Yuxuan Liang

USING QT TO IMPLEMENT A TRANSLATION APPLICATION

Thesis
CENTRIA UNIVERSITY OF APPLIED SCIENCES
Information Technology
December 2019

ABSTRACT

Centria University of Applied Sciences	**Date** December 2019	**Author** Yuxuan Liang
Degree programme Information Technology		
Name of thesis USING QT TO IMPLEMENT A TRANSLATION APPLICATION		
Instructor Jari Isohanni		**Pages** 32+1
Supervisor Jari Isohanni		

The aim of this thesis was to create a translation application with Qt. The main idea was to make an absolutely free translation application for users.

While learning a language, people have the option to add their own interpretation of words to the thesaurus, to share and learn other people's understanding of words. The thesis consists of two parts. The first part is to learn the basic knowledge of Qt procedures. Correspondingly, by learning the theoretical knowledge of Qt, and then discovering which functions of the translation software can be realized through related Qt programming knowledge. The entire software will use the relevant knowledge of Qt database, Qt interface design, modeling, C ++, Qt signal and slot connection.

As a result of the thesis work, this translation application can meet the daily learning needs of users and become an example of future development of Qt software.

Key words:
C++, SQL, Json, Qt Designer, Signal and Slots

CONCEPT DEFINITIONS

DOS	Disk Operating System
GUI	Graphical User Interface
MFC	Microsoft Foundation Class Library
VS	Visual Studio
API	Application programming interface
IDE	Integrated Development Environment
SQL	Structured Query Language
DDL	Data Definition Language
TCP	Transmission Control Protocol
UDP	User Data-gram Protocol
HTTP	Hyper Text Transfer Protocol

ABSTRACT

CONCEPT DEFINITIONS

CONTENTS

FIGURES

PICTURES

TABLES

1 INTRODUCTION

English is the most commonly used language in the world. Classic and authoritative dictionaries such as the Oxford English Dictionary and Long-man Dictionary can be said to be well-known. It is an auxiliary tool for many scholars to learn English. A dictionary needs to invest much manpower and material resources in the process of writing, and the electronic version of the online version is more convenient than the paper version. It can be quickly checked at any time, so more and more people use the electronic dictionary, becoming a kind of the new trend is playing an increasingly important role. The research scope of electronic dictionaries in Europe is relatively wide, and the beginning is relatively earlier than in China. For the types of dictionaries, which is not limited to the number of dictionaries. Domestic research on dictionary use has also gradually developed.

The aim for this thesis is to make a translation application with Qt which is a cross-platform C ++ application development framework that uses a variety of controls, such as line edit input, labels, buttons, text editing area, list controls and other design controls for the design of the interface, and uses the method of network communication to obtain the corresponding translation parsing function.
More specifically, the application includes a login system which allows users to log in based on their username and password. At the same time, it has basic translation function and pronunciation function, and can pronounce English for content that is parsed into English. The most innovative is the function of adding new words, which can add unfamiliar words to the library, and users can view them when they need them next time. In this function, users can view the new word list by viewing the new word book, they can also delete and modify the word.

2 QT BACKGROUND AND DEVELOPMENT

Qt is a cross-platform with C++ graphical user interface for application development framework developed by Troll-tech in 1991. It can be used to develop GUI -programs, as well as to develop non-GUI programs, such as console programs. It is similar as MFC on Windows. It provides all the functionality that an application developer needs to build a state-of-the-art graphical user interface. (Software 2019.) Qt Creator is a lightweight cross-platform integrated development environment for Qt development. Qt Creator delivers two key benefits: providing the first IDE designed to support cross-platform development and ensuring that developers who are new to the Qt framework can get up and running quickly.

Even without developing Qt applications, Qt Creator is an easy to use and powerful IDE (Qt Wiki 2019). Qt really refers to the Qt library, not Qt Creator. The Qt version number 5.7.0 on the official website also refers to the version of the Qt library. Developers can also use the visual studio integrated development environment to develop Qt-based applications.（Qt library 2019.) Qt encapsulates the same set of external programming interface API on each platform. Developers can write Qt programs on one platform, and can compile code without any modification to other platforms. Generating the application for the corresponding platform. Simply put, a code is compiled everywhere.（C++ wiki 2019.)

2.1 Qt Modules

Qt's modules are divided into three parts: Qt Essentials, Qt Add-Ons, and Qt Tools. The Qt base module includes the features of the Qt Core foundation. The Qt Extension module includes some mobile-related modules from previous QtMobility, such as Bluetooth QtBluetooth, sensor QtSensors. Also included are some of the previous Qt 4 modules, such as QtDBus, QtXML, QtScript. In addition, some new modules have been added, such as graphic effects QtGraphicalEffects, serial Qt Serial Port, and Qt3D appearing in the commercial version. These modules are used for special purposes and many of them need to be used on special platforms. In the expansion module it also saw the Qt Print Support print support module, which is a reorganization module of many previous classes; The Qt tool includes the Qt Designer, Qt Help, and Qt Interface Tools. (Marco 2018.)

2.1.1 Qt basic module framework

The Qt basic functionality for all platforms is defined in the QtBase, and the functionality provided in that module is required in most Qt applications. The bottom layer of the Qt base module is the QtCore module, and all other modules depend on it. (Marco 2019.) At the bottom is QtCore, which provides all the basic functions of meta-object systems, object trees, signal slots, threads, input and output, resource systems, containers, animation frameworks, JSON support, state machine frameworks, plug-in systems, event systems, and more. On top of it, QtCore is directly based on QtTest, QtSql, QtNetwork and QtGui. The test module QtTest and the database module QtSql are relatively independent, and more importantly, the network module QtNetwork and the graphics module QtGui are in them.

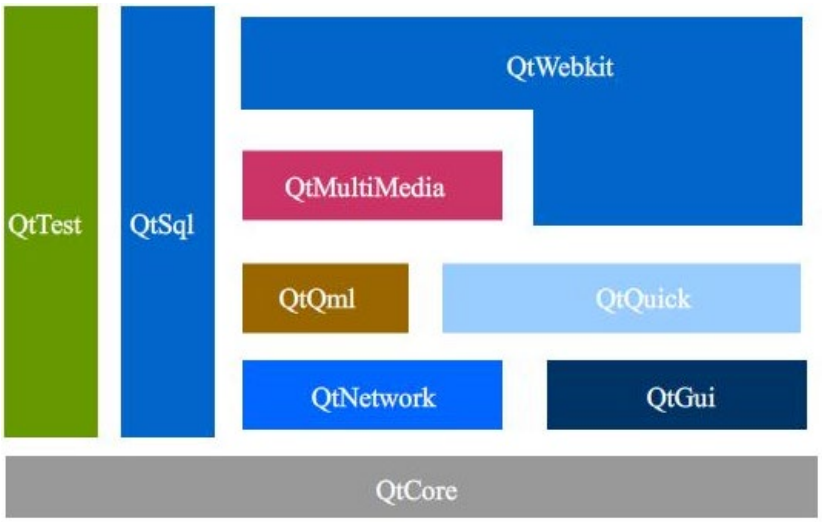


Figure 1. Framework of the entire basic module (Marco 2019)

At the bottom is QtCore, which provides all the basic functions of meta-object systems, object trees, signal slots, threads, input and output, resource systems, containers, animation frameworks, JSON support, state machine frameworks, plug-in systems, and event systems. On top of it, QtCore is directly based on QtTest, QtSql, QtNetwork and QtGui. The test module QtTest and the database module QtSql are relatively independent, and more importantly, the network module QtNetwork and the graphics module QtGui are in them. Above these are the important update parts of Qt 5, QtQml and QtQuick. The top layer is the newly added QtMultiMedia multimedia module, and the QtWebKit module on top of it. (Jamin 2018.) For the whole framework, with understanding the lower module provides support for the

upper module, or the upper module contains the function of the lower module. For example, the QtWebKit module, which has both graphical interface components and network capabilities, also supports multimedia applications. (Jamin 2018.)

2.1.2 Graphical interface library framework

In fact, QApplication is not in the QtGui module. Not only the base class QWidget of all user interfaces is not in the QtGui module, they are reassembled into a new module of QtWidgets. A major change in Qt 5 is the redefinition of the QtGui module, which is no longer a large and comprehensive graphical interface library, but provides a base class for GUI graphical user interface components, including window system integration, event handling, OpenGL, and OpenGL ES integration, 2D drawing, basic images, fonts, and text. (Mark 2018.)

In Qt 5, the graphics component classes from the previous QtGui module were moved to the QtWidgets module, and the print-related classes were moved to the Qt Print Support module. However, the QtOpenGL module was removed from Qt 5 and the OpenGL related classes were moved to the QtGui module. Some readers may find that there is still a QtOpenGL module in the Qt extension module. In fact, it is only reserved for Qt 4 porting to Qt 5. It is highly recommended to use the OpenGL class in the QtGui module when writing Qt 5 programs. (Packt 2018.)

