Personal Reflection (Student ID: 30012627)

Team working

In what ways was your contribution effective?
In the Envisioning stage, I took the responsibility of producing the design artifacts Use Case diagram and UML Class diagrams for all increments, as well as developing Front-End along with my front-end development team partner Dylan. In terms of programming, my implementations include developing the functionality to display metrics from a particular campaign, add different campaigns in a table inside the tab pane, create a listview component to add, edit filters, set up the graph associated with a particular metric, alter time granularity according to the day, week, month, or year, and open a new chart window to compare charts depicting different metrics and charts with different filters applied to them. In addition to the basic functionality of the system, I implemented the accessibility features which include variable font size and having a toggle between dark-mode or light-mode. My contribution was particularly effective as I was quick and transparent when generating ideas on how to develop and improve the user interface.

Describe how the team achievement would have been different without your input?
Overall my team has been very strong; everyone is skilled at programming and took full responsibility for their work. However, since I spent a significant amount of time learning JavaFx, using Scene Builder, planning and developing the User Interface, and developing UML Diagrams, the team may have suffered in terms of planning without my input. Towards the submission of the final increment, we were constrained with time, and most of the deliverable included adding features to the user interface. As a result, Dylan and I put in the most hours of work in the final increment to get the user interface final and ready - the team may have found it more difficult to complete the design of the user interface without my input, and this would have put more pressure on Dylan.

How does working in a team compare to working alone?
I, personally really enjoy working in a team, partially due to the fact that I am extroverted and prefer communicating my ideas out loud and working with others' ideas as well. Moreover, I believe that working in a team allows me to increase my efficiency, as I can consult other team members on how to solve difficult design problems. I believe that through collaboration, ideas develop further and productivity increases. Working together allows for unified perspectives, combined strength and collective brainstorming, which means that different approaches to solving the problem can be explored and difficult tasks can be finished off more easily. Peer programming more specifically has been very beneficial in the development process, as it made my partner Dylan and I recognise each other's mistakes when programming on one computer. Now, working in a team also requires many more meetings and decision-making takes longer. Working at your own pace allows for easier concentration and gives you the ability to make your own decisions since you are solely responsible for your job. Thus, in my opinion, it is necessary to work alone at times, even while working on a team project.

Did you encounter any conflicts, and how did you personally deal with them?
There were a few times when there was lack of clarity on who is doing what, as there is a lot of overlapping between the different components of the system. In such cases, the task and issue tracking software Gitkraken Glo was really beneficial as I could tick off what I have completed and effectively communicate with my teammates. While planning the layout of the GUI interface, we all had different ideas in mind about how the application would look but taking into account everyone's ideas and then generating a combined, well-thought solution for best user experience. In terms of programming, I

encountered various merge conflicts (when one of my peers had been working on the same class as me at the same time). To avoid this from happening, I pushed my code on to our git repository on a regular basis so that I could always retrieve previous versions of my work.

What would you change about the way you worked and interacted with your other team members if you were to start the project again?
Now that we know each other's strengths and weaknesses, I may have suggested distributing the work differently. Since William(mediator between front-end and back-end teams) has a strong experience in developing front-end, I would have approached him for help earlier in the final increment, as we struggled a lot to get everything done on time. After interacting a lot through messaging, I can definitely say that I would have proposed to organize more zoom calls, especially with my front-end developing partner Dylan, to get a better understanding of our progress in terms of programming, utilizing the screen-sharing feature.

What was the most valuable experience for you during this project?
The most special element about this project is that it simulates a real-time software engineering group project and allows us to use knowledge from lectures in the project. It involves planning and executing an actual software from scratch, and working under pressure as a team to deliver increments in a timely manner. Moreover, having discussions about the application with the supervisor gave an experience of how it would be to work with a client in the ‘real world’.

Personal Achievement

I am particularly proud of building the functionality for displaying and adjusting the graph data based on the bounce rate, time granularity and the type of chart (per hour of day vs per day of week etc.), as it took me the longest to implement. In order to carry out this implementation, the first step was planning. I would first need to create a class to represent points in the graph as instances, add functionality to load the graph points associated to a specific metric into the Controller to create the series and recreate the graph depending on adjustments recommended by the user. As a result, I discussed my plan with the rest of the team and updated the class diagram to accommodate for these changes.

Following the plan, I then created the GraphPoint class - so that I could create an instance of this class for every new point to be added to the chart, and set its location in terms of x and y coordinates to classify it as an outlier if the case. Next, I implemented loadCampaignPressed() method - which creates a new task to load the base data and queries for overall metrics (through the Model) associated with a campaign id. The next step was to implement the createSeries() method which creates a new series for a campaign based on the time granularity value, graph type and point data. All the data points were then plotted in the graph and the outliers were picked (whether a datapoint is an outlier or not depends on outlier strictness). If the time granularity value is altered, the recreateGraph() method takes as an argument (a time granularity value) and creates a corresponding series.

The interface also gives the user the ability to define the custom bounce rate/ number of pages visited, and check whether the custom bounce box is already selected. Upon selection of any of these, the data associated with a specific campaign is updated and the chart is again recreated to reflect this. In order to integrate the MainController with the system, I created an instance of the MainModel class to access campaign specific methods for querying of data, setting filters, computing mean etc. The last step was to link the newly added methods in the MainController to the mainScene.fxml file (through the @FXML tag) to actually display the components in the user interface. I used Scenebuilder for this, and decided to use the line chart and HBox to display the chart and put all the controls for Time Granularity, Outlier

Strictness and the type of graph respectively. I am quite satisfied with how the implementation has turned out to be visually - the graph takes up the top half of the GUI and graph points can be clearly read. The controls are also laid out in a simple manner along a row just below the graph.

The most challenging part of this implementation was figuring out how to create the series initially upon receiving the points, and then setting up the graph according to the specified time granulation value. Both of these required a significant amount of thought and planning beforehand.

Agile Methods (Envisioning: User Stories, Storyboards, Stakeholder analysis, personas)

A user story is based on how one would capture the description of a software feature from an end-user's point of view. More specifically, it describes the type of user, what they want from the software, for whom and why. User stories have been particularly effective because by nature, are simple and easy to use, and allow for the final product to be defined with clarity. Additionally, they allow for all members of a team to understand the application requirements regardless of their technical abilities, and help establish the customer priorities in terms of their needs and the delivery order. The basic form of a user story is: As a <type of user>, I want <some goal> so that <some reason>.

User stories have been an essential part of our group project. We generated ours by identifying the <type of user> as a primary/ secondary stakeholder, <some goal> as a particular requirement in the Ad Auction Specification, and <some reason> as a hypothetical scenario in which the stakeholder could validate that the requirement has been met.

Initially, we categorized the user stories as musts, shoulds and coulds, keeping in mind how much time each requirement in the story would take to be implemented, and its importance relative to the entire software. We then consulted the set of prioritised user stories with our supervisor to align them with their priorities and added them to the product backlog, which contains the set of prioritized user stories requirements that need to be implemented in the upcoming increments. Following this step,we selected stories to be delivered in each sprint and put them in the sprint backlog, with the condition being that basic requirements are picked first and that similar numbers of musts, shoulds and coulds are selected for each increment. Furthermore, the estimated task size for each user story in the sprint backlog(set of product backlog items selected for a sprint), in turn, also helped in graphically representing how much work is left to be done versus time through the burndown chart. For each increment, we prioritized musts over shoulds and coulds. We also distributed user stories in the sprint backlog, using task and issue tracking software GitKrakenGlo. As user stories capture how the software is meant to function, we used them for planning the layout of storyboards and used them to develop test criteria and scenarios for validation testing.