Software Engineering Group Project

Group 36: Project Increment 1

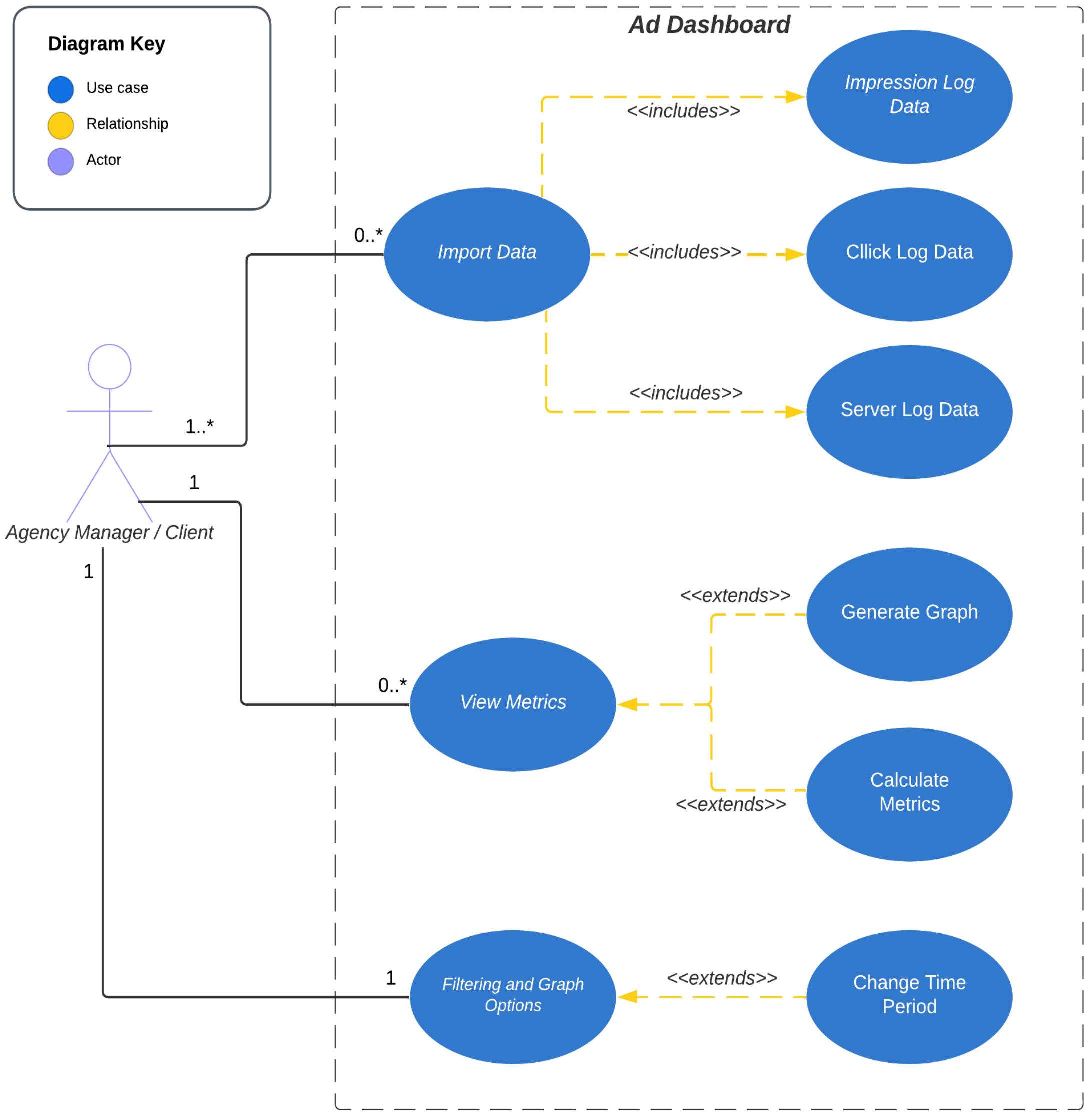
Connor Calkin, Daniel Braghis, Benjamin Lewis, Lucas Sayers, Pingding He

