**-----------------------------------------------------------------------------------**



**TYPING TUTOR**

**A PROJECT REPORT**



SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY

IN

**INFORMATION TECHNOLOGY**

Submitted by-

**IT- 1**

**G-3**

**Ishita Lather Ikroop Singh Kalsi**

**2K19/IT/063 2K19/IT/061**

Under the supervision of

[Geetanjali](http://www.dtu.ac.in/Web/Departments/CSE/faculty/rammurtirawat.php) Bhola

**Object Oriented Programming IT-203**

DELHI TECHNOLOGICAL UNIVERSITY

DEPARTMENT OF INFORMATION TECHNOLOGY

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

**CANDIDATE’S DECLARATION**

We, (Ishita Lather and Ikroop Singh Kalsi), Roll No – 2K19/IT/063 and 2K19/IT/061, students of B.Tech. (INFORMATION TECHNOLOGY), hereby declare that the project Dissertation titled “Typing Tutor” which is submitted by us to the Department of INFORMATION TECHNOLOGY, Delhi Technological University, Delhi in partial 2ulfilment of the requirement for the award of the degree of Bachelor of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Place :Delhi

Date: 23-11-2020

Ishita Lather

Ikroop Singh Kalsi

DEPARTMENT OF INFORMATION TECHNOLOGY

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

**CERTIFICATE**

I hereby certify that the Project Dissertation titled “Desktop application for structuring of online learning through Youtube” which is submitted by Ishita Lather and Ikroop Singh Kalsi; Roll No – 2K19/IT/063 and 2K19/IT/061; INFORMATION TECHNOLOGY, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology, is a record of the project work carried out by the students under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: Delhi

Date: 29-07-2020

**Geetanjali Bhola**

**SUPERVISOR**

**ACKNOWLEDGEMENT**

We would like to convey our heartfelt thanks to our supervisor Mrs. Geetanjali Bhola for her ingenious ideas, tremendous help and cooperation.

We are extremely grateful to our friends who gave valuable suggestions and guidance for completion of our project. The cooperation and healthy criticism came handy and useful with them.

Finally we would like to thank all the above mentioned people once again.

**ABSTRACT**

Typing is the basic command to run a computer and your typing speed plays vital role while working in computer to save time. Here we have presented a Typing Tutor Project in Qt in C++ to measure user’s typing speed and to enhance their typing speed.

Typing Tutor is perfect for individuals who lack any sort of fluency or speed when it comes to typing, but it can also be used by people who are very adept at typing but nonetheless want to push themselves to the restriction and see simply how many phrases they can type in a minute.

Some of the various features of the program are:

* A great program that will help you in improving your perusing rate.
* Well ordered way to deal with expert keyboarding
* Ease of use.
* Improve typing pace.

The aim of this typing tutor is to:

* **Get rid of typos**

Get rid of those annoying typing errors and increase the quality of your texts.

Save valuable work time

This typing tutor will help save dozens of working hours each year as you type faster and do not need to correct errors constantly.

* **Focus on your text and ideas**

With fluent keyboarding skills you can put your brain power into expressing your thoughts instead of finding the right keys.

* **Improve your ergonomics**

Natural, efficient finger movements and no need to look at the keyboard results in less stress in fingers, arms and neck.

* **Be confident with computers**

Improved typing skills will help take full advantage of your computer as you become more confident with the keyboard.

## Introduction to Qt

Qt (pronounced "cute") is a [free and open-source](https://en.wikipedia.org/wiki/Free_and_open-source) [widget toolkit](https://en.wikipedia.org/wiki/Widget_toolkit) for creating [graphical user interfaces](https://en.wikipedia.org/wiki/Graphical_user_interfaces) as well as [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) [applications](https://en.wikipedia.org/wiki/Application_software) that run on various software and hardware platforms such as [Linux](https://en.wikipedia.org/wiki/Linux), [Windows](https://en.wikipedia.org/wiki/Windows), [macOS](https://en.wikipedia.org/wiki/MacOS), [Android](https://en.wikipedia.org/wiki/Android_(operating_system)) or [embedded systems](https://en.wikipedia.org/wiki/Embedded_system) with little or no change in the underlying codebase while still being a native application with native capabilities and speed.

