Marathwada Shikshan Prasarak Mandal’s

**Deogiri Institute of Engineering and Management Studies,**

**Aurangabad**

**Report**

**On**

**Qt Designer**

Submitted By

**Tipare Bhagyashree Pravin**

**Dr. Babasaheb Ambedkar Technological University**

**Lonere (M.S.)**



Department of Computer Science and Engineering

**Deogiri Institute of Engineering and Management Studies,**

**Aurangabad**

(2019- 2020)

**Seminar Report**

**On**

**Qt Designer**

Submitted By

**Tipare Bhagyashree Pravin(36165)**

**In partial fulfillment of**

**Bachelor of Technology**

**(Computer Science & Engineering)**

Guided By

**S.B.Kalyankar**

Department of Computer Science & Engineering

**Deogiri Institute of Engineering and Management Studies,**

**Aurangabad**

(2019- 2020)

**CERTIFICATE**

This is to certify that, the Seminar entitled “**Smart Qt Designer**” submitted by **Tipare Bhagyashree Pravin** is a bonafide work completed under my supervision and guidance in partial fulfillment for award of Bachelor of Technology (Computer Science and Engineering) Degree of Dr. BabasahebAmbedkar Technological University, Lonere.

Place: Aurangabad

Date:14-10-19

**Mr S.B.Kalyankar Mr. S.B. Kalyankar**

**Guide Head**

**Dr. Ulhas D. Shiurkar**

**Director,**

**Deogiri Institute of Engineering and Management Studies,**

**Aurangabad**

Abstract

Qt Designer is the Qt tool for designing and building graphical user interfaces (GUIs) with [Qt Widgets](https://doc.qt.io/qt-5/qtwidgets-index.html). You can compose and customize your windows or dialogs in a what-you-see-is-what-you-get (WYSIWYG) manner, and test them using different styles and resolutions.

Widgets and forms created with Qt Designer integrate seamlessly with programmed code, using Qt's signals and slots mechanism, so that you can easily assign behavior to graphical elements. All properties set in Qt Designer can be changed dynamically within the code. Furthermore, features like widget promotion and custom plugins allow you to use your own components with Qt Designer.

You have the option of using [Qt Quick](https://doc.qt.io/qt-5/qtquick-index.html) for user interface design rather than widgets. It is a much easier way to write many kinds of applications. It enables a completely customizable appearance, touch-reactive elements, and smooth animated transitions, backed up by the power of OpenGL graphics acceleration.

Qt Creator includes a code editor and integrates Qt Designer for designing and building graphical user interfaces (GUIs) from Qt widgets.

The code editor in Qt Creator supports syntax highlighting for various languages. In addition to that, the code editor can parse code in C++,python and [QML](https://en.wikipedia.org/wiki/Qt_Meta_Language) languages and as a result code completion, context-sensitive help, semantic navigation are provided.

Qt Designer is a tool for designing and building graphical user interfaces (GUIs) from Qt widgets. It is possible to compose and customize the widgets or dialogs and test them using different styles and resolutions directly in the editor. Widgets and forms created with Qt Designer are integrated with programmed code, using the Qt signals and slots mechanism.

Qt Quick Designer is a tool for developing animations by using a declarative programming language [QML](https://en.wikipedia.org/wiki/Qt_Meta_Language).

Qt Creator provides support for building and running [Qt](https://en.wikipedia.org/wiki/Qt_(framework)) applications for desktop environments (Windows, Linux, FreeBSD and Mac OS), mobile devices ([Android](https://en.wikipedia.org/wiki/Android_(operating_system)), [BlackBerry](https://en.wikipedia.org/wiki/BlackBerry), [iOS](https://en.wikipedia.org/wiki/IOS" \o "IOS)) and embedded Linux devices. Build settings allow the user to switch between build targets, different Qt versions and build configurations. For mobile device targets, Qt Creator can generate an installation package, install it to a mobile device that is attached to the development computer and run it there.