1. DESIGN ARTIFACTS

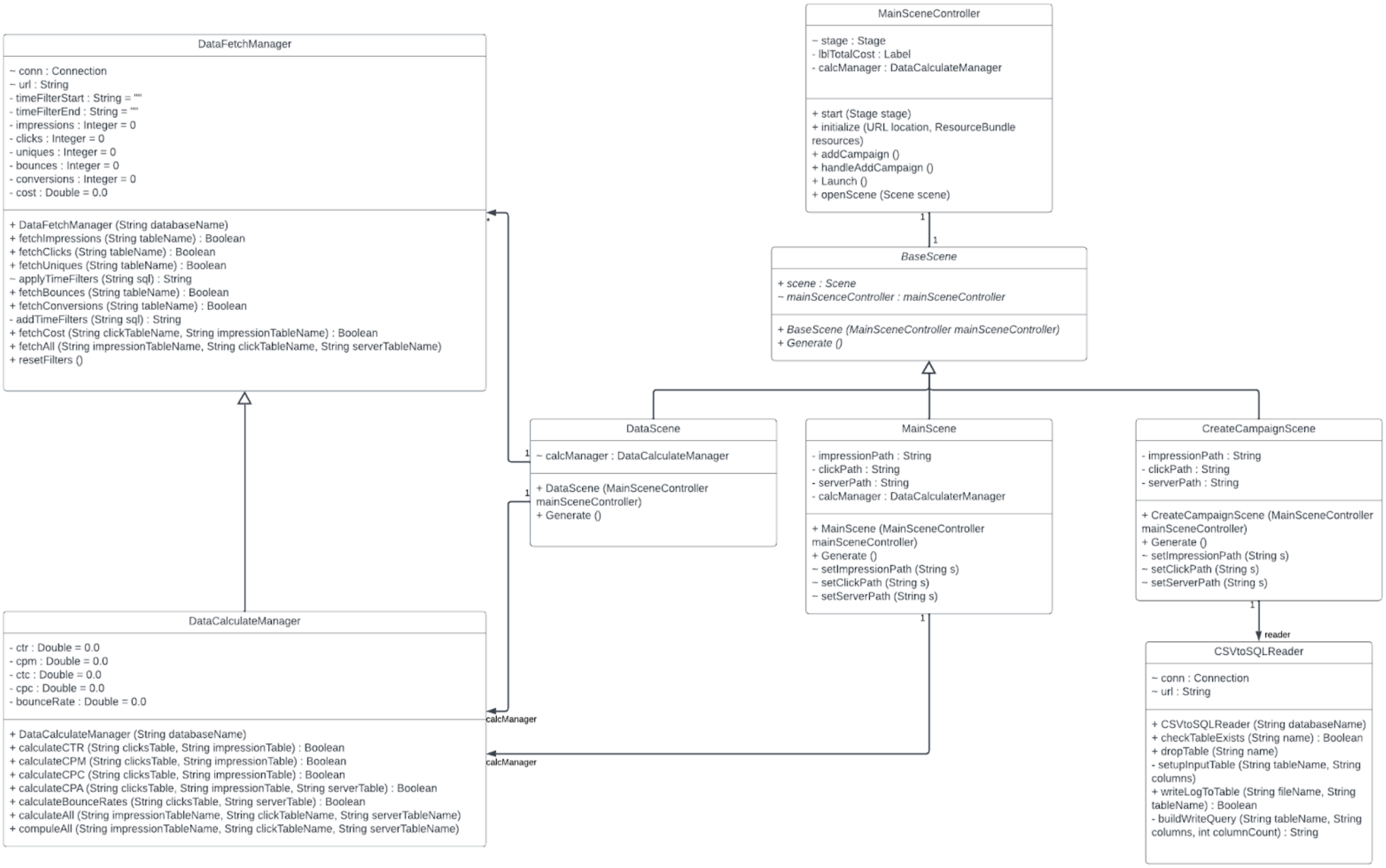
1.1 Use Case Diagram

The use case diagram below is based on the Sprint 1 artifacts implemented. Partially implemented use cases are included.

Use Case Diagram Sprint 1



1.2 UML Class Diagram


DataFetchManager
~ conn : Connection
~ url : String
- timeFilterStart : String = ""
- timeFilterEnd : String = ""
- impressions : Integer = 0
- clicks : Integer = 0
- uniques : Integer = 0
- bounces : Integer = 0
- conversions : Integer = 0
- cost : Double = 0.0
+ DataFetchManager (String databaseName)
+ fetchImpressions (String tableName) : Boolean
+ fetchClicks (String tableName) : Boolean
+ fetchUniques (String tableName) : Boolean
~ applyTimeFilters (String sql) : String
+ fetchBounces (String tableName) : Boolean
+ fetchConversions (String tableName) : Boolean
- addTimeFilters (String sql) : String
+ fetchCost (String clickTableName, String impressionTableName) : Boolean
+ fetchAll (String impressionTableName, String clickTableName, String serverTableName)
+ resetFilters ()
*
MainSceneController
~ stage : Stage
- lblTotalCost : Label
- calcManager : DataCalculateManager
+ start (Stage stage)
+ initialize (URL location, ResourceBundle resources)
+ addCampaign ()
+ handleAddCampaign ()
+ Launch ()
+ openScene (Scene scene)
1
1
BaseScene
+ scene : Scene
~ mainScenceController : mainScenceController
+ BaseScene (MainSceneController mainSceneController)
+ Generate ()
DataScene
~ calcManager : DataCalculateManager
+ DataScene (MainSceneController mainSceneController)
+ Generate ()
1
1
MainScene
- impressionPath : String
- clickPath : String
- serverPath : String
- calcManager : DataCalculaterManager
+ MainScene (MainSceneController mainSceneController)
+ Generate ()
~ setImpressionPath (String s)
~ setClickPath (String s)
~ setServerPath (String s)
1
CreateCampaignScene
- impressionPath : String
- clickPath : String
- serverPath : String
+ CreateCampaignScene (MainSceneController mainSceneController)
+ Generate ()
~ setImpressionPath (String s)
~ setClickPath (String s)
~ setServerPath (String s)
1
reader
CSVtoSQLReader
~ conn : Connection
~ url : String
+ CSVtoSQLReader (String databaseName)
+ checkTableExists (String name) : Boolean
+ dropTable (String name)
- setupInputTable (String tableName, String columns)
+ writeLogToTable (String fileName, String tableName) : Boolean
- buildWriteQuery (String tableName, String columns, int columnCount) : String
DataCalculateManager
- ctr : Double = 0.0
- cpm : Double = 0.0
- ctc : Double = 0.0
- cpc : Double = 0.0
- bounceRate : Double = 0.0
+ DataCalculateManager (String databaseName)
+ calculateCTR (String clicksTable, String impressionTable) : Boolean
+ calculateCPM (String clicksTable, String impressionTable) : Boolean
+ calculateCPC (String clicksTable, String impressionTable) : Boolean
+ calculateCPA (String clicksTable, String impressionTable, String serverTable) : Boolean
+ calculateBounceRates (String clicksTable, String serverTable) : Boolean
+ calculateAll (String impressionTableName, String clickTableName, String serverTableName)
+ compuleAll (String impressionTableName, String clickTableName, String serverTableName)
calcManager
calcManager

