A REPORT ON

**A SOCKET CLIENT FRAMEWORK FOR TESTING MONSTAR DAQ SERVER**

Submitted to: **Mr. AYAN BANERJEE**

**Computer Division**

**Vocational Training** at

**Variable Energy Cyclotron Centre, Kolkata**



From**: 18th June** To: **18th July**

Prepared By:

**Shalmoli Neogi-MCKVIE**

# **ACKNOWLEDGEMENT**

We would like to extend our heartfelt gratitude to my project guide, ***Mr. Ayan Banerjee***. Without his valuable support and guidance this project would not have got proper shape.

We are also grateful to our training coordinator ***Dr. P Y Nabhiraj*** for his guidance and support. Without him the training program would not have been possible.

# **CONTENTS**

## 1. Introduction

## 2. Objective

## 3. Program Flowchart

## 4. Output

## 5. Discussion

## 6. Limitations

## 7. Bibliography

## 

# **INTRODUCTION**

The project **A socket client framework for testing MONSTER DAQ server** has been made in **C++** language.

This project consists of the following parts:

* Server Port
* Server IP
* Connect
* Set buffer size
* Set data limit
* Send

Each part performs some defined function

# **OBJECTIVE**

Main objective of this project is:

To build a client for testing the multiclient server MONSTAR DAQ which will have the following properties:

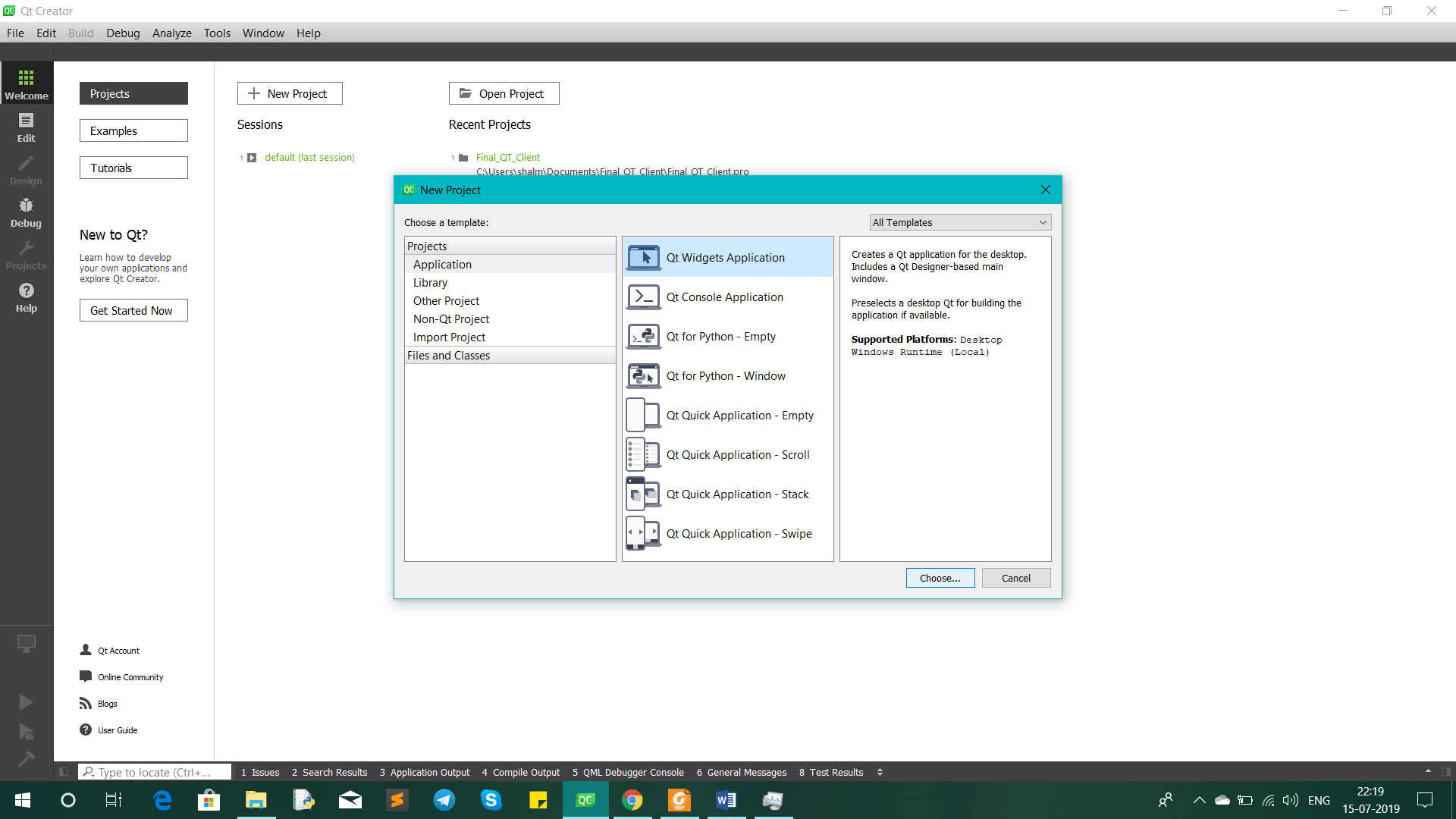
* It should write data as per size of user’s choice
* It should send data continuously until the limit specified by the user is reached
* Thread management needs to be implemented, i.e. one thread writes the data in the buffer, whereas the other one sends the buffer to the server.
* Writing and sending concurrency needs to be achieved.
* Locks needs to be implemented in order to prevent inconsistent results.

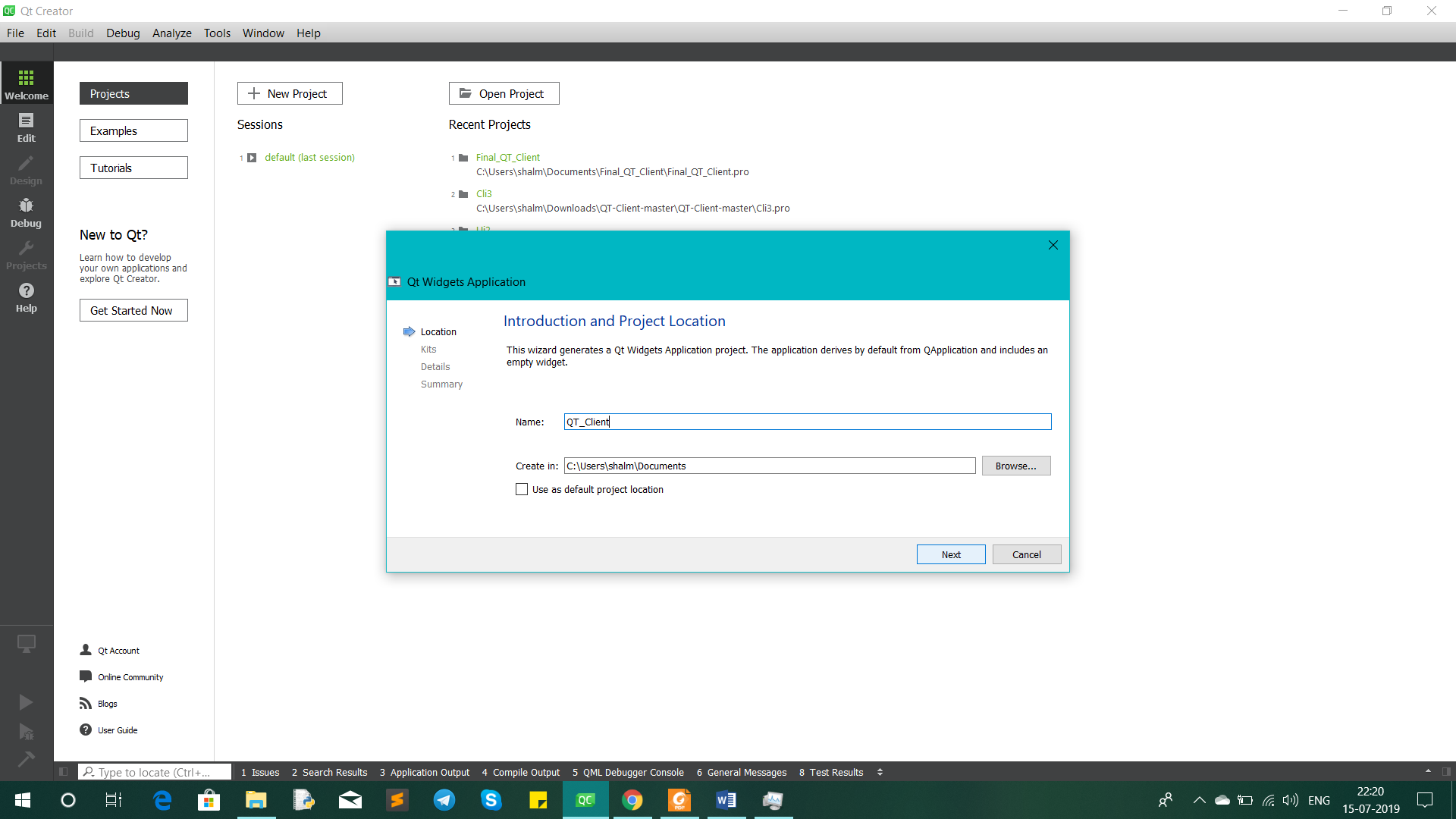
# **HARDWARE AND SOFTWARE USED**

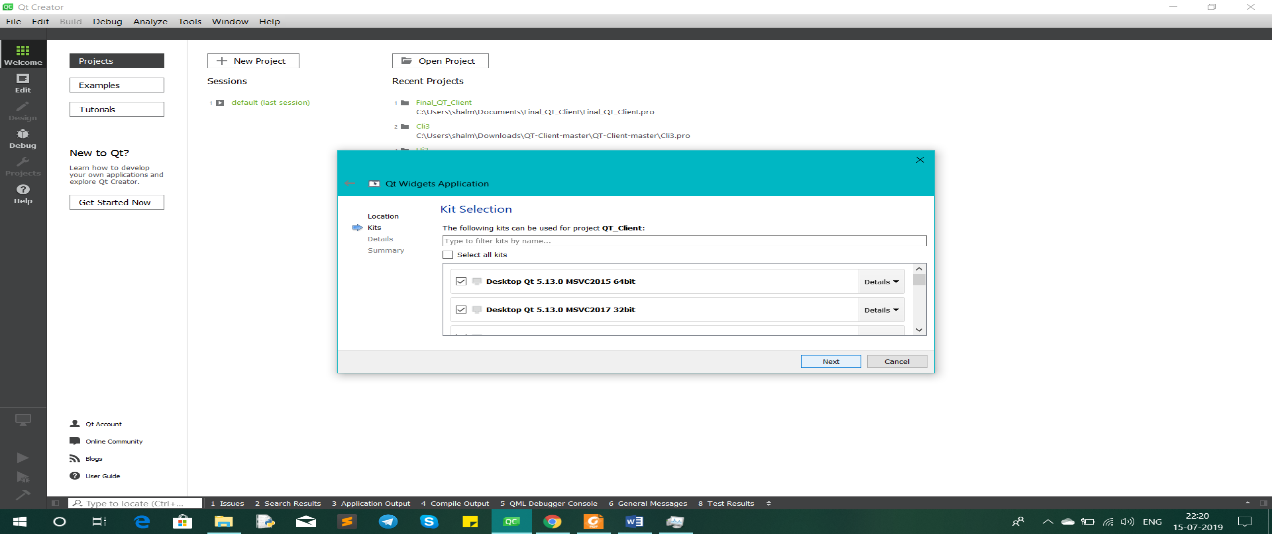
* Hardware used:
  + Memory: 3.7 GiB
  + Processor: Intel Core 2 Duo @3.00GHz\*2
  + Graphics: Intel Q35
  + OS Type: 64 bit
  + Disk: 241.9 GB
* Software required:
* Qt Creator
* g++ compiler

