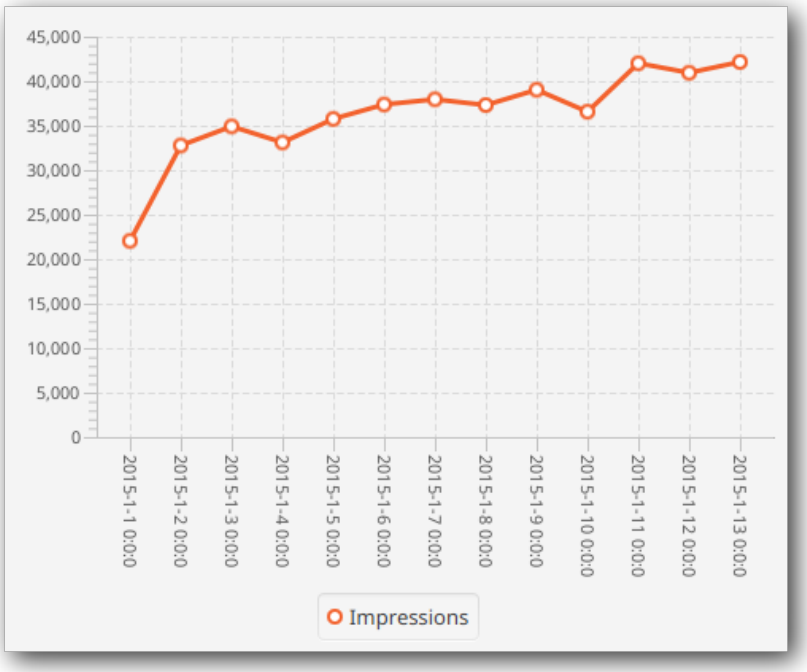
Ben Hadlington (bh9g22)
David Afonso Shepherd (das1g22)
Alejandro Andino Martinez (aam1g22)
Patrick Jfremov-Kustov (pjk1g22)
Kun Chen (kc4a22)

COMP2211
SEng Group Project

Ad Auction Dashboard
(2024)



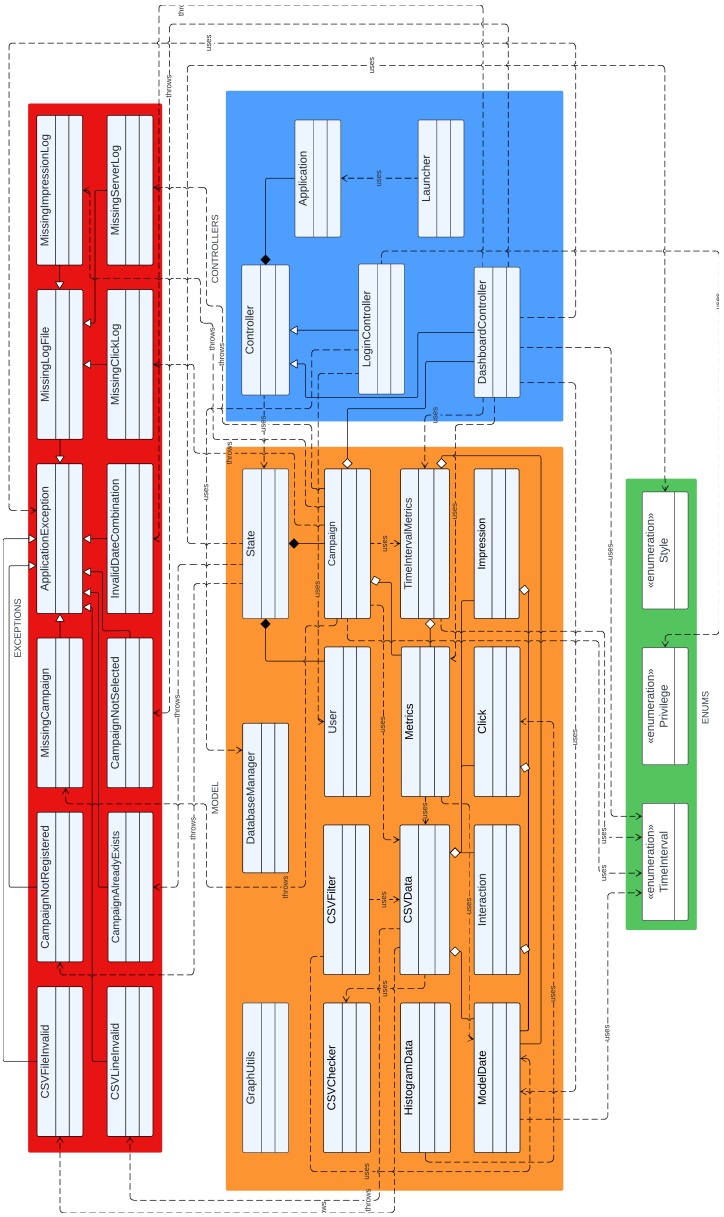
Group	24
Version	1
Hand-in	3

Part I.