1.3 Entity Relationship Diagram

Below are shown the three tables for storing imported CSV data – **impression_log**, **click_log**, **server_log.** They are part of a local SQLite database created at the import data user case. This way, we are able to import and process large datasets effectively and it allows data to be retrieved without reimporting after the application is restarted. We take advantage of SQLite's flexible typing, so the data types shown below are only the expected ones. This design choice, while convenient, will require more testing and error detection in the future.

impression_log	
item_id*	INTEGER
date	TEXT
user_id	TEXT
gender	TEXT
age	TEXT
income	TEXT
context	TEXT
impression_cost	REAL

click_log	
item_id*	INTEGER
date	TEXT
user_id	TEXT
click_cost	REAL

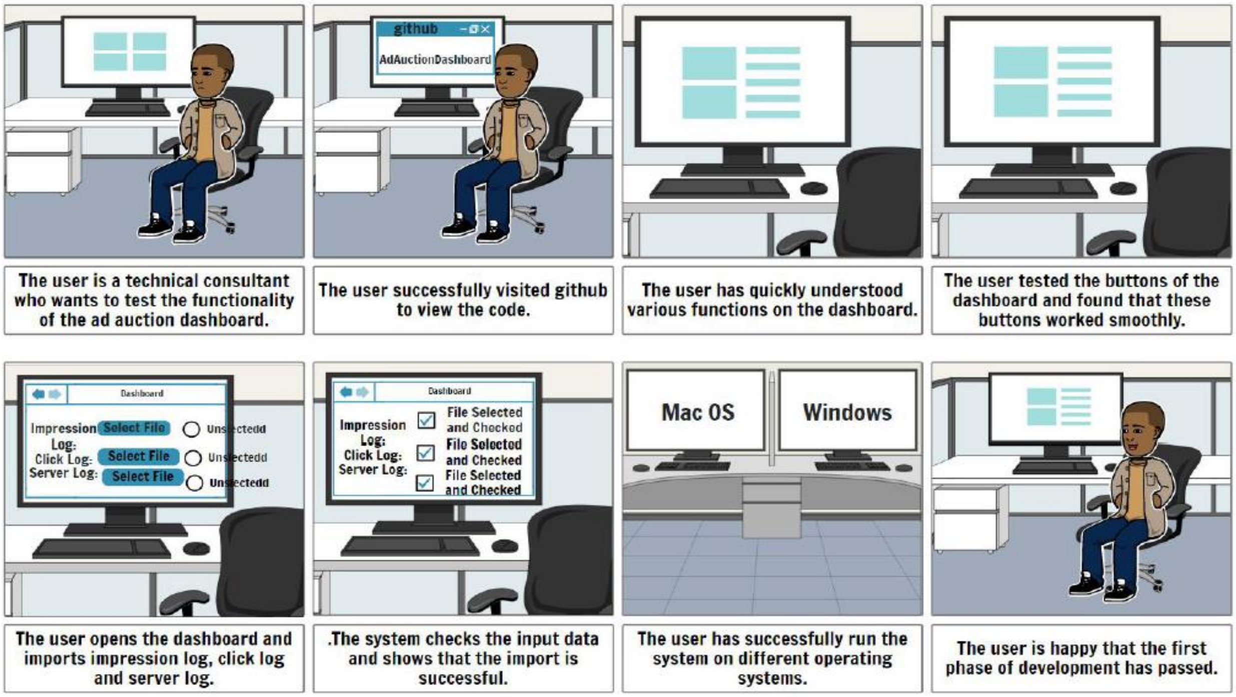
server_log	
item_id*	INTEGER
entry_date	TEXT
user_id	TEXT
exit_date	TEXT
pages_viewed	INTEGER
conversion	TEXT

* Primary key with autoincrementing

1.4 Storyboarding

Following our team meetings, we came up with the following artifacts related to the user stories for this sprint:

Storyboard: A user (Josh) checks the progress of the first development of the dashboard.


github
AdAuctionDashboard
Dashboard
Impression Log: Select File ○ Unselectedd
Click Log: Select File ○ Unselectedd
Server Log: Select File ○ Unselectedd
Dashboard
Impression Log: ☑ File Selected and Checked
Click Log: ☑ File Selected and Checked
Server Log: ☑ File Selected and Checked
Mac OS
Windows

The user is a technical consultant who wants to test the functionality of the ad auction dashboard.

The user successfully visited github to view the code.

The user has quickly understood various functions on the dashboard.

The user tested the buttons of the dashboard and found that these buttons worked smoothly.