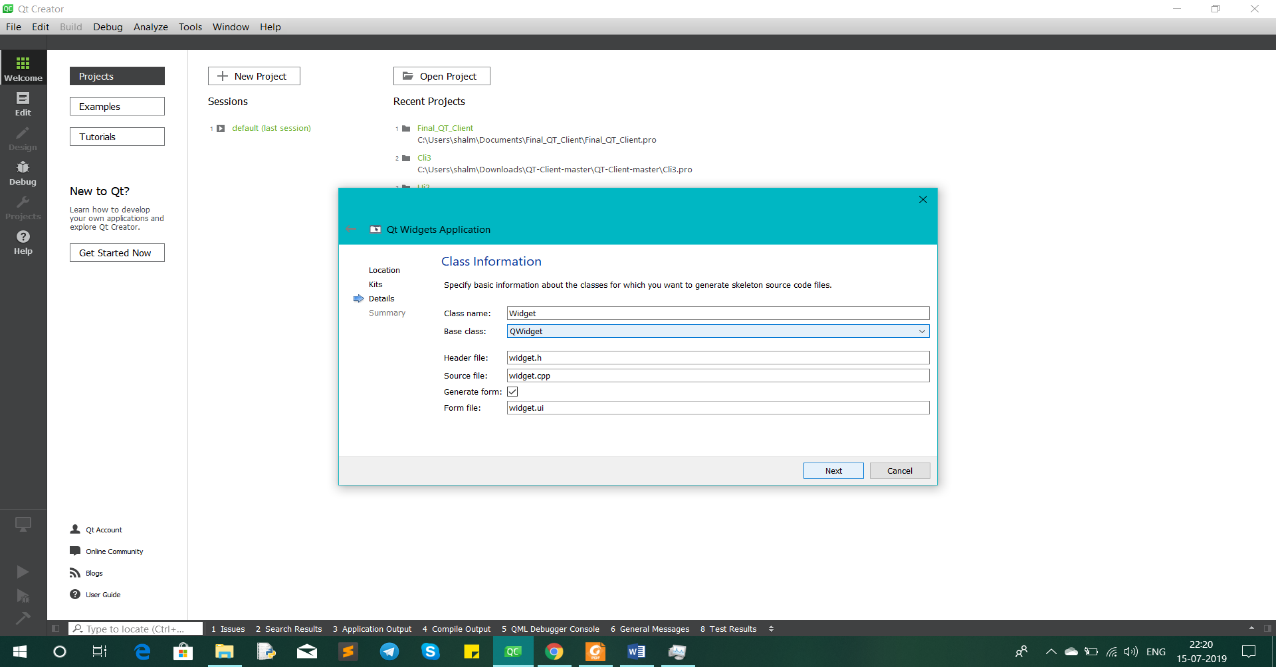
# **PROGRAM FLOWCHART**

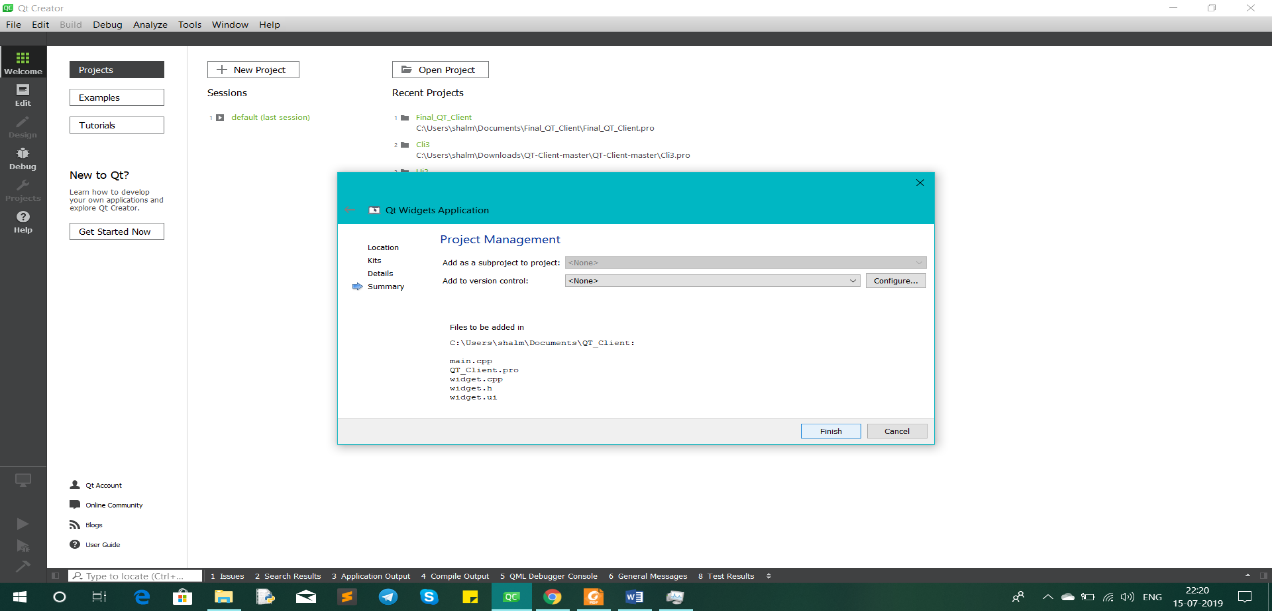
* **Step 1: Creating a project in QT**



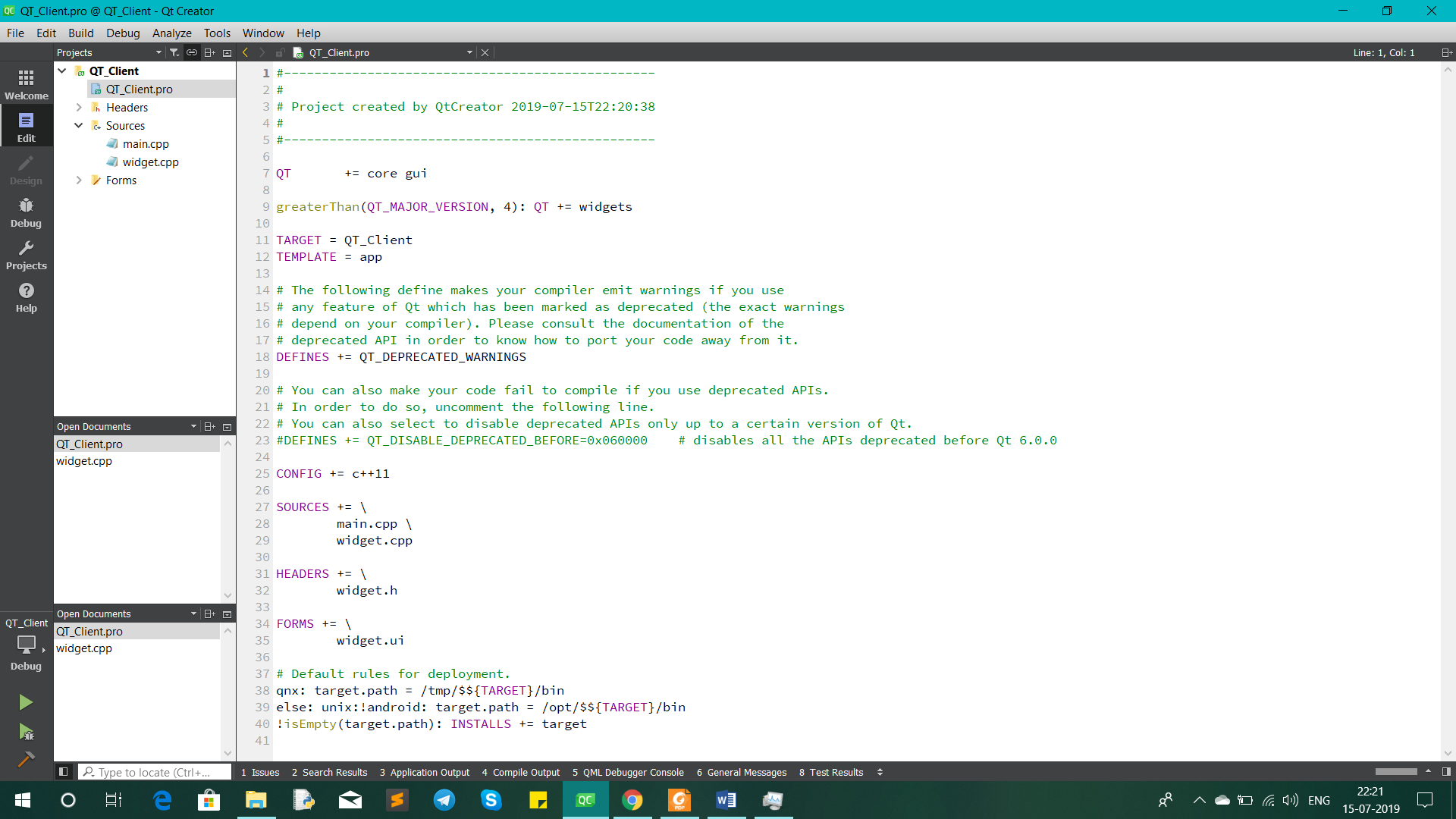


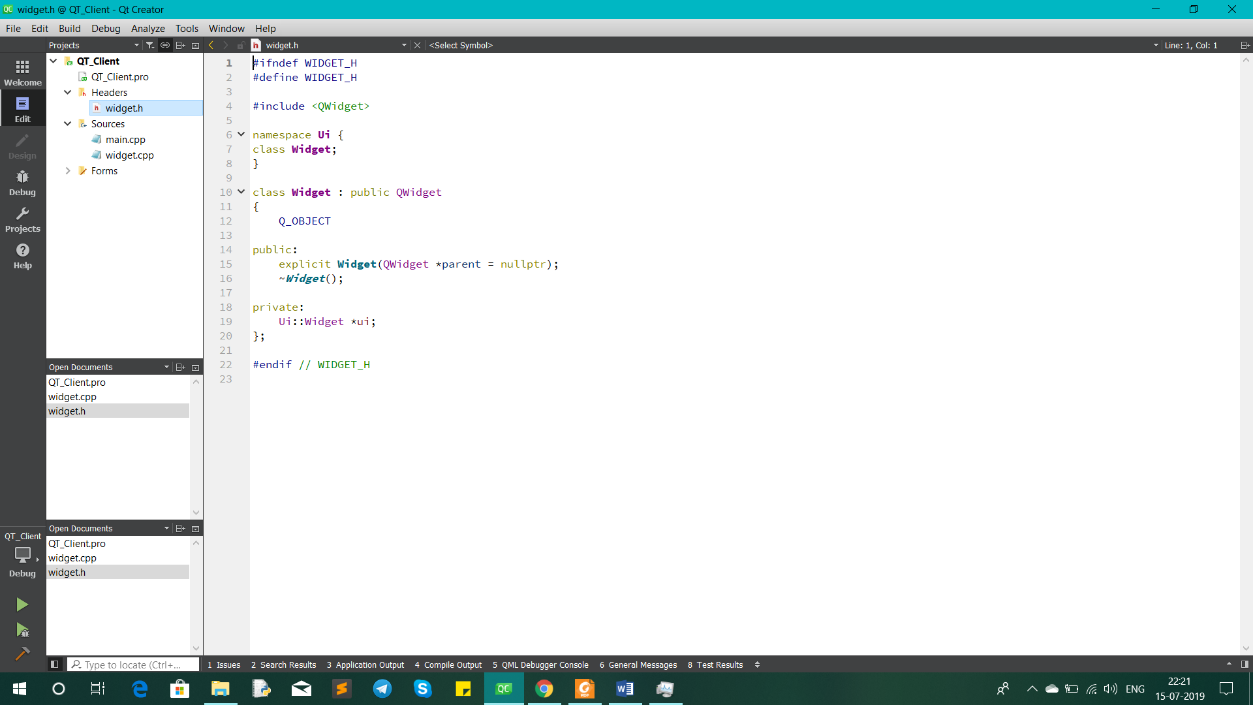