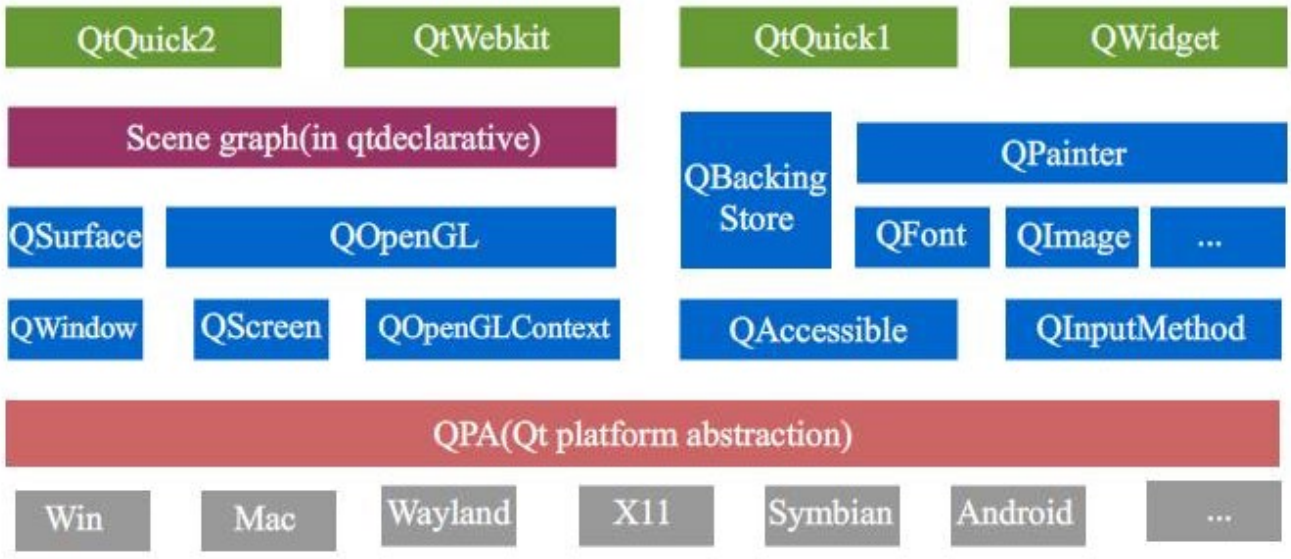


Figure 2. Framework of the Qt graphical interface library (Packt 2018)

Above the various supported platforms is the underlying platform abstraction layer QPA, a beacon project called LightHouse, which is the foundation on which Qt can be ubiquitous. All the blue blocks on it

are the contents of the QtGui module. They are divided into two categories. One is based on OpenGL, which is the basis of the latest QtQuick2 and QtWebkit; the other is auxiliary access and input. A generic-based graphical display class that is the basis for the classic QWidget widget class and QtQuick1. (Packt 2018.)

2.2 Qt with Database SQL

The QtSql module in Qt provides support for the database. The many classes in this module can be basically divided into three layers. The driver layer provides the underlying bridge between the specific database and the SQLinterface layer; the SQL interface layer provides access to the database, where the QSqlDatabase class is used to create the connection, and the QSqlQuery class can use the SQL statement to interact with the database. Several other classes provide support for this layer; several classes of the user interface layer implement linking the data in the database to the widgets. Which are implemented using the model/view framework from the previous chapter.

QtSql 模块的类分层

用户接口层
QSqlQueryModel、QSqlTableModel 和 QSqlRelationalTableModel
SQL 接口层
QSqlDatabase、QSqlQuery、QSqlError、QSqlField、QSqlIndex 和 QSqlRecord
驱动层
QSqlDriver、QSqlDriverCreator<T>、QSqlDriverCreatorBase、QSqlDriverPlugin 和 QSqlResult

Figure 3. Class hierarchy of QtSql module (Matthias 2002)

Which are higher Hierarchical abstraction, even if people are not familiar with SQL, people can manipulate the database. SQL is a language used to manipulate relational databases. SQL uses a combination of keywords, table names, column names. (SQL statement) to describe the content of the operation. Keywords are English words whose meanings or usage methods have been defined in advance, and there are keywords that contain various meanings such as "query the table" or "reference the table". SQL statements can be classified into the following three categories depending on the type of instructions given to the RDBMS. 90% of the actual SQL statements used are DML. DDL is used to create or delete objects for storing data and tables in the database. (Lee 2019.)

Driver Type	Description
QDB2	IBM DB2
QIBASE	Borland InterBase Driver
QMYSQL	MySQL Driver
QOCI	Oracle Call Interface Driver
QODBC	ODBC Driver (includes Microsoft SQL Server)
QPSQL	PostgreSQL Driver
QSQLITE	SQLite version 3 or above
QSQLITE2	SQLite version 2
QTDS	Sybase Adaptive Server

Figure 4. Qt database Driver type. (Matthias 2002)

2.3 QT Designer

PyQt is a toolkit for creating GUI applications. It is a successful fusion of the Python programming language and the Qt library. PyQt was developed by Phil Thompson. PyQt implements a Python module set. It has more than 300 classes and nearly 6,000 functions and methods. It is a mufti-platform toolkit that runs on all major operating systems, including UNIX, Windows and Mac. PyQt uses dual licenses and developers can choose between GPL and commercial licenses. Prior to this, the GPL version could only be used on Unix. Starting with version 4 of PyQt, the GPL license is available for all supported platforms. Because there are many classes available, they are divided into several modules. The QtCore module contains core non-GUI features. This module is used for time, files and directories, various data types, streams, URLs, MIME types, threads or processes. The QtGui module contains graphical components and related classes such as buttons, forms, status bars, toolbars, scroll bars, bitmaps, colors and fonts. The QtNetwork module includes classes for network programming that allow people to write TCP/IP and UDP clients and servers that make network programming simpler and lighter.

QtXml contains classes that use XML files. This module provides implementations of the SAX and DOM APIs. The QtSvg module provides classes for displaying SVG files. Scalable Vector Graphics (SVG) is an XML language used to describe two-dimensional graphics and graphics applications. The QtOpenGL module uses OpenGL libraries to render 3D and 2D graphics, and the module seamlessly integrates Qt's GUI libraries and OpenGL libraries. The QtSql module provides classes for the database.

(Pearson 2019.) With Qt Designer, developers can create both "dialog" style applications and applications that create a "main window" style with menus, tool-bars, balloon help, and other standard features. Qt Designer provides a variety of form templates, allowing developers to create their own templates to ensure consistency across an application or a range of application interfaces. Programmers can create their own custom forms that can be easily integrated with Qt Designer. (Pearson 2019.)

Qt Designer supports the development of applications using a form-based approach. Forms are represented by user interface (.ui) files that can be converted to C++ and compiled into an application, or processed at run-time to generate a dynamic user interface. Qt's build system automates the compilation process of the user interface, making the design process easier. (Nicholas 2017.) Qt Designer can easily integrate with many common IDEs Commercial license holders of the Windows platform that can gain the power of Qt Designer user interface design within Microsoft Visual Studio®. In Mac OS X, developers can use Qt Designer within the Apple's Xcode® environment. In addition, Nokia has developed Qt's integration plug-in for the cross-platform Eclipse integration environment to embed Qt Designer and other Qt technologies into the integrated environment framework. (Ray 2014.)

2.4 QT with HTTP

HTTP is a standard for client and server requests and responses. A network access interface is provided in the Qt network module to implement HTTP programming. A network access interface is a collection of classes that perform general network operations that provide an abstraction layer for specific operations and protocols used, exposing only classes to the outside world, functions and signals. (Marco 2019.) As mentioned in the previous section, Qt now uses the QNetworkAccessManager class and the QNetworkReply class for HTTP programming. The network request is represented by the QNetworkRequest class, which also serves as a container for information related to the request (for example, any header information and encryption). The URL specified when the request object is created determines the protocol used by the request. Currently, HTTP, FTP, and local file URLs are supported for uploading and downloading. The QNetworkAccessManager class is used to coordinate network operations. When a request is created, the class is used to schedule it and send a signal to report progress.

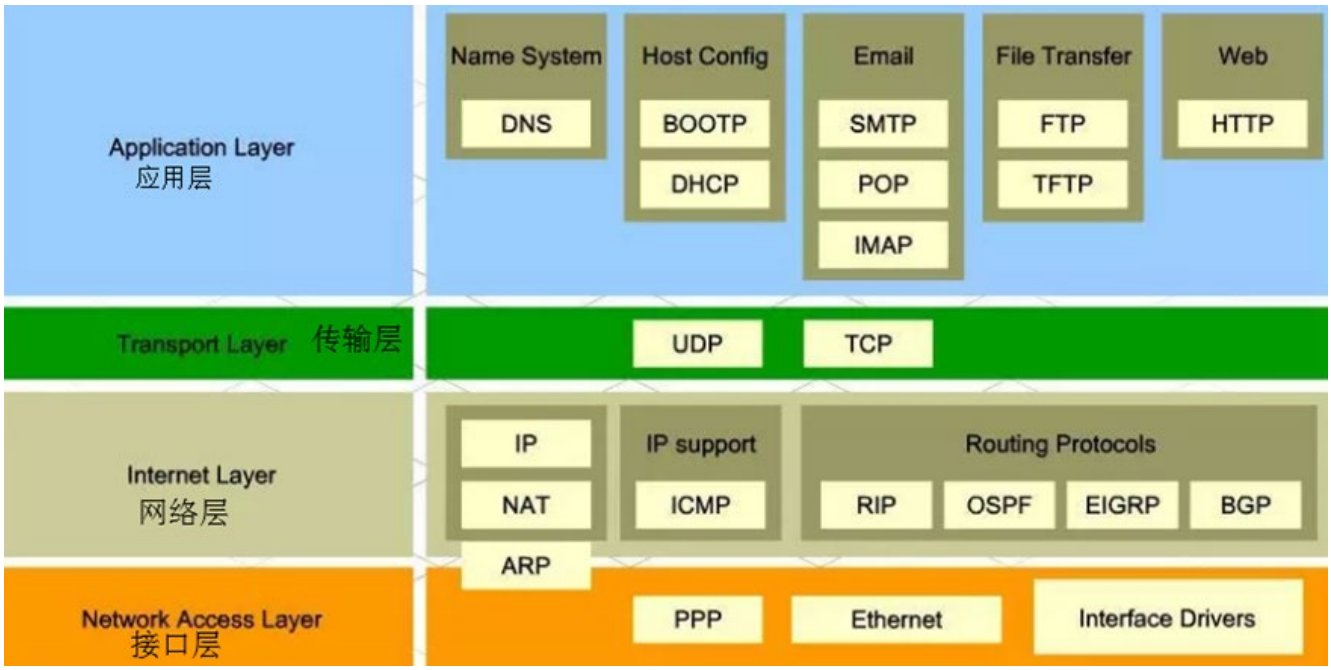


Figure 5. FTP Network protocol (Frank 2010)

This class also coordinates the use of cookies, authentication requests, and the use of their agents. The response to the network request is represented by the QNetworkReply class, which is created by the QNetworkAccessManager when the request is scheduled. The signals provided by QNetworkReply can be used to monitor each response individually. (John Wiley & Sons 2010.) As shown in the figure 5 above, it can be seen that the FTP protocol of the application layer implements file sharing transmission based on the TCP protocol of the transport layer. The TCP protocol of the transport layer is implemented based on the IP of the network layer. By default, the FTP protocol uses two ports, 20 and 21, in the TCP port, of which 20 is used to transmit data and 21 is used to transmit control information. TCP is a connection-oriented protocol, mainly used for large data applications, such as file transfer. A TCP connection must be established after 3 handshakes. (John & Sons 2010.)