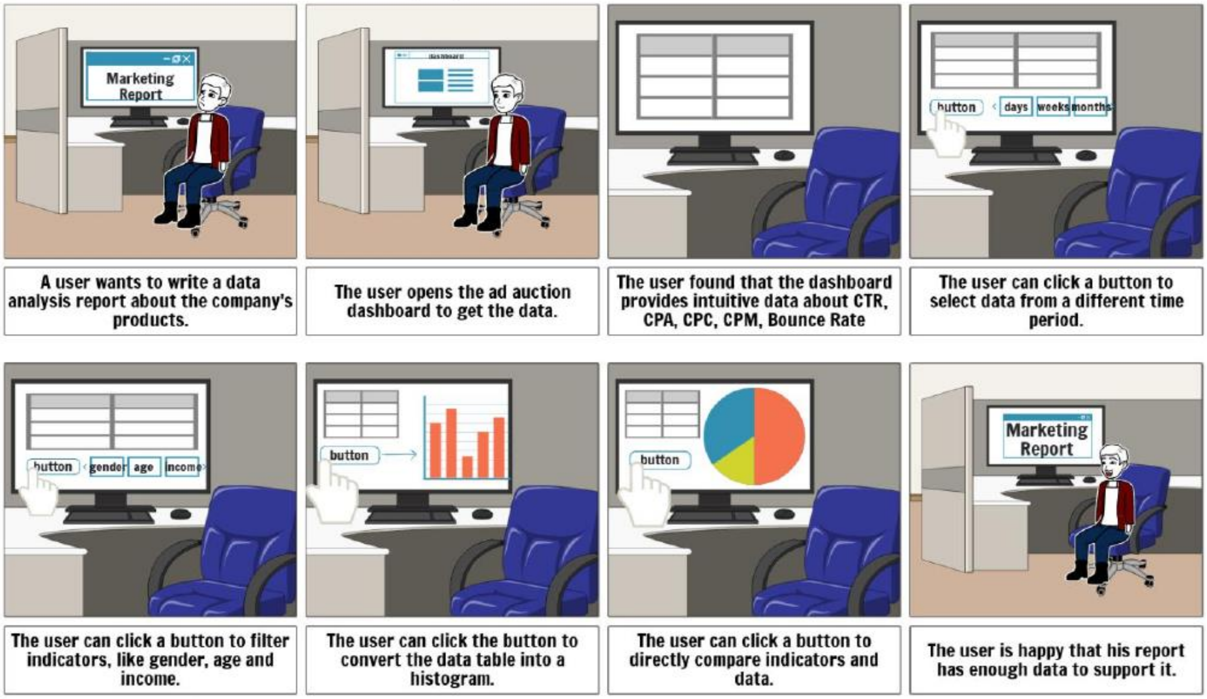
The user opens the dashboard and imports impression log, click log and server log.

.The system checks the input data and shows that the import is successful.

The user has successfully run the system on different operating systems.

The user is happy that the first phase of development has passed.

Storyboard: A user (Robert) uses the system to analyse data information so that he can write better marketing reports.



A user wants to write a data analysis report about the company's products.

The user opens the ad auction dashboard to get the data.

The user found that the dashboard provides intuitive data about CTR, CPA, CPC, CPM, Bounce Rate

The user can click a button to select data from a different time period.

The user can click a button to filter indicators, like gender, age and income.

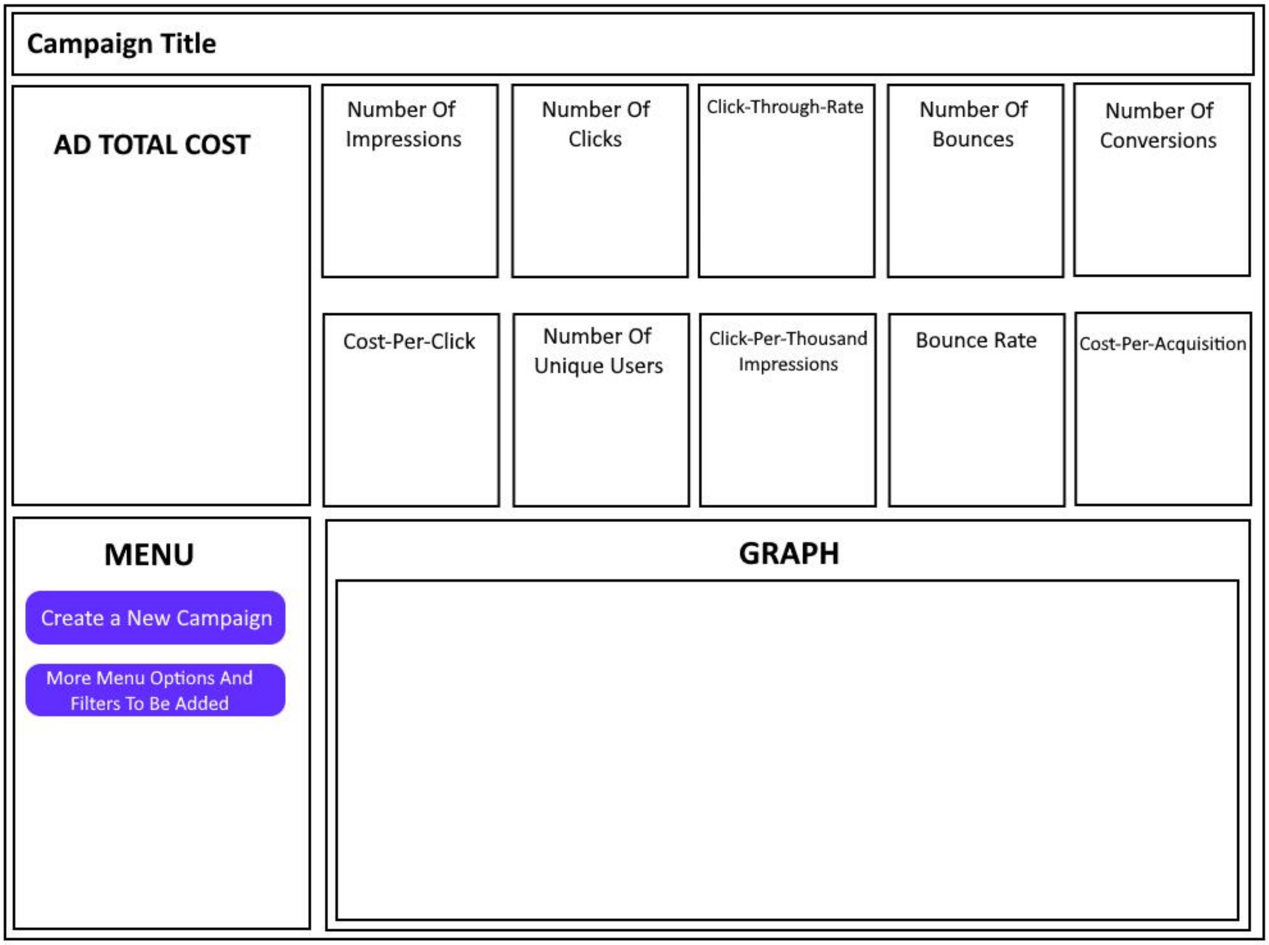
The user can click the button to convert the data table into a histogram.