###### Contents

**List of Figures**

1. **INTRODUCTION**
2. **HISTORY**
3. **PURPOSE AND ABILITY**
4. **LITERATURE SURVEY**
5. **CONCEPT OF SCRIPTING ENGINE**
6. **APPLICATION OF THE RESULTING APPROACH**
7. **IMAGES**
8. **USING CONTAINERS IN QT DESIGNER**
9. **CONCLUSION**

**REFERENCES**

**ACKNOWLEDGEMENT**

1. INTRODUCTION

With the advent of many programming languages, the programmer got the wide variety of choices for developing their applications. In the past, the entire software was written in high performance programming language such as C/C++. No matter, it provides an excellent performance as the code is precompiled; it becomes very difficult to program the application. High performance programming languages are very difficult to master in. On the other hand, scripting languages do not provide a high performance as, the conversion to machine code is on-fly, it is very easy to write an application in scripting language compared to compiler-based language. Also, programming in scripting language enables rapid development of an application. Thus, either the application developed will have low performance, or it will be very difficult to program or extend. There are many frameworks available in market, with Qt being used widely in industrial software development. Qt is a cross-platform application development framework, developed by Digia and Qt Projects (currently maintained by the Qt company), used for developing applications that can be run on various software and hardware platforms. It is available with both commercial and open source licence. The main development language for Qt is C++ and it supports GCC C++ compiler and MSVC. There are many third-party scripting extensions for Qt which enables extending the Qt application using scripts. The language includes but not limited to JavaScript, Lua, Python and Ruby. In order to balance the high performance provided languages such as C++/C and ease of programming provided by scripting language, the concept of scripting engine is proposed. The scripting engine is a tool/library, which is capable of loading, compiling and running script code. It can be used in an application to handle all the nonperformance critical and user-interface events. The scripting engine will enable to develop or extend an existing application easily using scripts, without hampering the performance of overall application. The further chapters explains the proposed concept of scripting engine as well as the comparative study of various scripting extensions available in Qt, which enable integrated development using high performance programming language(C++) and scripts.

1. HISTORY

Qt is a free and open-source widget toolkit for creating graphical user interfaces as well as cross-platform applications that run on various software and hardware platforms such as Linux, Windows, macOS, Android or embedded systems with little or no change in the underlying codebase while still being a native application with native capabilities and speed. Qt is currently being developed by The Qt Company, a publicly listed company, and the Qt Project under open-source governance, involving individual developers and organizations working to advance Qt. Qt is available under both commercial licenses[4] and open source GPL 2.0, GPL 3.0, and LGPL 3.0 licenses.

1. PURPOSE AND ABILITY

Qt is used for developing [graphical user interfaces](https://en.wikipedia.org/wiki/Graphical_user_interface) (GUIs) and multi-platform [applications](https://en.wikipedia.org/wiki/Application_software) that run on all major desktop platforms and most mobile or embedded platforms. 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).[[14]](https://en.wikipedia.org/wiki/Qt_(software)#cite_note-14)

Qt supports various compilers, including the [GCC](https://en.wikipedia.org/wiki/GNU_Compiler_Collection) C++ compiler and the [Visual Studio](https://en.wikipedia.org/wiki/Microsoft_Visual_Studio) suite 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
* network support.

1. LITERATURE REVIEW

The Qt application can be extended by scripting language using scripting modules provided by The Qt Company or the scripting extensions provided by thirdparty. The scripting languages for which, the scripting extension is available for Qt are JavaScript, Lua, Python and Ruby. The extensions for JavaScript are QtScript and QJSEngine , both provided by Qt framework. The extensions for Lua in Qt are QtLua by Savannah projects and Lqt . The extensions for Python in Qt are PyQt by Riverbank Computing , Pyside by Nokia , Pyotherside and PythonQt . The extensions for Ruby in Qt are QtRuby by Korundum and Ruby-QML by seanchas116 . Thus, with any of the extensions available, scripting engine can be developed, where the core part of the engine is written in C++ and non-performance critical part using scripts or the extension of the application in scripting language which can interact with the user interface.