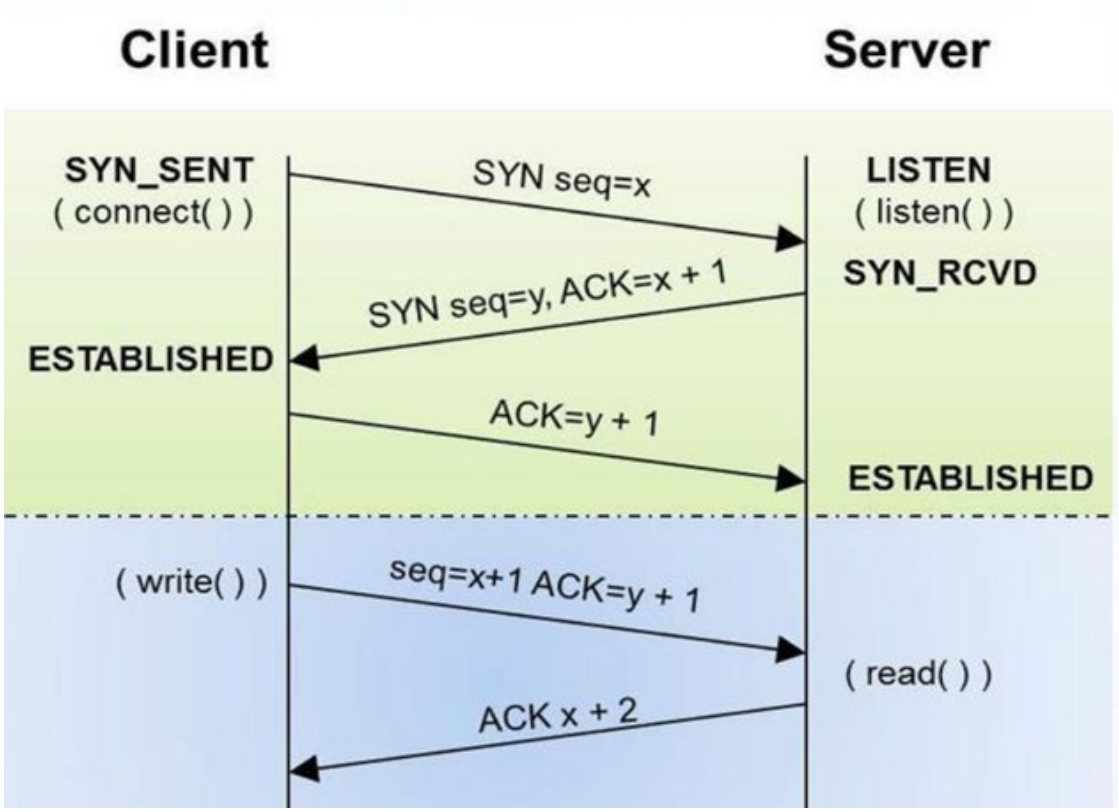


Figure 6. SYN connect with server (Chang 2019)

The client sends a signal to the server request. After receiving the signal server, the server sends a response signal to the client and provides the SEQ serial number (representing the number of each data packet, because the data is sent into multiple data packets). When informing the client, users' next packet sequence number received. The client sends an ACK again to determine the server's ACK request synchronization request. The client sends a FIN to close the client-to-server data transfer. The server receives this FIN, it sends back an ACK, confirming that the serial number is the received SEQ number.

The server closes the connection with the client and sends a FIN to the client A. The client sends back an ACK message confirmation, and sets the confirmation sequence number to the received sequence number plus one. (Chang 2019.) UDP with connection-less protocol is mainly used for the case where the packet sequence is not required to arrive and a small amount of data is transmitted. The data transmission efficiency is high. The disadvantage is that it is easy to drop the packet. The classes provided by the Qt Network module allow the writing of TCP/IP clients and servers, as well as common protocols such as FTP and HTTP. (Chang 2019.)

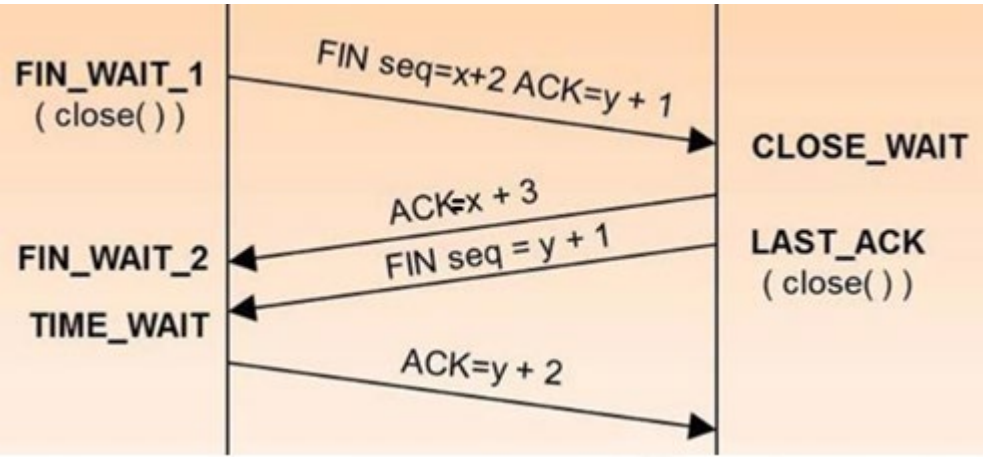


Figure 7. ACK connect with server (Chang 2019)

2.5 QT signals and slots

Signals and slots are used for communication between two objects. The signal and slot mechanism is a core feature of Qt and is the most prominent feature of Qt unlike other development frameworks. In GUI programming, when a component is changed, it is always desirable for other components to understand the change. More generally, people want any object to be able to communicate with other objects. For example, if the user clicks the close button, people want to be able to close the window by executing the window's close() function. In order to achieve communication between objects, some toolkits use a callback mechanism, while in Qt, signals and slots are used for communication between objects.

A special signal can be emitted when a special event occurs, such as a button being clicked; and a slot is a function that is called after the signal is transmitted to respond to the signal. Some signals and slots have been defined in Qt's component classes, but more importantly for users is to subclass this component and then signals and slots to achieve the desired functionality. (Pavel 2018).

The correlation between the signal and the slot used in the previous object is a signal corresponding to one slot. In fact, a signal can be associated with multiple slots. At the same time multiple signals can be associated with the same slot, and even one signal can be associated with another signal, as shown in figure 8. If there are multiple slots associated with a signal, then when this signal is transmitted, the slots will be executed one after the other, but the order in which they are executed is random and their execution order cannot be specified. (Packt 2019.)

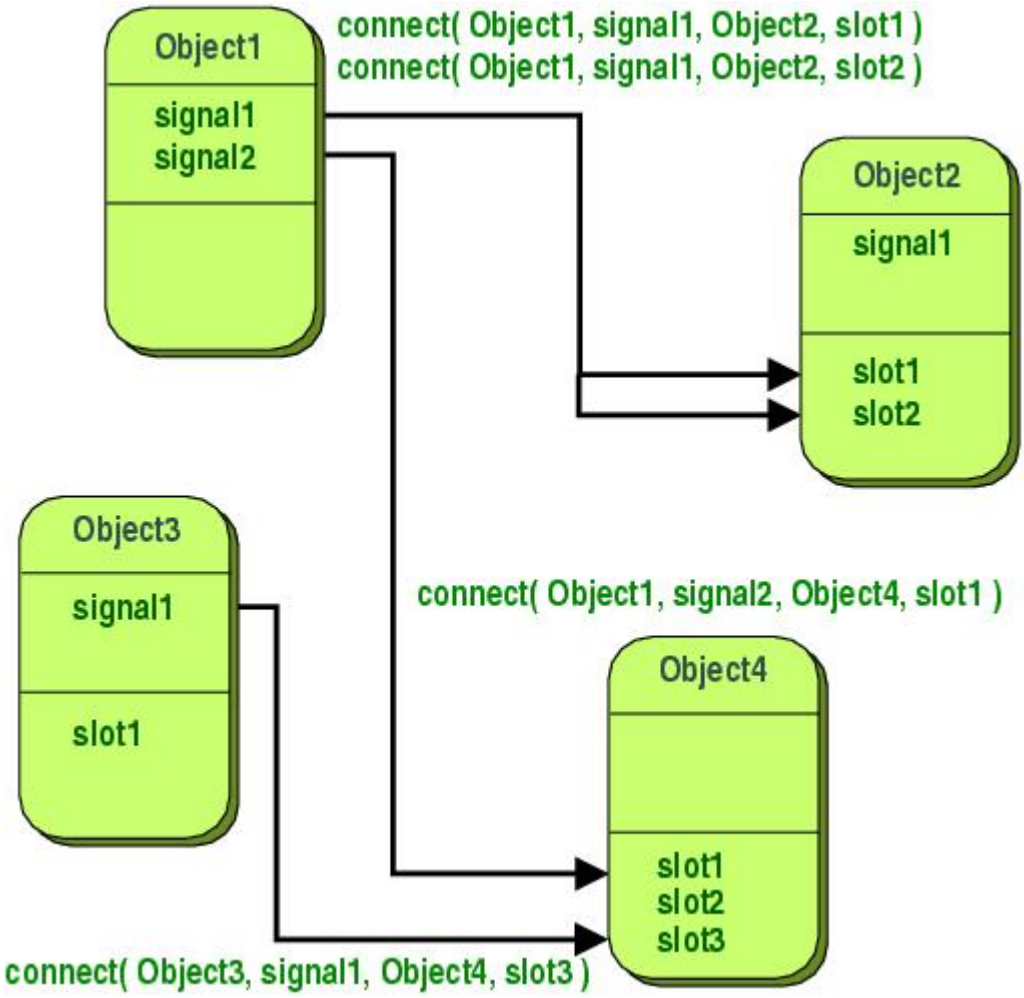


Figure 8. Signals with slots Structure (Pavel 2019)

The signal slot is one of the mechanisms that the Qt framework is proud of. The so-called signal slot is actually the observer mode. For example, when an event occurs, for example, the button detects that it has been clicked and it sends a signal. This kind of issuance has no purpose, Almost same to broadcasting. If an object is interested in this signal, it uses the connect function, which means that the signal to be processed is bound to its own function (called a slot) to process the signal. When the signal is sent, the connected slot function is automatically called back. This is similar to the observer mode: when an event of interest occurs, an operation is automatically triggered. (Witold 2019.)