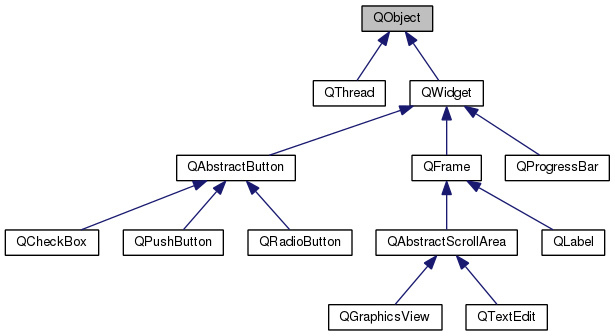
Most GUI programs created with Qt have a native-looking interface, in which case Qt is classified as a [widget toolkit](https://en.wikipedia.org/wiki/Widget_toolkit). Also non-GUI programs can be developed, such as [command-line](https://en.wikipedia.org/wiki/Command-line_interface) tools and consoles for servers. An example of such a non-GUI program using Qt is the Cutelyst [web framework](https://en.wikipedia.org/wiki/Web_framework).

Qt supports various compilers, including the [GCC](https://en.wikipedia.org/wiki/GNU_Compiler_Collection) C++ compiler, the [Visual Studio](https://en.wikipedia.org/wiki/Microsoft_Visual_Studio) suite, [PHP](https://en.wikipedia.org/wiki/PHP) via an extension for PHP5, and has extensive [internationalization](https://en.wikipedia.org/wiki/Internationalization_and_localization) support. Qt also provides [Qt Quick](https://en.wikipedia.org/wiki/Qt_Quick), that includes a [declarative](https://en.wikipedia.org/wiki/Declarative_programming) [scripting language](https://en.wikipedia.org/wiki/Scripting_language) called [QML](https://en.wikipedia.org/wiki/QML) that allows using [JavaScript](https://en.wikipedia.org/wiki/JavaScript) to provide the logic. With Qt Quick, rapid application development for mobile devices became possible, while logic can still be written with native code as well to achieve the best possible performance.

Other features include [SQL](https://en.wikipedia.org/wiki/SQL) database access, [XML](https://en.wikipedia.org/wiki/XML) parsing, [JSON](https://en.wikipedia.org/wiki/JSON) parsing, [thread](https://en.wikipedia.org/wiki/Thread_(computing)) management and network support.

Qt class hierarchy

Qt widely uses inheritance, especially in the Widgets module. The following graph shows some of these inheritances:



[QObject](http://doc.qt.io/qt-5/qobject.html) is the most basic class in Qt. Most of classes in Qt inherit from this class. QObject provides some very powerful capabilities like:

* object name : you can set a name, as a string, to an object and search for objects by names.
* parenting system (described in the following section)
* signals and slots (described in the next chapter)
* event management

Widgets are able to respond to events and use parenting system and signals and slots mechanism. All widgets inherit from QObject. The most basic widget is the [QWidget](http://doc.qt.io/qt-5/qwidget.html). QWidget contains most properties that are used to describe a window, or a widget, like position and size, mouse cursor, tooltips, etc.

## Widgets

Radio button is a standard GUI component. It is often used to make a unique choice from a list. In Qt, the [QRadioButton](http://doc.qt.io/qt-5/qradiobutton.html) is used to create radio buttons.

Thanks to a nice heritance, a QRadioButton behaves just like a QPushButton. All properties of the QPushButton are also the same in the QRadioButton, and everything that was learned in the second chapter can be reused here.

* text
* icon
* tooltip
* ...