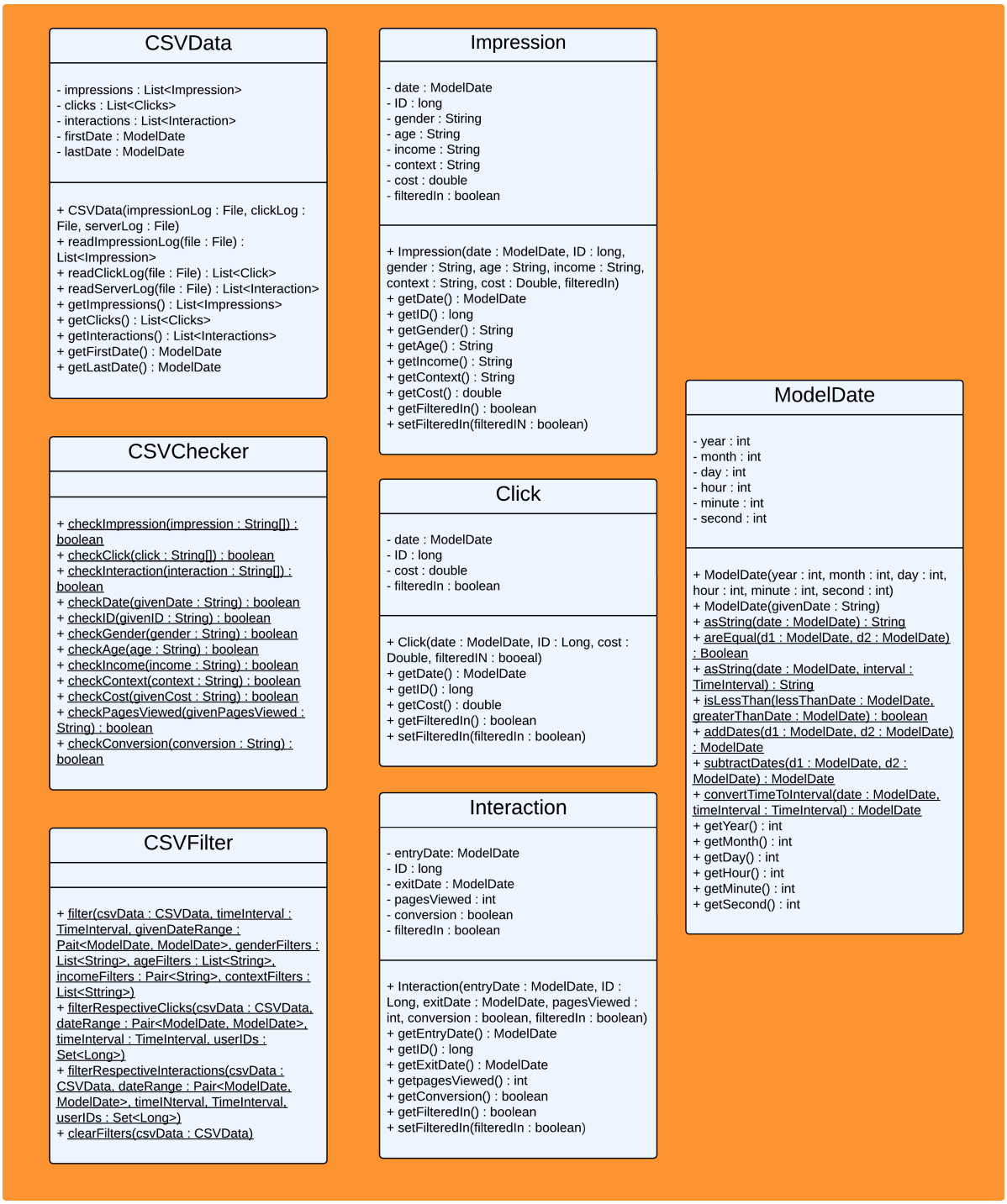
Increment 2

1. Design

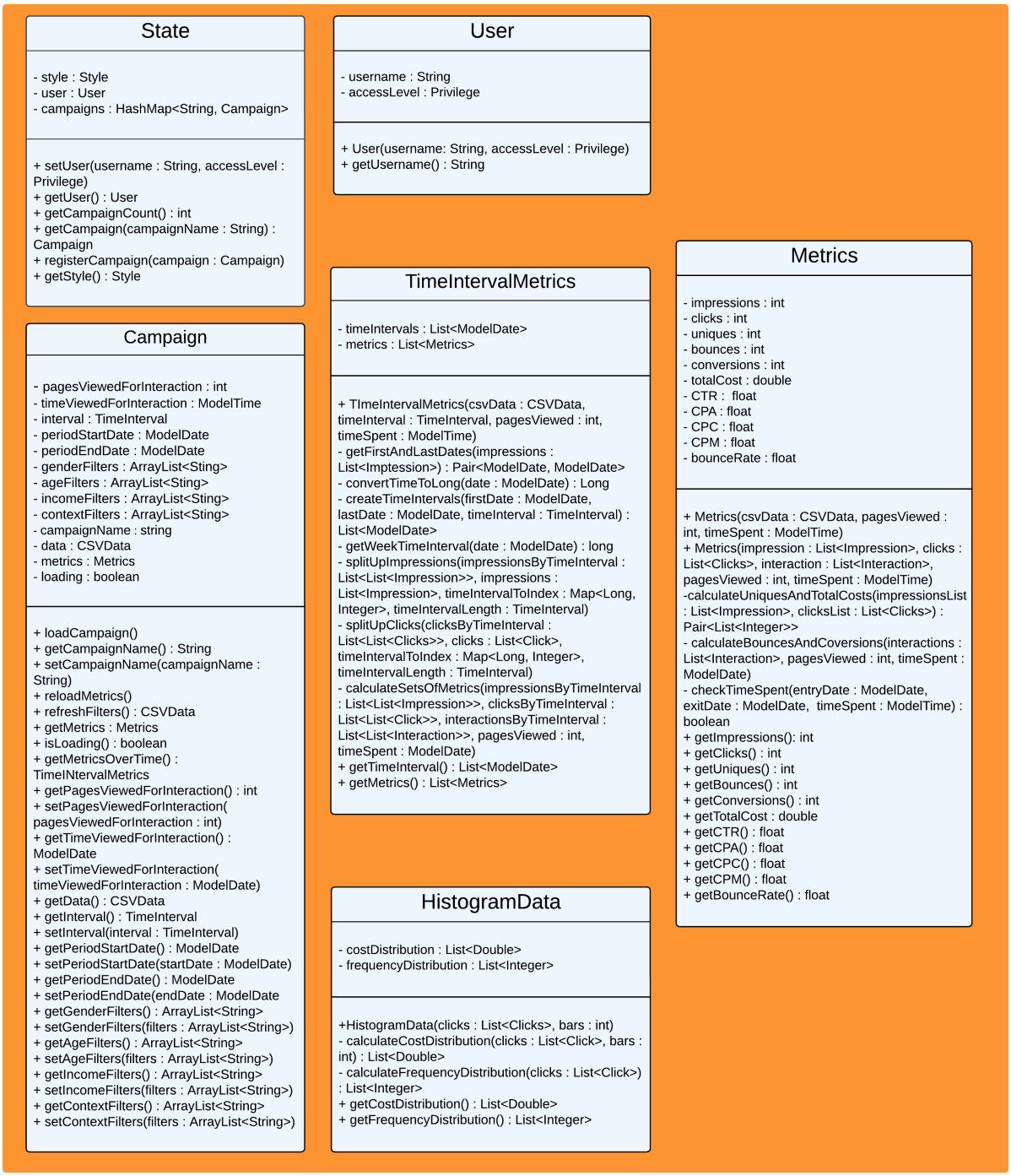
1.1. UML Class Diagram



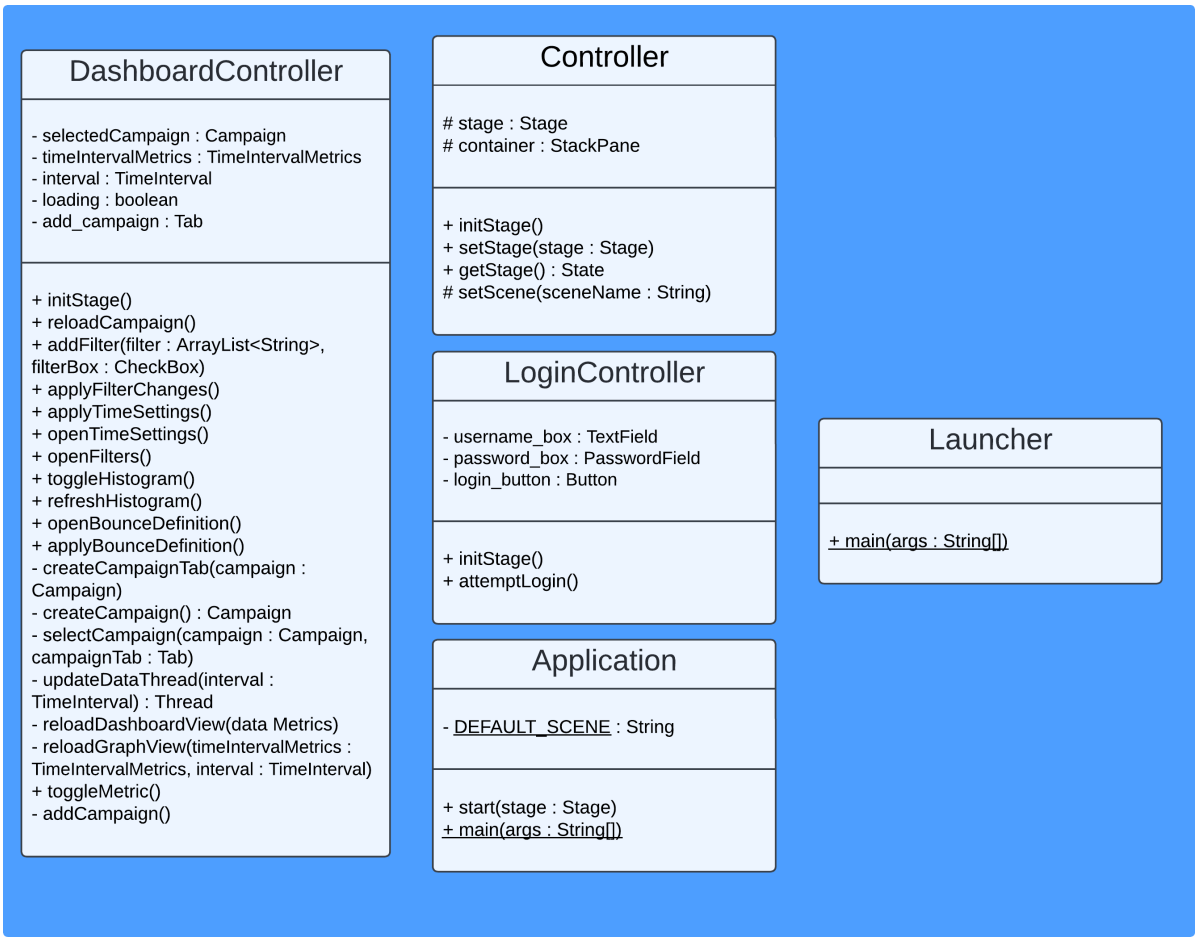
1.1.1. Model I


CSVData
- impressions : List<Impression>
- clicks : List<Clicks>
- interactions : List<Interaction>
- firstDate : ModelDate
- lastDate : ModelDate
+ CSVData(impressionLog : File, clickLog : File, serverLog : File)
+ readImpressionLog(file : File) : List<Impression>
+ readClickLog(file : File) : List<Click>
+ readServerLog(file : File) : List<Interaction>
+ getImpressions() : List<Impressions>
+ getClicks() : List<Clicks>
+ getInteractions() : List<Interactions>
+ getFirstDate() : ModelDate
+ getLastDate() : ModelDate
CSVChecker
+ checkImpression(impression : String[]) : boolean
+ checkClick(click : String[]) : boolean
+ checkInteraction(interaction : String[]) : boolean
+ checkDate(givenDate : String) : boolean
+ checkID(givenID : String) : boolean
+ checkGender(gender : String) : boolean
+ checkAge(age : String) : boolean
+ checkIncome(income : String) : boolean
+ checkContext(context : String) : boolean
+ checkCost(givenCost : String) : boolean
+ checkPagesViewed(givenPagesViewed : String) : boolean
+ checkConversion(conversion : String) : boolean
CSVFilter
+ filter(csvData : CSVData, timeInterval : TimeInterval, givenDateRange : Pait<ModelDate, ModelDate>, genderFilters : List<String>, ageFilters : List<String>, incomeFilters : Pair<String>, contextFilters : List<Sttring>)
+ filterRespectiveClicks(csvData : CSVData, dateRange : Pair<ModelDate, ModelDate>, timeInterval : TimeInterval, userIDs : Set<Long>)
+ filterRespectiveInteractions(csvData : CSVData, dateRange : Pair<ModelDate, ModelDate>, timeINterval, TimeInterval, userIDs : Set<Long>)
+ clearFilters(csvData : CSVData)
Impression
- date : ModelDate
- ID : long
- gender : Stiring
- age : String
- income : String
