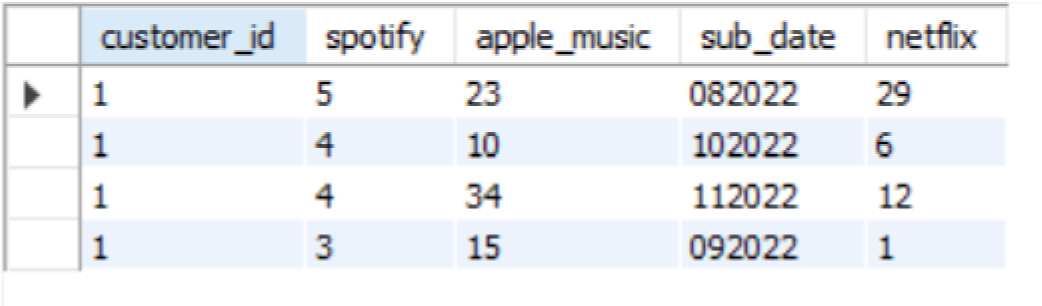
Scrum Master: Azam Shoaib
Jihoon Park
Ken Atta-Boakye

Sprint 3

ID	As a…	I want to be able to…	So that…	Priority
1	Administrator	Know what the user's subscriptions are	We know what apps to keep track of	Must
2	Administrator	See subscription app usage	See if the user is utilizing their subscriptions	Must
3	Administrator	Keep user data on spending history	Propose better budgeting for the user	Must
4	User	Be able to input my subscriptions	The program can see how much the user utilizes each subscription	Must

Sprint ID - 1: Know what the user's subscriptions are

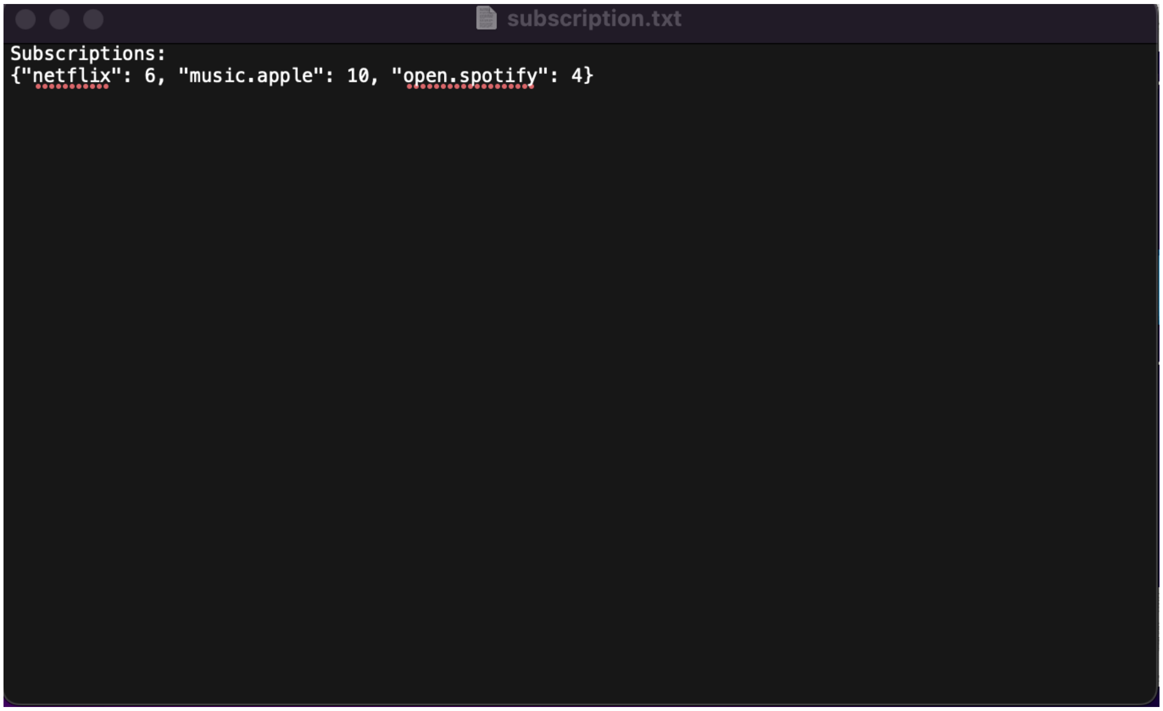


	customer_id	spotify	apple_music	sub_date	netflix
▶	1	5	23	082022	29
	1	4	10	102022	6
	1	4	34	112022	12
	1	3	15	092022	1

