The report on the application "No Shame"[1]

Khrapunov Pavel 202-2
Supervisor: Georgi Zhulikov

June 8, 2021

[1] https://github.com/Pavlida/dsba-itop2021-hw1

Problem statement

I was asked to create an application that counts the amount of time one needs to workout in order to burn all the calories they've gained after eating a meal. It had to be done by implemetning a way to load a menu as csv and interact with it's elements.
This required me to create a system that transfers the necessary data between widgets and is dynamically changing if required.

Project specification:

So that there is no shame.

This calculator allows you to calculate the calories and nutrition facts of each item from the McDonald's menu, and also shows how much time you should exercise after such a meal.

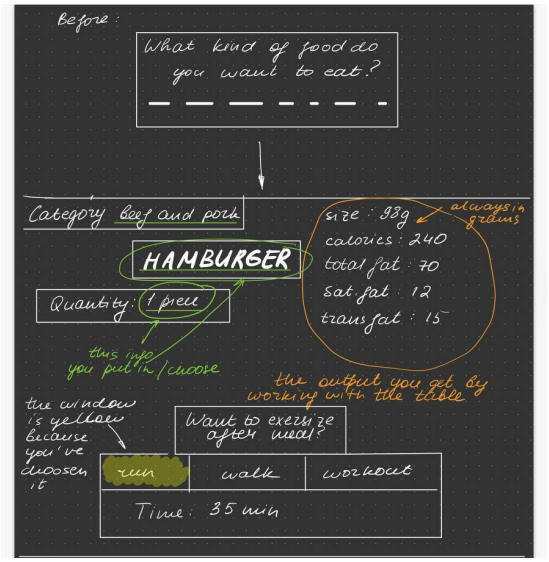
Go to Kaggle and find 'Nutrition facts for Macdonald's menu' data table. You will need only this columns: category, item, serving size, calories, total fat, saturated fay, trans fat.

https://www.kaggle.com/mcdonalds/nutrition-facts

1) Put in a category -> you get a list of all items from this category;
2) You can even search for the item in the list or type it in the 'search window';
3) You get list of nutrients of this item

Additionally:

1) you can choose several items. If so, as a result you get nutrients sum.
2) You can calculate the time you have to exercise for not to gain weight. There are 3 types of them: run, walk, workout. Make your own formula, taking into account that 1min of run burns 12cal, 1min of walk burns 4,5cal, 1min of run burns 30cal.



Implementation details

The task that i had before me was rather similar to what we have already done on our workshops so I decided to make it harder on me and have some additional features of my own and imlpement them in new ways.

For now, let's just focus on the specification.

Basic tasks

Firstly, I had to think of a ui and, since the project reminded me of choosing items and adding them to your shopping cart in any online store, I created something of that nature.

In order for the project to work, I had to implement a model that displays and properly stores the data. We have done so on our workshops so I practically copied the code from there. However, when parcing the model from a file of users selection, I check whether it has three main columns: Item Name, Calories, Category. This is the minimal requirement for a data to count as valid.

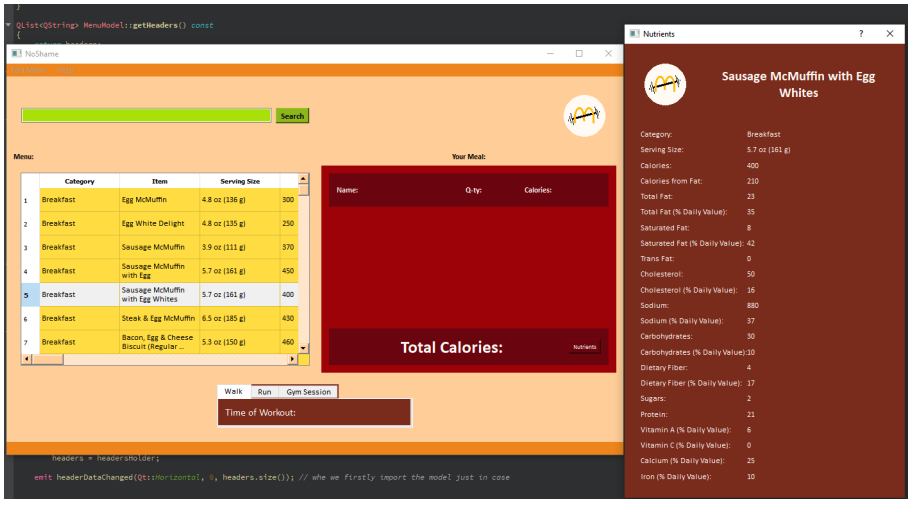
```
if(headers.isEmpty()) //if those are undefined we need their indices
{
    categoryColumn = headersHolder.indexOf("Category");
    nameColumn = headersHolder.indexOf("Item");
    caloryColumn = headersHolder.indexOf("Calories");
}

if (categoryColumn == -1 || nameColumn == -1 || caloryColumn == -1) //if unseccsesful throw error and refresh
{
    categoryColumn = -1;
    nameColumn = -1;
    caloryColumn = -1;
    throw std::domain_error("The Menu dosn't have the necessary columns to properly operate (Category, Item, Calories)");
}
```

