Health Services in Metropolitan Areas – GUI-based Data Management Application

I. Title Page

Title: "Health Services in Metropolitan Areas"
Author: *Student1613*
Group: DSBA
Submission Date: 10.06.2023

II. Problem Statement

The efficient management of health service data across metropolitan areas presents a considerable challenge, primarily due to the sheer volume and complexity of this data. Health service data contains vital information that can inform decision-making for healthcare providers, policymakers, patients, and other stakeholders. However, due to its often unorganized and overwhelming nature, leveraging this data effectively can be a complex task.

Current methods for managing such data may not provide the flexibility and functionality required for effective utilization. Specifically, there is a notable gap in user-friendly software that allows for interactive manipulation of this data - functionalities such as editing, removing, and adding data rows, filtering and sorting data based on various criteria, and the like.

My app aims to address these challenges by developing a Graphical User Interface (GUI) application designed to manage health service data, particularly focusing on data from the U.S. Census Bureau about health services and their ratings in 83 U.S. metropolitan areas. This application, developed using the QT framework, will allow users to seamlessly interact with the dataset, providing functionalities such as adding, removing, and editing data rows, filtering and sorting the data, and adjusting to screen size in real time.

Therefore, the core problem that this project aims to solve is the challenge of efficiently managing and utilizing health service data, by providing a user-friendly and feature-rich GUI application. This tool will aid in transforming raw, complex data into insightful, actionable information, thereby bridging the gap between data collection and informed decision-making in the realm of health services.

III. Implementation Details

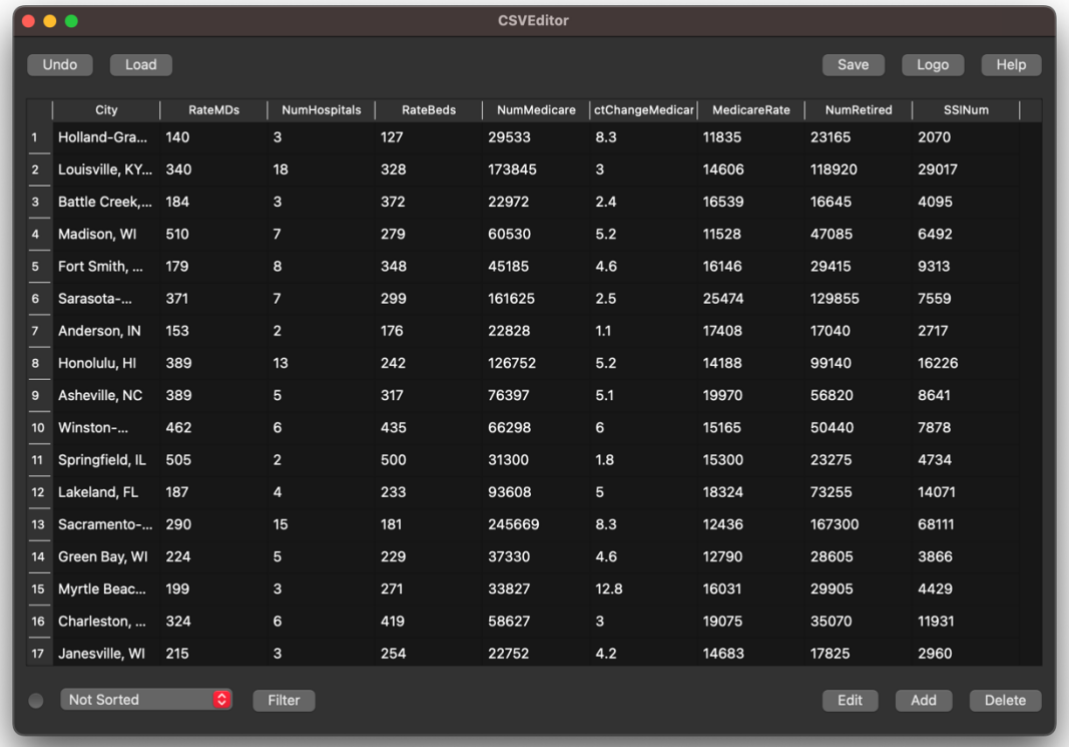
The project repository can be accessed by the link.

