A Visual GUI Tool for Auto-adjusting Text in Selected Image

Ravensburg-Weingarten University

Electrical Engineering and Information Technology

Bachelor Project Report
Period: 01.07.2020 - 30.09.2020

Supervisor:
Prof. Dr. Markus Pfeil

Author and Matrikel-Nr.:
Siyi Dai 29245
Daniel Heimman 29405

September 17, 2020

Contents

1. Introduction

Definition of Tasks

The project tasks are focused on developing graphical user interfaces (GUI) for auto-adjusting text to mouse selected area inside of an image using PyQt5.

GUI is a form of user interface that allows users to interact with electronic devices through graphical icons, instead of text-based user interfaces, typed command labels or text navigation.

The text itself should be able to modified with different fonts and sizes, and it should be equipped with wrapping function based on the content.

The space occupied by the input text should be evaluated and compared with the area the user set with mouse. If the text is overflowed out of the bounding box selected, the user should be able to choose to ignore it or clear the content.

2. Environment Specification

The work environment is composed of two sides. The hardware environment which presents the machines or components being used while running code. The software environment which represents the advanced programming interfaces, integrated development environments, editors, technologies and tools being used.

Hardware Environment

The table describes the used computer for work while two BenQ Corporation 32" monitors were used.

PC	HP ZBook 15 G5 (3AX13AV)
CPU	Intel(R) Core(TM) i7-8850H CPU @ 2.60GHz
RAM	32GB
OS	Ubuntu 18.04.3 LTS
Graphics Card	NVIDIA Quadro P1000 Mobile

Table 2.1: Characteristics of used computer

Software Environment

Python 3.6.10 :: Anaconda, Inc. *Python* is an interpreted, high-level, general-purpose programming language. The internship tasks are all coded in *Python 3.6.10* within help of *Anaconda*, which is a *Python* and *R* distribution including the core python language, 100+ Python libraries and package manager *conda*.

Visual Studio Code *Visual Studio Code* is a free source-code editor made by *Microsoft* for *Windows*, *Linux*, *macOS*. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

PyQt5 *PyQt5* is a GUI develop application provided by *Python*. It is cross-platform GUI toolkit, a set of python bindings for *Qt v5*. An interactive desktop application can be developed with much easier because of the tools and simplicity provided by this library.

MODULES AND CLASSES: While the designation of Exporter and Visualizer, various modules and classes from *PyQt5* have been used.

Modules	Classes
QtOpenGL	*QGLFormat*
QtCore	*pyqtSignal, Qt, QPoint, QRectF, QSize*
Qtgui	*QPainter, QColor, QBrush, QWheelEvent, QPixmap, QMouseEvent, QFont, QFontMetrics*
QtWidgets	*QGraphicsView, QGraphicsScene, QGraphicsPixmapItem, QAbstractScrollArea, QGraphicsRectItem, QGraphicsTextItem, QDesktopWidget, QMessageBox, QVBoxLayout, QFileDialog*

Table 2.2: Modules and Classes being used for developing UI tools

Qt Creator *Qt Creator* includes a code editor and integrates *Qt Designer* for designing and building GUIs from *Qt widgets*. After layout designing and objects naming in *Qt Creator*, a form implementation could be generated from reading ui file by *PyQt5 UI code generator 5.9.2*.

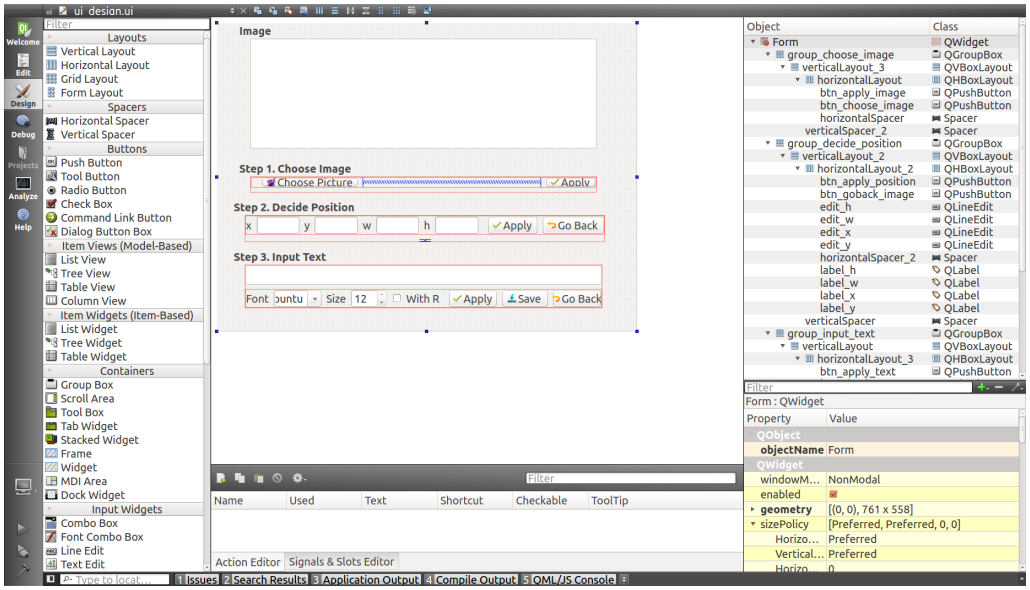

Figure 2.1: Qt Creator window in the design frame

3. Instruction

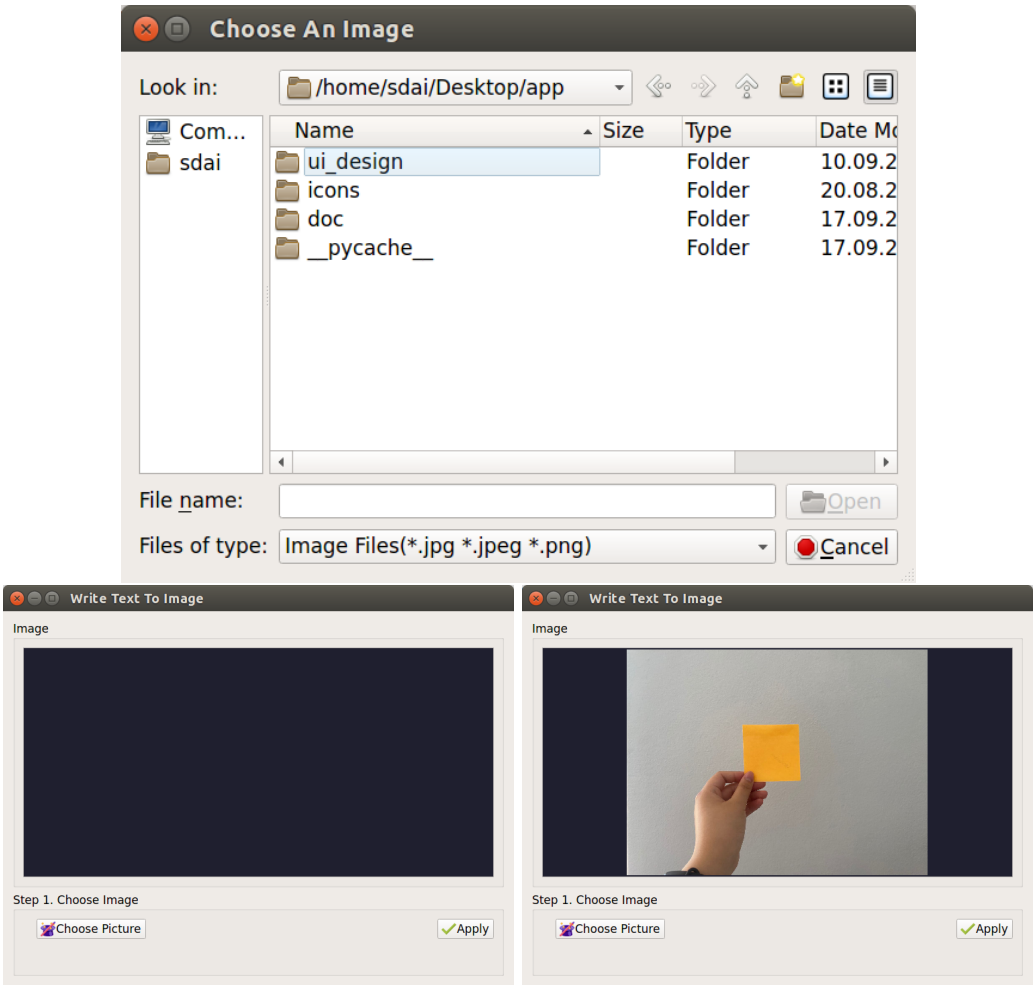

Figure 3.1: Selecting the picture to put text on

- **Step 1:** After the user selects a picture, click the Apply button and then go to step 2;