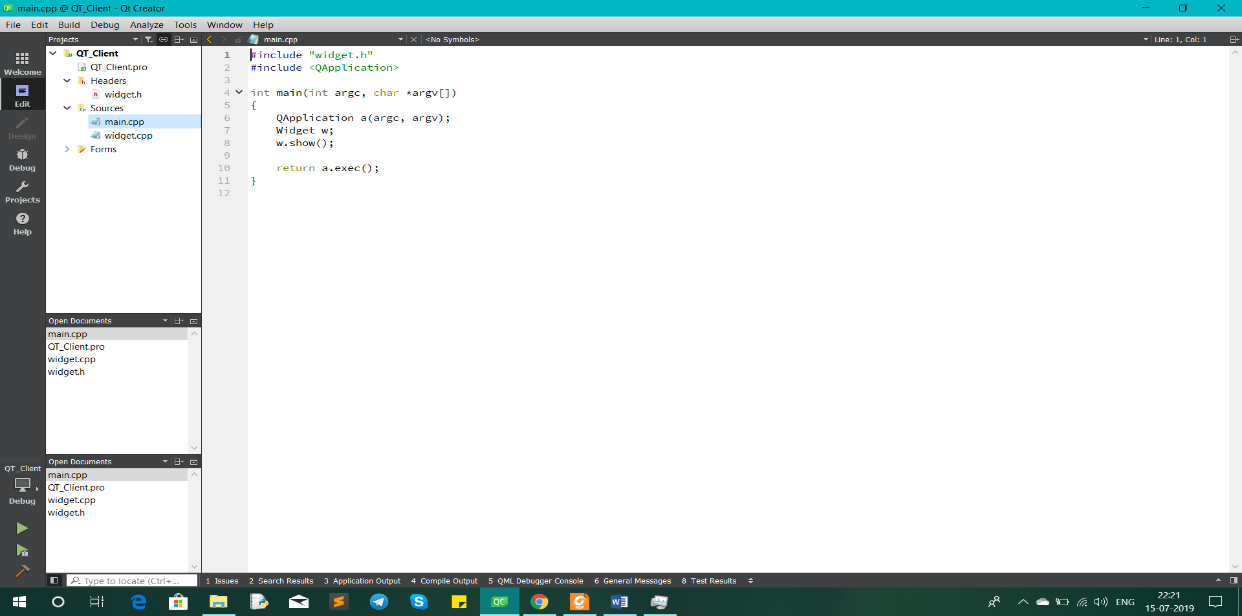


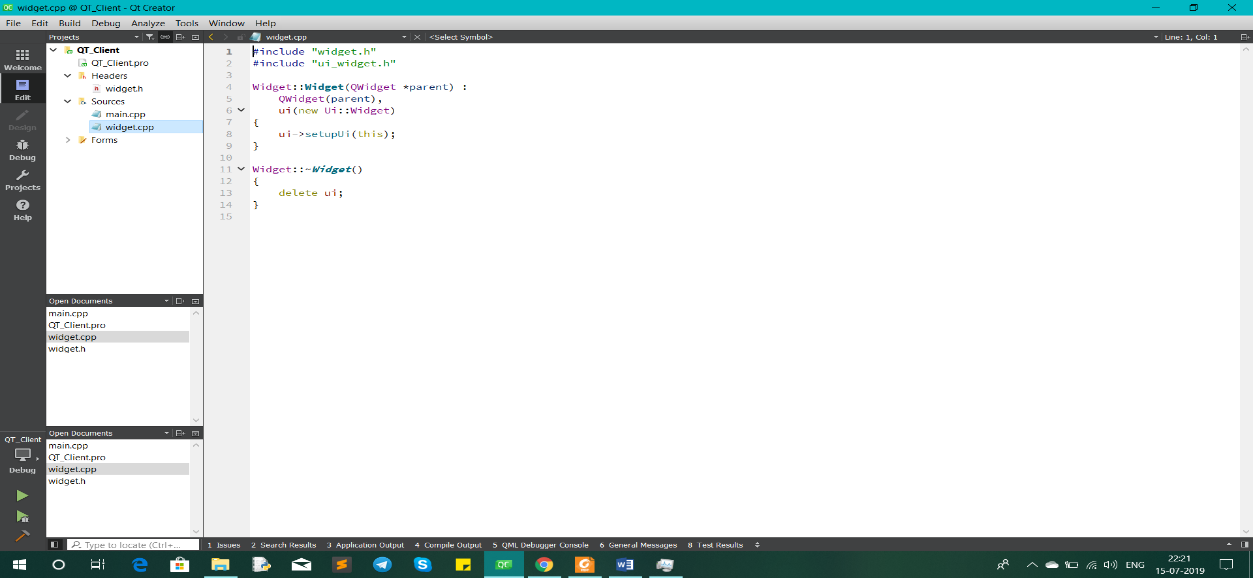


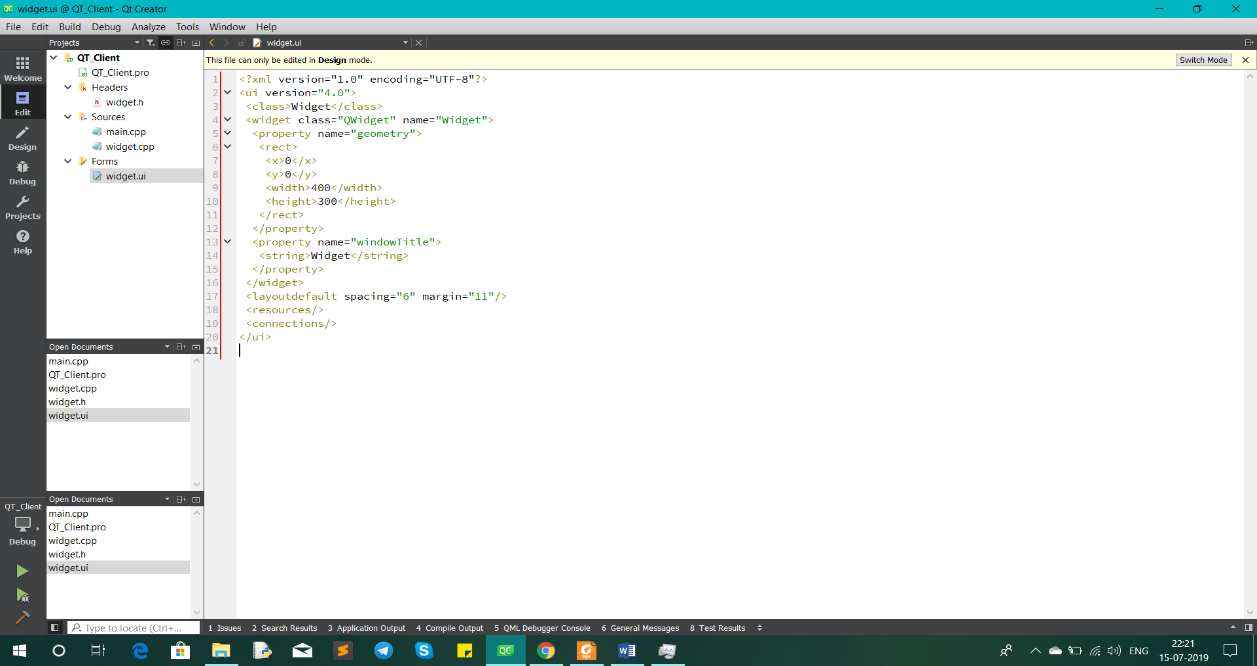
* **Step 2: Following 5 files are automatically created by QT**