1. Main Window

Key features:

- Add, Edit or Remove row.
- Edit any cell by double-clicking on it.
- Reload the file.
- Save As functionality.
- Undo the last action.
- Sort by combo box.
- Sort by clicking on the header.
- Filter by parameters.
- Display the logo.
- View info of a particular row or the entire table.

Main difficulties were to implement the Undo and SortFilter functionality. The undo functionality required me to create the QUndoStack and four separate cpp classes for undoing each operation. The main difficulties in implementing the SortFilter functionality were in the correspondence between the proxy model indexes and source model indexes as it required me to modify all the existing indexing with respect to new QSortFilterProxyModel inherited class.



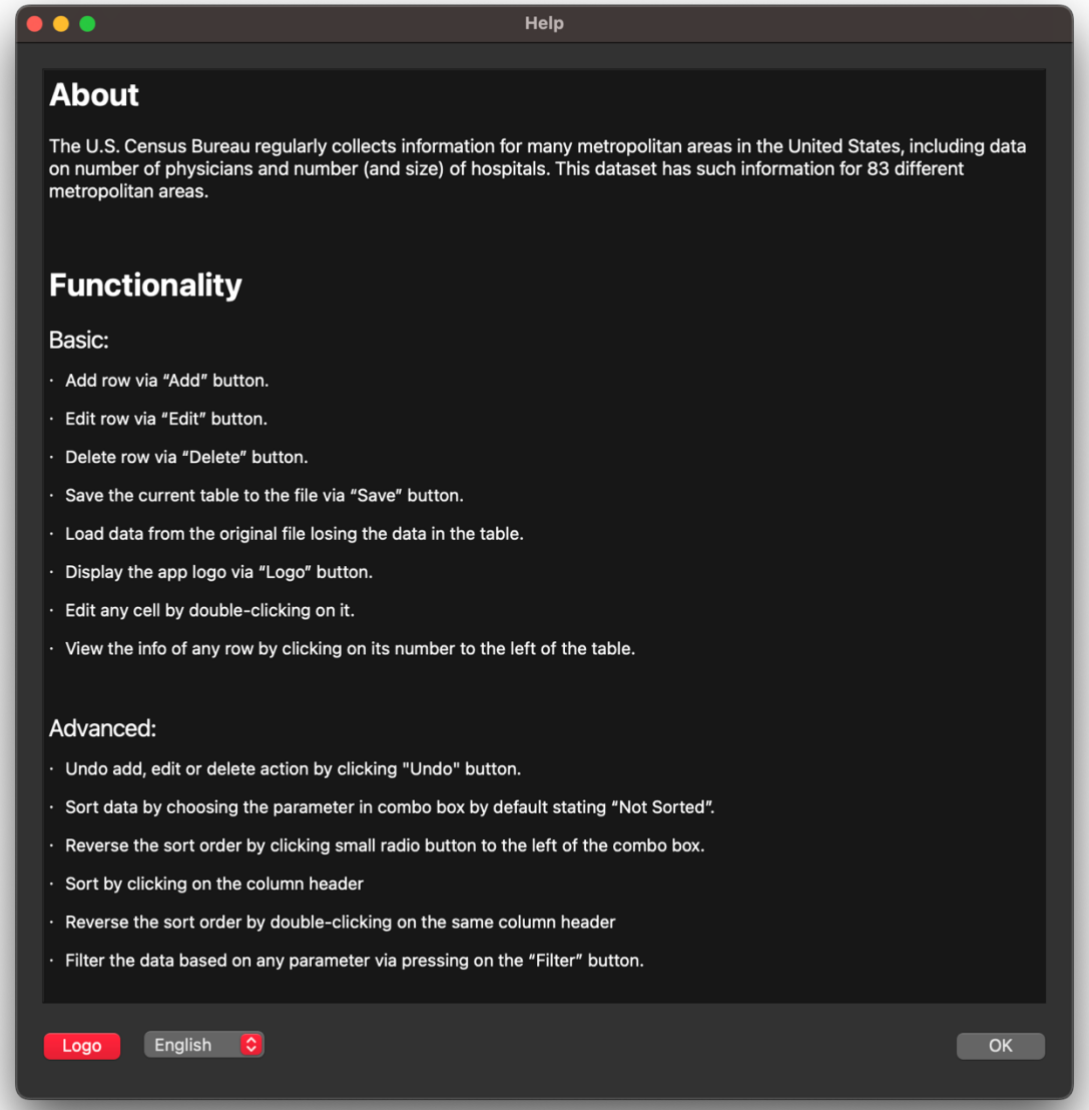
	City	RateMDs	NumHospitals	RateBeds	NumMedicare	ctChangeMedicar	MedicareRate	NumRetired	SSINum