By default, QRadioButtons are not grouped, so many of them can be checked at the same time. In order to have the "exclusive" behaviour of many radio buttons, we need to use [QButtonGroup](http://doc.qt.io/qt-5/qbuttongroup.html). This class can be used like this: We allocate a new button group and attach it to the parent object. Note that the parent object might be the mainwindow, or "this":

QButtonGroup \*buttonGroup = new QButtonGroup(object);

// Add buttons in the button group buttonGroup->addButton(button1); buttonGroup->addButton(button2); buttonGroup->addButton(button3); ...

What we want is to create a menu picker. In a window, a list of yummy plates should be displayed with radio buttons, and a push button that is used to select the chosen plate should be displayed.

Obviously, nothing will happen (now) when the buttons are clicked.

## Signals and slots

Instead of having observable objects and observers, and registering them, Qt provides two high level concepts: signals and slots.

* A signal is a message that an object can send, most of the time to inform of a status change.
* A slot is a function that is used to accept and respond to a signal.

Here are some examples of signals and slots from our well known [QPushButton](http://doc.qt.io/qt-5/qpushbutton.html) class.

* clicked
* pressed
* released

As you can see, their names are quite explicit. These signals are sent when the user clicked (pressed then released), pressed or released the button.

Here are some slots, from different classes

* QApplication::quit
* QWidget::setEnabled
* QPushButton::setText

In order to respond to a signal, a slot must be *connected* to a signal. Qt provides the method QObject::connect. It is used this way, with the two macros SIGNAL and SLOT

FooObjectA \*fooA = new FooObjectA(); FooObjectB \*fooB = new FooObjectB();

QObject::connect(fooA, SIGNAL (bared()), fooB, SLOT (baz()));

assuming that FooObjectA have a bared signal, and FooObjectB have a baz slot.

You have to write the signature of the signal and the slot inside the two macros *SIGNAL* and *SLOT*. If you want to get some information about what these macros do, please read the last section of this chapter.

Remark : Basically, signals and slots are methods, that might or might not have arguments, but that never return anything. While the notion of a signal as a method is unusual, a slot is actually a real method, and can be called as usual in other methods, or whilst responding to a signal.

## The Observer Pattern

Nearly all UI toolkits have a mechanism to detect a user action, and respond to this action. Some of them use *callbacks*, others use *listeners*, but basically, all of them are inspired by the [observer pattern](http://en.wikipedia.org/wiki/Observer_pattern).

Observer pattern is used when an *observable* object wants to notify other *observers* objects about a state change. Here are some concrete examples:

* A user has clicked on a button, and a menu should be displayed.
* A web page just finished loading, and a process should extract some information from this loaded page.
* An user is scrolling through a list of items (in an app store for example), and reached the end, so other items should be loaded.

Observer pattern is used everywhere in GUI applications, and often leads to some [boilerplate code](http://en.wikipedia.org/wiki/Boilerplate_code). Qt was created with the idea of removing this boilerplate code and providing a nice and clean syntax, and the signal and slots mechanism is the answer.

## Classes Used

1. **mainWindow()** - The Homescreen which contains 5 options:
   1. Play - Here user can play in three modes easy, medium, hard with the die time decrease in progressively in each mode.
   2. Highscore - The highest score is maintained for each of the play modes
   3. Help - Basic description of all the options present on main menu
   4. Practice - Typing speed gets returned at the end of timer that is running
   5. Exit - exits the program
2. **play()**
   1. on\_pushButton\_easy\_clicked() - triggers easy mode
   2. on\_pushButton\_normal\_clicked() - triggers normal mode
   3. on\_pushButton\_hard\_clicked() - triggers hard mode
3. **easy(), normal(), hard()**
   1. keyPressEvent() - handles the pause menu and running of game.
   2. timer() - resets the dietimer
   3. on\_lineEdit\_returnPressed() - process the line entered by user from the screen
   4. stopwatch() - handle the ui part part of the dietimer and displaying text
   5. highscore() - write the highscore in respective mode’s highscore files.
4. **practice()**
   1. stopwatch() - handles the timer
   2. keyPressEvent() - handles the pause menu and starting of game.
   3. checkChange() - gets triggered everytime user types something.
   4. update\_screen\_text() - practice text body updation on the screen happens here.
   5. changeColor() – this uses html tags to change color of desired string
   6. reset() – generates a new text body and resets all the properties of the class
   7. calculate\_result() – the speed and accuracy are calculated here.
5. **help()**