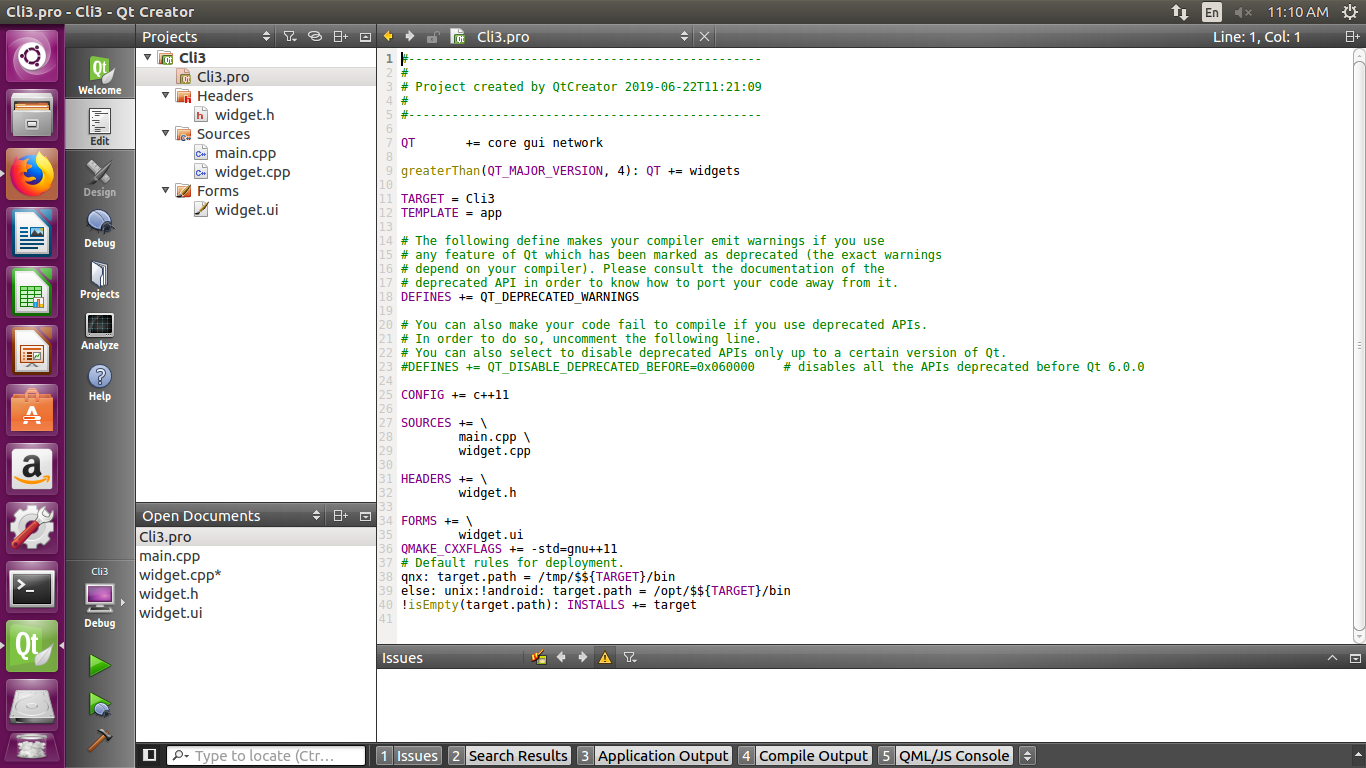




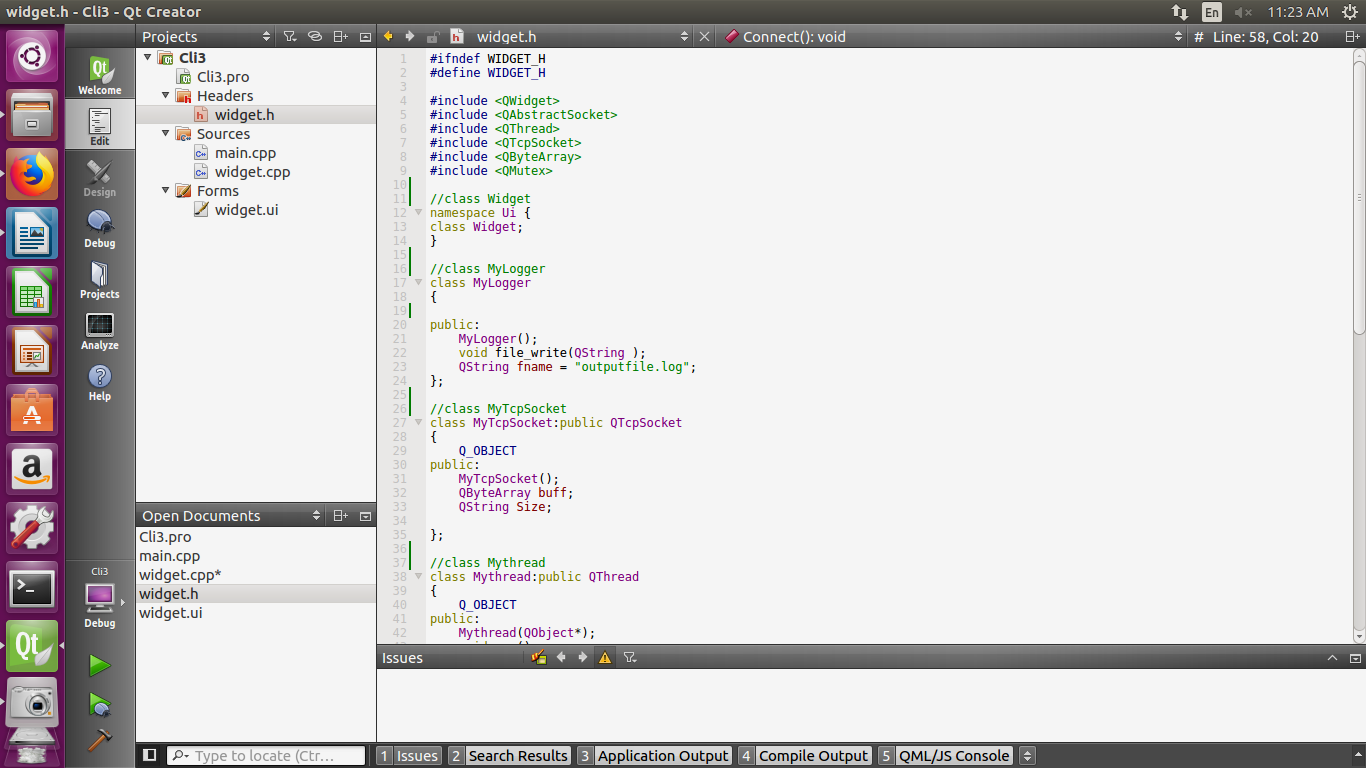


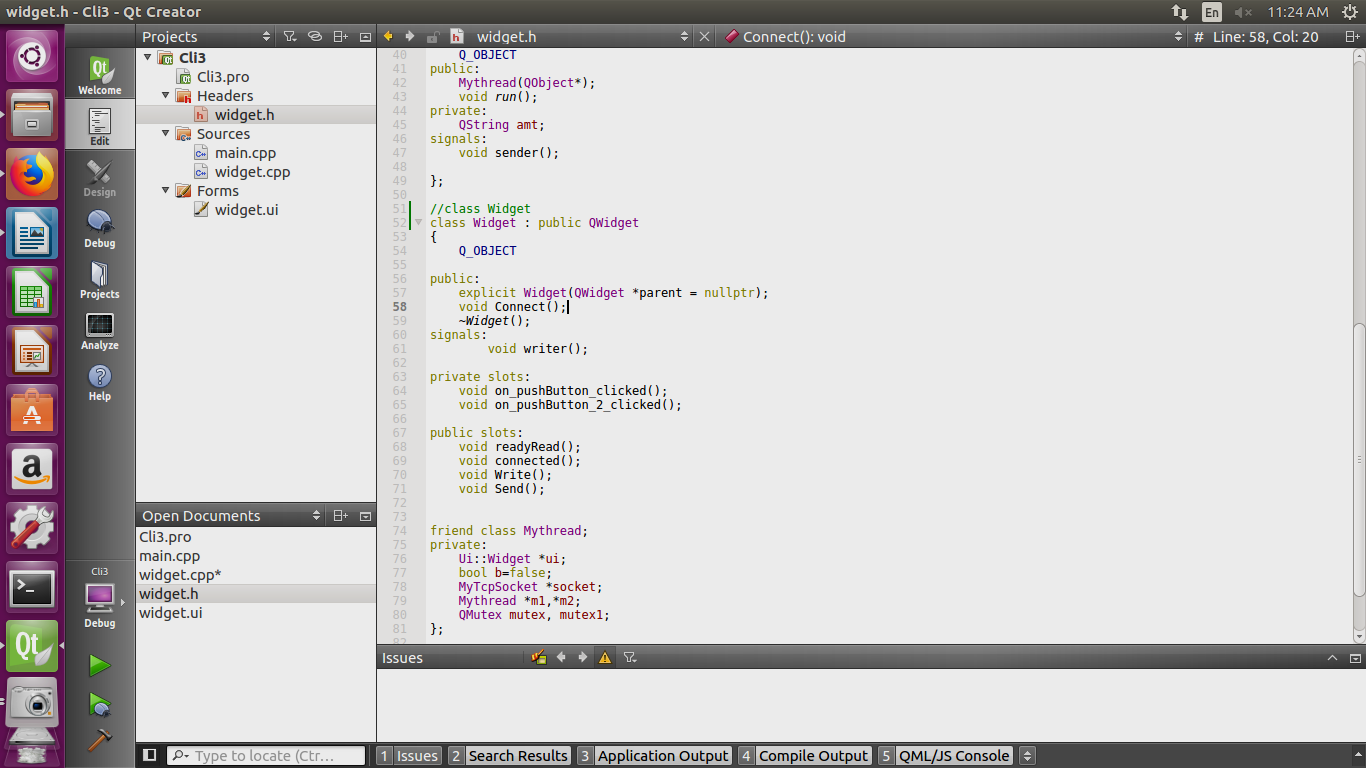
* **Step 3: Following header files are added**
* **QWidget**
* **QAbstractSocket**
* **QTcpSocket**
* **QByteArray**
* **QMutex**
* **QThread**
* **QTextStream**
* **QString**
* **QDebug**
* **QHostAddress**
* **QCoreApplication**
* **Qfile**