The user can click a button to directly compare indicators and data.

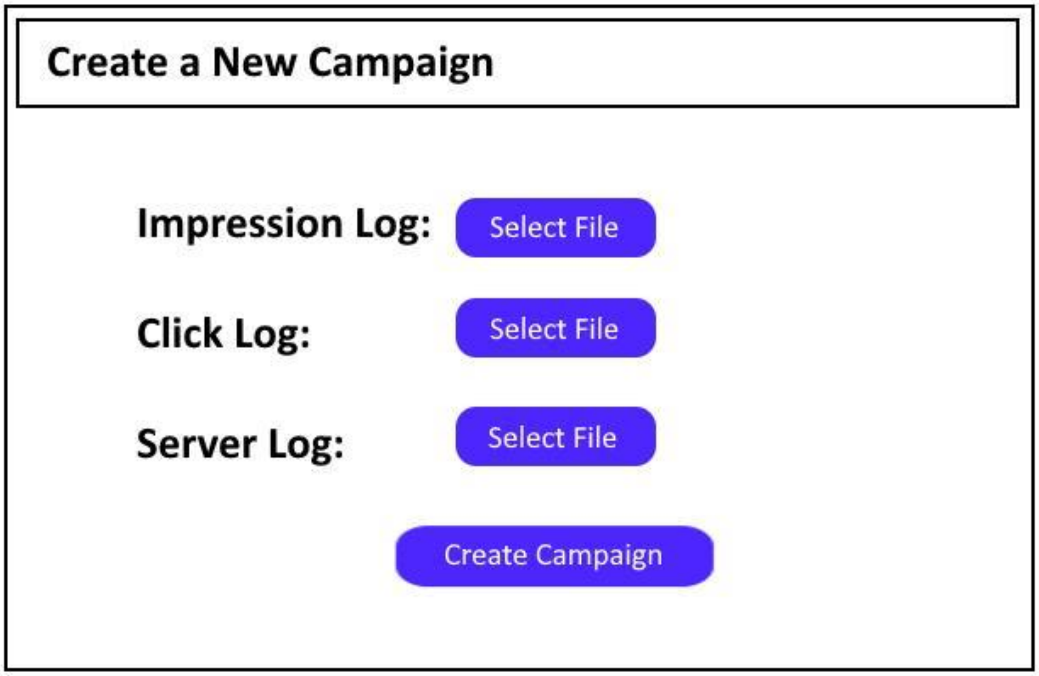
The user is happy that his report has enough data to support it.

These resulted in the following design concepts:

