Restaurant Management Program

Andres Eduardo Prato Bello

Department of Computer Science and Networking

Wentworth Institute of Technology

Boston, MA 02115, USA

pratoa@wit.edu

***Abstract*— The market in restaurant management programs is quite big, but there is always ways to make something better. The student accomplished most of his goals, specifically complete a working prototype of a restaurant management program. He learned by himself Qt and object oriented programming to accomplish this task. Lastly, this topic of the project was chosen because of many ideas the student has regarding this matter, which some will be discussed later in this report.**

I. Introduction

There are several restaurant management systems in the market nowadays, and it is a competitive market. These programs are made to keep track of every account there is in a restaurant, as well as to keep track of money and profits. Some of the systems are very good, but are missing intuitiveness and usability, even to the point to lack or have a weak security [1]. There is a lot of things that can be introduced into this field that will be discussed in this paper.

This paper reports the design of a restaurant management software created in an undergraduate programming course. The goal of the project was to accomplish creating the program and also to make possible the implementation of a GUI. The graphical user interface was built with Qt Creator, an application framework used to develop application software using C++. This program targets new restaurants that want to control their table checks easily and fast. At last, this program was made so that each table and seat in the restaurant can be monitored, and keep track of the check of each individual seat and table.

II. Project Plan

The project was started by seeing what it had to be done first. The first thing was to make a sketch of how the main window would look like. Here it can be observed how the main window looks like at the end of the project.

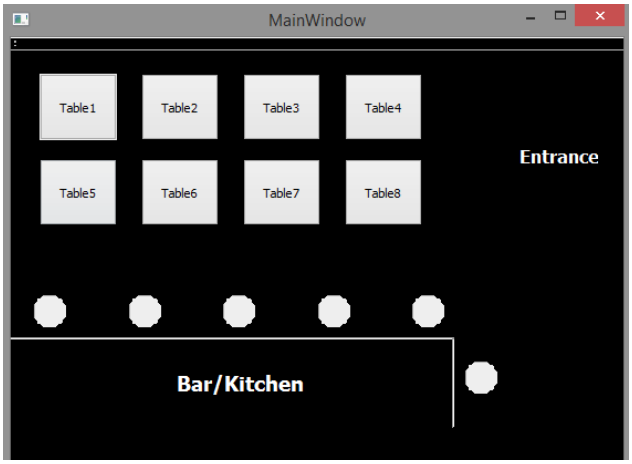

Figure 2. Main Window

As seen in Figure 1, each square is a table and each circle is a seat, they are all buttons that can be clicked. When a button is clicked for example Table 1, a new window will appear with the name of the table or seat that was clicked.

In order to make this happen every table and seat has to be connected with a function. Here is the sample code to make this connection. A pointer is created to

```
QPushButton *tb1 = ui->Table1;
QObject::connect(tb1,SIGNAL(clicked()),
                 this, SLOT(tableclick()));
```

Figure 3 Sample code connection

keep track of which table or seat was clicked, in this case is Table1. The connection is made by defining a signal and a slot. The signal is when the button is clicked, and the slot is the function that will be called when the signal happens. This function named "tableclick" will create a new window with six buttons Add Food, Subtract Food, Print Check, Clear Table, Return, and Refresh. These are the most important buttons in the program, because it is where the user will make every order and can see how the check is going.

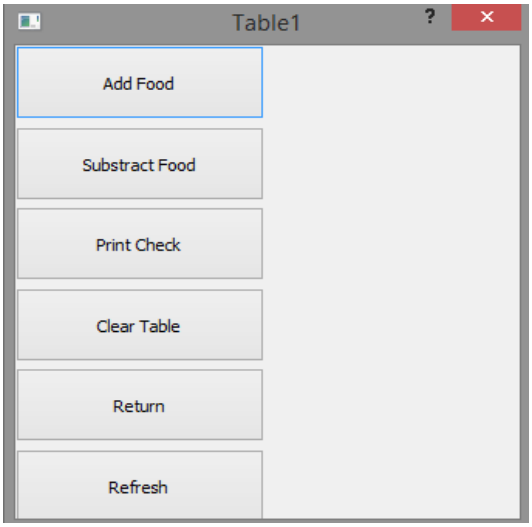

Figure 1 Table1 window

The second and last important thing that was thought before doing anything of the programming, was how to actually keep track of the tables and seats. This was done by creating two headers and two .cpp files. Headers tables and menu, in header tables there is a struct named table and each table will have an array of QString called name_products and a variable of type double named total_price. In the header menu, there is another struct called menu which has a QString called name and a variable type double for the price of the product. Then in the menu.cpp an array is defined name products with the 10 products of the restaurant.