* **Step 4: Set up is done in .pro file in order to run socket project**

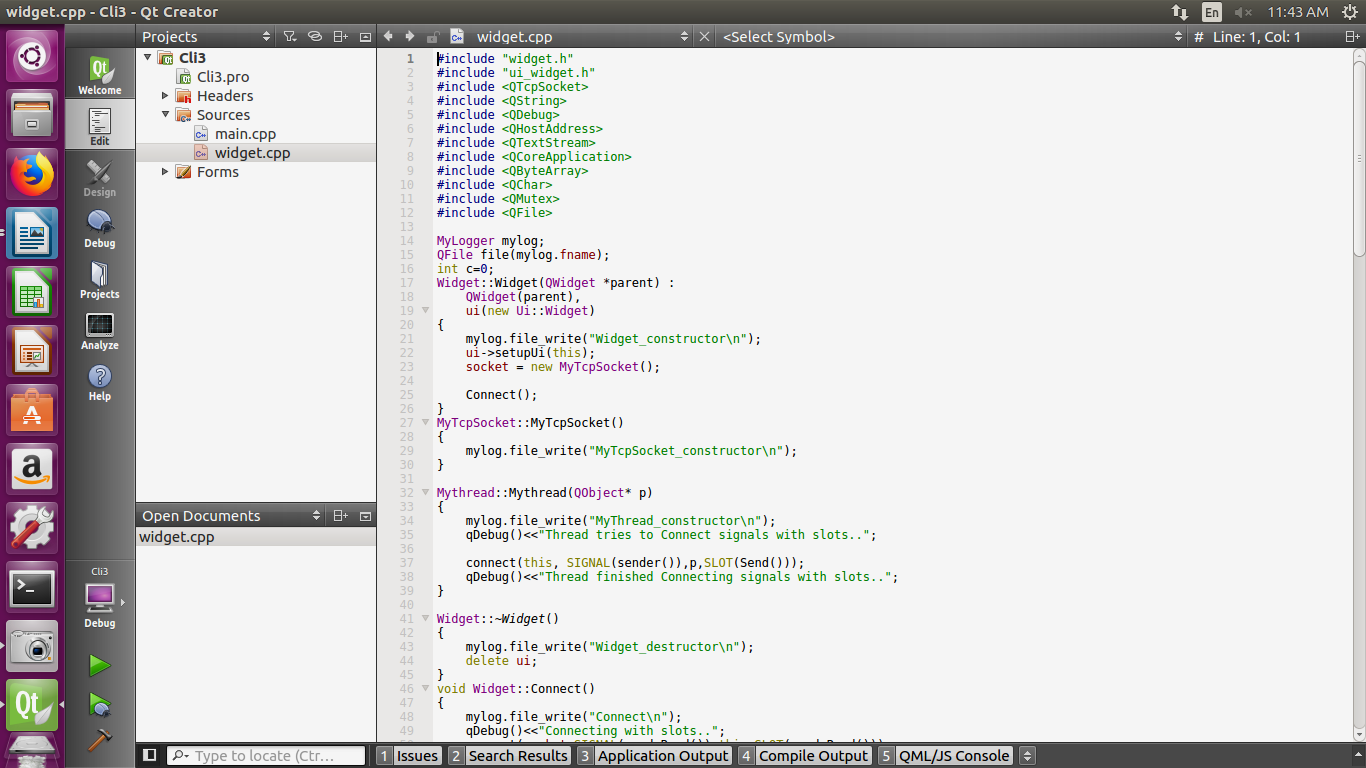
****

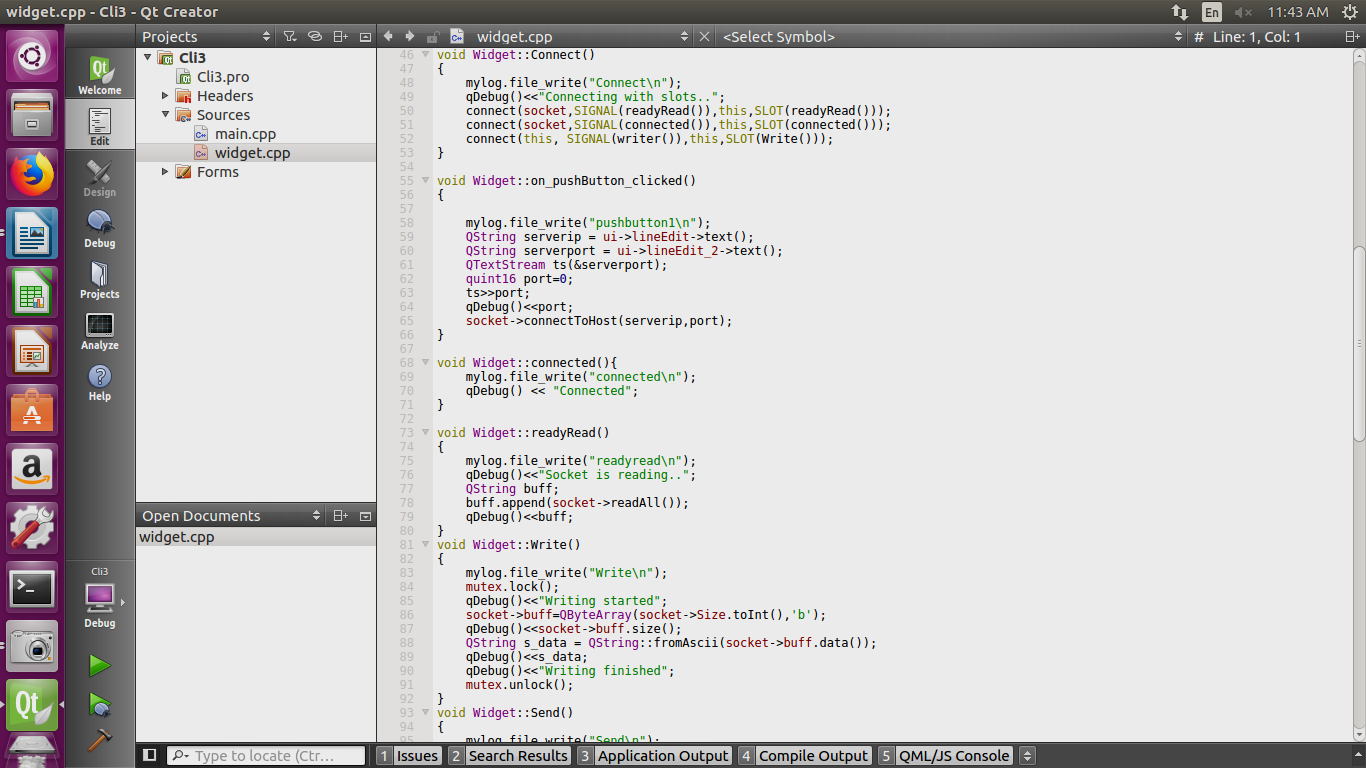
* **Step 5: Classes are declared in widget.h**