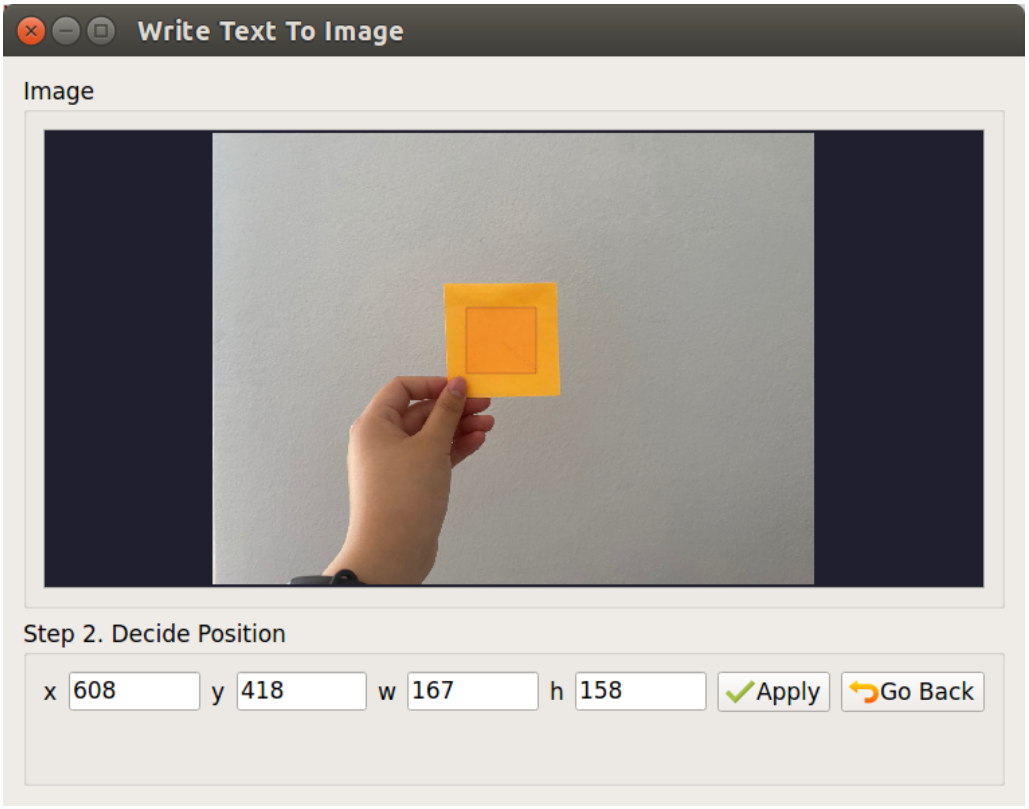

Figure 3.2: Selecting the text area in the image

- **Step 2:** The user selects the text block with the left mouse button, clicks Apply to enter step 3, and clicks the Go Back button to return to step 1;

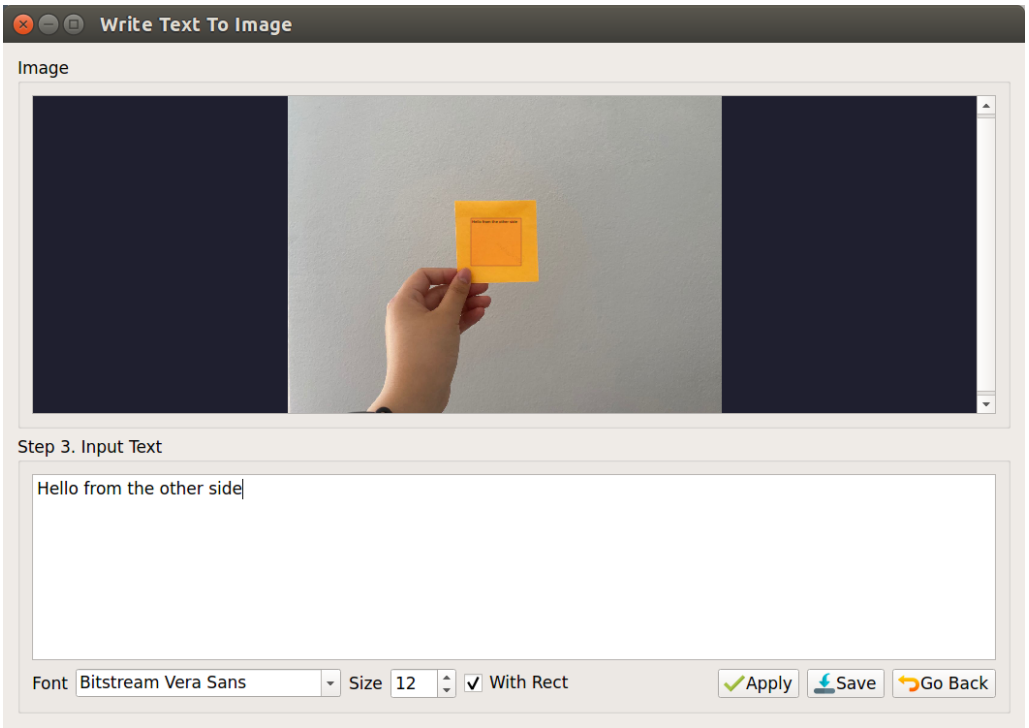

Figure 3.3: Input the text without font and size modified

- **Step 3:** The user can enter the text in the groupbox here.

- **Step 4:** After the user enters the text, select the desired font and font size and click the Apply button. The program will try to place the text in the specified block. The user can click the Go Back button to return to step 2.

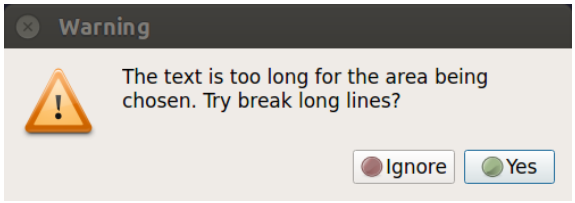

Figure 3.4: Message box asking for whether break the long lines

If the text is too long for the chosen area, a message box asking for whether break long lines will appear. Once the user clicked Yes, the text will be wrap into several lines.