III. Buttons

III.1 Add Food Button

When the add food button is clicked a new window pops up. This window is called menu, because it has listed all the products that the restaurant sell. In this window each button has a

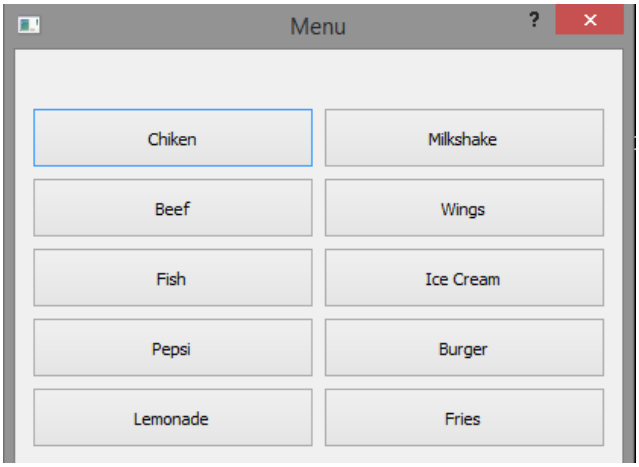

Figure 4 Menu

different product. When one of the buttons is clicked a function add is called which will first see in what table the user is currently at, and then add a QString with the name of the product into the array of QStrings of the table.

III.2 Subtract Food Button

This button work exactly the same way as the Add Food button. When it is clicked a new window is opened called menu, exactly the same as Figure 4. Instead of adding the product to the array of the table it will subtract the item from the array. When one of the products is clicked a function called subs will find the index of where that product is in the array, and then move everything that it is front of that index and one space less. So, for example, if the array has chicken, beef, and fish, and you want to subtract beef. What the function will do is find out that the index of beef is 1, and move fish from index 2 to index 1.

III.3 Print Check

When the button Print Check is clicked a QMessageBox is shown. This QMessageBox will have as title the table or seat that was clicked plus the word check. So, if table 1 was clicked then it will be “Table1 Check”. Also it will display the total amount of money that the table owes. This is done by printing the variable total_price that the table or seat has.

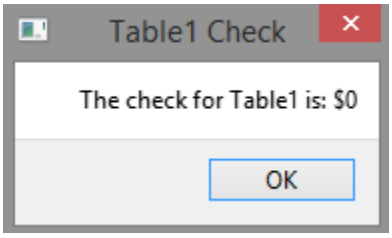

Figure 5. Print Check

Originally it was planned so that the QMessageBox would display all items with their respective price, and lastly the total amount. Unfortunately this wasn’t achieved because of time matters with the due date of the project.

III.4 Clear Table Button

This button was made to clear every item in a table or seat. When it is

```
void TableClick::on_ClearTable_clicked()
{
    //TABLE1
    if (this->objectName() == "Table1"){
        extern table Table1;
        for (int i = 0; i<14;i++){
            Table1.name_product[i] = " ";
            Table1.total_price = 0;
            extern int table1_counter;
            table1_counter = 0;
        }
    }
```

Figure 6. Clear Table Code

clicked it will first see in what table or seat the user is, and then set every item in the array name_products of the table or seat to be a space (“ “). This is done by creating a for loop and setting every item to be a space, and at the same time set the total_price to be 0. In Figure 6 it is shown the code implemented for the clear button. It first checks if the table or seat clicked is Table1, and then initiates a for loop and sets every value in Table1.name_product[] to a space and finally Table1.total_price to 0.