Dashboard Main Screen



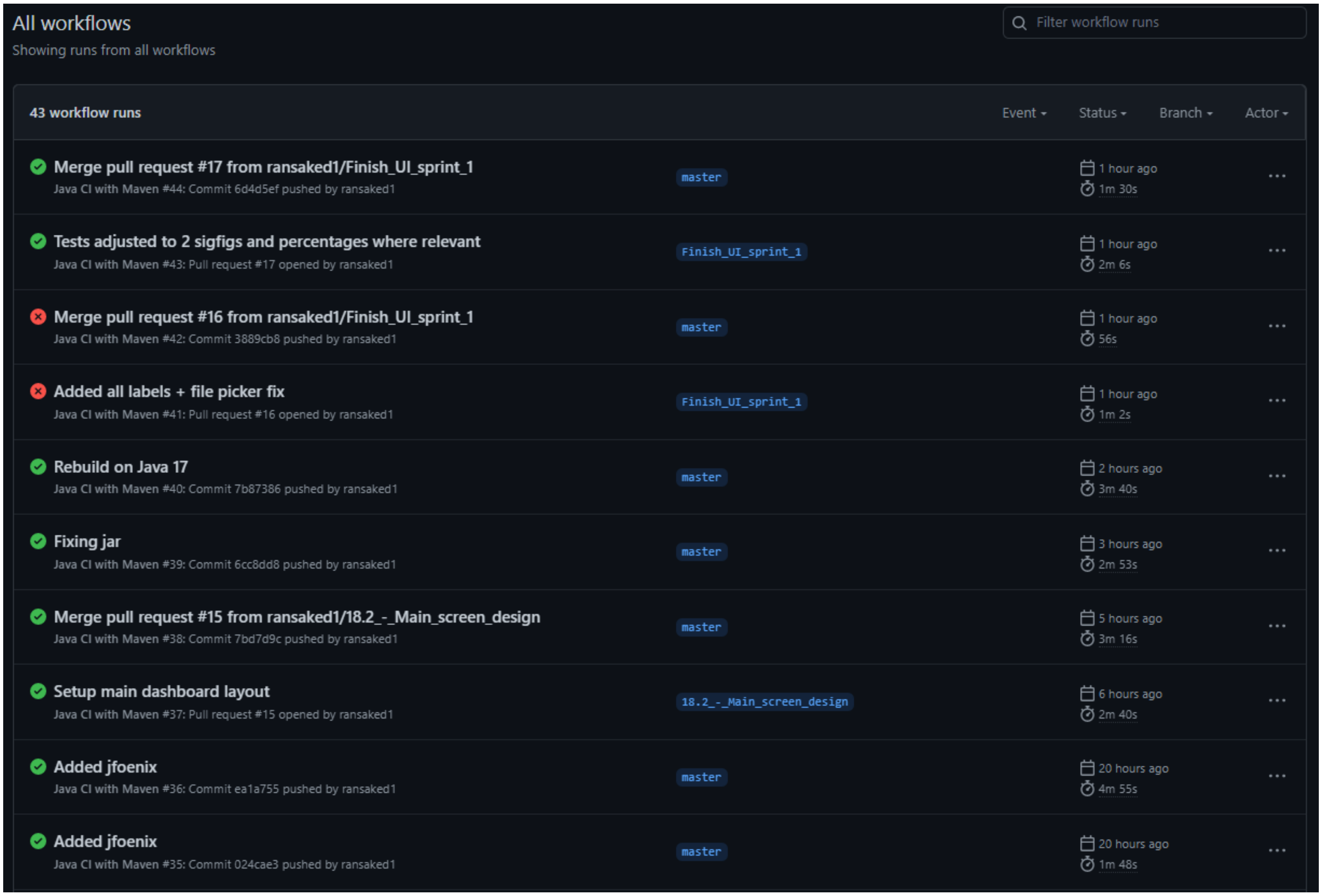
Adding New Campaign Screen



2. SCENARIOS AND TESTING

2.1 Continuous integration

Every commit on master is compiled and tested on all three latest releases of Windows, Ubuntu and macOS with GitHub Actions:



All workflows

Showing runs from all workflows

43 workflow runs

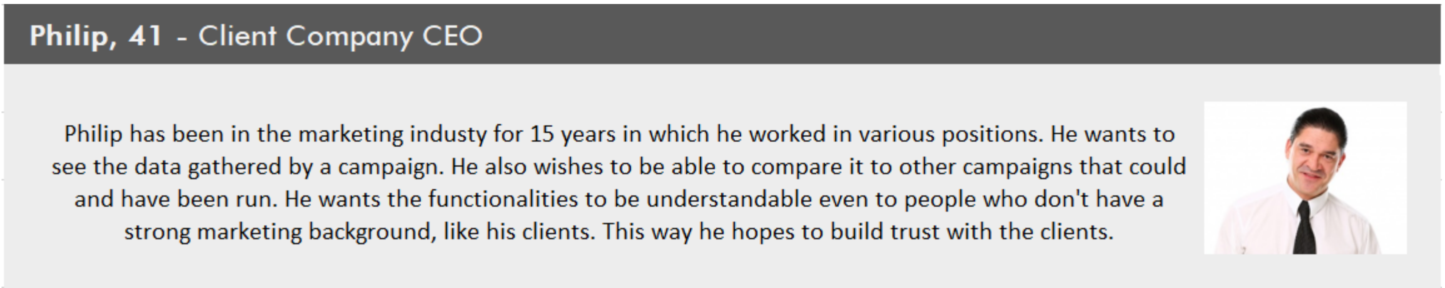
✓	**Merge pull request #17 from ransaked1/Finish_UI_sprint_1** Java CI with Maven #44: Commit 6d4d5ef pushed by ransaked1	master	1 hour ago 1m 30s	···
✓	**Tests adjusted to 2 sigfigs and percentages where relevant** Java CI with Maven #43: Pull request #17 opened by ransaked1	Finish_UI_sprint_1	1 hour ago 2m 6s	···
✗	**Merge pull request #16 from ransaked1/Finish_UI_sprint_1** Java CI with Maven #42: Commit 3889cb8 pushed by ransaked1	master	1 hour ago 56s	···
✗	**Added all labels + file picker fix** Java CI with Maven #41: Pull request #16 opened by ransaked1	Finish_UI_sprint_1	1 hour ago 1m 2s	···
✓	**Rebuild on Java 17** Java CI with Maven #40: Commit 7b87386 pushed by ransaked1	master	2 hours ago 3m 40s	···
✓	**Fixing jar** Java CI with Maven #39: Commit 6cc8dd8 pushed by ransaked1	master	3 hours ago 2m 53s	···
✓	**Merge pull request #15 from ransaked1/18.2_-_Main_screen_design** Java CI with Maven #38: Commit 7bd7d9c pushed by ransaked1	master	5 hours ago 3m 16s	···
✓	**Setup main dashboard layout** Java CI with Maven #37: Pull request #15 opened by ransaked1	18.2_-_Main_screen_design	6 hours ago 2m 40s	···
✓	**Added jfoenix** Java CI with Maven #36: Commit ea1a755 pushed by ransaked1	master	20 hours ago 4m 55s	···
✓	**Added jfoenix** Java CI with Maven #35: Commit 024cae3 pushed by ransaked1	master	20 hours ago 1m 48s	···

2.2 Scenarios and test results

Sarah, 30 - A Senior Marketing Leader

Sarah manages a marketing analyst team and reports directly to clients on the ad campaign performance. She struggles with analysing her client's data as they each have a unique custom campaign. She wants a dashboard that allows her to see campaigns and how they perform time. This could be aided with visuals as well as raw data. This would make her job easier and would allow her to focus on well performing campaigns.

