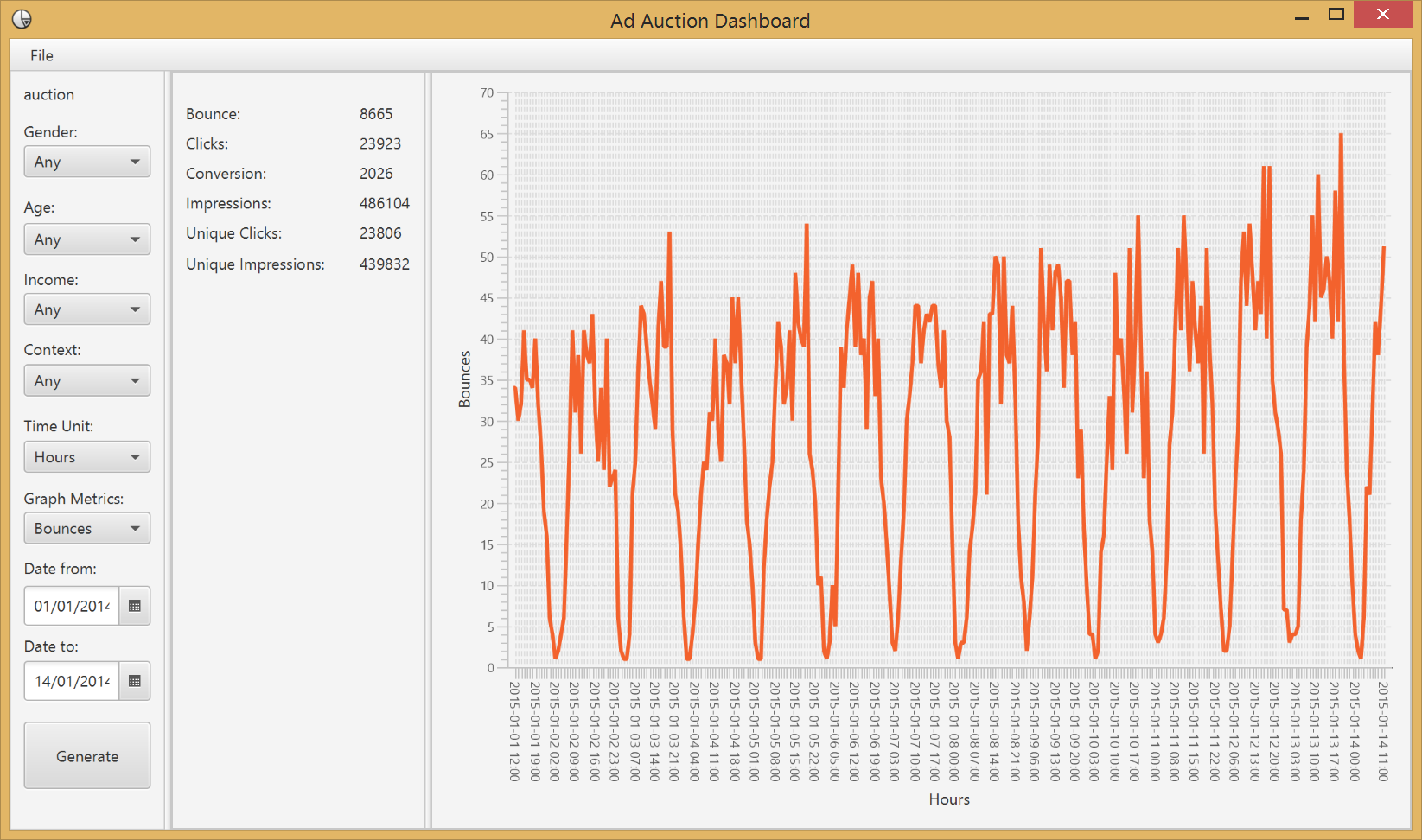
We decided to create a functional GUI that looks appealing to the user. That is why we created a few storyboards before jumping straight into the implementation stage. We chose to use CSS stylesheets to create a nice looking UI which will later on allow us to easily provide the users the option to change the colour scheme and the font settings