Moreover, I had to slightly alter the data by changing the separator to ';' because there were commas inside some of the item names.

The first task in the specification seemed a little bit off since the user might not know the exact name of categories in the menu, so i decided to execute the task a little differently. I created custom QTableView that makes it possible to show all the categories by clicking on the corresponding header. Furthermore, a custom QSortFilterProxyModel that I created makes it so that whenever a box of a category is unclicked, it's items are no longer shown. This is an easier and much more intuitive way to filter categories for the user.

Using the same QSortFilterProxyModel I make it possible to search for an item by it's name (2^{nd} task). Whenever the search button is pushed, I send the text in the searchBar to the model and invalidate the previous filter so that the model refreshes and checks not only the categories, but also the sear request. This seamed easier to do than creating multiple filter models + it would be easier to get lost in all those files.

For task 3, I decided to create a separate window that shows all the details of an item (and later meal), so it doesn't take up a lot of space in the mainwindow. I get all of the info about the item from the menu through the table, so I simply display it by inserting rows of headers (gathered earlier from the menuModel) and items of corresponding indices in the QFormLayout. Unfortuantely, if we have to look at another item we have to clear all the previous rows and insert the rows again (prevent the case when user has loaded a new menu).

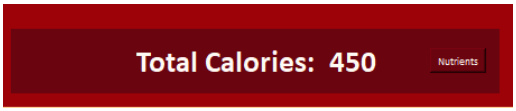


Aditional tasks

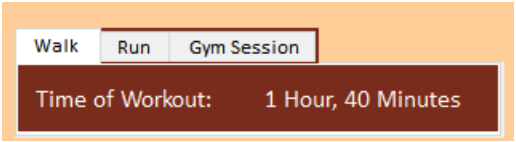
For getting the nutrient sum of a group of items I decided to approach the problem by creating a menu widget which you can use to add new items by right-clicking an item in the tableView. As it turns out, it wasn't so easy. I had to create a custom context menu for the tableView that is called any time a user right-clicks an item, and I used the menu for the details of an item as well so I don't overflow the ui with buttons.

The menu has Q-ty spinbox for all the items, which changes the calories of an item and the total calories of the menu. If the item is already in the Menu, which we check by it's name, that is set to not be changed, we just increase it's q-ty by one if possible.

If the user wants to see the nutrien sum of all the items in the menu he can easily do so by pushing the nutriens button that starts showing the same window used for displaying the details of a single item, with just a few thing missing because not everyhing, like category can eficciently be shown in a small window.



The total caloreis are connected to the tabWidget that has differnet categories walking, running, and working out in the gym every time the calories change, we also change the time of every workout according to the info provided in the specification.



My decisions

I've decided to add revert changes, delete row and disable sorting buttons, just to make it more convenient for the user. The first feature is done by saving the file path and reopening it when asked to.

I've decided to not change the values of the categories so the user doesn't tamper with the category menu. I've also left names unchangable because there is no reason for user to change them and it would make the addition to menu confusing for them. Sort of a dummy-block. I think it would be even better if the menu was uneditable, but I wanted to leave that freedom to the user.

For the other parameters like calories you need to re-insert the items in the meal or re-open the details in order for their properties to change. I've done it so that if the data is removed or overwriten by new menu the user still has acces to the correct old data as long as he needs to.

Results and discussion

I think I have created a great application that is functioning like an actual professional app. I didn't exactly follow the specification, because in some areas I believe I came up with a better solution. But the functioanality is all there and the app is user-friendly, compact and beautiful.

Conclusion

All in all, I am proud of all the things that I have learned and implemented along the way. I would say that I have pushed the limits of the specification that I had and there is not much room for improvement. The only thing that I can think of is extending the menu column-wise, manually adding an item and maybe some sort of export of your meal and workout or something of that nature. It took me approximately 20 hours to finish and I never would have thought that it takes so much time to create an application that it is not very complex. I can only imagine how long it takes to finish big projects.