- context : String
- cost : double
- filteredIn : boolean
+ Impression(date : ModelDate, ID : long, gender : String, age : String, income : String, context : String, cost : Double, filteredIn)
+ getDate() : ModelDate
+ getID() : long
+ getGender() : String
+ getAge() : String
+ getIncome() : String
+ getContext() : String
+ getCost() : double
+ getFilteredIn() : boolean
+ setFilteredIn(filteredIN : boolean)
Click
- date : ModelDate
- ID : long
- cost : double
- filteredIn : boolean
+ Click(date : ModelDate, ID : Long, cost : Double, filteredIN : booeal)
+ getDate() : ModelDate
+ getID() : long
+ getCost() : double
+ getFilteredIn() : boolean
+ setFilteredIn(filteredIn : boolean)
Interaction
- entryDate: ModelDate
- ID : long
- exitDate : ModelDate
- pagesViewed : int
- conversion : boolean
- filteredIn : boolean
+ Interaction(entryDate : ModelDate, ID : Long, exitDate : ModelDate, pagesViewed : int, conversion : boolean, filteredIn : boolean)
+ getEntryDate() : ModelDate
+ getID() : long
+ getExitDate() : ModelDate
+ getpagesViewed() : int
+ getConversion() : boolean
+ getFilteredIn() : boolean
+ setFilteredIn(filteredIn : boolean)
ModelDate
- year : int
- month : int
- day : int
- hour : int
- minute : int
- second : int
+ ModelDate(year : int, month : int, day : int, hour : int, minute : int, second : int)
+ ModelDate(givenDate : String)
+ asString(date : ModelDate) : String
+ areEqual(d1 : ModelDate, d2 : ModelDate) : Boolean
+ asString(date : ModelDate, interval : TimeInterval) : String
+ isLessThan(lessThanDate : ModelDate, greaterThanDate : ModelDate) : boolean
+ addDates(d1 : ModelDate, d2 : ModelDate) : ModelDate
+ subtractDates(d1 : ModelDate, d2 : ModelDate) : ModelDate
+ convertTimeToInterval(date : ModelDate, timeInterval : TimeInterval) : ModelDate
+ getYear() : int
+ getMonth() : int
+ getDay() : int
+ getHour() : int
+ getMinute() : int
+ getSecond() : int