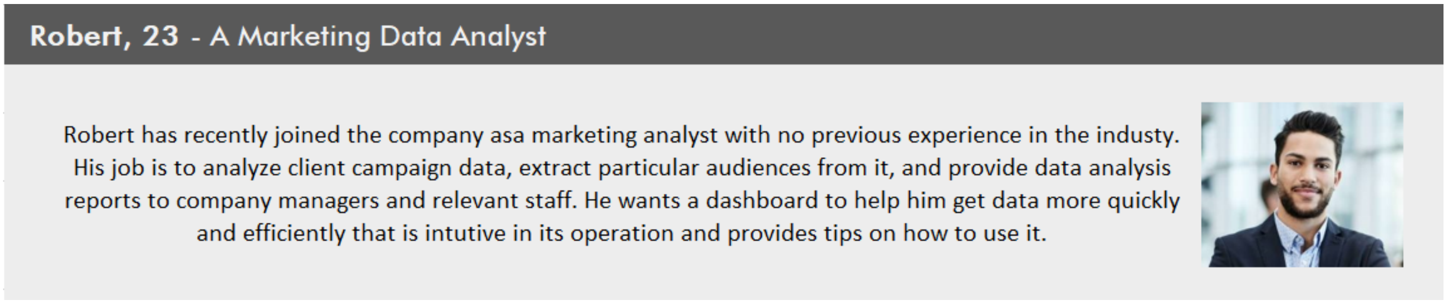
Sarah opens the Ad Auction dashboard and clicks "Add Campaign".
Sarah selects and inputs the campaign's CSV files.
After a brief buffering, the metrics are displayed at the top of the screen.
Sarah can easily read the different metrics and analyse them further as required.
It does not however offer visuals as she would like alongside the raw metrics.



Philip, 41 - Client Company CEO

Philip has been in the marketing industy for 15 years in which he worked in various positions. He wants to see the data gathered by a campaign. He also wishes to be able to compare it to other campaigns that could and have been run. He wants the functionalities to be understandable even to people who don't have a strong marketing background, like his clients. This way he hopes to build trust with the clients.

Philip opens the Ad Auction dashboard and clicks “Add Campaign”.
Philip selects and inputs the campaign’s CSV files.
After a brief buffering, the metrics are displayed at the top of the screen.
Philip analyses the metrics but then needs to analyse a separate campaign to compare them.
Philip cannot easily compare them; however, he can record the first set of metrics and then use the software to get the metrics from the second campaign and compare.



Robert, 23 - A Marketing Data Analyst

Robert has recently joined the company asa marketing analyst with no previous experience in the industy. His job is to analyze client campaign data, extract particular audiences from it, and provide data analysis reports to company managers and relevant staff. He wants a dashboard to help him get data more quickly and efficiently that is intutive in its operation and provides tips on how to use it.

Robert opens the Ad Auction dashboard and clicks “Add Campaign”.
Robert selects and inputs the campaign’s CSV files.
After a brief buffering, the metrics are displayed at the top of the screen.
Robert appreciates the speed in analysing the data automatically with accurate results.
He appreciates the beginnings of the dashboard, however, would like it to be further developed to make it more intuitive with tips on how to use it.

Real Scenarios:

Testing the dashboard using the 2-Week Campaign Data provided.

I open the Ad Auction dashboard and click “Add Campaign”. This button is self-explanatory and easy to understand.
A new window opens that allows me to input the three files as part of the campaign, each one clearly labelled including telling me if I haven’t yet selected a file.
Once I have inputted the files I click “Create Campaign”.
After a few seconds of buffering, all the metrics appear at the top of the screen correctly.

Testing the dashboard using the 2-Week Campaign Data provided, but this time submitting the incorrect files in each input.

I open the Ad Auction dashboard and click “Add Campaign”. This button is self-explanatory and easy to understand.
A new window opens that allows me to input the three files as part of the campaign, each one clearly labelled including telling me if I haven’t yet selected a file.
I input the wrong CSV files for each option and click “Create Campaign”
It incorrectly processes the results, rather than noticing the wrong files have been submitted. This needs to be corrected.

Testing the dashboard submitting empty CSV files each time.

I open the Ad Auction dashboard and click “Add Campaign”. This button is self-explanatory and easy to understand.
A new window opens that allows me to input the three files as part of the campaign, each one clearly labelled including telling me if I haven’t yet selected a file.
I input a blank CSV file each time.
It allows me to input the files but does not allow me to click “Create Campaign”. This works however could be improved by telling the user the reason “Create Campaign” does not work is because the files are empty.