1	Holland-Gra...	140	3	127	29533	8.3	11835	23165	2070
2	Louisville, KY...	340	18	328	173845	3	14606	118920	29017
3	Battle Creek,...	184	3	372	22972	2.4	16539	16645	4095
4	Madison, WI	510	7	279	60530	5.2	11528	47085	6492
5	Fort Smith, ...	179	8	348	45185	4.6	16146	29415	9313
6	Sarasota-...	371	7	299	161625	2.5	25474	129855	7559
7	Anderson, IN	153	2	176	22828	1.1	17408	17040	2717
8	Honolulu, HI	389	13	242	126752	5.2	14188	99140	16226
9	Asheville, NC	389	5	317	76397	5.1	19970	56820	8641
10	Winston-...	462	6	435	66298	6	15165	50440	7878
11	Springfield, IL	505	2	500	31300	1.8	15300	23275	4734
12	Lakeland, FL	187	4	233	93608	5	18324	73255	14071
13	Sacramento-...	290	15	181	245669	8.3	12436	167300	68111
14	Green Bay, WI	224	5	229	37330	4.6	12790	28605	3866
15	Myrtle Beac...	199	3	271	33827	12.8	16031	29905	4429
16	Charleston, ...	324	6	419	58627	3	19075	35070	11931
17	Janesville, WI	215	3	254	22752	4.2	14683	17825	2960