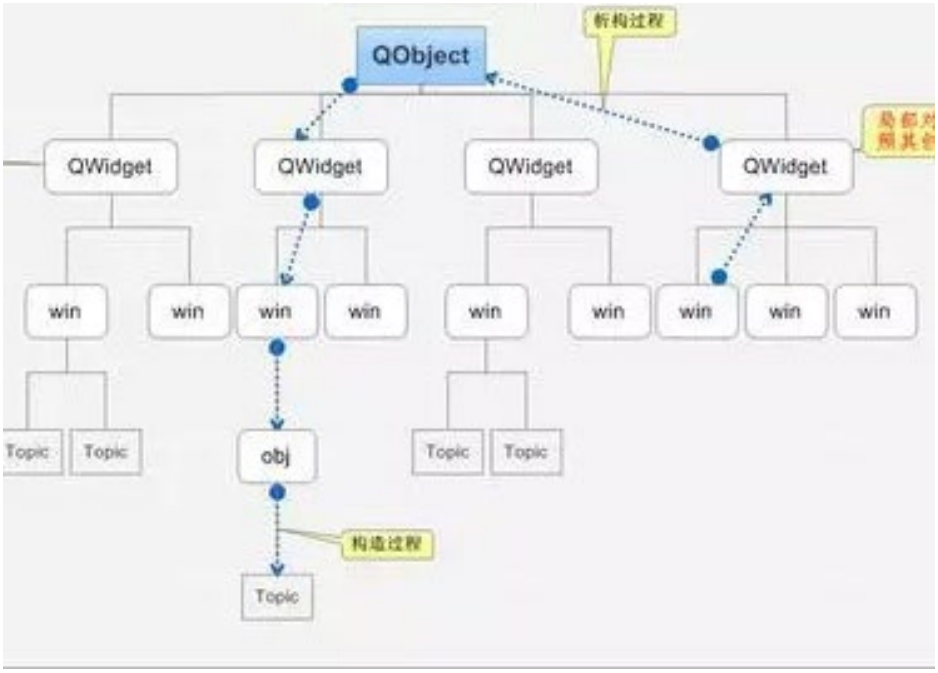


Figure 9. Qt transmitting Process (Pavel 2019)

Signals and slots are Qt-specific information transfer mechanisms and are an important foundation for Qt design programs. As showed in figure 9, they allow a connection between objects that do not interfere with each other. The essence of a slot is a member function of a class, and its parameters can be of any type. There is almost no difference from ordinary C++ member functions. It can be virtual functions; it can also be overloaded; it can be public, protected, private, or it can be called by other C++ member functions. The only difference is that the slot can be connected to the signal and is called whenever the signal connected to the slot is transmitted. (Witold 2019.)

Qt provides a mechanism to automatically and efficiently organize and manage Qt objects that inherit from QObject. This mechanism is the object tree. The Qt object tree is very useful for user interface programming. It can help programmers reduce the pressure of memory leaks. For example, when an application creates an object with a parent widget, the object is added to the child widget's child list. When the application destroys the parent widget, the objects in the child list below it are deleted one by one. This allows people to focus on the system's business when programming, improve programming efficiency, and also ensure the robustness of the system. (Witold 2019.)

3 IMPLEMENTATION OF TRANSLATION SOFTWARE IN QT

Before starting to prepare this project, there are some reasons to choose Qt as the platform for this program instead of Python and Java. First of all, Qt's code size is also among the best in the entire open source world. When choosing to translate software, the user ’s login interface and program display interface will be the main part, and Qt was first known for designing graphical interfaces. Secondly, because many dictionaries are developed using python using Linux, and the startup speed of Python has not reached the second start. Using C ++ and graphics library Deepin tool kit (based on Qt) to make a dictionary, which can log in to the translation system and implement queries Word interpretation and translation function. Qt also provides JSON parsing classes related to QJsonDocument and QJsonObject, which is very convenient, so users don't need to write the parsing class themselves. (Ray 2014.)

3.1 Architecture of QT translator

The architecture of qt translation software, which will be divided into three layers: GUI layer, Models layer, and Communication layer. Because the GUI layer is based on Qt, this layer is implemented using the Qt class library. All main interfaces will be from these three classes: QWiget, QMainWindow, and QDialog. One is inherited, and an instance is generated in the main function and shown, and it enters the main message loop. The model layer mainly implements the business logic of the software. The modules here are divided into modules due to the business logic of the entire software to achieve a good combination effect. It makes a bridge class so that all business logic classes are instantiated in this bridge class.

The Communication layer is used to collect data. There are various communication methods, including serial ports, network ports, and CAN ports. The data received by this layer is sent to the models layer, and user input is also sent to the executive agency through this layer. By instantiating in the bridge class, the bridge class is actually a class that has no business functions, but only provides a carrier that the various classes can connect to each other signal slots. After designing all levels of code, and knowing in which class and level to implement, users need to use threads to make the entire software run efficiently. This is the running architecture. Generally speaking, all communication classes will be separated into a single thread. In this way, it will not block the GUI thread, and it will be able to respond faster to collecting information and passing control information to the lower computer.

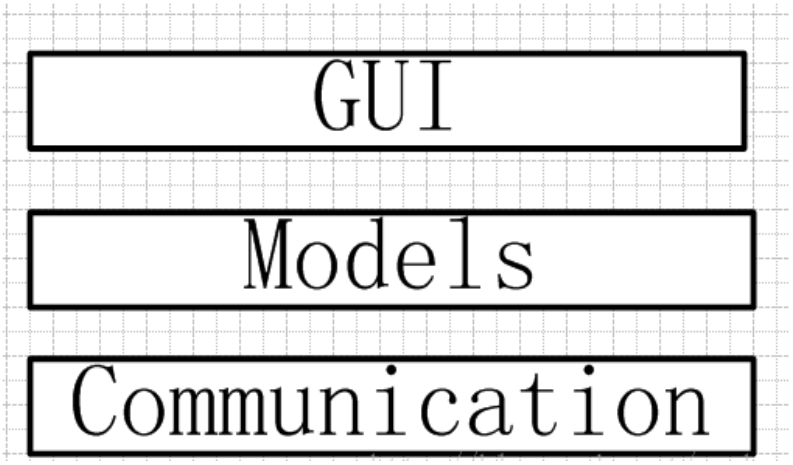


Figure 10. Basic Architecture of Qt Application (David 2012.)

The communication layer usually has several threads for several ports. Of course, some of the more lightweight that can be combined in one thread. The bridge class receives the GUI thread. When the interface pops up, it needs to get the latest value from the model and display it on the interface. It needs to call the get method, requirements are best to put all models in a thread-the GUI thread. In the figure 11, the main frame of such an information collection and control system is set up. Based on the composition and analysis of the translation dictionary software and achieving the desired functions, the plan is to divide the entire software composition into two major parts: the first part is the main user interface of the user login page, the design and word display of the main page of the translation software. The second part is to connect the click event of QPushButton to the on button clicked response function in the program through the unique signal slot mechanism in Qt. The translation function is implemented through the whole software, the function of deleting new words and viewing the new word book Functions.

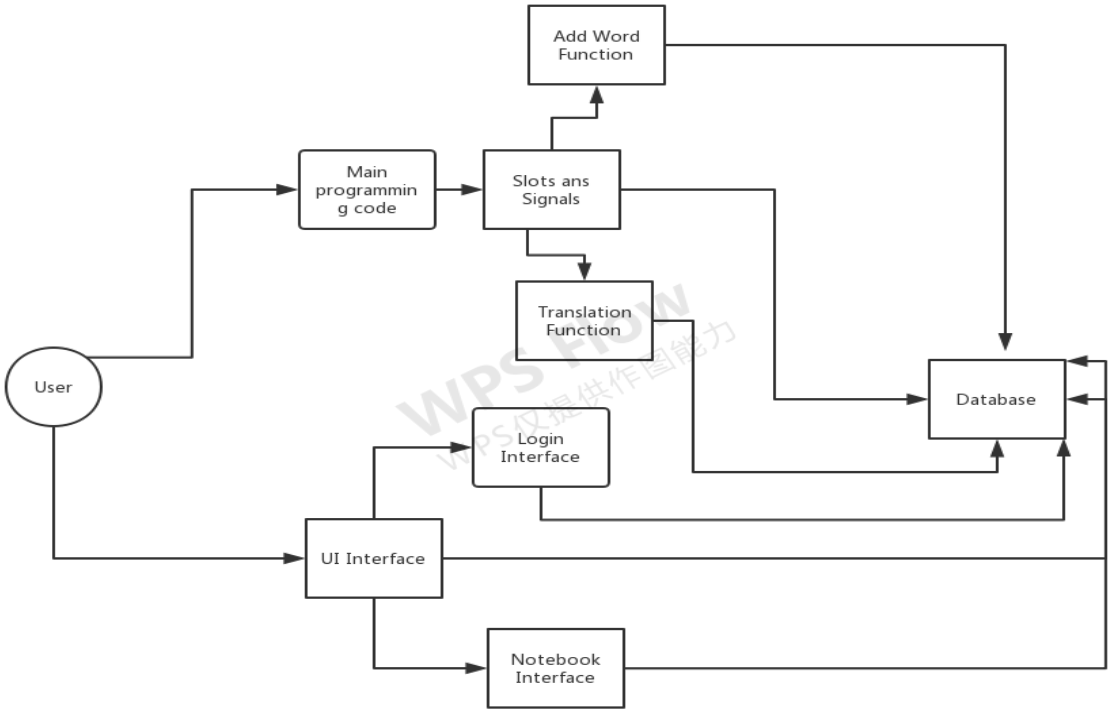


Figure 11. Program Case

3.2 GUI Design with Qt Designer

The application user interface was built by Qt designer. Qt Designer, also known as Qt Designer, is a comprehensive WYSIWYG GUI builder, and its designed user interface can be used on a variety of platforms. It is part of the Qt SDK and one of the most important development tools. With Qt Designer, people can drag and drop various Qt controls to construct a graphical user interface and preview the effect. (Jasmin 2006.) In this user login interface, in addition to the core Widget Box, Property Editor, and Object Inspector components. The main step is to create the interface window through the Widget, edit the page through the component, and use the property editor to change the window. Fill its body and properties of each control.

