National Research University Higher School of Economics

Faculty of Computer Science

Data Science and Business Analytics

Introduction to programming (C++)

Project review

C++ GUI application

“Nash Music”

By student:

Kornelyuk Egor

Group:

214-1

Submission date:

08.06.2022

Supervisor:

Carrasquel Julio

Moscow 2022

Problem

The main objective of the project was to design and code such application which would fit project specification with advised GUI design. During the process of coding, I've learnt to work with QT library and QT models such as QStandardItemModel or QStringListModel. Moreover, I faced some major inconveniences with memory management but still found a way to deal with them: for instance, passing a std::vector<Song> to another window without losing full functionality of editing/representing it.

Individual Problem Specification

This project will help your country overcome all the sanctions and restrictions. You are going to make an application that can search for a music, add one to customer's playlist and see the specifications of it. Overall, the domestic music application.

Implementation details section

Link to project's GitHub repository:

https://github.com/feddes/dsba-itop2022-hw

Application consists of 5 windows: main window, playlist window, about window, info window and add window. Playlist window has a direct connection with main window because it is the place where it gets const std::vector<Song> &plst and works with it. Also, application includes ProxyModel(.h/.cpp) files which is basically the code for managing filtering/sorting/searching system. Song.h file is just a header file with one struct inside of it, yet it is very convenient way of storing collection of objects. There are 2 QStandardItemModels in MainWindow for showcasing the data from csv file and one in PlaylistWindow for working with playlist. To get the data from the input I used the standard way of reading data line-by-line from file (using std::getline), worked a bit with finding a way to avoid bugs with input data (for example: comas in titles) and then I structed the songs by parameters and put them to std::vector<Song> vec. Each button which is implemented in the application has its own private slot in corresponding header file and implementation of it in cpp file. Save buttons implementation allows user to save table via new csv file. Logo was drawn using QPen. Some buttons were custom made meaning they weren't changed theirs form and change dynamically when being pressed.

Results and discussion

The obvious result of work is a simple GUI application which is capable of working with database with functions of editing it, adding new objects, creating sub-lists, and showcasing information of objects of the tables. Yet there is one more less physical result of this project. That is the obtained understanding and skill of working with QT and C++ on a certain level and using some knowledge of the course to build something general and relatively useful.

Conclusion

In the end, application is not perfect and may have some potential for upgrades such as better design, which could maybe imply menu with options, in which there could be such features as changing fonts and size of text. There also could be a feature which would allow user to save file in different formats. Another great feature could be adding ability to delete the songs directly from the playlist window but at the moment of facing this problem it still seems like there is no way to do it efficiently memory wise. Anyway, the project was quite educational and in some way fun to do.