Figure 1: Database info/date table

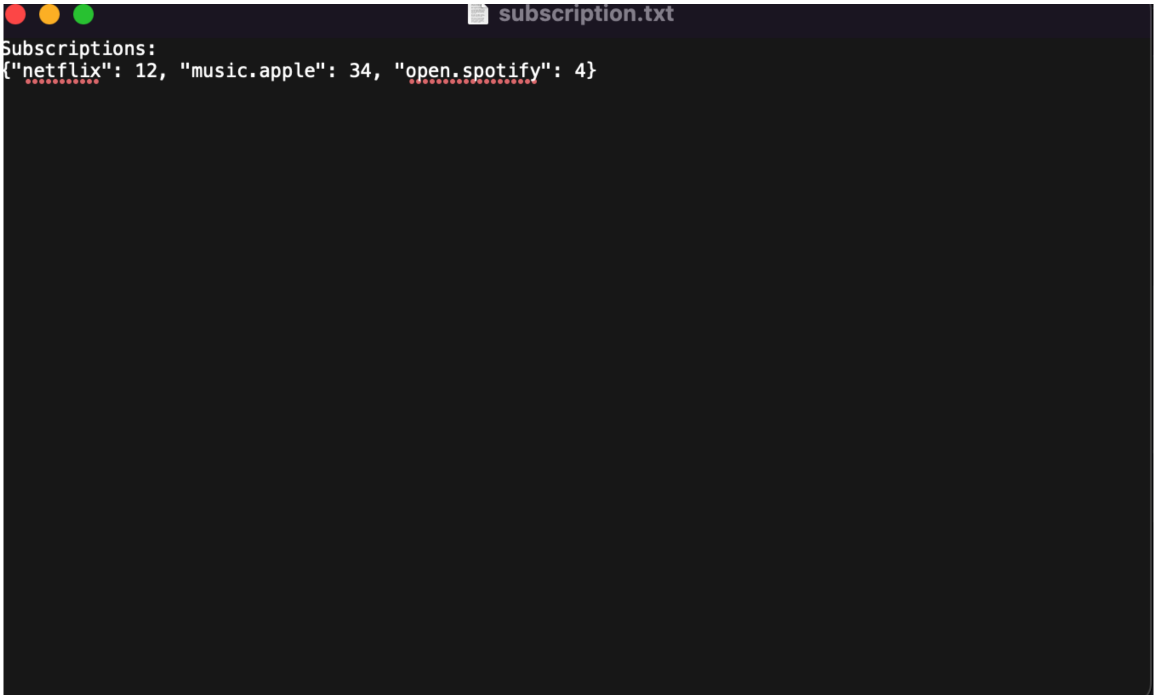
For this sprint ID we wanted the ability, as an administrator, to have knowledge on the user's subscription history. We have limited the subscriptions to spotify, apple music, and netflix. While their system is running we can see when the user visits these particular sites and this information is updated to the database for the admin to keep track of. This same information is reflected later in Sprint ID-2 and Sprint ID-4.

Sprint ID - 2: See subscription app usage



```
Subscriptions:
{"netflix": 6, "music.apple": 10, "open.spotify": 4}
```