Parts category	ObjectName	Text(WindowTitle)
Widget	myForm	Layout example
Label	label_user	User name
Label	label_password	User password
LineEdit	lineEdit_name	None
LineEdit	lineEdit_password	None
Horizontal Spacer	horizontalSpacer	None
PushButton	pushButton_login	Login to system
PushButton	pushButton_exit	Exit system

Table 1. Widget property settings

In the table 1, the core content of the login page is to create signals and slots and connect to the platform. The connection configuration window for signals and slots can be found by clicking the already created login button (signals and slots inherited from Qwidget). It needs to select the clicked () signal of the button and the close () slot of myForm and then accept with the login button. It comes from the main body, through the link of signals and slots. The basic functions of the main interface design have been realized.

3.3 User Login

Before starting the design, the goal of the application software is to implement the login function and the password will appear as an encrypted display. The connection between the signal and the slot is used, and the macros required by the signal and the slot are used through Q_OBJECT. Explicit is usually used to prevent ambiguity, clicking the login button is the executed slot function. In the registered slot function, the "username" label, the "Password" tab, username edit line, password edit line, login button, and logout button are the carriers that form part of it. The content of the second part is to get the text of the

userNameLEd input box, it trimmed () removes the leading and trailing spaces, and it uses the tr() function to prevent garbled characters when setting Chinese. When the user enters the password and user name, the close the form and the set the return value Accepted.

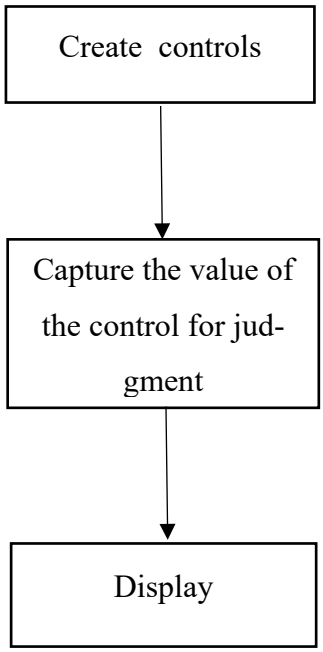


Figure 12. User login design (Matthias 2002)

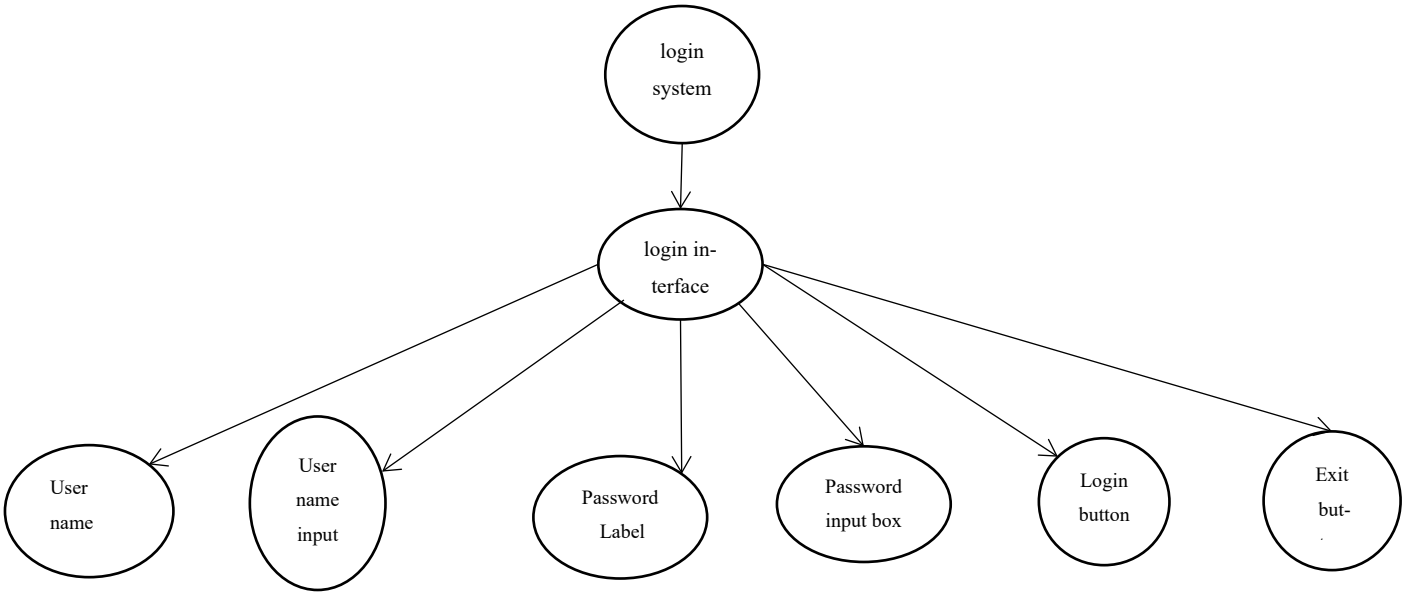


Figure 13. Login interface structure (Matthias 2002)

QMessageBox is later used to transmit the displayed information after login success or failure. The QMessageBox class in Qt provides a variety of commonly used dialog types, such as the warning dialogs here, as well as prompt dialogs and question dialogs. This uses a static function to set up a warning dialog, which is convenient. The parameters are: this indicates that the parent window is the login dialog; then the window title; then the content of the display; the last parameter is the button displayed, here a Yes button is used. Note that system also needs to add the header file for this class, namely: #include <QMessageBox>. At the same time, system needs to define a mainstream version that controls the file in the main.cpp file. It calls login.exec() to block the main control flow until the return is completed and it continues to execute the main control flow.

3.4 Function Realization

In order to implement the basic functions of translation, and the function of adding new words and viewing new word books. In the figure 14, the software needs to establish a database to retrieve and modify or view related information. Then it sends instructions to the related functions and implement them through the connection of signals and slots. It goes to the Qt button to trigger the action of the function, Qmessagebox prompts the related dialog box, and the code and interpretation uses it to read the database file into Json.

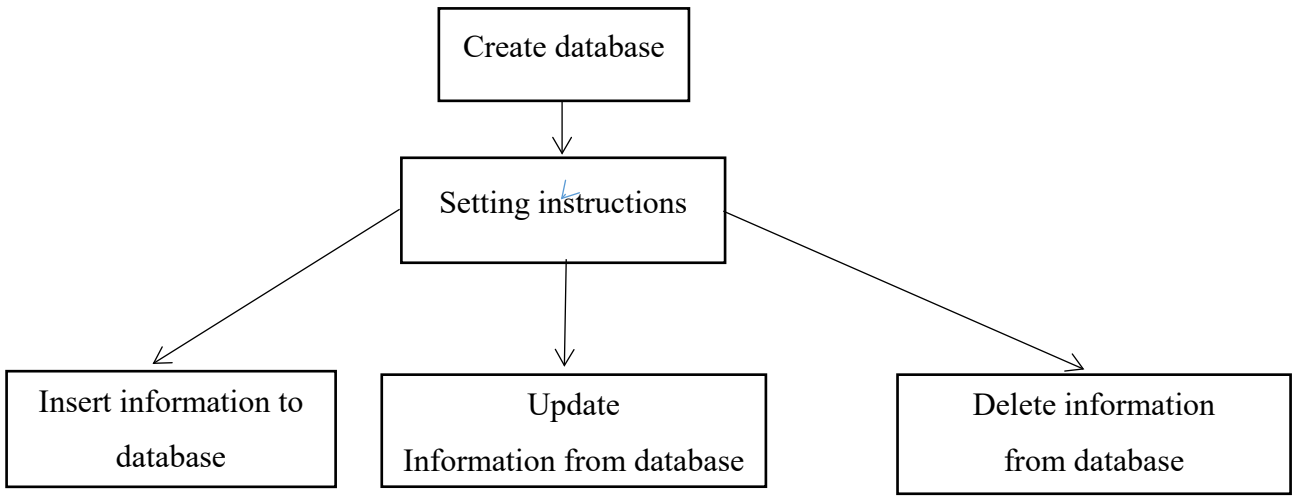


Figure 14. Structure of creating database

3.4.1 Create Database

To Begin with Introducing the SQL module. In the Qt project file (.pro file), add the SQL module: reference header file. In the class definition that requires SQL, reference the relevant header file. Then establish database Check connection, add database driver, and set database name. In the Figure 15, the aim is to create an object of QsqlDatabase, and define the name of the database and host. It uses open () to open the database and determine whether it is successful. Use the QSqlQuery class for database operations, which defines a QSqlQuery object, the auto statement is written directly in the parameter of the exec () function. Create table is the statement to create the table. The actual length of the varchar is changed. If sql_query.exec () is executed successfully, the table is created successfully.

```
CreateDb::CreateDb()
{

}

//initialized database
void CreateDb::initDB()
{
    //Get driver object
    QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");
    db.setHostName("HostName.db");//Set hostname
    db.setDatabaseName("DatabaseName");//Set the database name
    bool ok = db.open();
    if(ok)
    {
        qDebug()<<"Create DB";
```

Figure 15. Create Database (Screenshot from Qt Project)

```
bool CreateDb::insertWord(const QString &word, const QString &wordtr, const (
{
    QString order = QString("insert into notebook(word,wordtr,wordmean) value
            .arg(word).arg(wordtr).arg(wordmean);
    QSqlQuery m_query;
    bool success = m_query.exec(order);
    qDebug()<<"Add word information record:"<<order;
    if(success)
    {
        return true;
    }
    else
    {
        qDebug()<<m_query.lastError();
        return false;
    }
}
```

Figure 16. Insert information to the notebook database (Screenshot from Qt Project)

In the Figure 16, database information needs to insert the statement notebook (word, wordtr, wordmean) values ('% 1', '% 2', '% 3) into the previously created notebook table. This is to insert statement, notebook is the table name, and values () Is the data to be inserted. After the statement instruction is issued, it is written directly in the parameter of the exec () function in the form of qDebug. By following the same code function principle, using the statement UPDATE notebook SET wordtr = '% 1', wordmean = '% 2' WHERE word = '% 3.DELETE FROM notebook WHERE word ='% 1. The instruction can achieve the contents of the notebook database Update and delete.