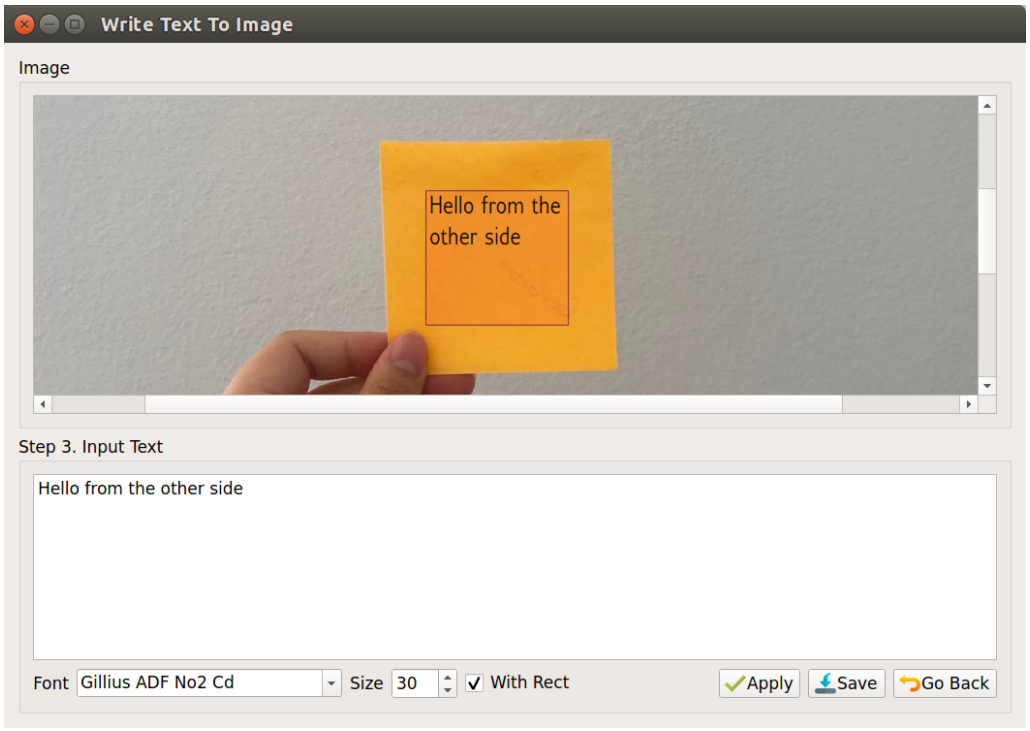

Figure 3.5: The text modified and wrapped into several lines

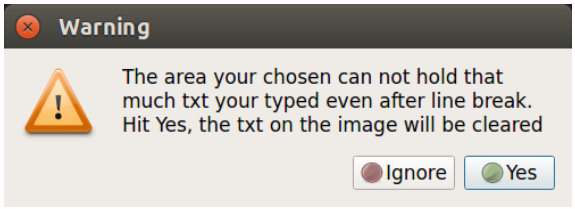

Figure 3.6: Message box asking for whether ignore the overflow or clear the content

If the text is still too much for the chosen area even after line wrap, a message box asking for whether ignore the overflow or clear the content will appear. Once the user clicked Yes, the text will be cleared out. Otherwise it will show the overflowed text.

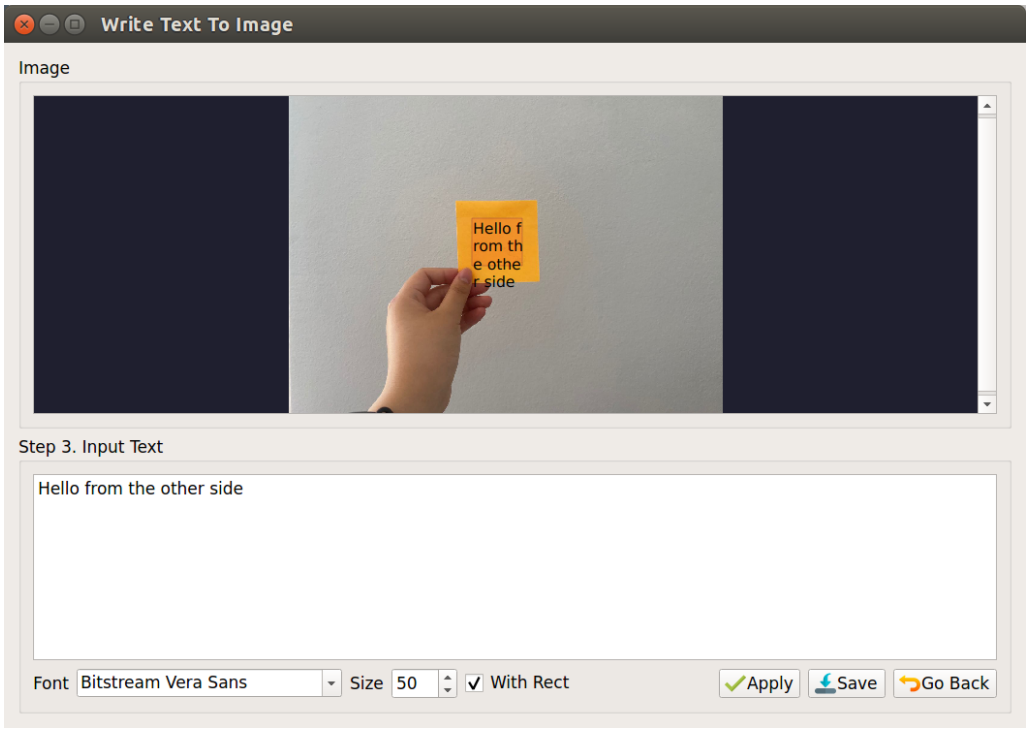

Figure 3.7: Overflowed text

- **Step 5:** The user can also click the Save button to save The picture containing the text is saved as a png format file. In addition, a selection button is provided, and the user can choose whether to display the text box.

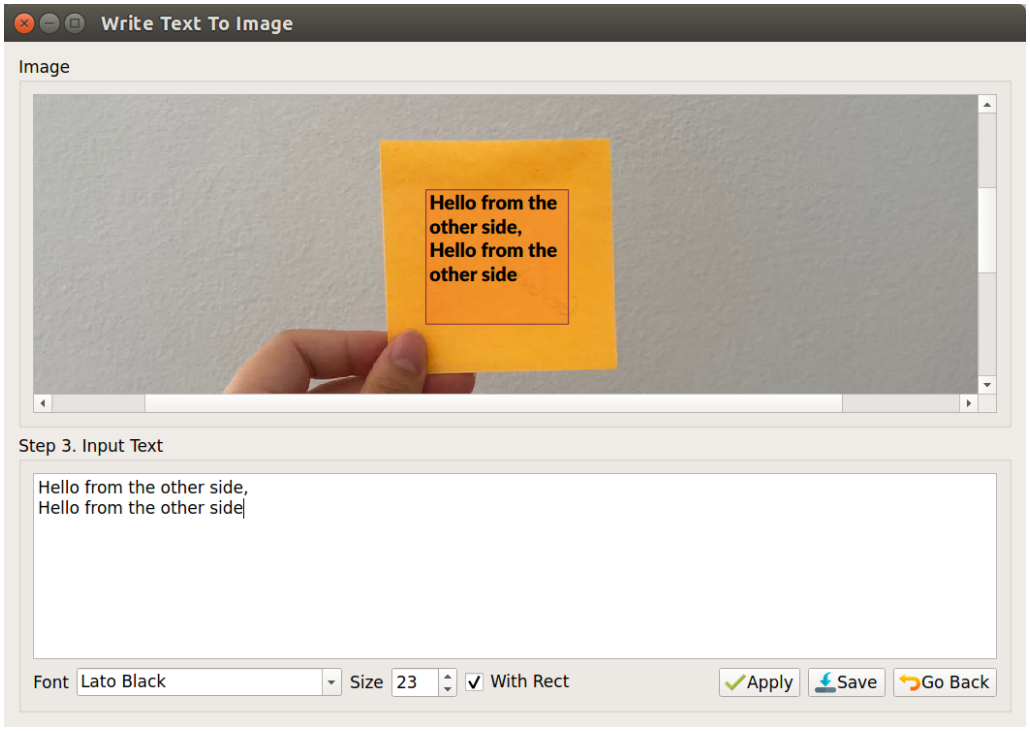

Figure 3.8: Modify the font and size to fit the text into the chosen area

Figure 3.9: Saved image with auto-adjusted text on it

4. Flowchart

Flowchart of the logic process of displaying text is attached as below.