The comparative study based on the requirement of product of each of the extensions available is as follows:

Advantages of the resulting approach

* Enables easy extension of the application using script. Thus, the core application can be extended using the scripting language easily without any significant performance difference compared to an application developed purely in python,handling performance critical task at python library level.
* No type-casting, memory management and complex syntax for extending an application.
* Automation of repetitive task based on the event generation at user-interface level or at library level.
* The application can be easily customised using the scripting engine, if the user interface and the interaction are managed by scripts.
* Rapid prototyping of an application

The comparative study based on the requirement of product of each of the extensions available is as follows:

A. Ease of using The scripting language to be used in scripting engine should be easy to use, as the purpose of its introduction in application is to extend the core part of library using script for rapid development. If an easy to learn and readable scripting language is selected, then the application can be extended easily by anyone, including a non-programmer or the person with least expertise in programming. Considering JavaScript, Lua, Python and Ruby, all are easy to use compared to high performance language like C/C++. Considering the readability of language, Lua and Python are readable. Ruby is bit confusing, as in Ruby there are more than one way of doing things in Ruby. eg. Conditional statements, aliasing methods and opposite methods .

B. Support provided for extensions As some of the extensions provided for scripting are third party extensions, support provided by this third party organization plans a vital role in development of this project. Support includes the forums and mailing list which addresses the issues related to these extensions. QtScript and QJSEngine are the part of Qt framework, thus all forums related to Qt addresses the issues. The forums/mailing list for QtScript and QJSEngine are . For Python, the extension PyQt has maximum support such as . PySide has a mailing list which addresses PySide related questions . QtLua, lqt and QtRuby

C. Licensing Licensing plays an important role, when the aim is to develop a proprietary application. QtScript and QJSEngine, both are licensed under GNU Library General Public License (LGPL). PyQt is available with both commercial license and GNU General Public License (GPL) with no functional difference between them. Informal support is provided for commercial license holder. PySide and PythonQt licensed under LGPL version 2.1 license. Pyotherside is licensed under ISC/BSD license. QtLua is licensed under LGPL version 3 license and Lqt is licensed under MIT(X11) license .QtRuby is licensed under LGPL version 2.1 license [.Ruby-QML is licensed under MIT license .

D. Roadmap to support future versions The extension should provide support to the recent versions of Qt and should support the maximum features of Qt. Considering QtScript and QJSEngine, both are the part of Qt framework. QtScript, which is based on JavaScriptCore engine, has been deprecated since Qt 4.7, since; it failed to give high performance as JavaScript V8 engine. Thus, QJSEngine was introduced, which is based on JavaScript V8 engine. It is much faster and present in Qt QML module, has better integration with C++ (and QML) and is more compliant with ECMA standard .But, it is still in developing phase and lacks features which are present in QtScript such as instantiating QObject from JavaScript, exposing individual native function to JavaScript and debugging API. PyQt supports Qt4 and Qt5.4.x. The support for Qt 5.5 is still not available . PySide supports only Qt4, there is no official support for Qt5 . Pyotherside currently supports Qt5 and is actively maintained one. PythonQt currently supports Qt4 and Qt5. It is actively maintained are developers are working to provide support for Qt 5.5 .The QtLua supports Qt 5.1, 5.2, 5.3, 5.4.x. The support for Qt 5.5 is not official . Lqt project is no longer maintained by the author and the existing Lqt contains many bugs and limited features. Thus, Lqt's support for Qt is stranded to Qt 4.7. QtRuby is no longer maintained by its developers and the existence QtRuby library supports only Qt 4.x. Ruby-QML provides binding between Ruby and QML. It currently provides support up to Qt 5.4 and is actively maintained [23].

E. Platform Supported The product should support multiple platforms, as it can be used by anyone on the desired platforms. The table below shows the platform supported by each of the extensions.

1. CONCEPT OF SCRIPTING ENGINE

The concept of scripting engine revolves around scripting languages and high performance programming language.High performance programming languages like C/C++ are performance efficient whereas scripting languages are easy to use, can achieve maximum functionality with minimum lines of code and provide the programmer the advantage of automatic memory management and typecasting. But scripting languages are weak in terms of performance, compared to programming languages such as C/C++. In C/C++, the code is compiled to native code and there is more control over memory layout, which is not possible using scripting language. And thus, scripting language consumes more CPU cycles which may compromise the overall performance of an application. An application developed using C++ programming language and scripting language .