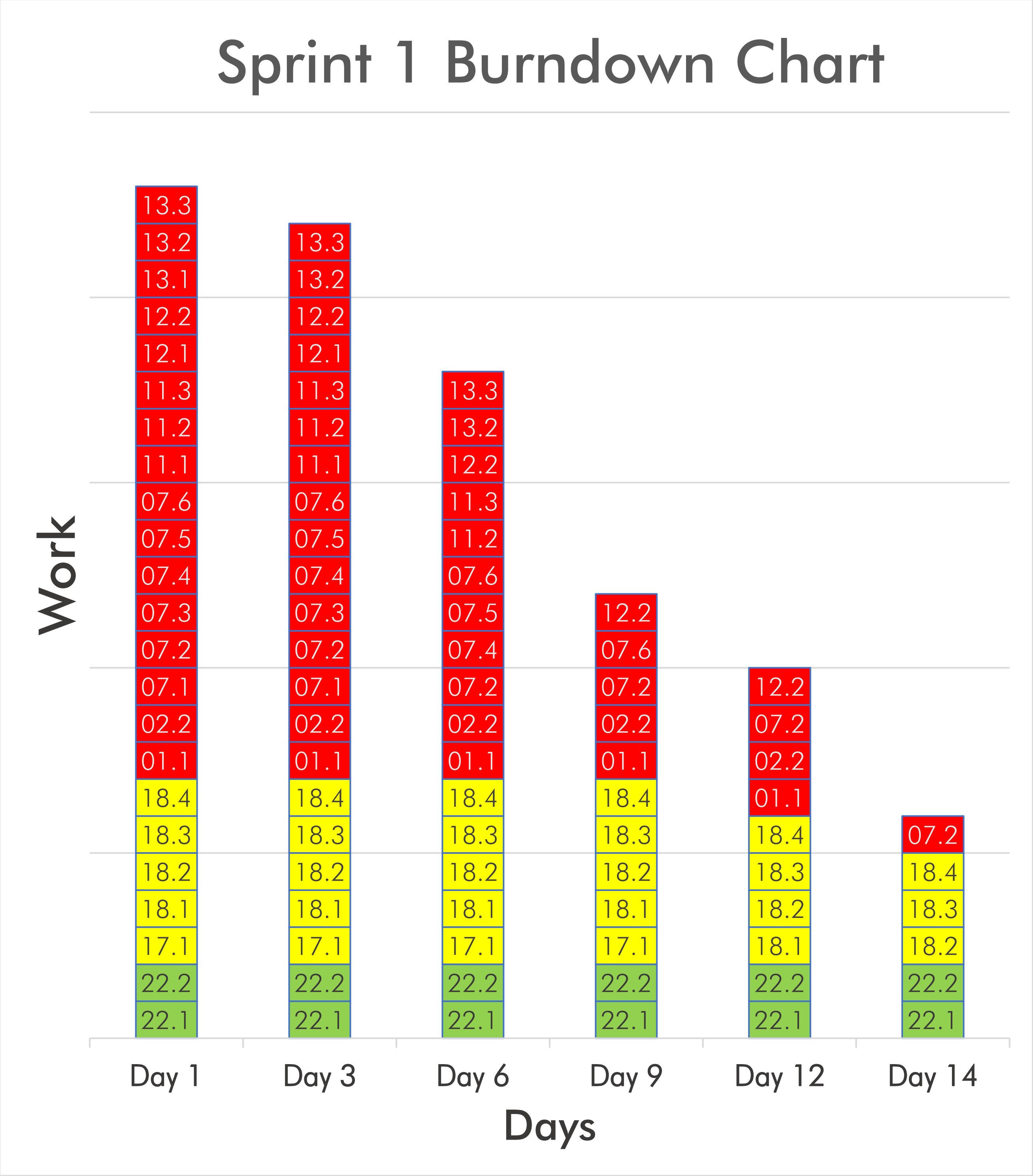
3. SPRINT RETROSPECTIVE AND SPRINT 2 PLANNING

3.1 Sprint 1 Retrospective

The goal of this sprint was to set up the processes around our project and the focus was on the software backend. While working on the data reading and database setup from epic 07 (Reading data from specified files) we realized that the majority of “must have’s” from sprint 2 that we saw as complex before where now trivialized to SQL requests. This made our team ambitious and wanting to provide something more tangible in increment 1 so towards the end of the sprint we decided to bring in some of the elements of the epics 01 and 02 (Most metrics) from sprint 2 into sprint 1 which will be reflected below. At the same time, because of this change in focus, we were left with much less time for epic 18 (Intuitive UI Design) which got carries over with task 07.2 and epic 22 (Automatic input file detection) to sprint 2. We added the epic 23 (Import optimization) which we found important as right now the app can run smoother and provide more visual feedback.

Overall, the team worked well and efficiently even if we did not manage to finish everything planned. Next sprint, we think that with improved team cohesion we will be able to achieve all goals.

3.2 Sprint 1 Burndown Chart



3.3 Sprint 2 Backlog

SPRINT 1				
Backlog ID & Description	**ID**	**Task Description**	**Size**	**Dependencies**
01 - Number of Impressions, Clicks, Uniques, Bounces	01.2	Display data adequately	3	-
	01.3	Graph data	5	01.2
	01.4	Graphing options	8	01.3
	01.5	Testing	3	01.4
02 - CTR, CPA, CPC, CPM, Bounce Rate	01.2	Display data adequately	3	-
	01.3	Graph data	3	02.2
	01.4	Graphing options	3	02.3
	01.5	Testing	3	02.4
10 - Large dataset handling	10.1	Make extremely large inputs	3	-
	10.2	Test handling input takes appropriate time	3	10.1
	10.3	Add feedback for the user while waiting	8	10.1
07 - Reading data from specified files	07.2	Error checking and recovery	5	-
23 - Import optimization	23.1	Make the CSV imports into independent threads to improve UX	3	-
18 - Intuitive UI Prototyping	18.2	Code dashboard home screen	5	-
	18.3	Code window flow and available controls	5	-
	18.4	Test basic dashboard functionality	3	18.2/18.3
15 - Saving report to various file formats	15.1	Data pre processing	5	-
	15.2	Generate report in PDF format	3	15.1
22 - Automatic input file detection	22.1	Implement automatic file detection	2	-
	22.2	Test correct data importing	1	22.1
SIZE ESTIMATE TOTAL			**77**	

3.4 Next sprint

For this sprint our priority was focusing on the basic functionality of displaying the metrics and making sure the backend was functioning before developing the frontend further. We felt this was beneficial to prioritize as we need the backend to by functioning to test the frontend works correctly. It was also a better task to complete first as it allowed us to greater understand each other's programming abilities, without the subjectivity of user interfaces getting in our way. We do have a basic frontend prepared that allows us to demonstrate the functionality of the backend, which we plan to improve more in the next sprint. It displays the metrics that have been implemented already.

For the next sprint, we intend to focus more on the frontend so that we can demonstrate more of the backend capabilities at the end of the second sprint. We need to label the different metrics along with other elements to make the UI more intuitive for the user. We also plan to optimize the backend to make it more efficient.