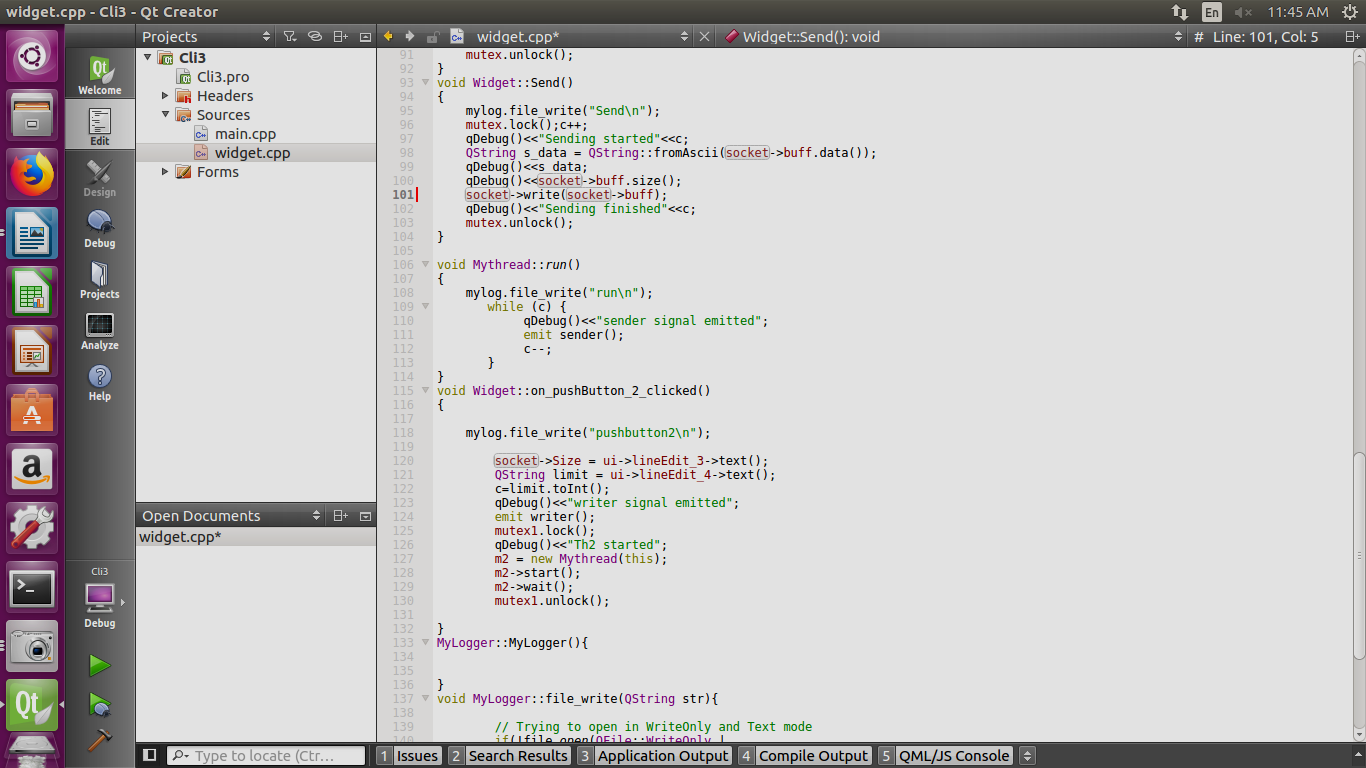
****

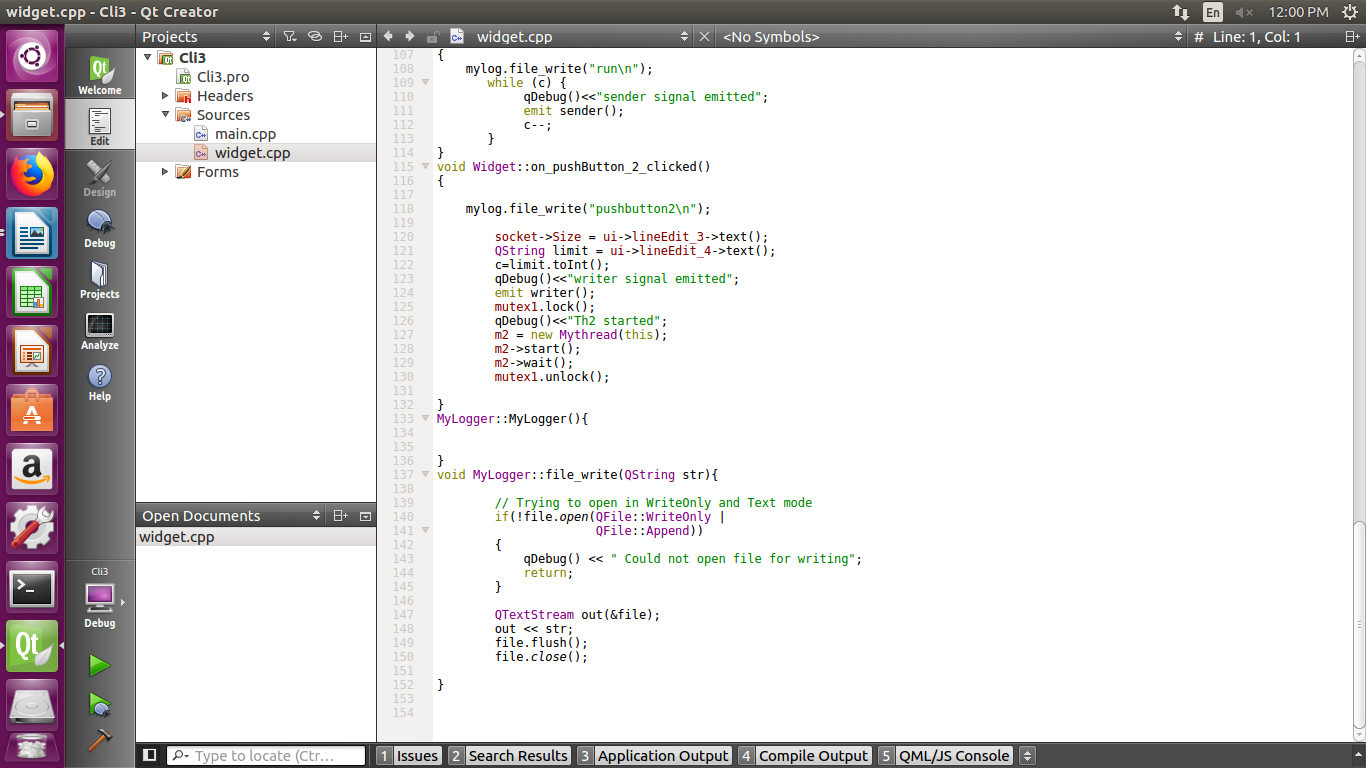


* **Step 6: Classes defined in widget.cpp**



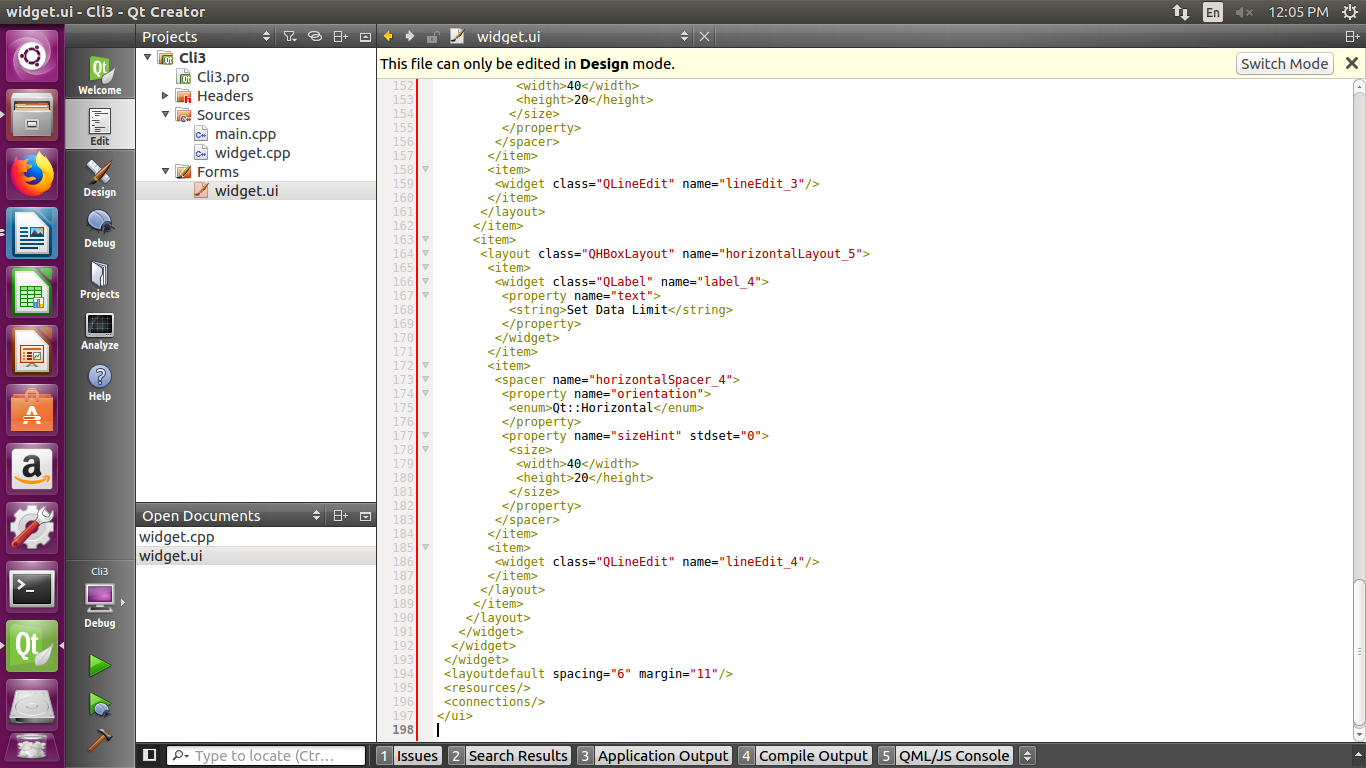


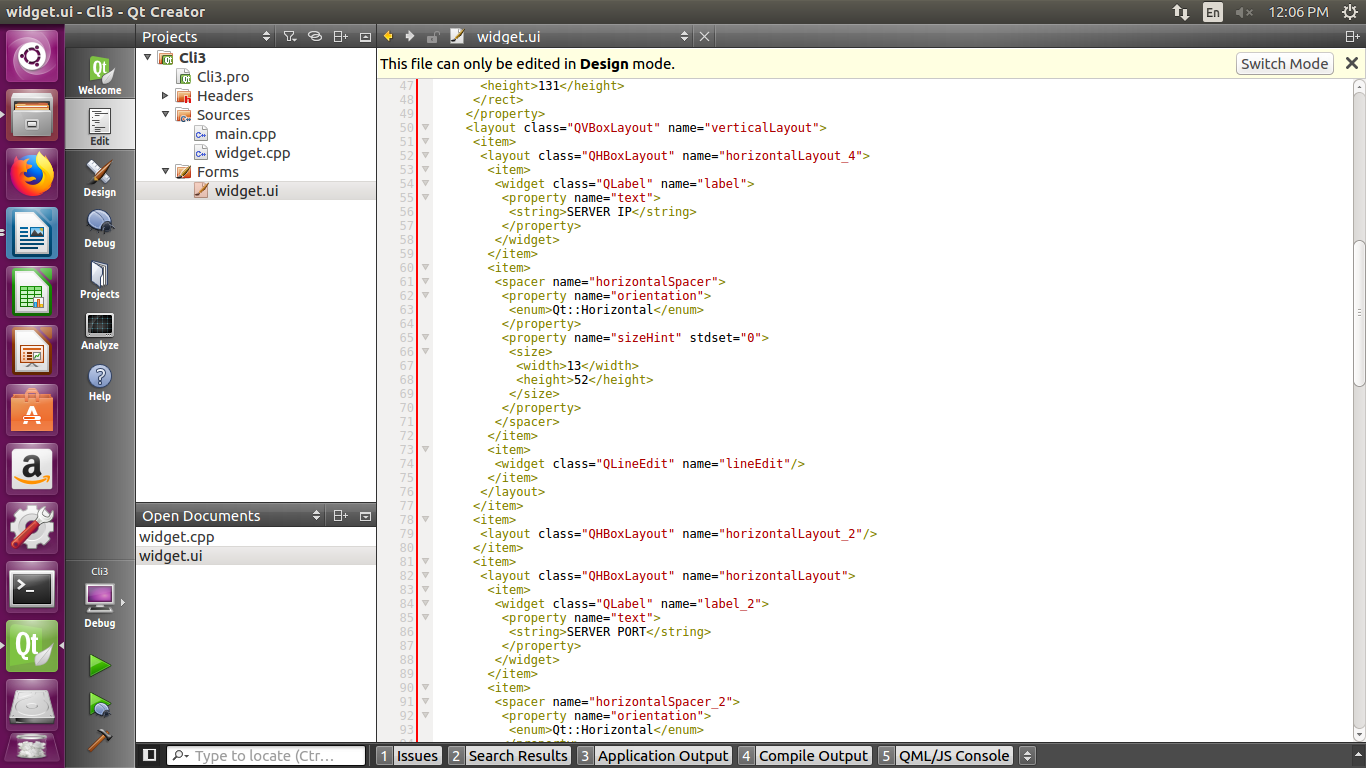


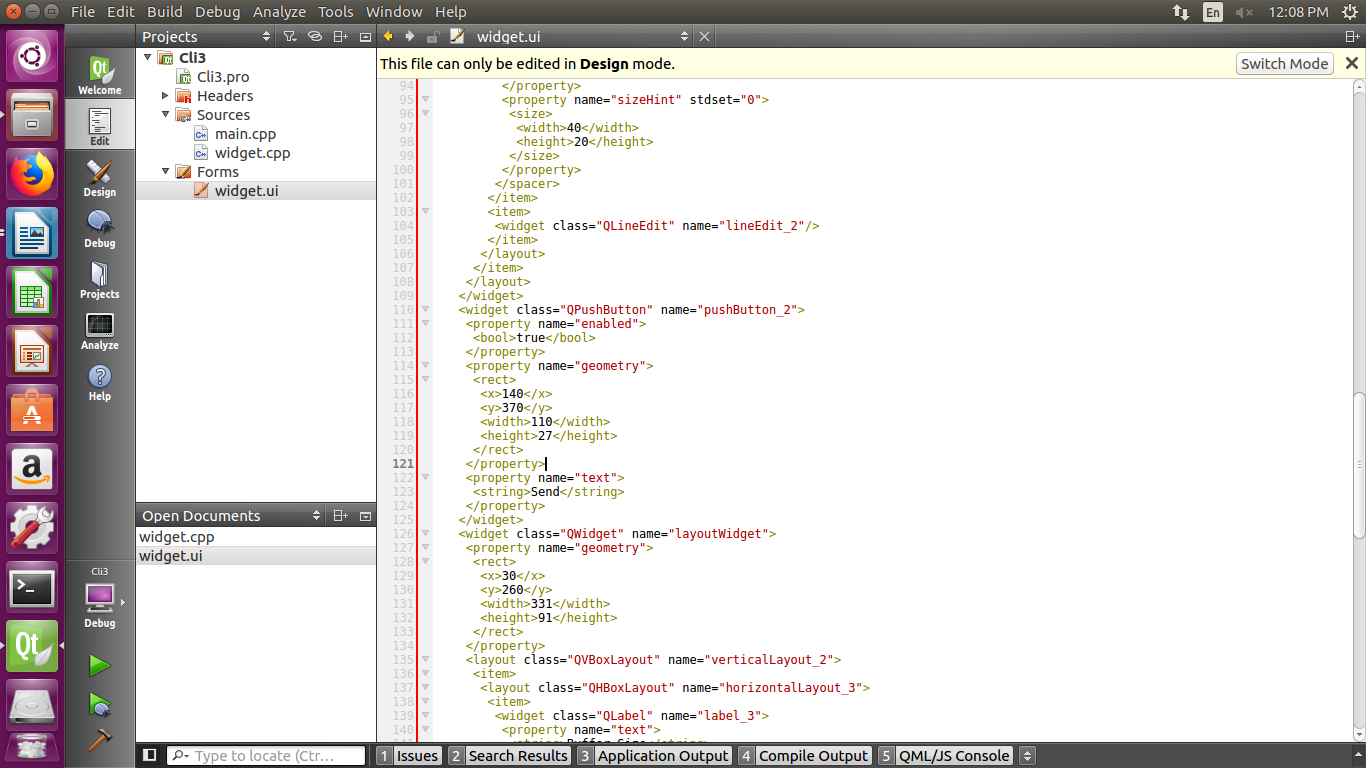


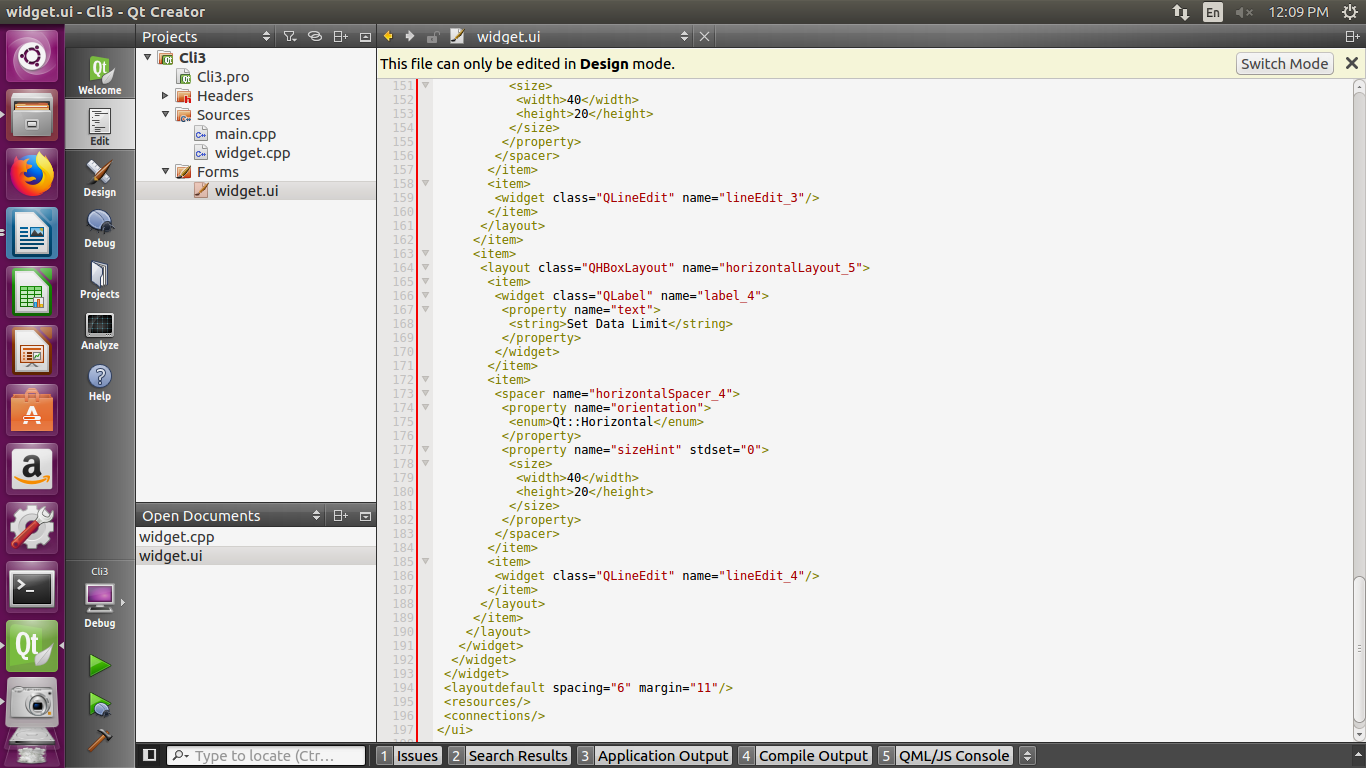