3.4.2 Signals and slots between objects

In the figure 17, a signal is a specific identifier. A slot is a function (only various from general functions). A slot function can be associated with a signal and can also be called directly like a normal function. In fact, the signal and the groove are very simple to understand. people can understand it as track and field competition. When the signal gun fires, the athlete starts to run, and the signal gun fires is the action to signal. The athlete starts to run is the groove function people wrote. The relationship between signals and slots can be divided into three types: one signal is associated with one slot, one signal is associated

with multiple slots, and multiple signals are associated with one slot. The full name of MOC in Qt is Meta-Object Compiler, also known as "meta-object compiler". When compiling C ++ file, if the class declaration contains the macro Q_OBJECT, another C ++ source file will be generated, which is the moc_xxx.cpp file.

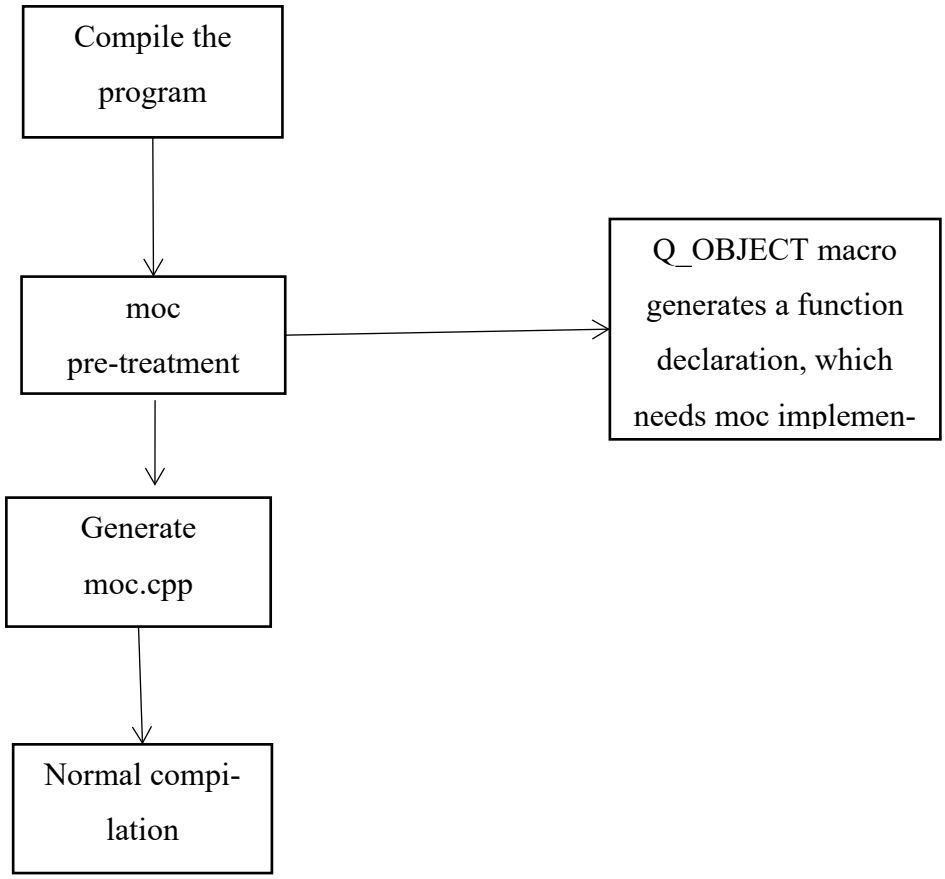


Figure 17. Execution process of MOC

Q_OBJECT is a very important macro. It is a key macro of Qt's meta compilation system. After this macro is expanded, it contains code that Qt helped people to write, including variable definitions and function declarations.

```
//Connect the signal to the slot
connect(manager,&QNetworkAccessManager::finished,this,[this](QNetworkRep
    reply->attribute(QNetworkRequest::HttpStatusCodeAttribute);
    reply->attribute(QNetworkRequest::RedirectionTargetAttribute);
```

Figure 18. Connect function(Screenshot from Qt project)

In the figure 19, when the connect function is executed, a Connection object is constructed and then stored in the sender's memory. The moc precompilation helps people to construct the beginning (signal function body) and end (qt_static_metacall callback function) of the signal slot callback. The intermediate callback process Qt has been implemented in the QOjbect function. Signals and slots are for the convenience of MOC to parse C ++ file and parse out signals and slots. The signal slot has a total of 5 connection modes. The first four are mutually exclusive and can be expressed as asynchronous and synchronous. The fifth unique connection uses in conjunction with the first four methods. Signals and slots are essentially the same, but for users, signals only need to be declared, MOC helps people implement them, slot function declarations and implementations need to be written by people. The connect method is to store the sender, signal, receiver, and slot for subsequent search when the signal is executed. A signal trigger is a series of function callbacks.

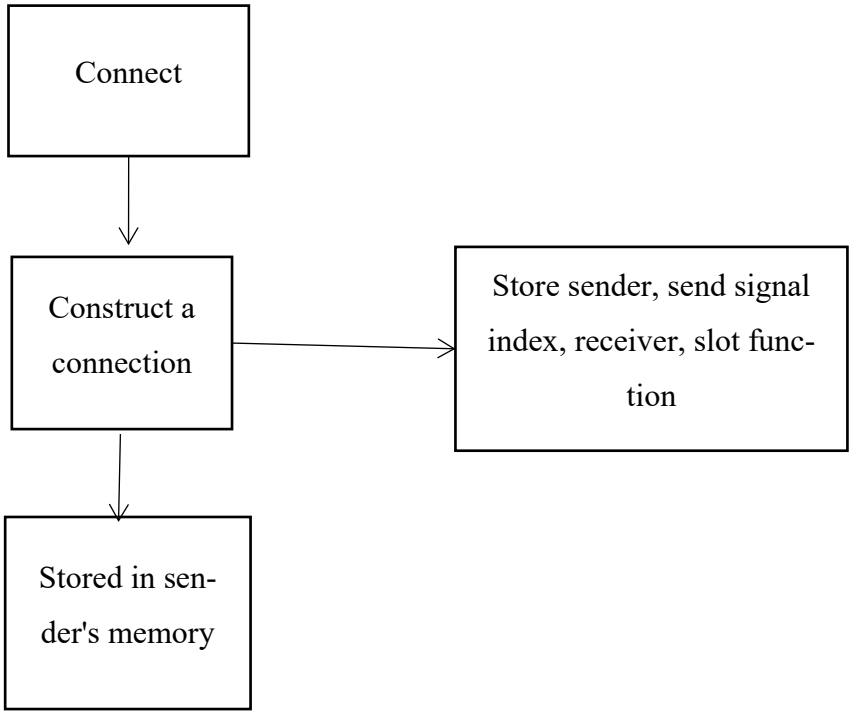


Figure 19. Connect structure

3.4.3 Word explanation function

In Qt, signals and slots are the most important part of the entire qt composition. It is similar to the bridge to establish a connection. people want to get the response of the instruction through the signal and the slot. people need to use the get method of qt. In general, the function is implemented through QNetworkAccessManager, QNetworkRequest and QNetworkReply. In the figure 20, here gives the official function and definition of them.

QNetworkAccessManager	Allows the application to send network requests and receive replie
QNetworkRequest	Holds a request to be sent with QNetworkAccessManager
QNetworkReply	Contains the data and headers for a request sent with QNetworkAccessManager

Figure 20. Definition of QNetworkfunctions (David 2012.)

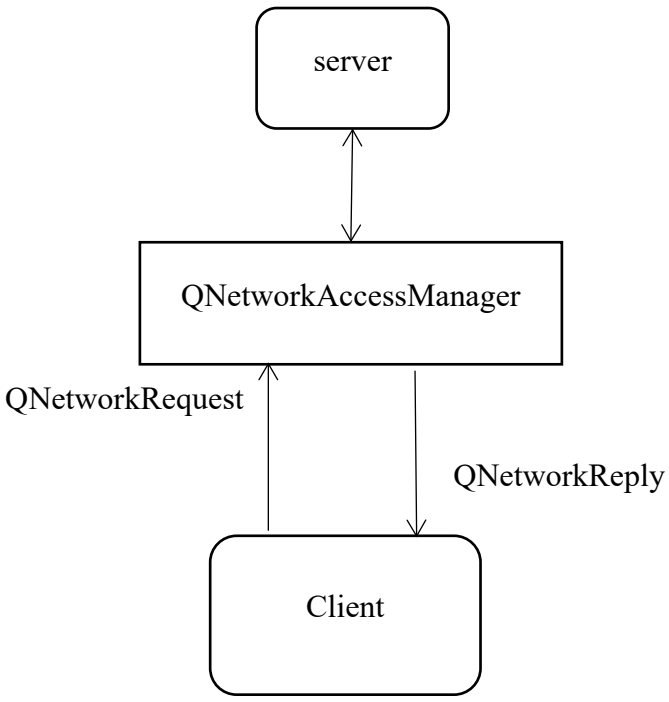


Figure 21. Relationship between Qnetwork functions

In the Figure 21, QNetworkRequest sets the parameters of a network request. QNetworkReply is used to receive data returned by the server. The QNetworkAccessManager acts as an intermediary, setting up

post, get, and other methods to send requests, and then monitoring whether the server returns data (stored in QNetworkReply), and delegates the returned reply to a slot for processing. When making a network request, there are some steps to do: generate a QNetworkRequest object, set the request header, the actual parameters required by the API used for the request, URL and other information. Prepare to parse and process the slot of the QNetworkReply; generate a QNetworkAccessManager, send the request, and return it Reply is delegated to the slot processing. (implementation is a connect statement)

In the figure 22, by QJsonDocument Json, QJsonObject and QJsonArray to achieve the function of reading database content and translating words. In the figure 23, QJsonDocument provides a way to read and write Json documents. QJsonDocument is a class that contains a complete JSON document that supports reading and writing JSON documents in UTF-8 encoded text and QT's own binary format. Regardless of the parameter information required to set the request or the reply that is returned, the data is expressed in the form of QJson. Therefore, to achieve network requests, the generation and analysis of QJson is important. The text-based representation should refer to the form ["Qt ”,” Version ”, true], and the whole is a JSON document based on text-based representation. This whole can be stored in QByteArray in Qt. QJsonDocument is a tool for reading and writing JSON documents in Qt. This corresponds to the concept of arrays and objects in Json. Correspondingly, Qt provides two classes related to objects and arrays: QJsonObject and QJsonArray.

```
QJsonDocument json;
QJsonObject middle, basic0;

QByteArray byte=reply->readAll();//read All Data
qDebug()<<byte;
//Convert to json Document format
json = QJsonDocument::fromJson(byte);

if (!json.isNull())
{
    middle = json.object();
```