Application written in C/C++ is first compiled which can be run any number of time, on any platform, thus providing high performance. Applications written in scripting language are loaded and interpreted on-fly at runtime by scripting engine and thus are not performance efficient. The concept of scripting engine is to develop the part of an application which is performance critical, in high performance language and to manage the non-performance critical code, User-interface components and interaction in scripting language. The performance critical library can be exposed to script-based application, thus, it provides an advantage of accessing the code with minimum scripts and the core library can be modified independently, without affecting the interaction and GUI module. Thus, the resulting application will give almost equal performance compared to the application developed in C/C++ and enables rapid development.

1. APPLICATION OF THE RESULTING APPROACH

The resulting scripting engine can be used in various approaches, such as:

* The approach can be used for rapid development of embedded systems, where the engineer can use the resulting scripting engine for generating panel user-interface with ease. The development of user interface will be independent of the changes or modification in the core libraries.
* Development of games, where the common actions based on events are defined in the core libraries and the UI can be developed rapidly using those libraries through script, thus, automating the action of tiles in the game.
* Development of User Interface of Direct TV, where the scripts can be defined to load the functions in library based on the event from user.
* Development of applications, where the development time is critical and the user interface developer has to experiment with the designs.

1. IMAGES

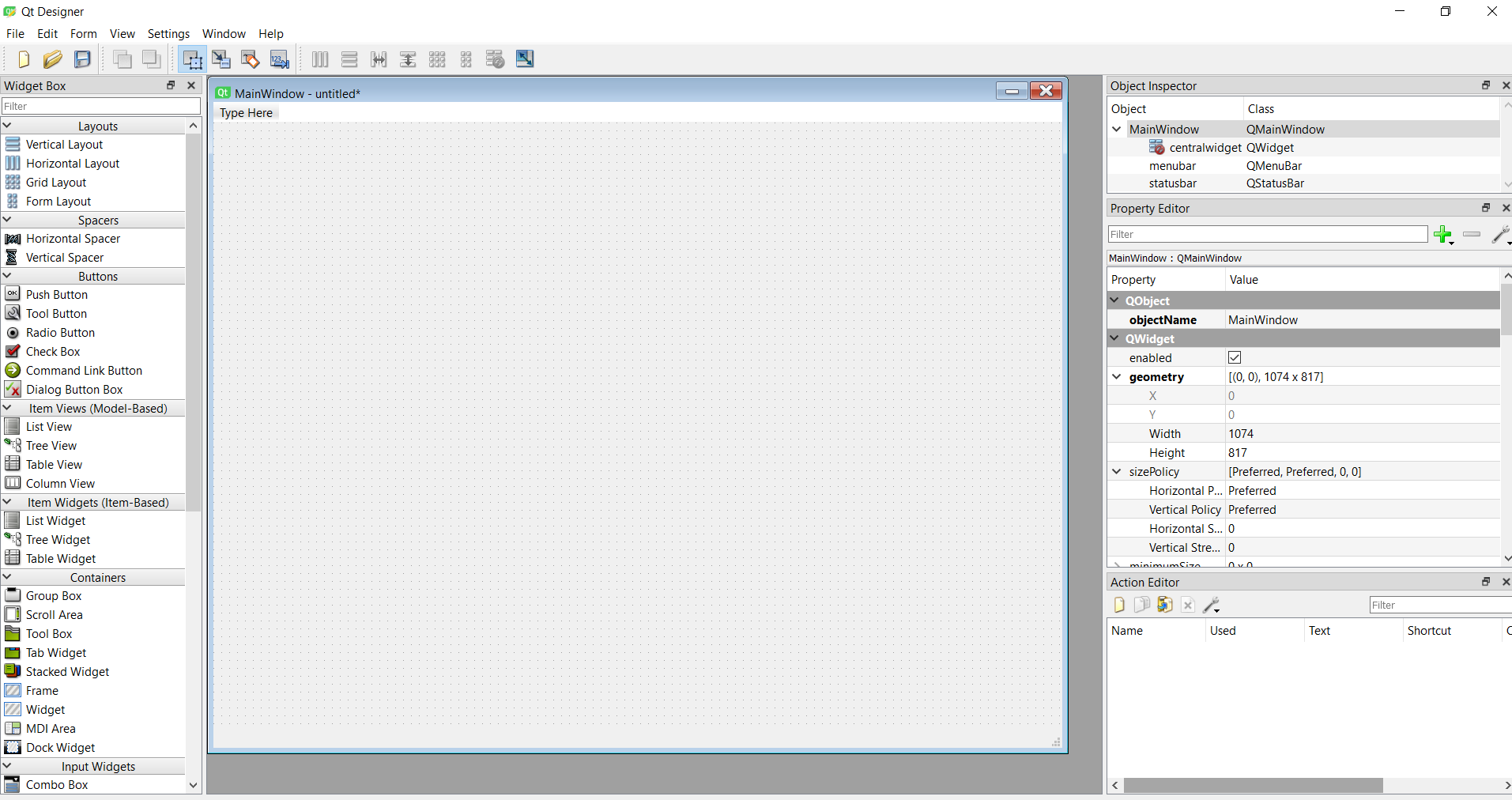
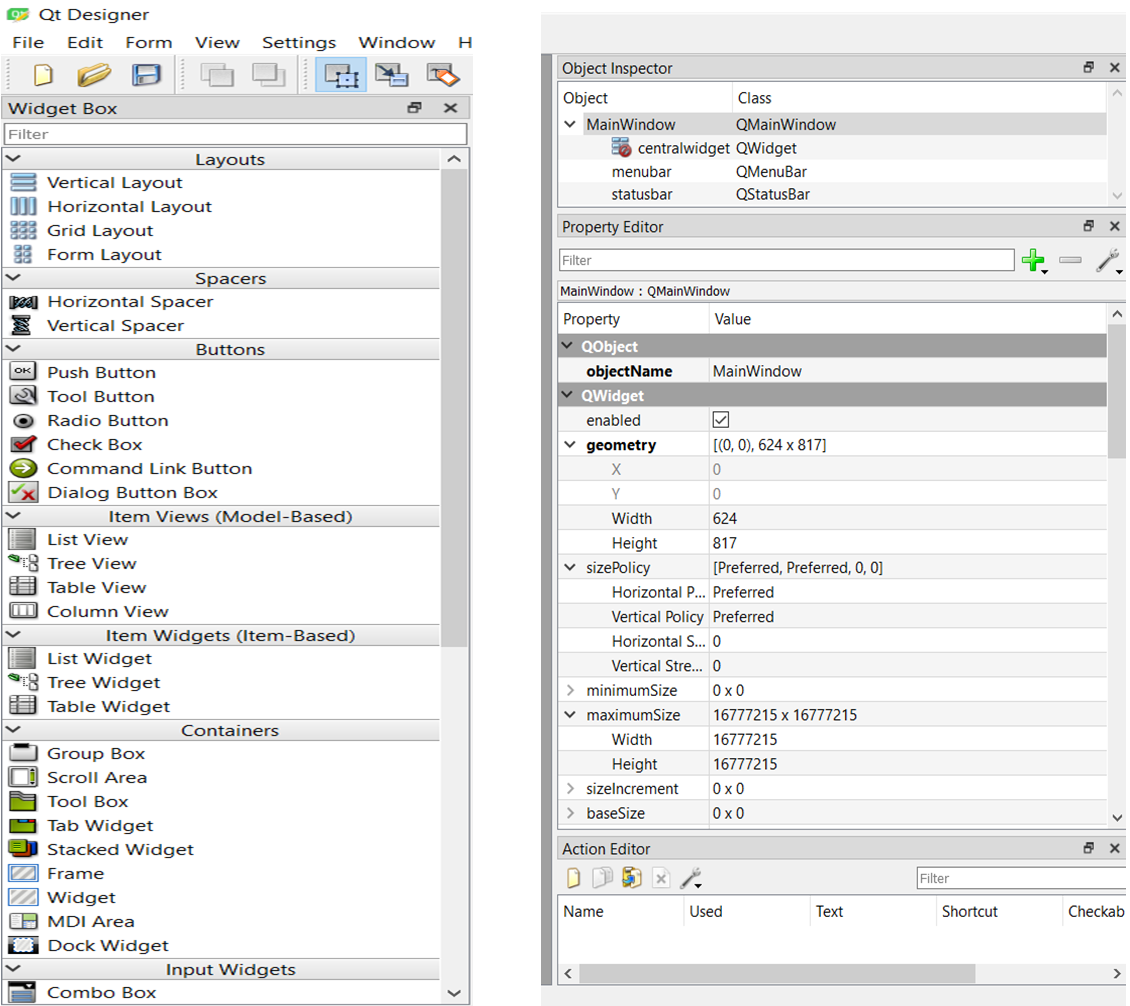