* **Step 7: UI Design**

# 

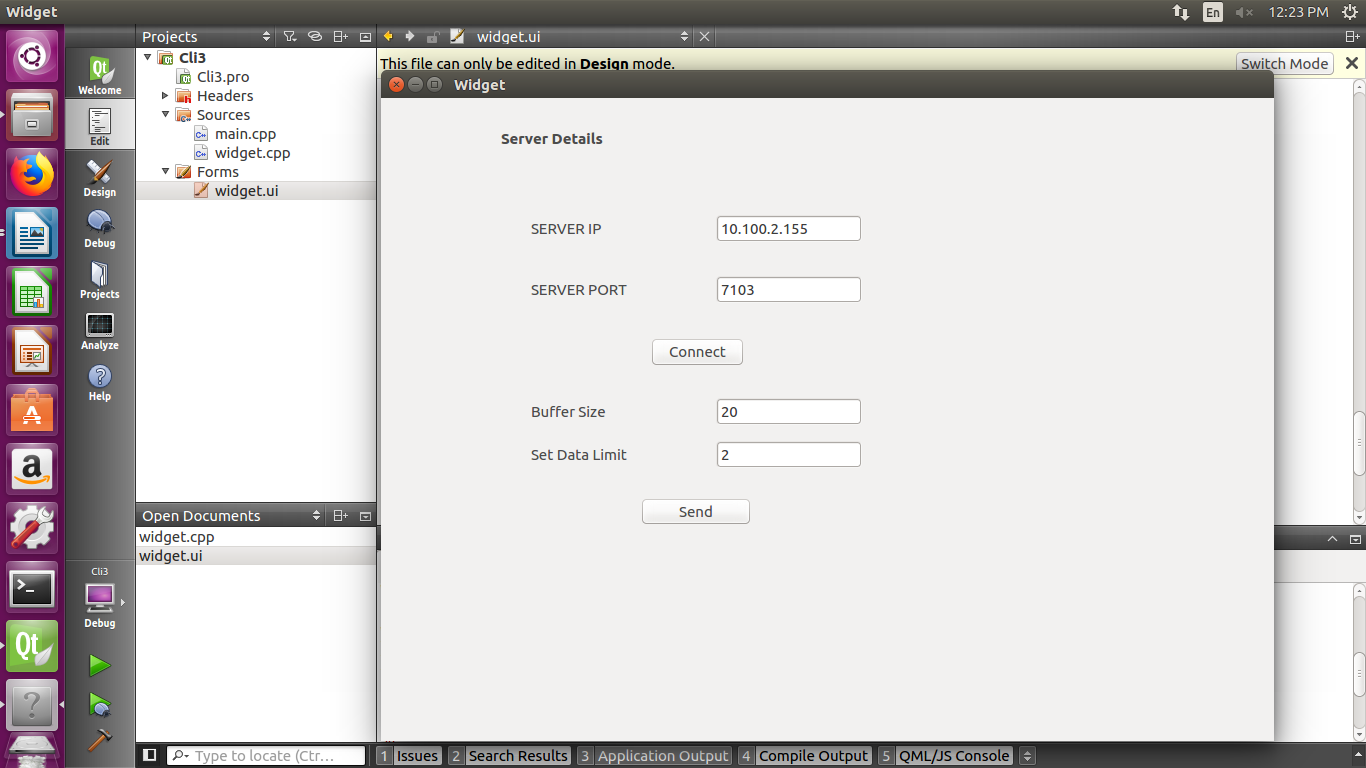
****

****

****

****

# **OUTPUT**

****

****

# **DISCUSSION**

* **Description of different Header files Used:**

**1.QWidget:**

The [QWidget](https://doc.qt.io/qt-5/qwidget.html) class is the base class of all user interface objects.