III.5 Return Button

This is the simplest button in the program. The only thing that does is to return from the window of the table or seat to the main window. This is done simply by closing the window. This is the code that performs this action. It basically says that when the ReturnButton is clicked the window TableClick, which is the window of the table or seat that was clicked, will be closed.

```
void TableClick::on_ReturnButton_clicked(){
    close();
}
```

Figure 8. Return code

III.6 Refresh Button

What this button does was originally supposed to be atomically done, but that unfortunately was not achieved. When the Refresh button is clicked, it checks what table the user has clicked. After that it creates a for loop with the number of maximum products a table or seat can have which is 14, this number was chosen randomly (it can be changed). Then it creates another for inside the previous one with the number of products in the menu which is 10, and finally it grabs the name of the product stored in Table1.name_product[i] and it looks the price for that product in the menu and it adds the value of the product to the total price of the table or seat.

```
if (this->objectName() == "Table1"){
    for (int i=0;i<14;i++){
        for (int j=0;j<10;j++){
            if (Table1.name_product[i] == products[j].name){
                Table1.total_price = Table1.total_price + products[j].price;
            }
        }
    }
    ui->label0->setText(Table1.name_product[0]);
    ui->label1->setText(Table1.name_product[1]);
    ui->label2->setText(Table1.name_product[2]);
    ui->label3->setText(Table1.name_product[3]);
    ui->label4->setText(Table1.name_product[4]);
    ui->label5->setText(Table1.name_product[5]);
    ui->label6->setText(Table1.name_product[6]);
    ui->label7->setText(Table1.name_product[7]);
    ui->label8->setText(Table1.name_product[8]);
    ui->label9->setText(Table1.name_product[9]);
    ui->label10->setText(Table1.name_product[10]);
    ui->label11->setText(Table1.name_product[11]);
    ui->label12->setText(Table1.name_product[12]);
    ui->label13->setText(Table1.name_product[13]);
```

Figure 7. Refresh code

Finally it will set each label in the window to display the name each product in the table, and at last the total price of the table.

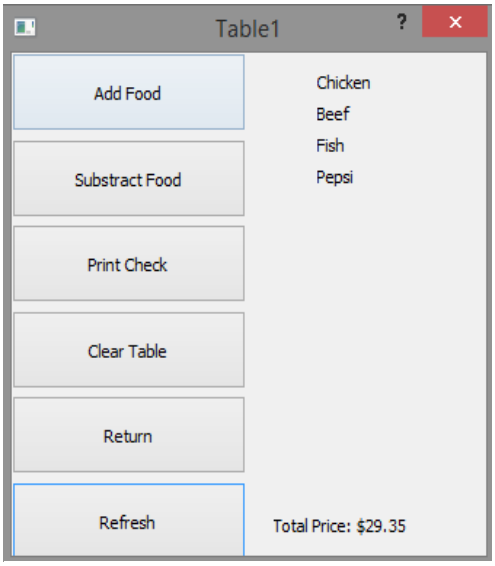

Figure 10. After Refresh is Clicked

IV. Discussion and Lessons Learned

While programming this project, it was learned that before starting the actual programming, first there should be a clear and explicit way of how the project it is going to be. Before starting to code, there was little thought of how it should be done, and a lot of things had to be improvised while programming which took more time than what it was expected.

Also the student learned how to work with Qt by his own doing an online course in Lynda.com and actually programming with Qt itself by himself. This was the biggest challenge that the student had when doing this project plus time constraints.

V. Future Plan

The major expectations of the project were met when it was finalized, which were to make a working prototype, and to be able to keep track of each table and seat. For the future, this project can be fully developed into an application.

A goal that can be achieved is to create an app in smartphones so that the waiters in a restaurant won't lose time going to the machine to put all of the orders, instead it could be done right in front of the customer in their smartphone. The major problem with this idea is the battery of the smartphone of the waiters, which will last a shorter time and will need to be recharged. At the time the restaurant wouldn't have to buy those expensive machines and expensive software's, instead they could simply download the app in their waiters and managers smartphones.

Lastly, the most important thing of this project is that it was made to learn new things and that was accomplished. The student wanted to push himself to accomplish something that he wanted to learn, which was Qt and some object-oriented programming.

References

[1] **Wesly, Williams, and Devon Simmonds. "A Case Study in the Design of a Restaurant Management System." (n.d.): 1-7. Web. <http://people.uncw.edu/simmondsd/research/PUBLICATIONS/rms.pdf>**