2. Help Window

Key features:

- Provide information about the app.
- Display the logo.
- Switch languages.

Main difficulties here were to implement the switch language functionality. It required me to create multiple functions to retranslate each element of Ui of each window of the whole app.

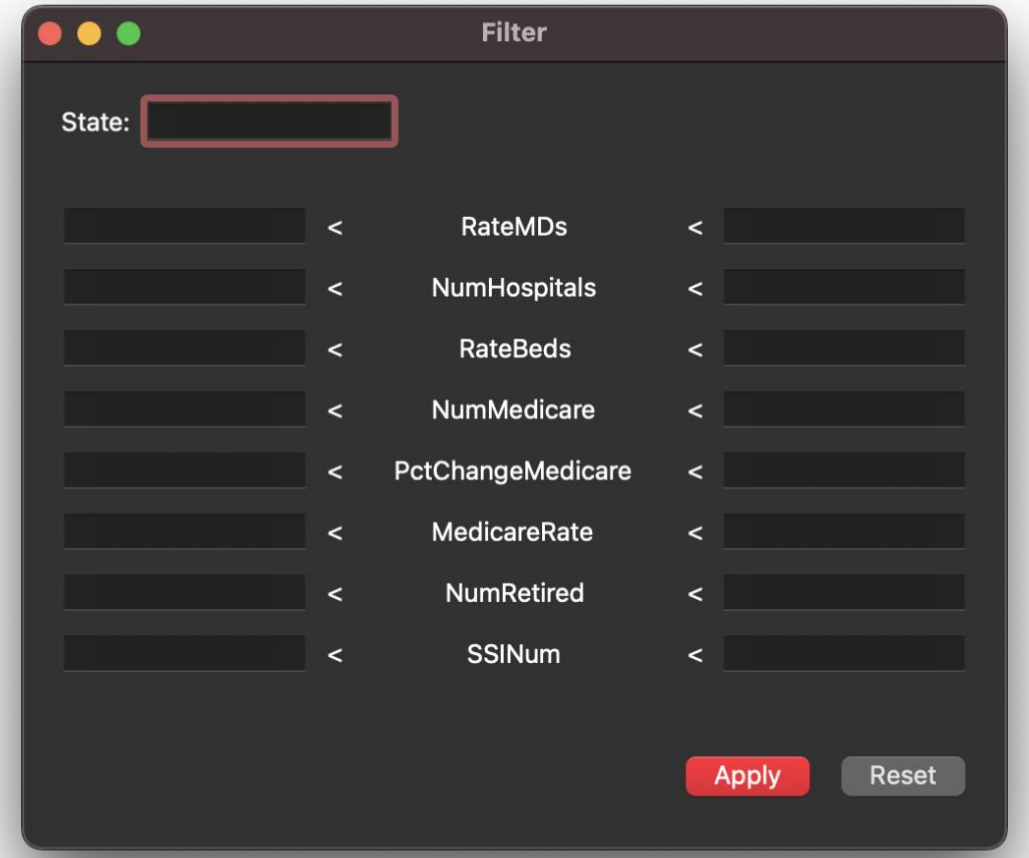


3. Filter Window

Key features:

- Display the cities from chosen state.
- Filter the data by any number of parameters.
- Reset the filters with one button.
- Doesn't break the initial order of data.

Main difficulties here were to adjust existing functionality to new QSortFilterProxyModel inherited class and to properly handle the filtering request. Adjusting the functionality involved working with source indexes, proxy indexes and resulted in using mapToSource function to solve the problem. To properly handle the data from the filter window I used elegant solution: duet of findChild and special naming system of each line edit. To propely handle any number of filters I by default set the filter parameter to ±inf.