Fig 4.1 Qt5 Designer Window



4.2 Inbuild Widget Box 4.3 Inbulid Property Editor

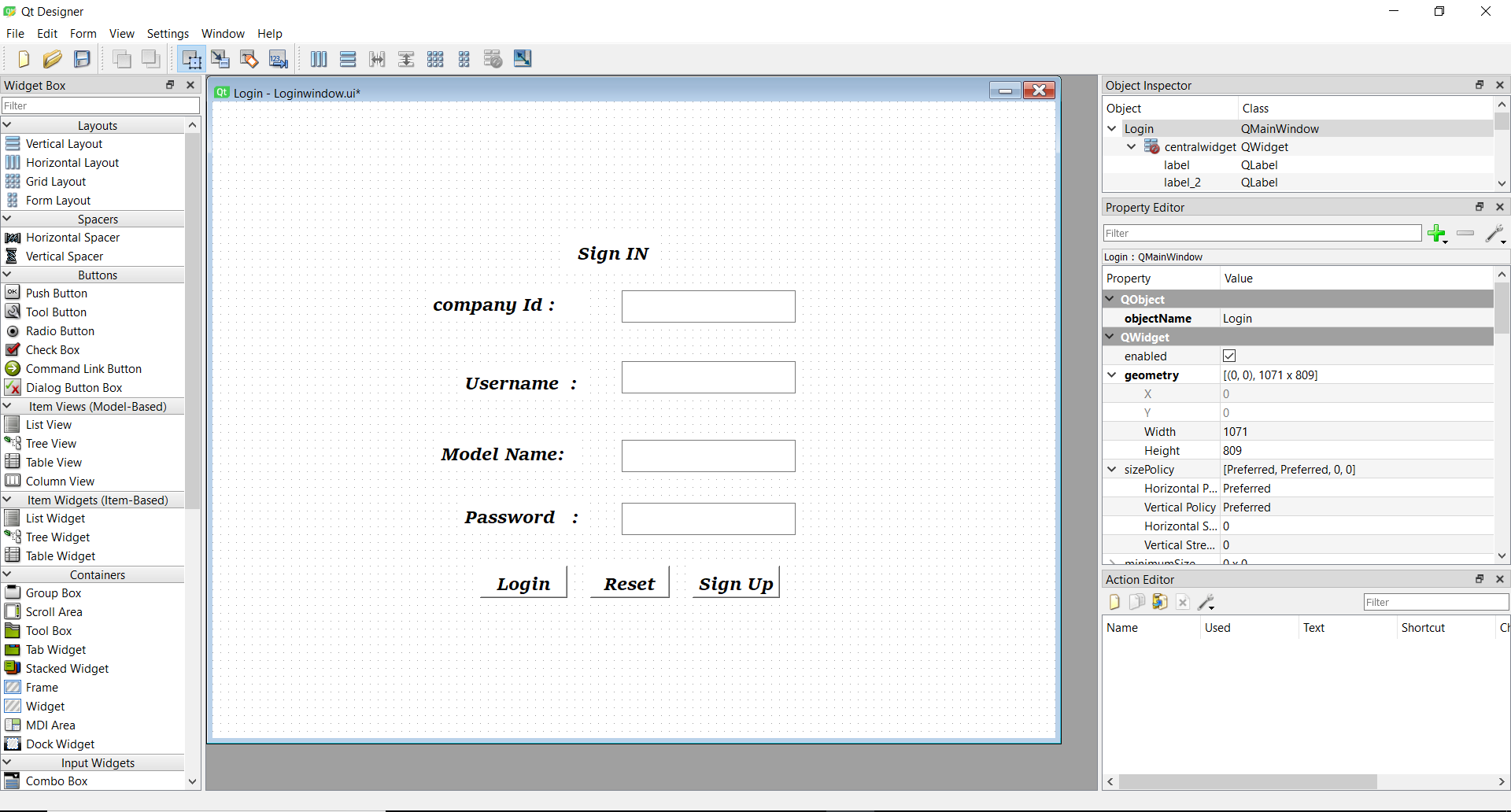
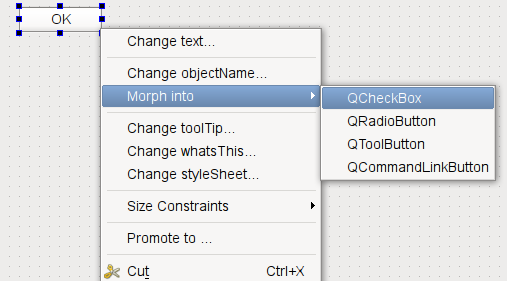