Figure 2: txt file 1



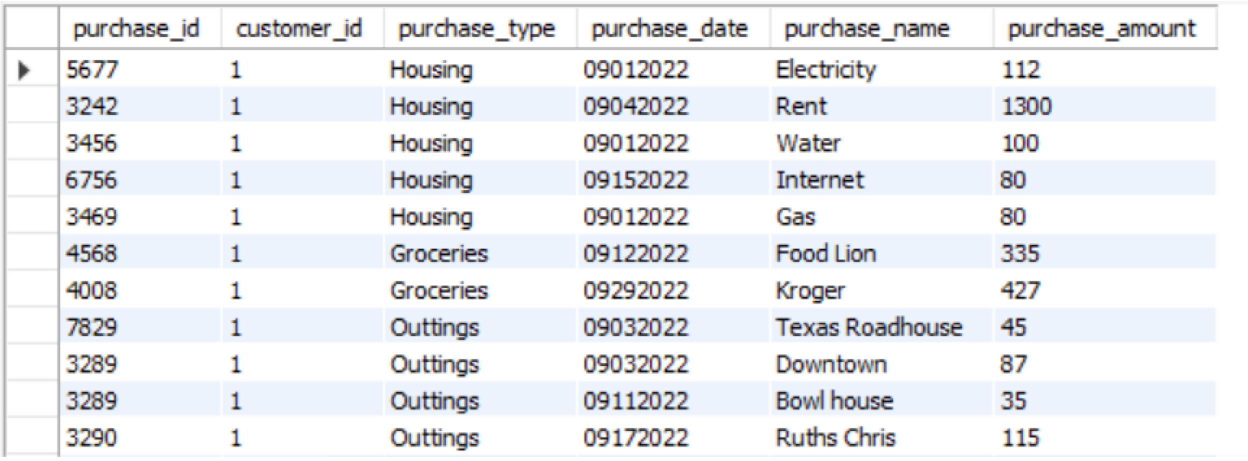
```
Subscriptions:
{"netflix": 12, "music.apple": 34, "open.spotify": 4}
```

Figure 3: txt file 2

The first picture shows which we will assume the first month of the person using the application to track their subscriptions and the second picture shows the second month. In our database we will do the calculation for the difference between month 2 and 1 because the data we get from the DB file adds on to each to the current count. To get the correct data value for the time period we need to exclude the previous month's count. This will be done for x amount of months or time intervals.

We as of right now are tracking three subscription sites: Netflix, Apple Music, and Spotify. For Netflix we are keeping a safe range to keep the subscription at 20 visits per month. For Apple Music and Spotify we are keeping a safe range to keep the subscription at 35 visits per month. As this is currently a desktop application, these values were reasonable to us as a group, because nowadays we tend to use our phones to watch videos and listen to music but occasionally we will use our laptop/desktops for it.

Sprint ID - 3: Keep user data on spending history



	purchase_id	customer_id	purchase_type	purchase_date	purchase_name	purchase_amount
▸	5677	1	Housing	09012022	Electricity	112
	3242	1	Housing	09042022	Rent	1300
	3456	1	Housing	09012022	Water	100
	6756	1	Housing	09152022	Internet	80
	3469	1	Housing	09012022	Gas	80
	4568	1	Groceries	09122022	Food Lion	335
	4008	1	Groceries	09292022	Kroger	427
	7829	1	Outtings	09032022	Texas Roadhouse	45
	3289	1	Outtings	09032022	Downtown	87
	3289	1	Outtings	09112022	Bowl house	35
	3290	1	Outtings	09172022	Ruths Chris	115

Figure 4: database table information

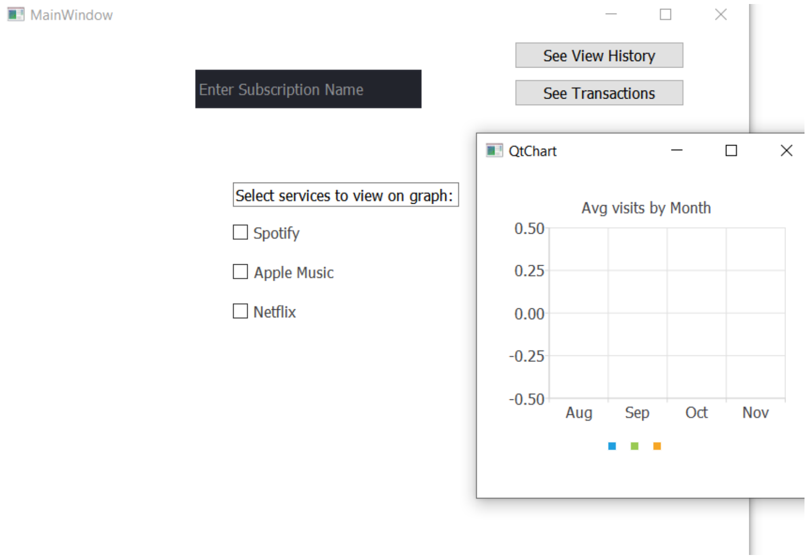
For this project because we do not have access to the user's actual bank information, we prompt the user to input their transaction information in the GUI. This information is sent to the database as shown in the table above. Our system is able to pull the data from the database and see whether the user went over their proposed budget for the month in any category. In this example, the user exceeded their budget for outings.