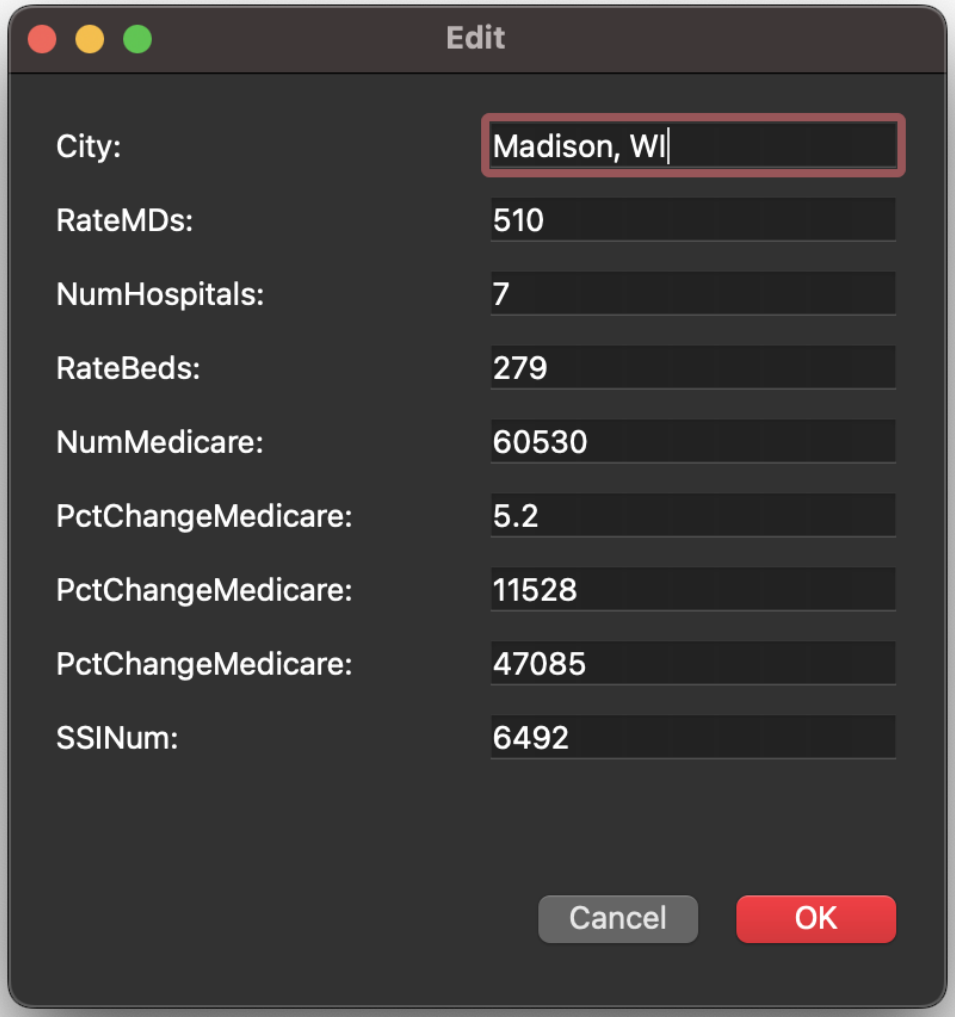


4. Edit Window

Key features:

- Edit the selected row.
- Save or discard changes to the table.
- Works with filtered and sorted table.
- Supports the Undo action.
- Doesn't automatically save changes.
- Filters entered data for appropriate format.

The implementation of the Edit Window involved creating a separate class for undoing the editing. Main problem here was to modify the indexing and track the undo stack.