Fig 4.4 Login page design Example using Different fields

1. USING CONTAINERS IN QT DESIGNER

Container widgets provide high level control over groups of objects on a form. They can be used to perform a variety of functions, such as managing input widgets, providing paged and tabbed layouts, or just acting as decorative containers for other objects.



Qt Designer provides visual feedback to help you place objects inside your containers. When you drag an object from the widget box (or elsewhere) on the form, each container will be highlighted when the cursor is positioned over it. This indicates that you can drop the object inside, making it a child object of the container. This feedback is important because it is easy to place objects close to containers without actually placing them inside. Both widgets and spacers can be used inside containers.

Stacked widgets, tab widgets, and toolboxes are handled specially in Qt Designer. Normally, when adding pages (tabs, pages, compartments) to these containers in your own code, you need to supply existing widgets, either as placeholders or containing child widgets. In Qt Designer, these are automatically created for you, so you can add child objects to each page straight away.

Each container typically allows its child objects to be arranged in one or more layouts. The type of layout management provided depends on each container, although setting the layout is usually just a matter of selecting the container by clicking it, and applying a layout. The table below shows a list of available containers.

|  |  |
| --- | --- |
| https://doc.qt.io/qt-5/images/designer-containers-frame.png | Frames  Frames are used to enclose and group widgets, as well as to provide decoration. They are used as the foundation for more complex containers, but they can also be used as placeholders in forms.  The most important properties of frames are frameShape, frameShadow, lineWidth, and midLineWidth. These are described in more detail in the [QFrame](https://doc.qt.io/qt-5/qframe.html) class description. |
| https://doc.qt.io/qt-5/images/designer-containers-groupbox.png | Group Boxes  Group boxes are usually used to group together collections of checkboxes and radio buttons with similar purposes.  Among the significant properties of group boxes are title, flat, checkable, and checked. These are demonstrated in the [Group Box](https://doc.qt.io/qt-5/qtwidgets-widgets-groupbox-example.html) example, and described in the [QGroupBox](https://doc.qt.io/qt-5/qgroupbox.html) class documentation. Each group box can contain its own layout, and this is necessary if it contains other widgets. To add a layout to the group box, click inside it and apply the layout as usual. |
| https://doc.qt.io/qt-5/images/designer-containers-stackedwidget.png | Stacked Widgets  Stacked widgets are collections of widgets in which only the topmost layer is visible. Control over the visible layer is usually managed by another widget, such as combobox, using signals and slots.  Qt Designer shows arrows in the top-right corner of the stack to allow you to see all the widgets in the stack when designing it. These arrows do not appear in the preview or in the final component. To navigate between pages in the stack, select the stacked widget and use the Next Page and Previous Page entries from the context menu. The Insert Page and Delete Page context menu options allow you to add and remove pages. |
| https://doc.qt.io/qt-5/images/designer-containers-tabwidget.png | Tab Widgets  Tab widgets allow the developer to split up the contents of a widget into different labelled sections, only one of which is displayed at any given time. By default, the tab widget contains two tabs, and these can be deleted or renamed as required. You can also add additional tabs.  To delete a tab:  Click on its label to make it the current tab.  Select the tab widget and open its context menu.  Select Delete Page.  To add a new tab:  Select the tab widget and open its context menu.  Select Insert Page.  You can add a page before or after the current page. Qt Designer will create a new widget for that particular tab and insert it into the tab widget.  You can set the title of the current tab by changing the currentTabText property in the Property Editor. |
| https://doc.qt.io/qt-5/images/designer-containers-toolbox.png | ToolBox Widgets  Toolbox widgets provide a series of pages or compartments in a toolbox. They are handled in a way similar to stacked widgets.  To rename a page in a toolbox, make the toolbox your current pange and change its currentItemText property from the Property Editor.  To add a new page, select Insert Page from the toolbox widget's context menu. You can add the page before or after the current page.  To delete a page, select Delete Page from the toolbox widget's context menu. |
| https://doc.qt.io/qt-5/images/designer-containers-dockwidget.png | Dock Widgets  Dock widgets are floating panels, often containing input widgets and more complex controls, that are either attached to the edges of the main window in "dock areas", or floated as independent tool windows.  Although dock widgets can be added to any type of form, they are typically used with forms created from the [main window template](https://doc.qt.io/qt-5/designer-creating-mainwindows.html). |