|  |  |
| --- | --- |
| Header: | #include <QWidget> |
| qmake: | QT += widgets |
| Inherits: | [QObject](https://doc.qt.io/qt-5/qobject.html) and [QPaintDevice](https://doc.qt.io/qt-5/qpaintdevice.html) |
| Inherited By: | [QAbstractButton](https://doc.qt.io/qt-5/qabstractbutton.html), [QAbstractSlider](https://doc.qt.io/qt-5/qabstractslider.html), [QAbstractSpinBox](https://doc.qt.io/qt-5/qabstractspinbox.html), [QCalendarWidget](https://doc.qt.io/qt-5/qcalendarwidget.html), [QComboBox](https://doc.qt.io/qt-5/qcombobox.html), [QDesktopWidget](https://doc.qt.io/qt-5/qdesktopwidget.html), [QDialog](https://doc.qt.io/qt-5/qdialog.html), [QDialogButtonBox](https://doc.qt.io/qt-5/qdialogbuttonbox.html), [QDockWidget](https://doc.qt.io/qt-5/qdockwidget.html), [QFocusFrame](https://doc.qt.io/qt-5/qfocusframe.html), [QFrame](https://doc.qt.io/qt-5/qframe.html), [QGroupBox](https://doc.qt.io/qt-5/qgroupbox.html), [QKeySequenceEdit](https://doc.qt.io/qt-5/qkeysequenceedit.html), [QLineEdit](https://doc.qt.io/qt-5/qlineedit.html), [QMacCocoaViewContainer](https://doc.qt.io/qt-5/qmaccocoaviewcontainer.html), [QMacNativeWidget](https://doc.qt.io/qt-5/qmacnativewidget.html), [QMainWindow](https://doc.qt.io/qt-5/qmainwindow.html), [QMdiSubWindow](https://doc.qt.io/qt-5/qmdisubwindow.html), [QMenu](https://doc.qt.io/qt-5/qmenu.html), [QMenuBar](https://doc.qt.io/qt-5/qmenubar.html), [QOpenGLWidget](https://doc.qt.io/qt-5/qopenglwidget.html), [QProgressBar](https://doc.qt.io/qt-5/qprogressbar.html), [QRubberBand](https://doc.qt.io/qt-5/qrubberband.html), [QSizeGrip](https://doc.qt.io/qt-5/qsizegrip.html), [QSplashScreen](https://doc.qt.io/qt-5/qsplashscreen.html), [QSplitterHandle](https://doc.qt.io/qt-5/qsplitterhandle.html), [QStatusBar](https://doc.qt.io/qt-5/qstatusbar.html), [QTabBar](https://doc.qt.io/qt-5/qtabbar.html), [QTabWidget](https://doc.qt.io/qt-5/qtabwidget.html), [QToolBar](https://doc.qt.io/qt-5/qtoolbar.html), and [QWizardPage](https://doc.qt.io/qt-5/qwizardpage.html) |

**2.QAbstractSocket:**

The [QAbstractSocket](https://doc.qt.io/qt-5/qabstractsocket.html) class provides the base functionality common to all

Socket types.

|  |  |
| --- | --- |
| Header: | #include <QAbstractSocket> |
| qmake: | QT += network |
| Inherits: | [QIODevice](https://doc.qt.io/qt-5/qiodevice.html) |
| Inherited By: | [QTcpSocket](https://doc.qt.io/qt-5/qtcpsocket.html) and [QUdpSocket](https://doc.qt.io/qt-5/qudpsocket.html) |

**3.QTcpSocket:**

The [QTcpSocket](https://doc.qt.io/qt-5/qtcpsocket.html) class provides a TCP socket.

|  |  |
| --- | --- |
| Header: | #include <QTcpSocket> |
| qmake: | QT += network |
| Inherits: | [QAbstractSocket](https://doc.qt.io/qt-5/qabstractsocket.html) |
| Inherited By: | [QSctpSocket](https://doc.qt.io/qt-5/qsctpsocket.html) and [QSslSocket](https://doc.qt.io/qt-5/qsslsocket.html) |

**4.QByteArray:**

The [QByteArray](https://doc.qt.io/qt-5/qbytearray.html) class provides an array of bytes.

|  |  |
| --- | --- |
| Header: | #include <QByteArray> |
| qmake: | QT += core |

**5.QMutex:**

The [QMutex](https://doc.qt.io/qt-5/qmutex.html) class provides access serialization between threads.

|  |  |
| --- | --- |
| Header: | #include <QMutex> |
| qmake: | QT += core |
| Inherits: | [QBasicMutex](https://doc.qt.io/qt-5/qbasicmutex.html) |

**6.QThread:**

The [QThread](https://doc.qt.io/qt-5/qthread.html) class provides a platform-independent way to manage threads.

|  |  |
| --- | --- |
| Header: | #include <QThread> |
| qmake: | QT += core |
| Inherits: | [QObject](https://doc.qt.io/qt-5/qobject.html) |

**7.QTextStream:**

The [QTextStream](https://doc-snapshots.qt.io/qt5-5.12/qtextstream.html) class provides a convenient interface for reading and writing

text..

|  |  |
| --- | --- |
| Header: | #include <QTextStream> |
| qmake: | QT += core |

**8.QString:**

The [QString](https://doc.qt.io/qt-5/qstring.html) class provides a Unicode character string.

|  |  |
| --- | --- |
| Header: | #include <QString> |
| qmake: | QT += core |

**9.QDebug:**

The [QDebug](https://doc.qt.io/qt-5/qdebug.html) class provides an output stream for debugging information.

|  |  |
| --- | --- |
| Header: | #include <QDebug> |
| qmake: | QT += core |

**10.QHostAddress:**

The [QHostAddress](https://doc.qt.io/qt-5/qhostaddress.html) class provides an IP address.

|  |  |
| --- | --- |
| Header: | #include <QHostAddress> |
| qmake: | QT += network |

**11.QCoreApplication:**

The [QCoreApplication](https://doc.qt.io/qt-5/qcoreapplication.html) class provides an event loop for Qt applications without UI.

|  |  |
| --- | --- |
| Header: | #include <QCoreApplication> |
| qmake: | QT += core |
| Inherits: | [QObject](https://doc.qt.io/qt-5/qobject.html) |

**12.QFile:**

The [QFile](https://doc.qt.io/qt-5/qfile.html) class provides an interface for reading from and writing to files.

|  |  |
| --- | --- |
| Header: | #include <QFile> |
| qmake: | QT += core |
| Inherits: | [QFileDevice](https://doc.qt.io/qt-5/qfiledevice.html) |
| Inherited By: | [QTemporaryFile](https://doc.qt.io/qt-5/qtemporaryfile.html) |

* **Description of Important pre-defined Functions used:**

**1.connect(obj1,signal,obj2,slot):-**

The signals and slots mechanism is type safe: The signature of a signal must match the signature of the receiving slot. (In fact a slot may have a shorter signature than the signal it receives because it can ignore extra arguments.) Since the signatures are compatible, the compiler can help us detect type mismatches. Signals and slots are loosely coupled: A class which emits a signal neither knows nor cares which slots receive the signal. Qt's signals and slots mechanism ensures that if you connect a signal to a slot, the slot will be called with the signal's parameters at the right time. Signals and slots can take any number of arguments of any type. They are completely type safe.

All classes that inherit from [QObject](https://doc.qt.io/archives/qt-4.8/porting4.html" \l "qobject) or one of its subclasses (e.g., [QWidget](https://doc.qt.io/archives/qt-4.8/qwidget.html)) can contain signals and slots. Signals are emitted by objects when they change their state in a way that may be interesting to other objects. This is all the object does to communicate. It does not know or care whether anything is receiving the signals it emits. This is true information encapsulation, and ensures that the object can be used as a software component.

Slots can be used for receiving signals, but they are also normal member functions. Just as an object does not know if anything receives its signals, a slot does not know if it has any signals connected to it. This ensures that truly independent components can be created with Qt.

You can connect as many signals as you want to a single slot, and a signal can be connected to as many slots as you need. It is even possible to connect a signal directly to another signal. (This will emit the second signal immediately whenever the first is emitted.)

Together, signals and slots make up a powerful component programming mechanism.

**2.connectToHost(const QString &*hostName*, quint16 *port*, QIODevice::OpenMode *openMode* = ReadWrite, QAbstractSocket::NetworkLayerProtocol *protocol* = AnyIPProtocol):-**

Attempts to make a connection to *hostName* on the given *port*. The *protocol* parameter can be used to specify which network protocol to use (eg. IPv4 or IPv6).

The socket is opened in the given *openMode* and first enters [HostLookupState](https://doc.qt.io/qt-5/qabstractsocket.html" \l "SocketState-enum), then performs a host name lookup of *hostName*. If the lookup succeeds, [hostFound](https://doc.qt.io/qt-5/qabstractsocket.html" \l "hostFound)() is emitted and [QAbstractSocket](https://doc.qt.io/qt-5/qabstractsocket.html) enters [ConnectingState](https://doc.qt.io/qt-5/qabstractsocket.html" \l "SocketState-enum). It then attempts to connect to the address or addresses returned by the lookup. Finally, if a connection is established, [QAbstractSocket](https://doc.qt.io/qt-5/qabstractsocket.html) enters [ConnectedState](https://doc.qt.io/qt-5/qabstractsocket.html" \l "SocketState-enum) and emits [connected](https://doc.qt.io/qt-5/qabstractsocket.html" \l "connected)().

At any point, the socket can emit [error](https://doc.qt.io/qt-5/qabstractsocket.html" \l "error)() to signal that an error occurred.

*hostName* may be an IP address in string form (e.g., "43.195.83.32"), or it may be a host name (e.g., "example.com"). [QAbstractSocket](https://doc.qt.io/qt-5/qabstractsocket.html) will do a lookup only if required. *port* is in native byte order.

**3.write(const char \*, qint64 ):-**

Writes at most *maxSize* bytes of data from *data* to the device. Returns the number of bytes that were actually written, or -1 if an error occurred.

**LIMITATIONS**

* It can’t send data infinitely
* If the data limit is set much higher(>1KB), it takes significant amount of time.

# **BIBLIOGRAPHY**

* [**https://www.qt.io/**](https://www.qt.io/)
* [**https://riptutorial.com/qt/example/29873/tcp-client**](https://riptutorial.com/qt/example/29873/tcp-client)