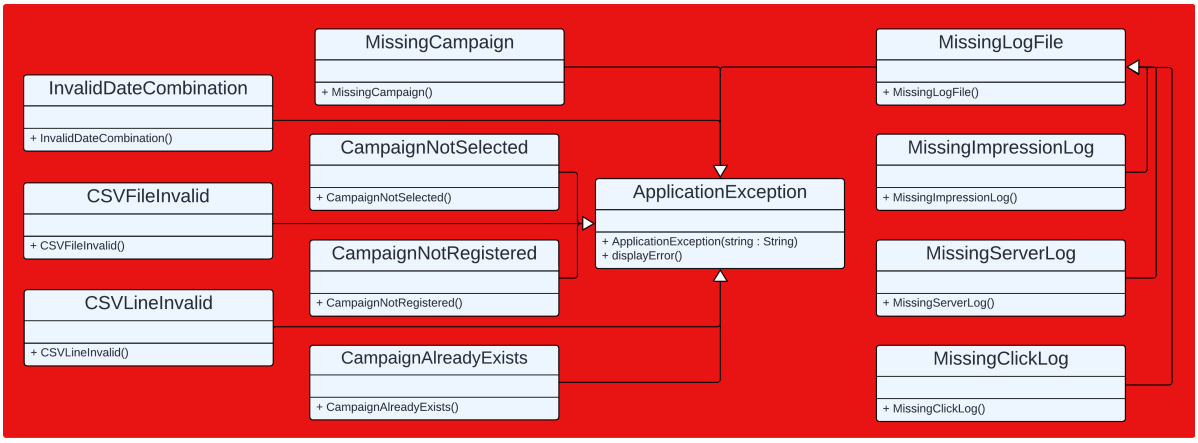
1.1.2. Model II


State
- style : Style
- user : User
- campaigns : HashMap<String, Campaign>
+ setUser(username : String, accessLevel : Privilege)
+ getUser() : User
+ getCampaignCount() : int
+ getCampaign(campaignName : String) : Campaign
+ registerCampaign(campaign : Campaign)
+ getStyle() : Style
User
- username : String
- accessLevel : Privilege
+ User(username: String, accessLevel : Privilege)
+ getUsername() : String
Campaign
- pagesViewedForInteraction : int
- timeViewedForInteraction : ModelTime
- interval : TimeInterval
- periodStartDate : ModelDate
- periodEndDate : ModelDate
- genderFilters : ArrayList<Sting>
- ageFilters : ArrayList<Sting>
- incomeFilters : ArrayList<Sting>
- contextFilters : ArrayList<Sting>
- campaignName : string
- data : CSVData
- metrics : Metrics
- loading : boolean
+ loadCampaign()
+ getCampaignName() : String
+ setCampaignName(campaignName : String)
+ reloadMetrics()
+ refreshFilters() : CSVData
+ getMetrics : Metrics
+ isLoading() : boolean
+ getMetricsOverTime() : TimeINtervalMetrics
+ getPagesViewedForInteraction() : int
+ setPagesViewedForInteraction(pagesViewedForInteraction : int)
+ getTimeViewedForInteraction() : ModelDate
+ setTimeViewedForInteraction(timeViewedForInteraction : ModelDate)
+ getData() : CSVData
+ getInterval() : TimeInterval
+ setInterval(interval : TimeInterval)
+ getPeriodStartDate() : ModelDate
+ setPeriodStartDate(startDate : ModelDate)
+ getPeriodEndDate() : ModelDate
+ setPeriodEndDate(endDate : ModelDate
+ getGenderFilters() : ArrayList<String>
+ setGenderFilters(filters : ArrayList<String>)
+ getAgeFilters() : ArrayList<String>
+ setAgeFilters(filters : ArrayList<String>)
+ getIncomeFilters() : ArrayList<String>
+ setIncomeFilters(filters : ArrayList<String>)
+ getContextFilters() : ArrayList<String>
+ setContextFilters(filters : ArrayList<String>)
TimeIntervalMetrics
- timeIntervals : List<ModelDate>
- metrics : List<Metrics>
+ TImeIntervalMetrics(csvData : CSVData, timeInterval : TimeInterval, pagesViewed : int, timeSpent : ModelTime)
- getFirstAndLastDates(impressions : List<Imptession>) : Pair<ModelDate, ModelDate>
- convertTimeToLong(date : ModelDate) : Long
- createTimeIntervals(firstDate : ModelDate, lastDate : ModelDate, timeInterval : TimeInterval) : List<ModelDate>
- getWeekTimeInterval(date : ModelDate) : long
- splitUpImpressions(impressionsByTimeInterval : List<List<Impression>>, impressions : List<Impression>, timeIntervalToIndex : Map<Long, Integer>, timeIntervalLength : TimeInterval)
- splitUpClicks(clicksByTimeInterval : List<List<Clicks>>, clicks : List<Click>, timeIntervalToIndex : Map<Long, Integer>, timeIntervalLength : TimeInterval)
- calculateSetsOfMetrics(impressionsByTimeInterval : List<List<Impression>>, clicksByTimeInterval : List<List<Click>>, interactionsByTimeInterval : List<List<Interaction>>, pagesViewed : int, timeSpent : ModelDate)
+ getTimeInterval() : List<ModelDate>
+ getMetrics() : List<Metrics>
HistogramData
- costDistribution : List<Double>
- frequencyDistribution : List<Integer>
+HistogramData(clicks : List<Clicks>, bars : int)
- calculateCostDistribution(clicks : List<Click>, bars : int) : List<Double>
- calculateFrequencyDistribution(clicks : List<Click>) : List<Integer>
+ getCostDistribution() : List<Double>
+ getFrequencyDistribution() : List<Integer>
Metrics
- impressions : int
- clicks : int
- uniques : int
- bounces : int
- conversions : int
- totalCost : double
- CTR : float
- CPA : float
- CPC : float
- CPM : float
- bounceRate : float
+ Metrics(csvData : CSVData, pagesViewed : int, timeSpent : ModelTime)
+ Metrics(impression : List<Impression>, clicks : List<Clicks>, interaction : List<Interaction>, pagesViewed : int, timeSpent : ModelTime)
-calculateUniquesAndTotalCosts(impressionsList : List<Impression>, clicksList : List<Clicks>) : Pair<List<Integer>>
- calculateBouncesAndCoversions(interactions : List<Interaction>, pagesViewed : int, timeSpent : ModelDate)
- checkTimeSpent(entryDate : ModelDate, exitDate : ModelDate, timeSpent : ModelTime) : boolean
+ getImpressions(): int
+ getClicks() : int
+ getUniques() : int
+ getBounces() : int
+ getConversions() : int
+ getTotalCost : double
+ getCTR() : float
+ getCPA() : float
+ getCPC() : float
+ getCPM() : float
+ getBounceRate() : float