1. CONCLUSION

The concept of scripting engine using Qt framework aims to make an application development or extension easy, balancing the performance requirement desired for the resulting application. Using Qt extensions for scripting, it enables to develop rich user-interface and event-handling of the interaction with user-interface using scripts. Considering the requirement of the application, different extension can be. As one of the objective of Scripting Engine is the ease of extending an existing application by a non-programmer, scripting engine like Lua and Python can be selected which are easily readable and unambiguous to use. If the aim is to develop a commercial application, PyQt provides commercial licence and informal support for the commercial licence holder. If the speed of the application is priority, QJSEngine can be used for scripting with the fastest JavaScript version 8 engine, provided few of the functions can be compromised. Pyotherside or Ruby-QML can be considered, if the developer does not wish to purchase the commercial license to develop proprietary software, as they are licensed under ISC/BSD and MIT respectively. The online support provided by exensions such as QtScript, QJSEngine and PyQt will provide the developer an added bene\_t. The Scripting Engine developed using extensions such as QtScript, QJSEngine, PyQt and PySide can be extended on many platforms including the non-conventional ones, whereas, the other extensions provide support for only conventional ones. Thus, using a strong framework like Qt, an appropriate extension can be selected to develop a Scripting Engine for rapid application development, meeting the high performance requirement. The resulting approach will enable any non-programmer to extend an application with ease and the product can be marketed within the stipulated time.

VII. REFERENCES

[1] M. McLaughlin and M. Katchabaw, A Reusable Scripting Engine for Automating Cinematics and Cut-Scenes in Video Games, Proceedings of CGSA 2006 Symposium.

[2] Milivoj Bozic, Dusan Zivkov, Istvan Pap and Goran Miljkovic, Scriptable Graphical user Interface Engine for embedded platform, 21st Telecommunication Forum TELFOR 2013.

[3] Hans Christian Woithe and Ulrich Kremer, A Light Weight Scripting for the Slocum Glider, IEEE Oceans 2010 Conference, Sydney, Australia, May 2010.

**ACKNOWLEDGEMENT**

I would like to place on record my deep sense of gratitude to Prof. S.B.Kalyankar, HOD-Dept. of Computer Science and Engineering, Deogiri Institute of Engineering and management Studies Aurangabad,for his generous guidance, help and useful suggestions.

I express my sincere gratitude to Prof. S.B.Kalyankar, Dept. of Computer Science and Engineering, Deogiri Institute of Engineering and management Studies Aurangabad, for his stimulating guidance, continuous encouragement and supervision throughout the course of present work.

I am extremely thankful toUlhas D. Shiurkar, Director, Deogiri Institute of Engineering and management Studies Aurangabad, for providing me infrastructural facilities to work in, without which this work would not have been possible.

**Signature of Student**

Tipare Bhagyashree Pravin Sign