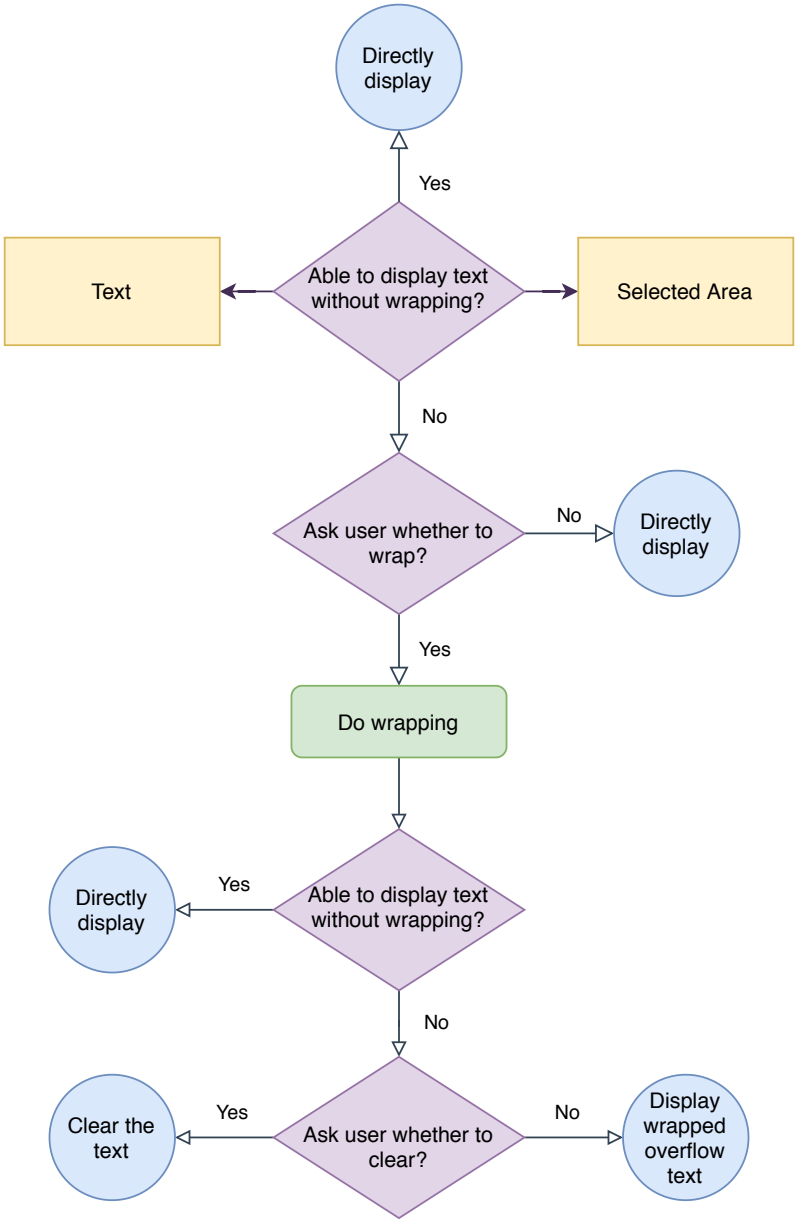

Figure 4.1: Flowchart of process of displaying text

5. Text Adjuster

Project Structure

A graphical user interfaces (GUI) for auto-adjusting text to mouse selected area inside of an image using PyQt5 has been successfully developed in this project.

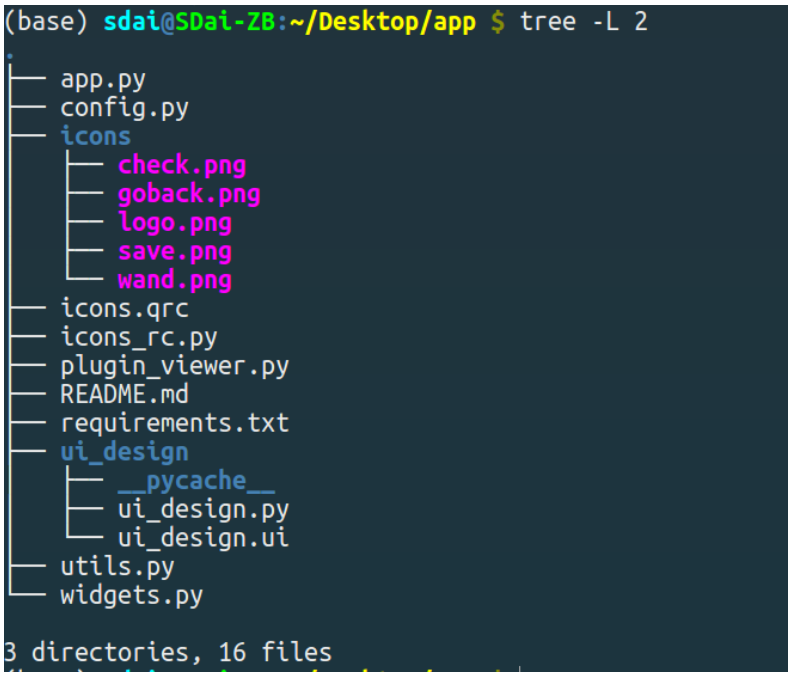

Figure 5.1: The project structure in level 2

The project folder is divided as:

- **app.py** executive file to run the text adjuster
- **config.py** config file for saving information of the app and the default directory path image loading from
- **icon** folder for storing icons used in the mainwindow
- **icons.qrc and icons_rc.py** icon related script auto-generated by *the Resource Compiler for PyQt5 (Qt v5.13.0)*
- **plugin_viewer.py** general setting relates to the viewer, including the wheel event and mouse event setup for selecting text area in image.
- **README.md** instruction for using the app

- **requirement.txt** environment requirement
- **ui_design** folder for storing the design of the mainwindow from Qt Creator
- **utils.py** some util functions for message boxes and load directory set
- **widgets.py** main script with handlers for choosing images, input texts, position storing, and decide whether break line

Code

The scripts being used in this project has been attached as below.

```python
#app.py

# coding: utf-8
from widgets import QtWidgets, WidgetMain
import sys

if __name__ == '__main__':
    app = QtWidgets.QApplication(sys.argv)
    m = WidgetMain()
    m.show()
    sys.exit(app.exec_())

```

```python
#config.py

# coding: utf-8

import sys
import os
import json

app_name_en = "txt_image"
app_name_cn = "Write Text To Image"
author_name = "Daniel Heimman(29405) and Siyi Dai(29245)"
author_org = "Hochschule Ravensburg Weingatyen"

base_dir = os.path.abspath(os.path.dirname(__file__))
home_dir = os.path.expanduser("~")
app_config_fp = os.path.join(home_dir, ".{}".format(app_name_en))