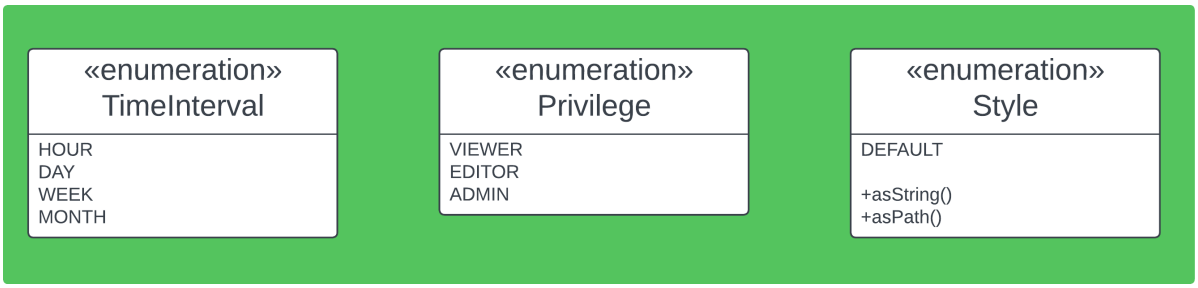
1.1.3. Controller



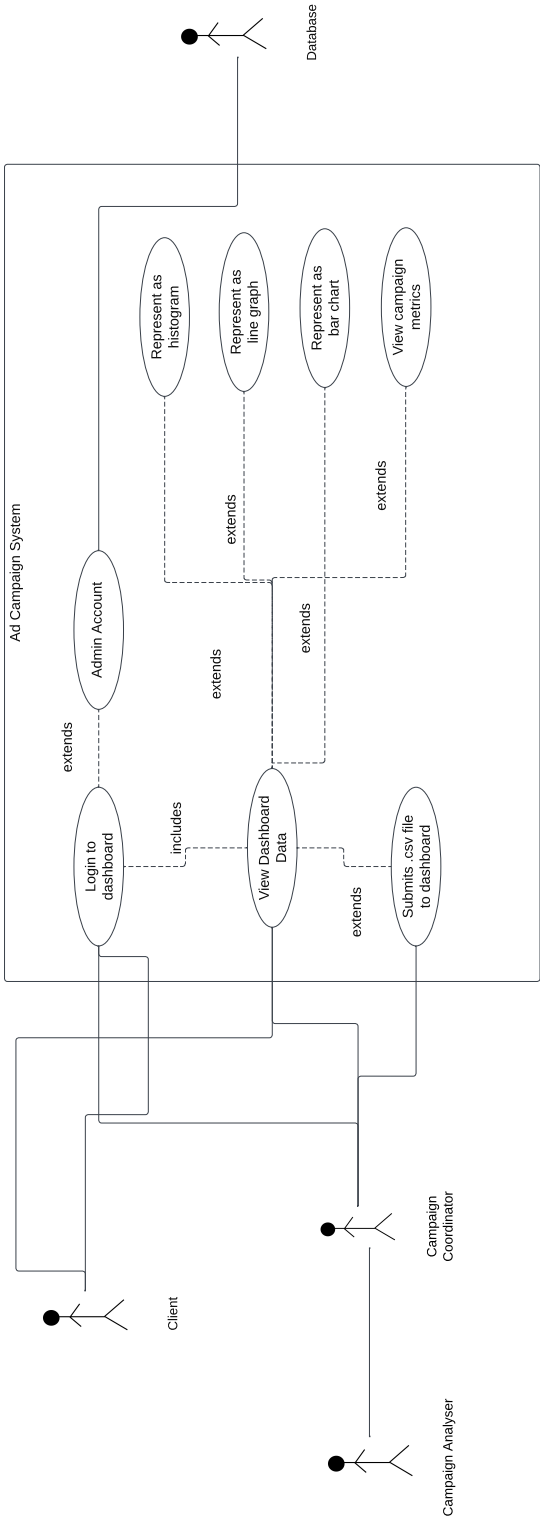
1.1.4. Exceptions



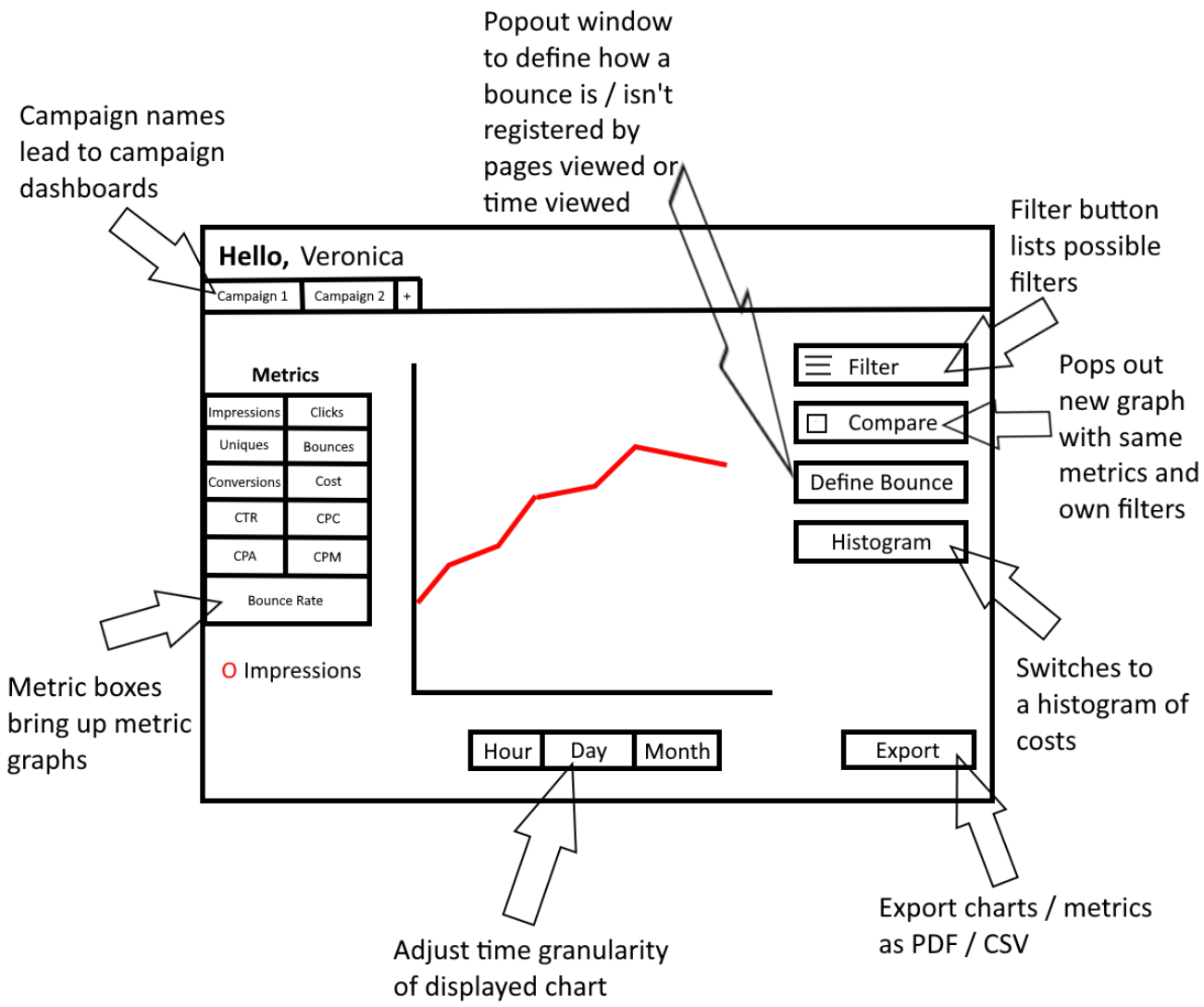
1.1.5. Enums



1.2. UML Use-Case Diagram



1.3. Storyboard



1.4. Scenarios

[A] Veronica Haileys is a first-time entrepreneur:

1. Veronica opens the application
2. She logs in using the credentials provided by the marketing agency
3. She submits the CSV data directory provided by her agency
4. Veronica can view key information about her campaign
5. She can check what terms mean, how they are calculated, and what unit of measurement they use
6. She is notified of any accidental misuse of the software
7. She can also view instructions on how to use the software if needed

[B] Sharon McGee is an Etsy store owner:

1. Sharon opens the application
2. She logs in using the credentials provided by the marketing agency
3. She submits the CSV data directory provided by her agency
4. Sharon can switch to graph view
5. She can view trends of campaigns for different Etsy listings overtime

6. She can easily compare these trends visually using filters

[C] Bill Hawks is a marketing agency manager:

1. Bill opens the application
2. He logs in using the credentials provided by the software administrators