Figure 22. Json Convert (Screenshot from Qt Project)

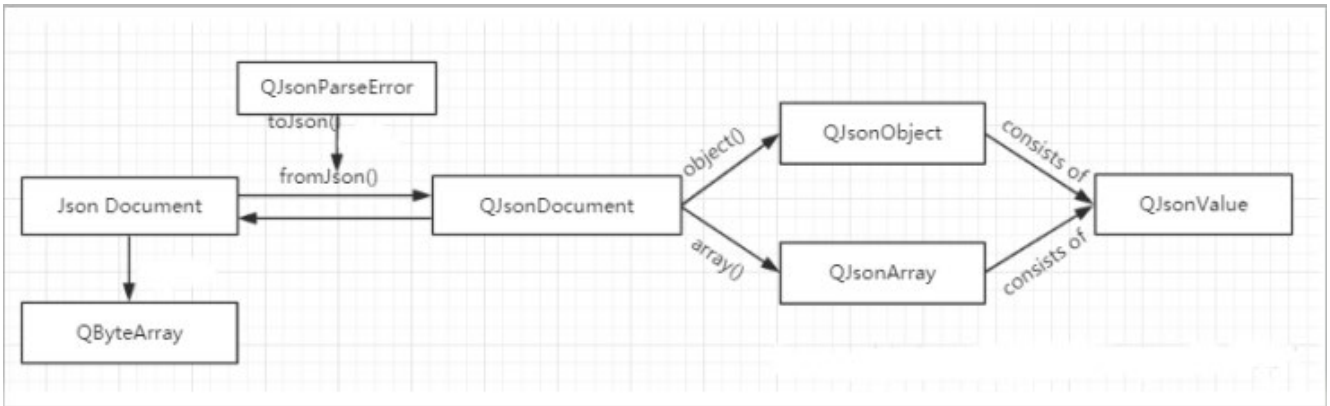


Figure 23. Json class relationship

3.4.4 Add word Function with QMessageBox and QPushButton

The QMessageBox class provides a modal dialog for notifying users or asking users questions and receiving answers. The message box displays the main text to remind the user of the situation, the informative text to further explain the alert or ask the user a question, and optional details text to provide more data when requested by the user. The message box can also display icons and standard buttons for accepting user responses.

```
//Add word function
void Dialog::on_addpushButton_clicked()
{
    if(this->ui->InputtextEdit->toPlainText().isEmpty()||this->ui->OutputEdit
    {
        QMessageBox::information(this,"Add the error","Words that cannot be a
        return ;
    }
    //Get information about the current translation of new words
    auto word=this->ui->InputtextEdit->toPlainText();
    auto wordtr=this->ui->OutputEdit->toPlainText();
    auto wordmean=this->ui->analysistextEdit->toPlainText();

    //Add to the database
    if(mydb.insertWord(word,wordtr,wordmean))
    {
        QMessageBox::information(this,"successfully added","Add new words to
    }else
    {
        QMessageBox::information(this,"fail to add ","Database entry failed")
    }
}
```

Figure 24. Messages with add words success or fail (Screenshot from Qt Project)

In figure 24, it describes a dialog box for information prompts through function calls to determine whether a new word has been successfully added to the database. The QPushButton component is used to accept user click events, it can display prompt strings, it is a functional component, requires the parent component as a container, it can be positioned in the parent component, it uses to execute commands or trigger events. It is also a command button. Click on it to perform some action or respond to some questions. Whenever an instruction is issued, it triggers the previous step of the corresponding function implementation. In the figure 25, the main interface button slot function analysis, add a word function to determine whether there is a message box prompt after empty. Get information about the current translation of the new word, add it to the database. Then view the current word book, get the word, query the word details and start to initialize the display dialog box. This will get updated data and update the database.

```
void Dialog::on_TrpushButton_clicked()
```

Figure 25. Trigger Button (Screenshot from Qt Project)

3.4.5 Qt event programming

Events are issued by the system or the QT platform itself at different times. When the user presses the mouse, hits the keyboard, or when the window needs to be redrawn, a corresponding event is emitted. Some events are emitted in response to user actions, such as keyboard events; others are automatically issued by the system. In the figure 24, for example, for QPushButton mouse clicks, events won't need to care about this mouse click event, but about the clicked () signal. But events and signal slots in QT are not interchangeable. The signal is emitted by a specific object. Once the signal is emitted, it will be immediately handed over to the slot connected by the connect () function for processing. For events, QT uses an event queue to maintain all emitted events. When new events occur information will be appended to the end of the event queue. After the previous event is completed, the subsequent events are taken out for processing, and QT events can also be directly processed without entering the event queue. Throughout the entire program, log in to the button slot, close the button slot, translate the button slot, and add a new word button slot to the pronunciation button slot. Check the current word list button slot. qt clicked triggers events and implements corresponding functions through the link of signals and slots.

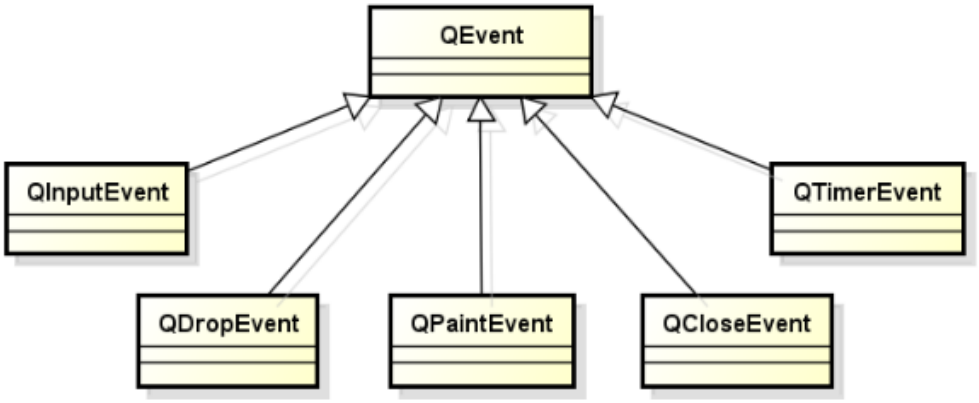
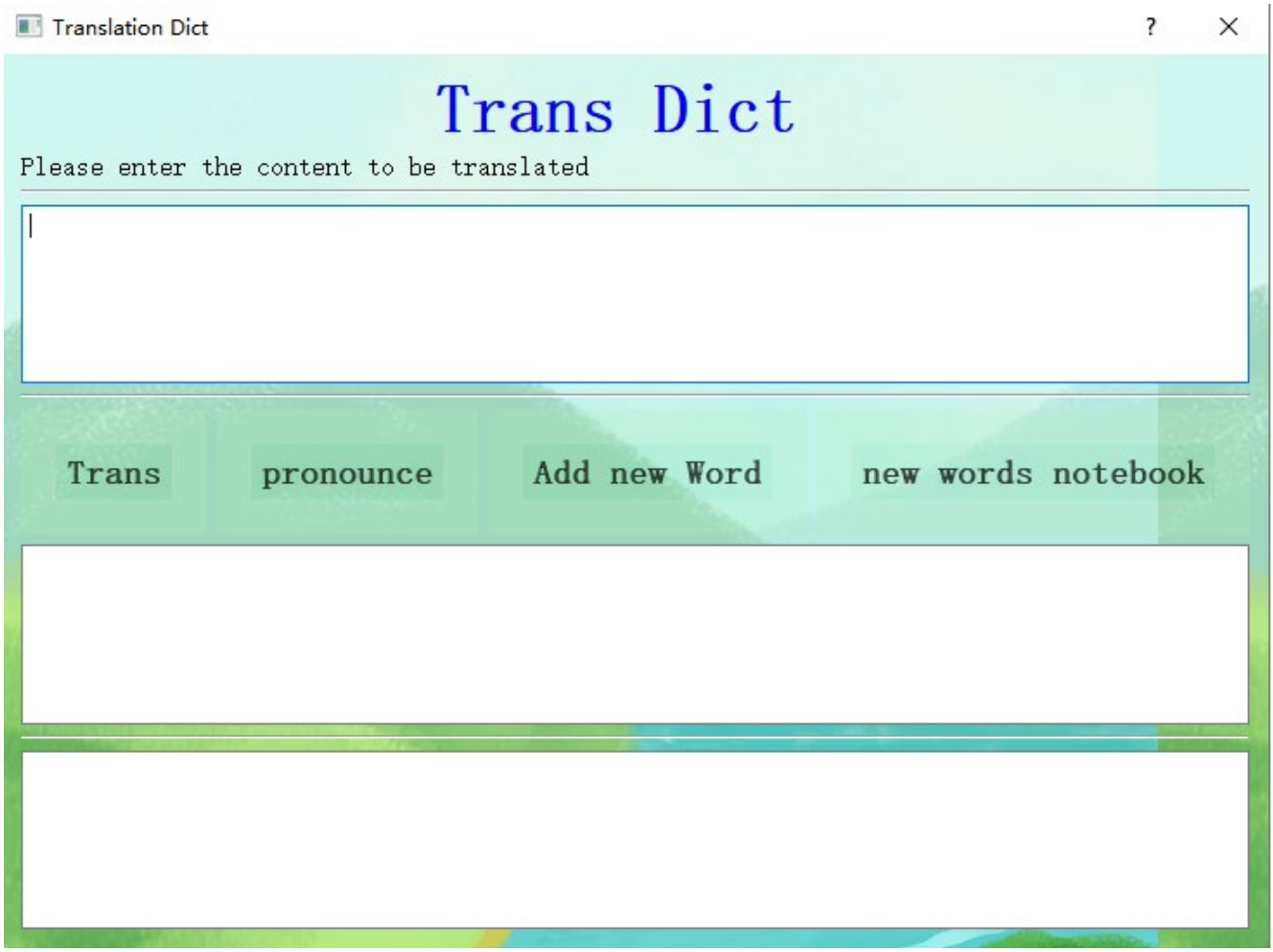