*Fig. 4 More user friendly interface*



*Fig. 5 More detailed Database View and more options in the left pane*



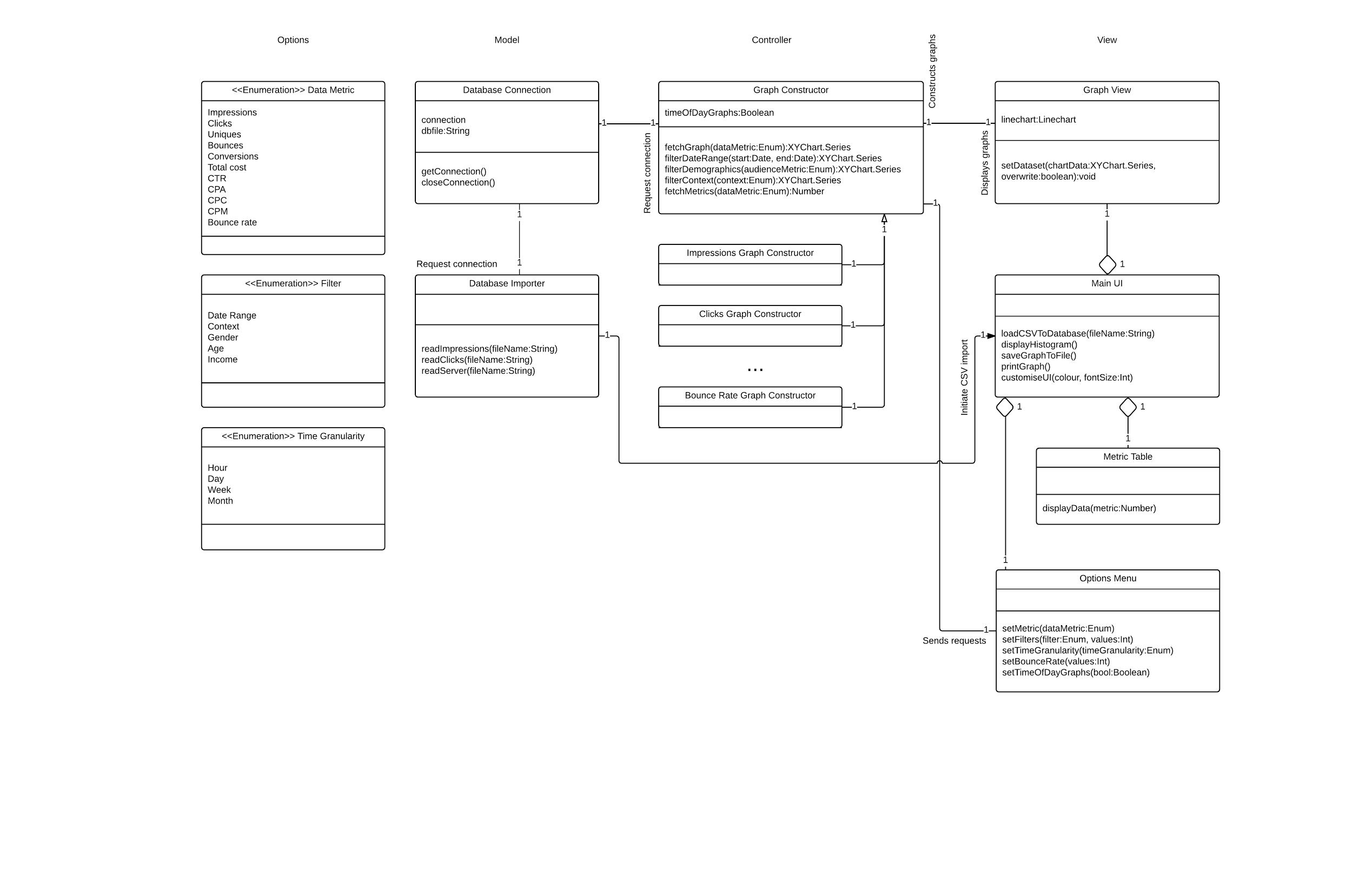
*Fig. 6 Current stage of the UI implementation*