```

```python
#plugin_viewer.py

# coding: utf-8

from PyQt5 import QtCore, QtGui, QtWidgets, QtOpenGL
import math


class Viewer(QtWidgets.QGraphicsView):
    # color of frame background
    backgroundColor = QtGui.QColor(31, 31, 47)
    # color of frame border
    borderColor = QtGui.QColor(58, 58, 90)

    sig_position = QtCore.pyqtSignal(tuple)

    def __init__(self, parent=None, *args, **kwargs):
        super().__init__(parent=parent, *args, **kwargs)

        self.setBackgroundBrush(self.backgroundColor)  # set background color

        self.setOptimizationFlag(self.DontSavePainterState)

        self.setRenderHints(
            QtGui.QPainter.Antialiasing | QtGui.QPainter.TextAntialiasing |
QtGui.QPainter.SmoothPixmapTransform
        )

        if QtOpenGL.QGLFormat.hasOpenGL():
            self.setRenderHint(QtGui.QPainter.HighQualityAntialiasing)

        self.setResizeAnchor(self.AnchorUnderMouse)  # set the current position of
mouse as anchor

        self.setRubberBandSelectionMode(QtCore.Qt.IntersectsItemShape)

        self.setTransformationAnchor(self.AnchorUnderMouse)

        self.setViewportUpdateMode(self.SmartViewportUpdate)

        self._scene = QtWidgets.QGraphicsScene(self)
        self.setScene(self._scene)

        self._pix_item = QtWidgets.QGraphicsPixmapItem()
        self._scene.addItem(self._pix_item)

        # self.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOff)
        # self.setVerticalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOff)


self.setSizeAdjustPolicy(QtWidgets.QAbstractScrollArea.AdjustToContentsOnFirstShow)
        self.setAlignment(QtCore.Qt.AlignJustify | QtCore.Qt.AlignVCenter |
QtCore.Qt.AlignHCenter)

        # enable zoom in
        self.setAcceptDrops(True)
        self.setDragMode(QtWidgets.QGraphicsView.NoDrag)

        self.last = "Click"

        self.canDraw = False
```

```
        self.item_rect = QtWidgets.QGraphicsRectItem()
        self.scene().addItem(self.item_rect)
        self.item_rect.setBrush(QtGui.QBrush(QtGui.QColor(255, 99, 45, 60)))
        self.item_rect.setPen(QtGui.QColor(109, 25, 75))

        self.item_text = QtWidgets.QGraphicsTextItem()
        # self.item_text.setPlainText('hello world')
        self.scene().addItem(self.item_text)

        self.setAllowDrawRect(False)

        self.begin = QtCore.QPoint()
        self.end = QtCore.QPoint()
        self.point_a = 0, 0
        self.point_b = 0, 0

    def wheelEvent(self, event: QtGui.QWheelEvent) -> None:
        """press 'ctrl' with wheel event to zoom in/out"""
        if event.modifiers() & QtCore.Qt.ControlModifier:
            self.scaleView(math.pow(2.0, event.angleDelta().y() / 240.0))
            return event.accept()
        super(QtWidgets.QGraphicsView, self).wheelEvent(event)

    def scaleView(self, scaleFactor):
        factor = self.transform().scale(scaleFactor,
scaleFactor).mapRect(QtCore.QRectF(0, 0, 1, 1)).width()
        if factor < 0.07 or factor > 100:
            return
        self.scale(scaleFactor, scaleFactor)

    def display_pix(self, pix: QtGui.QPixmap):
        """update the pic to display"""
        if not isinstance(pix, QtGui.QPixmap):
            raise TypeError
        self._pix_item.setPixmap(pix)
        w, h = pix.width(), pix.height()
        self.fitInView(QtCore.QRectF(0, 0, w, h), QtCore.Qt.KeepAspectRatio)
        self._scene.update()

    def clear_pix(self):
        """clear out the pic"""
        self._pix_item.setPixmap(QtGui.QPixmap())

    def mousePressEvent(self, event: QtGui.QMouseEvent) -> None:
        self.last = "Click"
        if self.canDraw and event.buttons() & QtCore.Qt.LeftButton:
            self.begin = event.pos()
            self.end = event.pos()
            p = self._pix_item.mapFromScene(self.mapToScene(event.pos()))
            self.point_a = self.point_b = p.x(), p.y()

    def mouseMoveEvent(self, event: QtGui.QMouseEvent) -> None:
        self.end = event.pos()
        if self.canDraw and (event.buttons() & QtCore.Qt.LeftButton):
            rect = QtCore.QRectF(self.mapToScene(self.begin),
self.mapToScene(self.end))
            self.item_rect.setRect(rect)
            self.item_text.setPos(self.mapToScene(self.begin))
            p = self._pix_item.mapFromScene(self.mapToScene(event.pos()))
            self.point_b = p.x(), p.y()

            x1, y1 = self.point_a
```

```
            x2, y2 = self.point_b

            w = x2 - x1
            h = y2 - y1

            if w == 0 or h == 0:
                text_pos = 0, 0, 0, 0
            elif w > 0 and h > 0:
                text_pos = x1, y1, w, h
            elif w > 0 and h < 0:
                text_pos = x1, y2, w, -h
            elif w < 0 and h > 0:
                text_pos = x2, y1, -w, h
            elif w < 0 and h < 0:
                text_pos = x2, y2, -w, -h
            else:
                text_pos = 0, 0, 0, 0


self.item_text.setPos(self._pix_item.mapToScene(QtCore.QPointF(text_pos[0],
text_pos[1])))

            self.sig_position.emit(text_pos)

    def mouseReleaseEvent(self, event: QtGui.QMouseEvent) -> None:
        if self.last == "Click":
            print("click")
        else:
            print("Double Click")

    def mouseDoubleClickEvent(self, event: QtGui.QMouseEvent) -> None:
        self.last = "Double Click"

    def setAllowDrawRect(self, b: bool):
        """"""
        if b is True:
            # self.item_rect.show()
            self.canDraw = True
        else:
            # self.item_rect.hide()
            self.canDraw = False

    def display_text(self, txt: str, font: QtGui.QFont):
        self.item_text.setFont(font)
        self.item_text.setPlainText(txt)

    def clear_text(self):
        self.item_text.setPlainText("")

    def hide_rect(self):
        self.item_rect.hide()

    def display_rect(self):
        self.item_rect.show()

    def save_image(self) -> QtGui.QPixmap:
        size = self._pix_item.boundingRect().size()
        pix = QtGui.QPixmap(QtCore.QSize(int(size.width()), int(size.height())))
        painter = QtGui.QPainter(pix)
        painter.setRenderHint(QtGui.QPainter.Antialiasing, True)
        self.scene().render(
```

```
177             painter, self._pix_item.boundingRect(), self._pix_item.boundingRect(),
    QtCore.Qt.KeepAspectRatio,
178         )
179         painter.end()
180         return pix
181
```

```python
#ui_design.py

# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'ui_design.ui'
#
# Created by: PyQt5 UI code generator 5.9.2
#
# WARNING! All changes made in this file will be lost!

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(761, 558)
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap(":/icons/logo.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
        Form.setWindowIcon(icon)
        self.group_viewer = QtWidgets.QGroupBox(Form)
        self.group_viewer.setGeometry(QtCore.QRect(40, 0, 661, 241))
        self.group_viewer.setObjectName("group_viewer")
        self.gridLayout = QtWidgets.QGridLayout(self.group_viewer)
        self.gridLayout.setObjectName("gridLayout")
        self.graphics_viewer = Viewer(self.group_viewer)
        self.graphics_viewer.setObjectName("graphics_viewer")
        self.gridLayout.addWidget(self.graphics_viewer, 0, 0, 1, 1)
        self.group_choose_image = QtWidgets.QGroupBox(Form)
        self.group_choose_image.setGeometry(QtCore.QRect(40, 250, 661, 71))
        self.group_choose_image.setObjectName("group_choose_image")
        self.gridLayout_2 = QtWidgets.QGridLayout(self.group_choose_image)
        self.gridLayout_2.setObjectName("gridLayout_2")
        self.verticalLayout_3 = QtWidgets.QVBoxLayout()
        self.verticalLayout_3.setObjectName("verticalLayout_3")
        self.horizontalLayout = QtWidgets.QHBoxLayout()
        self.horizontalLayout.setContentsMargins(20, -1, -1, -1)
        self.horizontalLayout.setObjectName("horizontalLayout")
        self.btn_choose_image = QtWidgets.QPushButton(self.group_choose_image)
        icon1 = QtGui.QIcon()
        icon1.addPixmap(QtGui.QPixmap(":/icons/wand.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.btn_choose_image.setIcon(icon1)
        self.btn_choose_image.setObjectName("btn_choose_image")
        self.horizontalLayout.addWidget(self.btn_choose_image)
        spacerItem = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)
        self.horizontalLayout.addItem(spacerItem)
        self.btn_apply_image = QtWidgets.QPushButton(self.group_choose_image)
        icon2 = QtGui.QIcon()
        icon2.addPixmap(QtGui.QPixmap(":/icons/check.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.btn_apply_image.setIcon(icon2)
        self.btn_apply_image.setObjectName("btn_apply_image")
        self.horizontalLayout.addWidget(self.btn_apply_image)
        self.verticalLayout_3.addLayout(self.horizontalLayout)
        spacerItem1 = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Expanding)
        self.verticalLayout_3.addItem(spacerItem1)
        self.gridLayout_2.addLayout(self.verticalLayout_3, 0, 0, 1, 1)
        self.group_decide_position = QtWidgets.QGroupBox(Form)
        self.group_decide_position.setGeometry(QtCore.QRect(30, 320, 686, 91))
```

```
        self.group_decide_position.setObjectName("group_decide_position")
        self.gridLayout_3 = QtWidgets.QGridLayout(self.group_decide_position)
        self.gridLayout_3.setObjectName("gridLayout_3")
        self.verticalLayout_2 = QtWidgets.QVBoxLayout()
        self.verticalLayout_2.setObjectName("verticalLayout_2")
        self.horizontalLayout_2 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_2.setObjectName("horizontalLayout_2")
        self.label_x = QtWidgets.QLabel(self.group_decide_position)
        self.label_x.setObjectName("label_x")
        self.horizontalLayout_2.addWidget(self.label_x)
        self.edit_x = QtWidgets.QLineEdit(self.group_decide_position)
        self.edit_x.setObjectName("edit_x")
        self.horizontalLayout_2.addWidget(self.edit_x)
        self.label_y = QtWidgets.QLabel(self.group_decide_position)