Figure 26. Qt event programming structure

4 FINAL APPLICATION

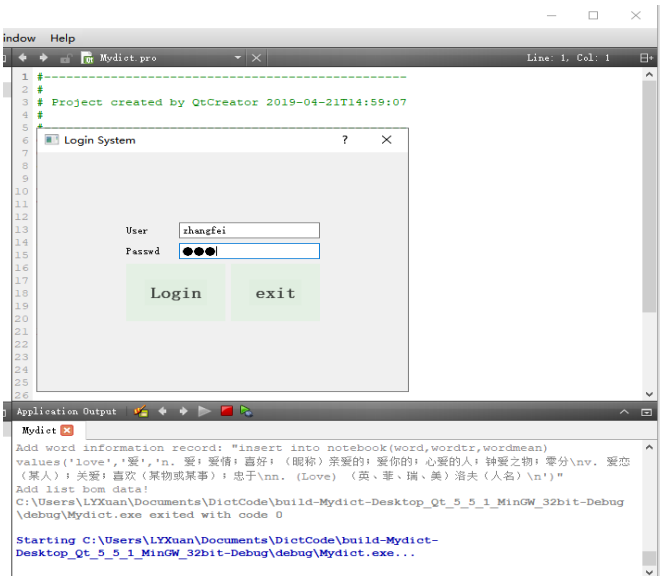
The Final application has been made successfully. It has been tested with windows 10 platform using Qt crater. The main project is based on main translation interface, user login interface and notebook interface. In the main interface, there are 4 functions that have been achieved. Translation, pronunciation, adding new book and checking notebook.



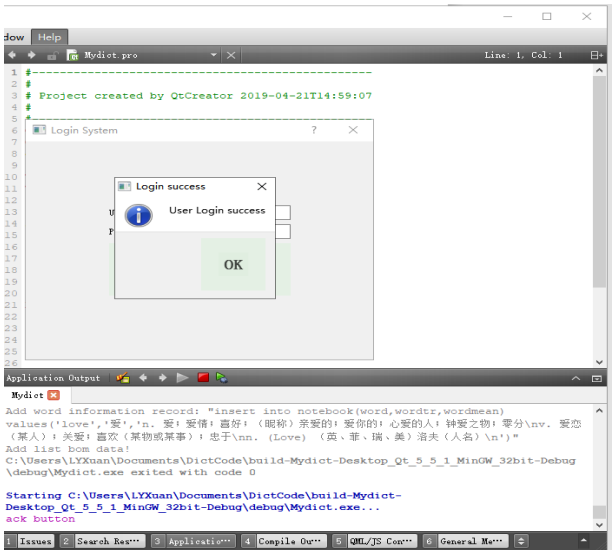
Picture 1. Main Interface translation

4.1 Login Test

As can been seen in Pictures 2 and 3, they show that document data comes to Qt with correct username and password which made login success. User login module: On the homepage login page, users fill in their login account and password. If the information is correct, they can enter the system for operation. Logout: After entering the system, the user can click the logout button in the upper right corner of the interface to log out of the system.



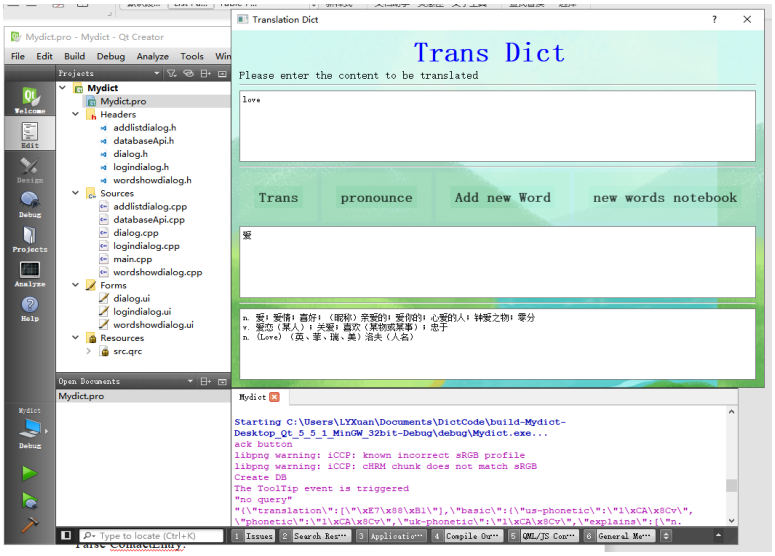
Picture 2. Login Interface



Picture 3. User login successfully

4.2 Translation Function Test

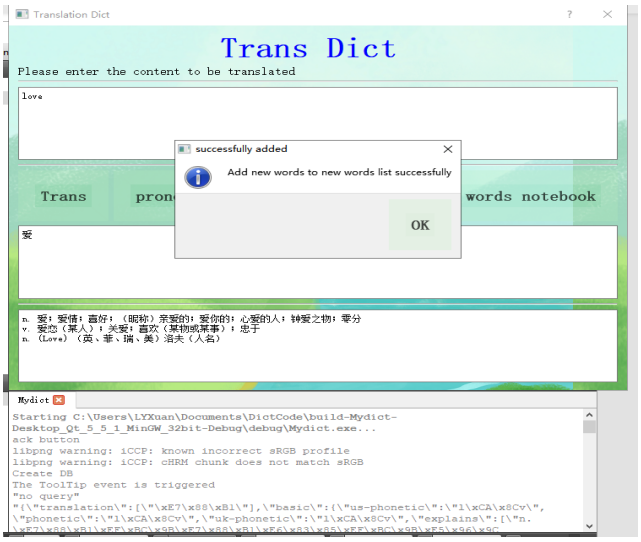
As Picture 4 shows, when command request to take related translation meaning of word. It successfully appeared from database. After inputting the translation, the user can use the language to be translated, and click the translation button to perform the automatic translation language function. Word translation is a direct translation of words, and other interpretations will also appear. Sentence translation can translate long sentences.



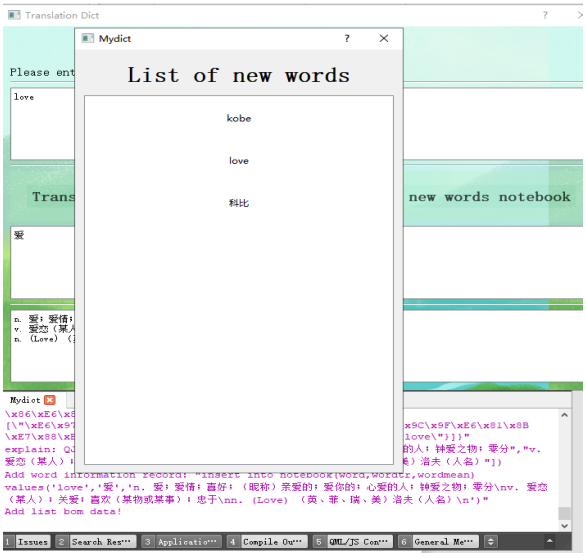
Picture 4 .Word Translated Successfully (screenshot from Qt project)

4.3 Notebook adding Test

From Pictures 5 and 6, it shows that the result of adding word notebook seemed to work well. By reading the list of words users can add and delete, which means the word translation will be more open to accept. Every user has the right to change the mistakes and improve with new vocabularies. That is the point makes users to have a totally different experience, which triggers the application meaningful.



Picture 5. Notebook Interface



Picture 6. Result of adding successfully to notebook

5 CONCLUSIONS

The idea of making a translation application is to create a new model of a digital dictionary. Basically, all the functions have been realized except pronunciation. It seemed there are some mistakes when creating request from pronunciation related API. Users can login with their account and add their favorite words, users can even change the word whey want. But in general, this is just a basic function translation application, there are many functions that can be added to it and make it become more useful. In this Qt production program, the user login, the connection with the new word database, the addition and removal of the new word information, the wording function, and the Chinese and English translation of words and sentences were achieved.

But at the same time, there are also many problems that can be solved in later development. For example, users can increase the connection web page query, so that the information found will be more comprehensive, detailed and authoritative. And users can add the relevant usage of each part of the word. At the same time, the design of UI is a bit simplistic, users can add the function of the custom UI interface. More importantly, adding the ability to retain the query history of the word will give the user a better experience and is more convenient. Just like many search engines, hopefully that the translation software will have relevant vocabulary when people input a few letters and sort according to the rating of the search. The addition of the final phonetic symbol will be a finishing touch, because most Chinese people learn to read English from the phonetic symbols. In general, there are still many to improve in infields, but with current level of knowledge and ability.

REFERENCES

Blanchette, J.& Summerfield, M. 2006**.** C++ GUI Programming with Qt 4. Pearson:Prentice Hall Professional.

Piccolino, M. 2018. Qt 5 Projects. Birmingham: Packt Publishing Ltd.

Fitzek, F. & Mikkonen, T. & Torp ,T. 2010. Qt for Symbian. Willey:John Wiley & Sons.

Dalheimer, M. 2002. Programming with Qt. Sebastopol :O'Reilly Media.

Lazer, G.& Penea,P.2018. Mastering Qt 5 .Birmingham: Packt Publishing Ltd.

Summerfield, M. 2010. Advanced Qt Programming. New Jersey :Pearson Education.

Baka, B. 2019.Getting Started with Qt 5 .Birmingham: Packt Publishing Ltd.

Weller, J . 2015. Re-factoring my Qt database code. Available at:https://www.meetingcpp.com/blog/items/refactoring-my-qt-database-code.html. Accessed:14 March 2015.

Qt: the C++ framework for developing cross-platform software. Article on the website of 1&1 IONOS Inc. https://www.ionos.com/digitalguide/server/know-how/qt/. Accessed:23 January 2019.

Admin,T .2016. Qt Tutorials For Beginners – Creating Simple Login Form in QT. Available：http://www.codebind.com/cpp-tutorial/qt-tutorial/qt-tutorials-beginners-creating-simple-login-form-qt/. Accessed:13 June 2016.

Fitzpatrick, M . 2019. Actions — Tool-bars and Menus.Available:https://www.learnpyqt.com/courses/start/actions-toolbars-menus/. Accessed:15 April 2019.