The previous figure shows the current features of the UI, though not all of them are functional, also we did not yet include the Preferences tab as we are going to implement features such as changing the colour scheme and the font style in the next increments, this also applies for the Help tab which will be later on added to include tutorials on how to use the software and definitions of key terminology.

We have had two meetings during the past two weeks in which we mostly had positive feedback. Some of the changes we intend to implement in the later increments is the possibility of changing the type of graph (i.e. from line to scatter) and a login application for the marketing agency and their clients. Something else we discussed was whether the application should have a fixed size or not. We decided in the end that a resizable window is a better option as the users could have different types of monitors.

# UML Diagrams and MVC approach

An MVC approach seemed the most suitable strategy so to illustrate this we drew a class diagram. This includes how our classes are going to be partitioned in the project and the connections between them.

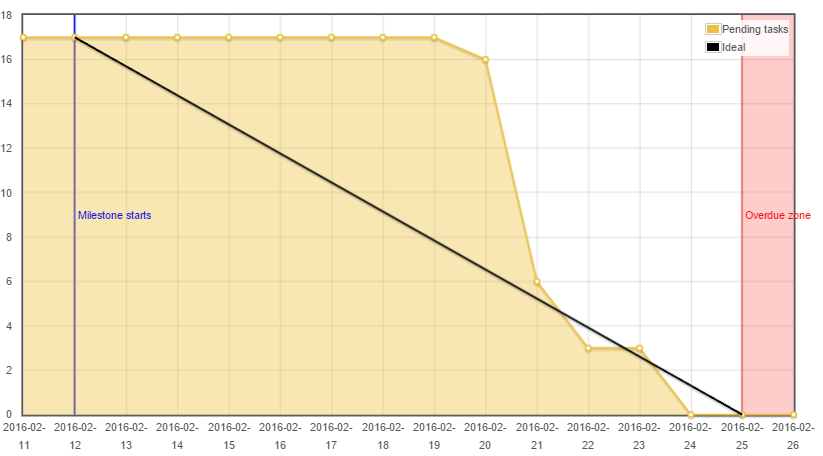


*Fig. 7 An illustration of how the classes are partitioned by the MVC approach*

The structure of the program is based on the passive MVC model: The ‘model’ components cannot communicate with the view directly, but in one instance the reverse is possible: the UI can send a direct request to the Database Importer class to import the campaign .csv files into the SQLite Database.

# Burndown Chart

As we focused on the larger tasks first, we didn’t manage to burn down tasks until about halfway through the increment. Once that was completed, we gained momentum and completed our set tasks a day before our deadline. We used the time to improve the report and improve the quality and user experience of our product.



*Fig. 8 Illustration of the implementation progress over time for the first increment*

# Testing stages

In order to test the correctness of the CSV importer class, we created a shell script to compare the entries in the CSV to the generated SQLite Database. We tested that the number of entries for each date were correct, to ensure that all of the tuples were imported:

## Impressions Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Entries CSV** | **Entries Database** | **Correct?** |
| 01/01/2015 | 22049 | 22049 | True |
| 02/01/2015 | 32773 | 32773 | True |
| 03/01/2015 | 34919 | 34919 | True |
| 04/01/2015 | 33111 | 33111 | True |
| 05/01/2015 | 35758 | 35758 | True |
| 06/01/2015 | 37379 | 37379 | True |
| 07/01/2015 | 37958 | 37958 | True |
| 08/01/2015 | 37311 | 37311 | True |
| 09/01/2015 | 39031 | 39031 | True |
| 10/01/2015 | 36562 | 36562 | True |
| 11/01/2015 | 42014 | 42014 | True |
| 12/01/2015 | 40945 | 40945 | True |
| 13/01/2015 | 42159 | 42159 | True |
| 14/01/2015 | 14135 | 14135 | True |

*Fig. 9 Table of correctness for counting the entries added to the Impressions table in the database.*

Tables for the Clicks and Server databases can be found in the appendix.