        self.label_y.setObjectName("label_y")
        self.horizontalLayout_2.addWidget(self.label_y)
        self.edit_y = QtWidgets.QLineEdit(self.group_decide_position)
        self.edit_y.setObjectName("edit_y")
        self.horizontalLayout_2.addWidget(self.edit_y)
        self.label_w = QtWidgets.QLabel(self.group_decide_position)
        self.label_w.setObjectName("label_w")
        self.horizontalLayout_2.addWidget(self.label_w)
        self.edit_w = QtWidgets.QLineEdit(self.group_decide_position)
        self.edit_w.setObjectName("edit_w")
        self.horizontalLayout_2.addWidget(self.edit_w)
        self.label_h = QtWidgets.QLabel(self.group_decide_position)
        self.label_h.setObjectName("label_h")
        self.horizontalLayout_2.addWidget(self.label_h)
        self.edit_h = QtWidgets.QLineEdit(self.group_decide_position)
        self.edit_h.setObjectName("edit_h")
        self.horizontalLayout_2.addWidget(self.edit_h)
        spacerItem2 = QtWidgets.QSpacerItem(40, 20,
QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)
        self.horizontalLayout_2.addItem(spacerItem2)
        self.btn_apply_position = QtWidgets.QPushButton(self.group_decide_position)
        self.btn_apply_position.setIcon(icon2)
        self.btn_apply_position.setObjectName("btn_apply_position")
        self.horizontalLayout_2.addWidget(self.btn_apply_position)
        self.btn_goback_image = QtWidgets.QPushButton(self.group_decide_position)
        icon3 = QtGui.QIcon()
        icon3.addPixmap(QtGui.QPixmap(":/icons/goback.png"), QtGui.QIcon.Normal,
QtGui.QIcon.Off)
        self.btn_goback_image.setIcon(icon3)
        self.btn_goback_image.setObjectName("btn_goback_image")
        self.horizontalLayout_2.addWidget(self.btn_goback_image)
        self.verticalLayout_2.addLayout(self.horizontalLayout_2)
        spacerItem3 = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Expanding)
        self.verticalLayout_2.addItem(spacerItem3)
        self.gridLayout_3.addLayout(self.verticalLayout_2, 0, 0, 1, 1)
        self.group_input_text = QtWidgets.QGroupBox(Form)
        self.group_input_text.setGeometry(QtCore.QRect(30, 410, 681, 121))
        self.group_input_text.setObjectName("group_input_text")
        self.gridLayout_4 = QtWidgets.QGridLayout(self.group_input_text)
        self.gridLayout_4.setObjectName("gridLayout_4")
        self.verticalLayout = QtWidgets.QVBoxLayout()
        self.verticalLayout.setObjectName("verticalLayout")
        self.edit_txt = QtWidgets.QPlainTextEdit(self.group_input_text)
        self.edit_txt.setObjectName("edit_txt")
        self.verticalLayout.addWidget(self.edit_txt)
        self.horizontalLayout_3 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_3.setObjectName("horizontalLayout_3")
```

```
        self.label_font = QtWidgets.QLabel(self.group_input_text)
        self.label_font.setObjectName("label_font")
        self.horizontalLayout_3.addWidget(self.label_font)
        self.font_chooser = QtWidgets.QFontComboBox(self.group_input_text)
        self.font_chooser.setObjectName("font_chooser")
        self.horizontalLayout_3.addWidget(self.font_chooser)
        self.label_size = QtWidgets.QLabel(self.group_input_text)
        self.label_size.setObjectName("label_size")
        self.horizontalLayout_3.addWidget(self.label_size)
        self.size_chooser = QtWidgets.QSpinBox(self.group_input_text)
        self.size_chooser.setMaximum(300)
        self.size_chooser.setProperty("value", 12)
        self.size_chooser.setObjectName("size_chooser")
        self.horizontalLayout_3.addWidget(self.size_chooser)
        self.btn_check = QtWidgets.QCheckBox(self.group_input_text)
        self.btn_check.setObjectName("btn_check")
        self.horizontalLayout_3.addWidget(self.btn_check)
        spacerItem4 = QtWidgets.QSpacerItem(40, 20,
QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)
        self.horizontalLayout_3.addItem(spacerItem4)
        self.btn_apply_text = QtWidgets.QPushButton(self.group_input_text)
        self.btn_apply_text.setIcon(icon2)
        self.btn_apply_text.setObjectName("btn_apply_text")
        self.horizontalLayout_3.addWidget(self.btn_apply_text)
        self.btn_save_image = QtWidgets.QPushButton(self.group_input_text)
        icon4 = QtGui.QIcon()
        icon4.addPixmap(QtGui.QPixmap(":/icons/save.png"), QtGui.QIcon.Normal,
QtGui.QIcon.Off)
        self.btn_save_image.setIcon(icon4)
        self.btn_save_image.setObjectName("btn_save_image")
        self.horizontalLayout_3.addWidget(self.btn_save_image)
        self.btn_goback_position = QtWidgets.QPushButton(self.group_input_text)
        self.btn_goback_position.setIcon(icon3)
        self.btn_goback_position.setObjectName("btn_goback_position")
        self.horizontalLayout_3.addWidget(self.btn_goback_position)
        self.verticalLayout.addLayout(self.horizontalLayout_3)
        self.gridLayout_4.addLayout(self.verticalLayout, 0, 0, 1, 1)

        self.retranslateUi(Form)
        QtCore.QMetaObject.connectSlotsByName(Form)

    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "Form"))
        self.group_viewer.setTitle(_translate("Form", "Image"))
        self.group_choose_image.setTitle(_translate("Form", "Step 1. Choose
Image"))
        self.btn_choose_image.setText(_translate("Form", "Choose Picture"))
        self.btn_apply_image.setText(_translate("Form", "Apply"))
        self.group_decide_position.setTitle(_translate("Form", "Step 2. Decide
Position"))
        self.label_x.setText(_translate("Form", "x"))
        self.label_y.setText(_translate("Form", "y"))
        self.label_w.setText(_translate("Form", "w"))
        self.label_h.setText(_translate("Form", "h"))
        self.btn_apply_position.setText(_translate("Form", "Apply"))
        self.btn_goback_image.setText(_translate("Form", "Go Back"))
        self.group_input_text.setTitle(_translate("Form", "Step 3. Input Text"))
        self.label_font.setText(_translate("Form", "Font"))
        self.label_size.setText(_translate("Form", "Size"))
        self.btn_check.setText(_translate("Form", "With Rect"))
        self.btn_apply_text.setText(_translate("Form", "Apply"))
```

```
        self.btn_save_image.setText(_translate("Form", "Save"))
        self.btn_goback_position.setText(_translate("Form", "Go Back"))

from plugin_viewer import Viewer
import icons_rc

if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Form = QtWidgets.QWidget()
    ui = Ui_Form()
    ui.setupUi(Form)
    Form.show()
    sys.exit(app.exec_())
```

```
#ui_design.ui

<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>Form</class>
 <widget class="QWidget" name="Form">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>761</width>
    <height>558</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>Form</string>
  </property>
  <property name="windowIcon">
   <iconset resource="../icons.qrc">
    <normaloff>:/icons/logo.png</normaloff>:/icons/logo.png</iconset>
  </property>
  <widget class="QGroupBox" name="group_viewer">
   <property name="geometry">
    <rect>
     <x>40</x>
     <y>0</y>
     <width>661</width>
     <height>241</height>
    </rect>
   </property>
   <property name="title">
    <string>Image</string>
   </property>
   <layout class="QGridLayout" name="gridLayout">
    <item row="0" column="0">
     <widget class="Viewer" name="graphics_viewer"/>
    </item>
   </layout>
  </widget>
  <widget class="QGroupBox" name="group_choose_image">
   <property name="geometry">
    <rect>
     <x>40</x>
     <y>250</y>
     <width>661</width>
     <height>71</height>
    </rect>
   </property>
   <property name="title">
    <string>Step 1. Choose Image</string>
   </property>
   <layout class="QGridLayout" name="gridLayout_2">
    <item row="0" column="0">
     <layout class="QVBoxLayout" name="verticalLayout_3">
      <item>
       <layout class="QHBoxLayout" name="horizontalLayout">
        <property name="leftMargin">
         <number>20</number>
        </property>
        <item>
         <widget class="QPushButton" name="btn_choose_image">
```

```
          <property name="text">
           <string>Choose Picture</string>
          </property>
          <property name="icon">
           <iconset resource="../icons.qrc">
            <normaloff>:/icons/wand.png</normaloff>:/icons/wand.png</iconset>
          </property>
         </widget>
        </item>
        <item>
         <spacer name="horizontalSpacer">
          <property name="orientation">
           <enum>Qt::Horizontal</enum>
          </property>
          <property name="sizeHint" stdset="0">
           <size>
            <width>40</width>
            <height>20</height>
           </size>
          </property>
         </spacer>
        </item>
        <item>
         <widget class="QPushButton" name="btn_apply_image">
          <property name="text">
           <string>Apply</string>
          </property>
          <property name="icon">
           <iconset resource="../icons.qrc">
            <normaloff>:/icons/check.png</normaloff>:/icons/check.png</iconset>
          </property>
         </widget>
        </item>
       </layout>
      </item>
      <item>
       <spacer name="verticalSpacer_2">
        <property name="orientation">
         <enum>Qt::Vertical</enum>
        </property>
        <property name="sizeHint" stdset="0">
         <size>
          <width>20</width>
          <height>40</height>
         </size>
        </property>
       </spacer>
      </item>
     </layout>
    </item>
   </layout>
  </widget>
  <widget class="QGroupBox" name="group_decide_position">
   <property name="geometry">
    <rect>
     <x>30</x>
     <y>320</y>
     <width>686</width>
     <height>91</height>
    </rect>
   </property>
```

```
  <property name="title">
   <string>Step 2. Decide Position</string>
  </property>
  <layout class="QGridLayout" name="gridLayout_3">
   <item row="0" column="0">
    <layout class="QVBoxLayout" name="verticalLayout_2">
     <item>
      <layout class="QHBoxLayout" name="horizontalLayout_2">
       <item>