We want to be able to give the user useful feedback to help them continue to budget, that is why we also created an assistant using python to suggest reallocation of the budget. This part of the project is still under development, we are trying to find the best way to propose the reallocation. The method right now will only work if money was overspent in one area and there is less than 80% of a budget used elsewhere.

Sprint ID - 4: Be able to input my subscriptions

After the user tracking history has been uploaded into the database, the information is then pulled into a separate file so that our program can read the information into a visual format. Since the database will keep track of user history, we can say that this will be the most accurate and up to date information that the user can see.

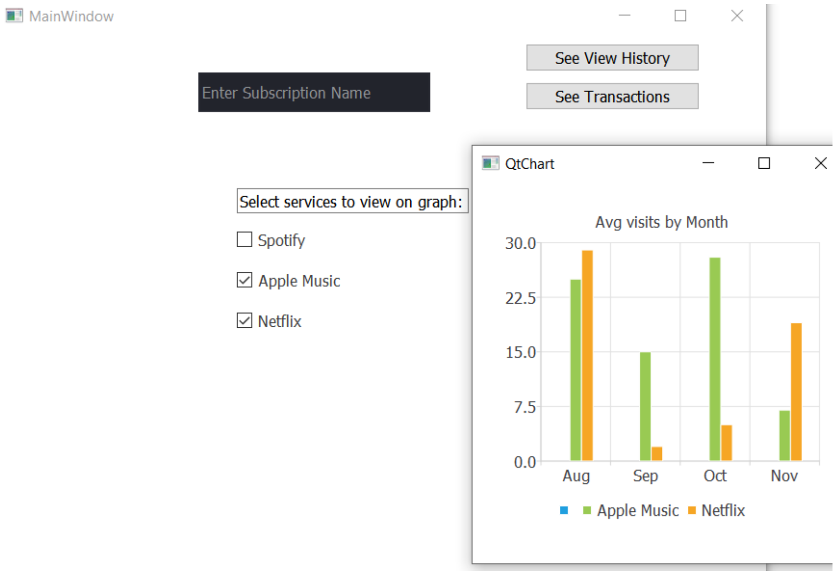
The new implementation does not require the user input for subscriptions. For testing purposes we made three focal subscriptions the user can have, which would be netflix, spotify and apple music. This can definitely be improved upon before the final deliverable where we do ask the user for which things they are subscribed to but would be highly difficult. Reason being is that each subscription site has keywords that would need to be added to the array in the code so the program can acknowledge if the subscription site is in use and not someone just browsing.



☐ Spotify
☐ Apple Music
☐ Netflix

Figure 5: User checks no boxes

In the GUI, the user is greeted with a few interactable buttons from which they can choose to see their subscription visits per month or to see their transaction history. This will open up a separate window where a graph will be displayed. The user is also able to check some of their subscribed subscriptions to see only data regarding that service.



☐ Spotify
☑ Apple Music
☑ Netflix

Figure 6: User checks boxes

Sprint Retrospective

Sprint 3 delved deeper into communication between our database to GUI to separate files. Here, we needed to make sure that the database is able to receive user subscription visit information. The information then gets filtered and needs to get sorted by month. On the GUI side, the user will have the choice to pick and choose which subscription they would like to see as an informative graph. Our next step would be to integrate everything over to Python as it will be easier for us to connect with our database. This will mean changing our GUI implementation such that it will function in python.