We also created a script to check that the tuples that were read into the database contained the correct data. Due to size and time constraints, it would be impractical to test every single tuple, so the compromise was to look at the first 1000 entries. For each table in the database, we compare the differences between the columns that contain numerical data and the list of differences is written to a file. These files are currently empty, meaning that the CSV and SQLite Database contain the same data as far as the tests check.

To help test our SQLite queries, we used a program called SQLite Database Browser(SDB). It allowed us to open .db files that our application creates while importing a campaign. We then used the execute query functionality of SDB to help us in writing and testing multiple query variations, to see which queries allowed the functionality we wanted - such as filtering based on other tables not used in calculating that key metric - alongside being quick as well as returning the correct data values.

In the next increment we will use JUnit to create Unit/Regression tests using dependency injection.

# Design choices

In order to create a nice looking application, we have used JavaFX for our user interface. It is part of the Java 8 standard library, and has fantastic support for graphs. We have also used ControlsFX to add the MultiComboBox.

The UI was mostly designed using the NetBeans scene builder because of its simplicity and flexibility, also because of its integration with the IDE (NetBeans). It allows us to simply drag and drop, and edit the interface in a fast and easy way.

We chose a simple layout to keep the application focused, so a new user can quickly learn how to use our application. The generate button is disabled until the user either imports or opens up a campaign. The campaign name will display at the top left of the GUI, the campaign name is also used as the database filename for simplicity.

Appendix A: Additional tables for correctness tests

Clicks Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Entries CSV** | **Entries Database** | **Correct?** |
| 01/01/2015 | 1079 | 1079 | True |
| 02/01/2015 | 1638 | 1638 | True |
| 03/01/2015 | 1754 | 1754 | True |
| 04/01/2015 | 1631 | 1631 | True |
| 05/01/2015 | 1747 | 1747 | True |
| 06/01/2015 | 1833 | 1833 | True |
| 07/01/2015 | 1857 | 1857 | True |
| 08/01/2015 | 1797 | 1797 | True |
| 09/01/2015 | 1957 | 1957 | True |
| 10/01/2015 | 1785 | 1785 | True |
| 11/01/2015 | 2042 | 2042 | True |
| 12/01/2015 | 2024 | 2024 | True |
| 13/01/2015 | 2053 | 2053 | True |
| 14/01/2015 | 726 | 726 | True |

*Fig. 10 Table of correctness for counting the entries added to the Clicks table in the database.*

Server Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Entries CSV** | **Entries Databse** | **Correct?** |
| 01/01/2015 | 1079 | 1079 | True |
| 02/01/2015 | 1638 | 1638 | True |
| 03/01/2015 | 1754 | 1754 | True |
| 04/01/2015 | 1631 | 1631 | True |
| 05/01/2015 | 1747 | 1747 | True |
| 06/01/2015 | 1833 | 1833 | True |
| 07/01/2015 | 1857 | 1857 | True |
| 08/01/2015 | 1797 | 1797 | True |
| 09/01/2015 | 1957 | 1957 | True |
| 10/01/2015 | 1785 | 1785 | True |
| 11/01/2015 | 2042 | 2042 | True |
| 12/01/2015 | 2024 | 2024 | True |
| 13/01/2015 | 2053 | 2053 | True |
| 14/01/2015 | 726 | 726 | True |

*Fig. 11 Table of correctness for counting the entries added to the Server table in the database.*