3. Bill can create new dashboard viewing accounts for his clients
4. Bill can also submit and view campaign data for his team's campaign analyser to determine trends
5. Agency members can now better-target campaigns based on trends and audience data

[D] Marcus Hemmering is a returning entrepreneur who wants to look into other markets:

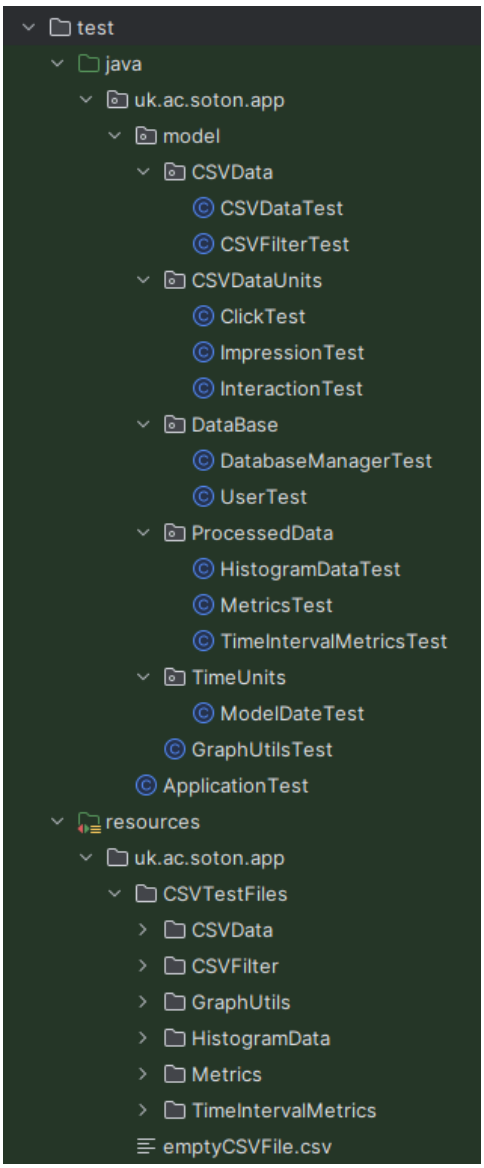
1. Marcus opens the application
2. He logs in using the credentials provided by the marketing agency
3. He submits the CSV data directory provided by his agency
4. Marcus can view a histogram of his total costs on campaigns overtime
5. He can define what is / isn't considered a bounce
6. He can also export charts to share with his employees

2. Testing

2.1. Development Testing

2.1.1. JUnit

We have set up unit test harnesses to ensure the correctness of the major "model" components of our application. Please find these below:



Class	Description
ApplicationTests	Smaller test class that ensures that the test harness and application run succesfully.
CSVDataGettersTest	Test that takes in sample .csv files (such as the empty data files and valid data files), reads in this data, and checks that the output click / impression / server objects align with the contents of the file. This is done both with and without filtering applied.