Displays all the information about all the modes

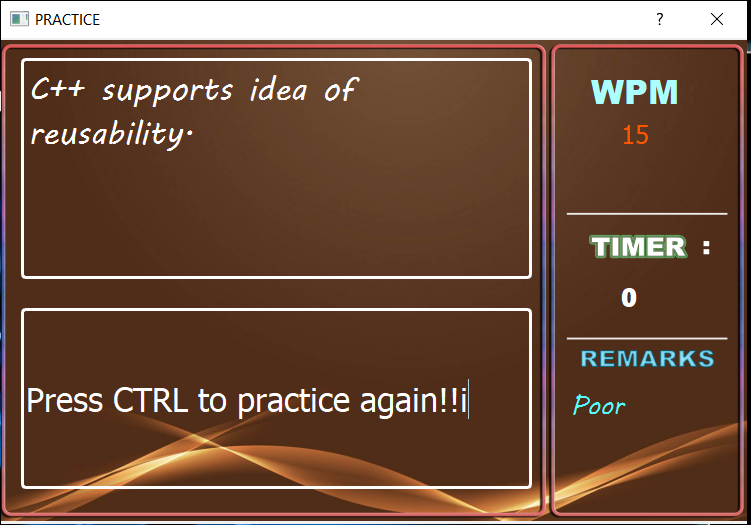
## Result and Analysis

The program works perfectly fine with the implementation of Qt environment, it calculates the speed in WPM accurately and saves the high scores as well. All the functions present execute appropriately data text files and background images are showing as well.

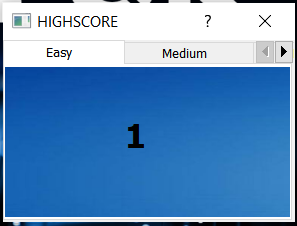
ANALYSIS(OUTPUT):

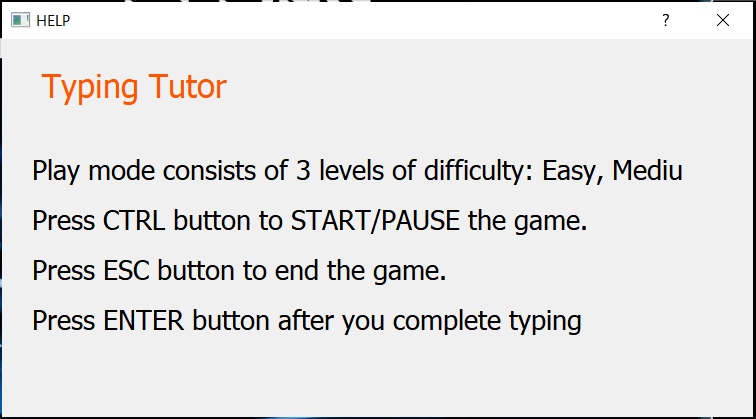












## Conclusion

Our project Typing Tutor, in current state can achieve most of our desired objectives with a major hope of future developments as well.

It is capable of: -

* Getting rid of those annoying typing errors and increase the quality of your texts.
* Saving valuable work time.
* saving dozens of working hours each year as you type faster and do not need to correct errors constantly.
* Improving typing skills will help take full advantage of your computer as you become more confident with the keyboard.

## Future Scope

There is a lot of scope for future for our project:-

* We could add a more organized leaderboard which would save data of more than one person for all the difficulty levels.
* We could add an onscreen keyboard for feedback, that which key has been pressed.
* We could add a word-finger assist to teach the user, the most efficient way to type using both hands.
* We could integrate a game with our project as well to make it more interactive.