        <widget class="QLabel" name="label_x">
         <property name="text">
          <string>x</string>
         </property>
        </widget>
       </item>
       <item>
        <widget class="QLineEdit" name="edit_x"/>
       </item>
       <item>
        <widget class="QLabel" name="label_y">
         <property name="text">
          <string>y</string>
         </property>
        </widget>
       </item>
       <item>
        <widget class="QLineEdit" name="edit_y"/>
       </item>
       <item>
        <widget class="QLabel" name="label_w">
         <property name="text">
          <string>w</string>
         </property>
        </widget>
       </item>
       <item>
        <widget class="QLineEdit" name="edit_w"/>
       </item>
       <item>
        <widget class="QLabel" name="label_h">
         <property name="text">
          <string>h</string>
         </property>
        </widget>
       </item>
       <item>
        <widget class="QLineEdit" name="edit_h"/>
       </item>
       <item>
        <spacer name="horizontalSpacer_2">
         <property name="orientation">
          <enum>Qt::Horizontal</enum>
         </property>
         <property name="sizeHint" stdset="0">
          <size>
           <width>40</width>
           <height>20</height>
          </size>
         </property>
        </spacer>
       </item>
```

```
        <item>
         <widget class="QPushButton" name="btn_apply_position">
          <property name="text">
           <string>Apply</string>
          </property>
          <property name="icon">
           <iconset resource="../icons.qrc">
            <normaloff>:/icons/check.png</normaloff>:/icons/check.png</iconset>
          </property>
         </widget>
        </item>
        <item>
         <widget class="QPushButton" name="btn_goback_image">
          <property name="text">
           <string>Go Back</string>
          </property>
          <property name="icon">
           <iconset resource="../icons.qrc">
            <normaloff>:/icons/goback.png</normaloff>:/icons/goback.png</iconset>
          </property>
         </widget>
        </item>
       </layout>
      </item>
      <item>
       <spacer name="verticalSpacer">
        <property name="orientation">
         <enum>Qt::Vertical</enum>
        </property>
        <property name="sizeHint" stdset="0">
         <size>
          <width>20</width>
          <height>40</height>
         </size>
        </property>
       </spacer>
      </item>
     </layout>
    </item>
   </layout>
  </widget>
  <widget class="QGroupBox" name="group_input_text">
   <property name="geometry">
    <rect>
     <x>30</x>
     <y>410</y>
     <width>681</width>
     <height>121</height>
    </rect>
   </property>
   <property name="title">
    <string>Step 3. Input Text</string>
   </property>
   <layout class="QGridLayout" name="gridLayout_4">
    <item row="0" column="0">
     <layout class="QVBoxLayout" name="verticalLayout">
      <item>
       <widget class="QPlainTextEdit" name="edit_txt"/>
      </item>
      <item>
       <layout class="QHBoxLayout" name="horizontalLayout_3">
```

```
        <item>
         <widget class="QLabel" name="label_font">
          <property name="text">
           <string>Font</string>
          </property>
         </widget>
        </item>
        <item>
         <widget class="QFontComboBox" name="font_chooser"/>
        </item>
        <item>
         <widget class="QLabel" name="label_size">
          <property name="text">
           <string>Size</string>
          </property>
         </widget>
        </item>
        <item>
         <widget class="QSpinBox" name="size_chooser">
          <property name="maximum">
           <number>300</number>
          </property>
          <property name="value">
           <number>12</number>
          </property>
         </widget>
        </item>
        <item>
         <widget class="QCheckBox" name="btn_check">
          <property name="text">
           <string>With Rect</string>
          </property>
         </widget>
        </item>
        <item>
         <spacer name="horizontalSpacer_3">
          <property name="orientation">
           <enum>Qt::Horizontal</enum>
          </property>
          <property name="sizeHint" stdset="0">
           <size>
            <width>40</width>
            <height>20</height>
           </size>
          </property>
         </spacer>
        </item>
        <item>
         <widget class="QPushButton" name="btn_apply_text">
          <property name="text">
           <string>Apply</string>
          </property>
          <property name="icon">
           <iconset resource="../icons.qrc">
            <normaloff>:/icons/check.png</normaloff>:/icons/check.png</iconset>
          </property>
         </widget>
        </item>
        <item>
         <widget class="QPushButton" name="btn_save_image">
          <property name="text">
```

```
           <string>Save</string>
          </property>
          <property name="icon">
           <iconset resource="../icons.qrc">
            <normaloff>:/icons/save.png</normaloff>:/icons/save.png</iconset>
          </property>
         </widget>
        </item>
        <item>
         <widget class="QPushButton" name="btn_goback_position">
          <property name="text">
           <string>Go Back</string>
          </property>
          <property name="icon">
           <iconset resource="../icons.qrc">
            <normaloff>:/icons/goback.png</normaloff>:/icons/goback.png</iconset>
          </property>
         </widget>
        </item>
       </layout>
      </item>
     </layout>
    </item>
   </layout>
  </widget>
 </widget>
 <customwidgets>
  <customwidget>
   <class>Viewer</class>
   <extends>QGraphicsView</extends>
   <header>plugin_viewer.h</header>
  </customwidget>
 </customwidgets>
 <resources>
  <include location="../icons.qrc"/>
 </resources>
 <connections/>
</ui>
```

```python
#utils.py

# coding: utf-8

import os
import config
from PyQt5 import QtWidgets, QtCore, QtGui
import json


class AbstractFunction(object):
    def move_to_center(self):
        qtRectangle = self.frameGeometry()
        centerPoint = QtWidgets.QDesktopWidget().availableGeometry().center()
        qtRectangle.moveCenter(centerPoint)
        self.move(qtRectangle.topLeft())

    @classmethod
    def show_warning_message(
        cls,
        message: str,
        title: str = "Warning",
        detail: str = None,
        extra: str = None,
        parent=None,
        only_yes: bool = False,
    ):
        """show warning msg"""
        msg_box = QtWidgets.QMessageBox(parent=parent)
        msg_box.setIcon(QtWidgets.QMessageBox.Warning)
        msg_box.setText(message)
        msg_box.setWindowTitle(title)
        if isinstance(extra, str) and detail:
            msg_box.setInformativeText(extra)
        if isinstance(detail, str) and detail:
            msg_box.setDetailedText(detail)
        if only_yes is True:
            msg_box.setStandardButtons(QtWidgets.QMessageBox.Yes)
            btn_yes = msg_box.button(QtWidgets.QMessageBox.Yes)
            btn_yes.setText("Yes")
        else:
            msg_box.setStandardButtons(QtWidgets.QMessageBox.Yes |
QtWidgets.QMessageBox.No)
            btn_yes = msg_box.button(QtWidgets.QMessageBox.Yes)
            btn_yes.setText("Yes")
            btn_no = msg_box.button(QtWidgets.QMessageBox.No)
            btn_no.setText("Ignore")

        msg_box.setDefaultButton(QtWidgets.QMessageBox.Yes)
        msg_box.setEscapeButton(QtWidgets.QMessageBox.No)
        msg_box.setTextInteractionFlags(QtCore.Qt.TextSelectableByMouse)  #
QtCore.Qt.NoTextInteraction
        r = msg_box.exec_()
        if r == QtWidgets.QMessageBox.Yes:
            return True
        else:
            return False

    @classmethod
    def show_info_message(
        cls,
```

```
        message: str,
        title="Notification",
        detail: str = None,
        extra: str = None,
        parent=None,
        only_yes: bool = False,
    ):
        """show notification msg"""
        msg_box = QtWidgets.QMessageBox(parent=parent)
        msg_box.setIcon(QtWidgets.QMessageBox.Information)
        msg_box.setText(message)
        msg_box.setWindowTitle(title)
        if isinstance(extra, str) and detail:
            msg_box.setInformativeText(extra)
        if isinstance(detail, str) and detail:
            msg_box.setDetailedText(detail)
        if only_yes is True:
            msg_box.setStandardButtons(QtWidgets.QMessageBox.Yes)
            btn_yes = msg_box.button(QtWidgets.QMessageBox.Yes)
            btn_yes.setText("Yes")
        else:
            msg_box.setStandardButtons(QtWidgets.QMessageBox.Yes |
QtWidgets.QMessageBox.No)
            btn_yes = msg_box.button(QtWidgets.QMessageBox.Yes)
            btn_yes.setText("Yes")
            btn_no = msg_box.button(QtWidgets.QMessageBox.No)
            btn_no.setText("No")

        msg_box.setDefaultButton(QtWidgets.QMessageBox.Yes)
        msg_box.setEscapeButton(QtWidgets.QMessageBox.No)
        msg_box.setTextInteractionFlags(QtCore.Qt.TextSelectableByMouse)  #
QtCore.Qt.NoTextInteraction
        r = msg_box.exec_()
        if r == QtWidgets.QMessageBox.Yes:
            return True
        else:
            return False

    @classmethod
    def get_last_directory(cls):
        data = cls.__load_default_config()
        return data.get("last_dir", config.base_dir)

    @classmethod
    def save_last_directory(cls, dir_path):
        if not isinstance(dir_path, str):
            raise TypeError
        if not os.path.exists(dir_path):
            raise FileNotFoundError
        if not os.path.isdir(dir_path):
            raise ValueError
        data = cls.__load_default_config()
        data["last_dir"] = dir_path
        cls.__save_default_config(data=data)

    @classmethod
    def __save_default_config(cls, data: dict):
        if not isinstance(data, dict):
            raise TypeError
        with open(config.app_config_fp, "w", encoding="utf-8") as f:
            json.dump(data, f)