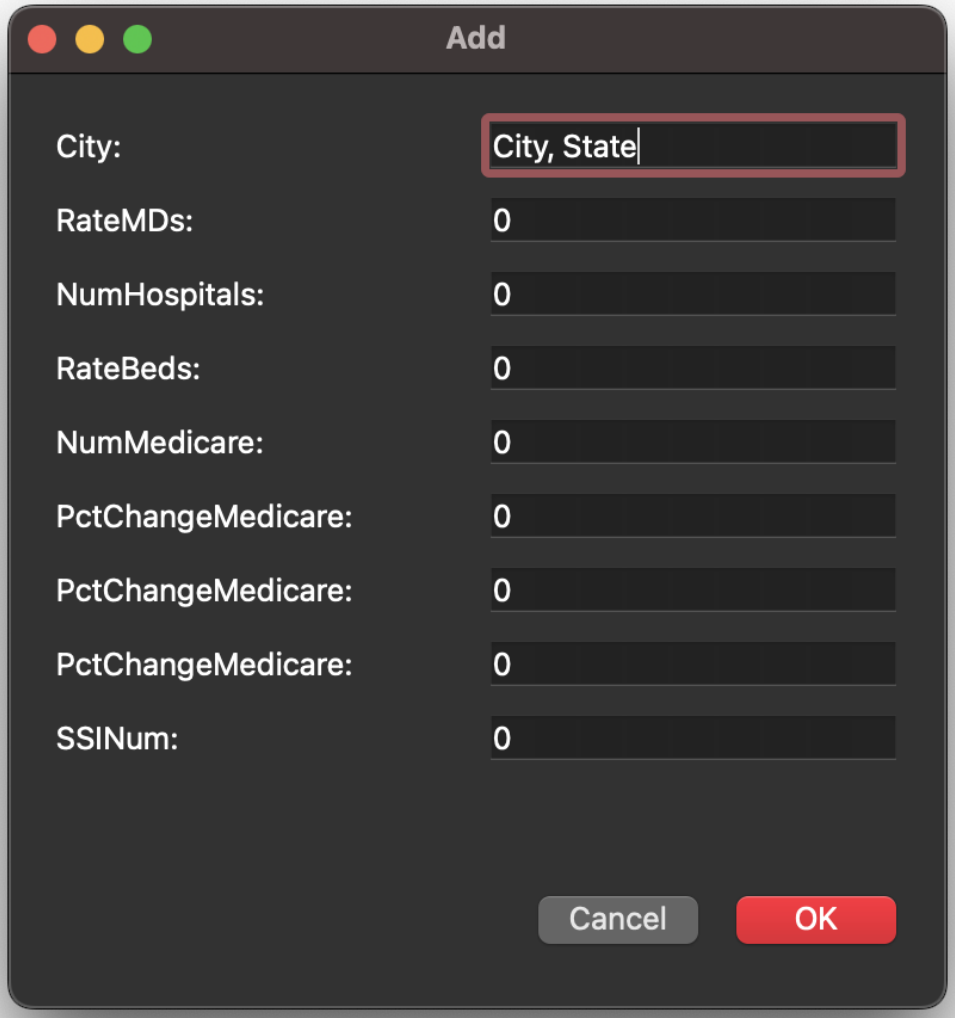


5. Add Window

Key features:

- Add row below the selected row.
- Add row on confirmation.
- Works with filtered and sorted table.
- Supports the Undo action.
- Doesn't automatically save changes.
- Includes default values for each field.
- Filters entered data for appropriate format.

The implementation of the Add Window involved creating a separate class for undoing the adding. Main problem here was to modify the indexing and track the undo stack.



IV. Results and Discussion

Implemented software offers a comprehensive set of features designed to facilitate data manipulation, user guidance, and interactive data filtering, amongst others. While these key features performed well, certain functionalities posed significant implementation challenges. Overpassing these challenges with different approaches showed me that rewriting the same functionality many times is normal and sometimes can take enormous amount of time.

Despite these challenges, all mandatory features were implemented, all the features mentioned in the specification were also implemented, even some additional features, e.g., support of Russian language, were added. The best approach that I've found out during the development process is to constantly try to simplify existing code, make it easier to understand and try to act absolutely inadequate while testing the app GUI.

V. Conclusion

The accomplished project demonstrates a significant and successful application of Qt C++ in the creation of a user-friendly interface for data management. The diverse set of functionalities provided, such as the ability to add, edit, remove, undo actions, sort, and filter data, underscores the robustness of the designed software. Furthermore, the implementation of advanced features like switch language functionality and the maintenance of initial data order, irrespective of the applied filters, adds additional layers of complexity and usability to the application.

Despite facing challenges in developing the Undo and SortFilter functionality and adapting the software to the QSortFilterProxyModel inherited class, effective solutions were developed. These involved the creation of multiple classes and functions, demonstrating the adaptability of the Qt C++ framework and the potential for overcoming difficulties inherent in the design process.

Moreover, the developed software is capable of handling both sorted and filtered data, and the incorporation of an undo stack in the Edit and Add windows underscores the versatility of the design. The project outcomes affirm that through consistent refinement and testing, software can be made not only functional but also intuitive and user-friendly.

The added language support for Russian demonstrates the software's potential for further enhancement and customizability to cater to diverse user bases. Overall, the project solidified the importance of continuously simplifying and testing the codebase, thereby ensuring the maintainability and robustness of the application.

In conclusion, the project serves as an excellent testament to the power and flexibility of Qt C++ in creating comprehensive, user-oriented software applications, providing valuable insights and experiences that can be applied in future software development endeavors.