Class	Description
readClickLogTest	Reads through the click log, testing that all click objects align with the contents of the file. Also checks that the correct exceptions are thrown for: • an empty file • the lack of a file • a file with headings, but no contents • a file with one (or more) missing column(s) • a file with one (or more) extra column(s) • a file with invalid data in one (or more) column(s)
readImpressionLogTest	Similar to "readClickLogTest" but specific to impressions.
readServerLogTest	Similar to "readClickLogTest" but specific to the server log.
ModelDateTests	Provides extensive checking on functions used for comparing / representing dates, such as checking that one date comes before another.
ClickTests	Checks that methods can be used to receive specific click components, such as the click cost.
ImpressionTests	Checks that methods can be used to receive specific impression components, such as the ID.
InteractionTests	Checks that methods can be used to receive specific interaction components, such as the entry date.
CSVCheckerTests	Ensures that any passed CSV data that's in the incorrect format is flagged as erroneous. Tested on all forms, including impressions, clicks, and interactions.

2.2. Acceptance Testing

To perform acceptance testing, we have been going through our aforementioned scenarios and using them as a validation criteria in order to ensure that our goals have been achieved. Each scenario has been given an identifier (A, B, C, D), with each task being given a number. These are used to form task-unique identifiers for us to use to assess our criteria (e.g. is A1 complete?). Please see our acceptance test table below:

Scenario ID(s)	Passed	Description
A1, B1, C1, D1	Yes	Users should be able to open the application
A2, B2, C2, D2	No	Users should be able to log into the application with different privilege levels
A3, B3, D3	Yes	Users should be able to submit campaign CSV data to be processed
A4	Yes	Users should be able to view a summary of key campaign statistics
A5	Yes	Users should be able to view what different statistics mean, how they are calculated, and what units of measurement they are represented in
A6	Yes	Users should be provided concise error messages explaining issues
A7	No	Users should be able to view detailed instructions on how to use the application
B4	Yes	Users should be able to view graphs of different metrics over time
B5	Yes	Users can view trends over time by adjusting time granularity of graphs
B6	No	Users should be able to filter graphs and compare similar graphs with different filters applied
C3	No	Agency members should be able to create viewer accounts for their clients
C4, C5	No	Agency members should be able to view campaign data like viewers can
D4	Yes	Users should be able to view a histogram of their click costs
D5	Yes	Users should be able to provide their own definitions for a bounce
D6	No	Users should be able to export graphs to PDF / image

2.3. System Testing

Tests that have been converted into unit test equivalents from our previous list of test cases have been marked with a *:

ID	Test	Test Data	Expected	Actual	Action
0	Can switch from login page to dashboard page	Press login and select a directory with an impression, click and server logs	Should switch to dashboard when login	Switched to dashboard when login	N/A
1	Selecting an invalid directory is handled	Select directory with no impression log	Should display warning	Displayed warning	N/A
2	Selecting an invalid directory is handled	Select a directory with no click log	Should display warning	Displayed warning	N/A
3	Selecting an invalid directory is handled	Select directory with no server log	Should display warning	Displayed warning	N/A
4	Closing directory selector without selection	Select no directory	Should display warning	Error	Catch error and display warning popup
5	Closing directory selector without selection	Select no directory	Should display warning	Displayed warning	N/A
6*	Correct number of impressions is calculated and displayed	Press login and select a valid directory	486104	486104	N/A
7*	Correct number of clicks is calculated and displayed	Press login and select a valid directory	23923	23923	N/A

ID	Test	Test Data	Expected	Actual	Action
8*	Correct number of uniques is calculated and displayed	Press login and select a valid directory	23806	23806	N/A
9*	Correct number of bounces is calculated and displayed	Press login and select a valid directory	23867	23867	N/A
10*	Correct number of conversions is calculated and displayed	Press login and select a valid directory	2026	2026	N/A
11*	Correct total cost is calculated and displayed	Press login and select a valid directory	1180.98	1180.98	N/A
12*	Correct CTR is calculated and displayed	Press login and select a valid directory	0.0492	0.0492	N/A
13*	Correct CPA is calculated and displayed	Press login and select a valid directory	0.583	0.583	N/A
14*	Correct CPC is calculated and displayed	Press login and select a valid directory	0.049	0.049	N/A
15*	Correct CPM is calculated and displayed	Press login and select a valid directory	2.430	2.430	N/A
16*	Correct bounce rate is calculated and displayed	Press login and select a valid directory	0.9977	0.9977	N/A
17	Can switch from dashboard view to graph view	Click graph on view drop-down	Should switch to graph page on select	Switched to graph page on select	N/A
18	Can switch from graph view to dashboard view	Click dashboard on view drop-down	Should switch to dashboard page on select	Switched to dashboard page on select	N/A
19	Metric graph is displayed	Click graph on view drop-down	Should display metric graph	Displayed metric graph	N/A

ID	Test	Test Data	Expected	Actual	Action
20	Metric graph has correct y-axis	Click graph on view drop-down	Should have a y-axis of the metric's values	Had a y-axis of the metric's values	N/A
21	Metric graph has correct x-axis	Click graph on view drop-down	Should have a x-axis of dates	Had a x-axis of dates	N/A
22*	Correct values are displayed on the metric graph	Click graph on view drop-down	94, 96, 115, 105, 110, ...	4, 6, 16, 1, 68, ...	Fix error in grouping entries in the log files according to a time interval.
23*	Correct values are displayed on the metric graph	Click graph on view drop-down	94, 96, 115, 105, 110, ...	94, 96, 115, 105, 110, ...	N/A
24	Can open/close bounce menu to update definition	Click bounce button to open, apply to close & update	Open and close menu	Open and close menu	N/A
25	Can open/close filters menu to update filters	Click filters button to open, apply to close & update	Open and close menu	Open and close menu	N/A
26	Can open/close time menu to update time settings	Click time settings button to open, apply to close & update	Open and close menu	Open and close menu	N/A
27	Can toggle between cost histogram and metric chart	Click histogram button to toggle	Switches graph view and updates button text	Switches graph view and updates button text	N/A
28	Can switch displayed metrics	Press metric checkboxes	Updates graph instantaneously	Updates graph instantaneously	N/A

3. Planning

3.1. This Increment

3.1.1. Burndown Chart

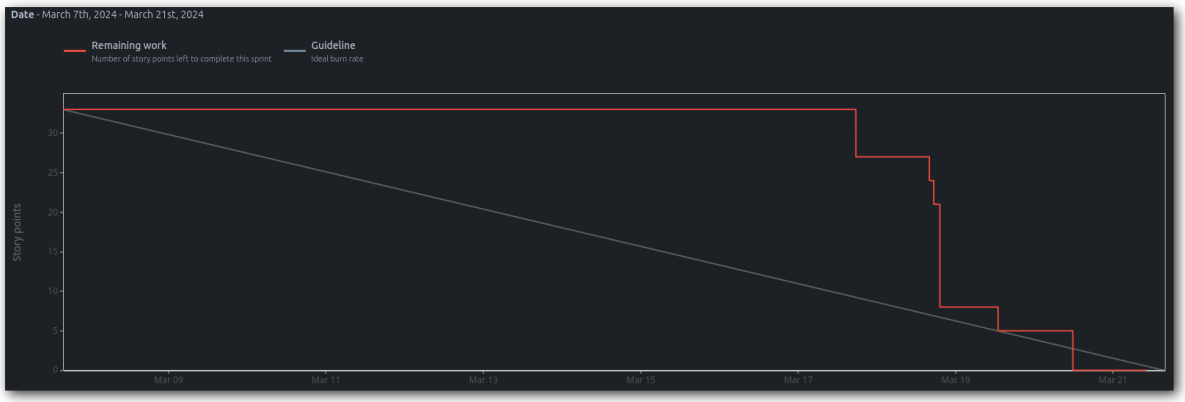


Figure 1: Final burndown chart for increment 2

3.1.2. Sprint Plan Review

N/A: moved back to next sprint

Story Description	Member	Expected	Actual
Clients should be able to view historical data by adjusting time periods of displayed data	AM	2	3
Clients should be able to view metric charts to track performance trends	BH	3	3
Clients should be able to view multiple charts at once to compare data	BH	5	N/A
Clients should be able to view metrics per hour, day, week, or month	PJ	5	5
Clients should be able to filter metrics	DS	8	8
Clients should be able to provide a custom definition for bounces	AM	3	3
Users should be able to receieve custom error messages for troubleshooting issues	KC	2+2	2+2

3.2. Next Increment

3.2.1. Burndown Chart

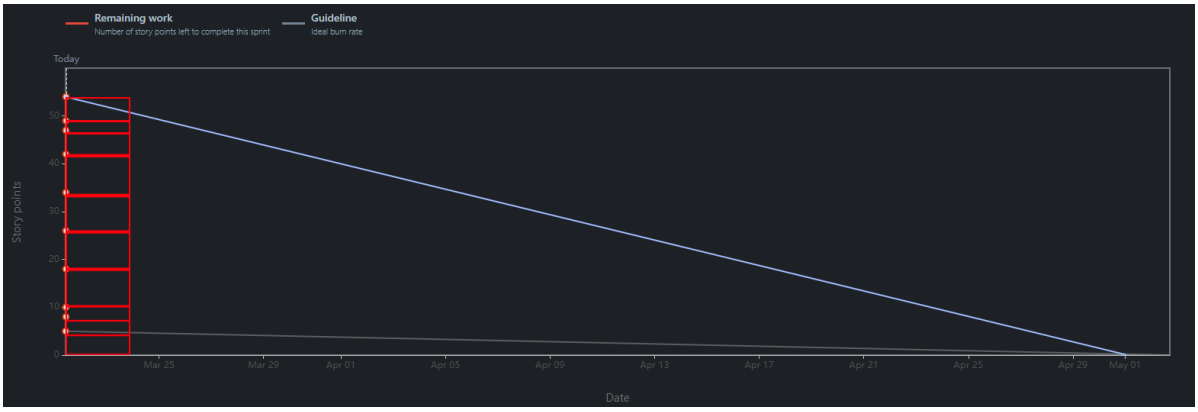


Figure 2: Day-zero burndown chart for increment 3

3.2.2. Sprint Plan

Please see our sprint plan for the upcoming increment (Figure 3). This sprint contains a lot of high-effort stories (with multiple instances of 8-point assignments) due to the fact that the duration of the sprint is much longer (6 weeks). Balancing of story points for each members has been made to its best effort. Moreover, as we have already began the implementation of graph comparison (as it was pushed back from the previous increment), this story has been marked as such (shown as "in progress").

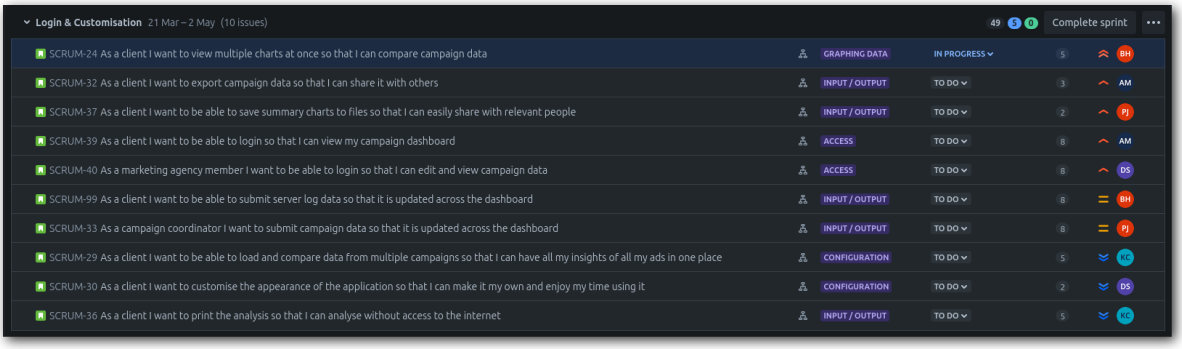


Figure 3: Sprint plan for increment 3