```

```
120     @classmethod
121     def __load_default_config(cls):
122         data = {}
123         if not os.path.exists(config.app_config_fp):
124             return data
125         if not os.path.isfile(config.app_config_fp):
126             return data
127         with open(config.app_config_fp, "r", encoding="utf-8") as f:
128             data = json.load(f)
129         return data
130 
```

```python
#widgets.py

# coding: utf-8

from utils import QtCore, QtWidgets, QtGui, AbstractFunction
import config
from ui_design.ui_design import Ui_Form
import os
import typing


class WidgetMain(QtWidgets.QWidget, Ui_Form, AbstractFunction):
    def __init__(self, parent=None):
        super().__init__(parent=parent)
        self.setupUi(self)

        layout = QtWidgets.QVBoxLayout(self)

        layout.addWidget(self.group_viewer)
        layout.insertWidget(1, self.group_choose_image)

        layout.insertWidget(1, self.group_decide_position)
        self.group_decide_position.hide()

        layout.insertWidget(1, self.group_input_text)
        self.group_input_text.hide()

        layout.setStretchFactor(self.group_viewer, 1)

        self.setLayout(layout)

        self.current_image = ""
        self.current_pix = QtGui.QPixmap()
        self.text_pos = 0, 0, 0, 0

        self.graphics_viewer.sig_position.connect(self.handle_user_pick_position)

        self.btn_choose_image.clicked.connect(self.handle_choose_a_image)
        self.btn_apply_image.clicked.connect(self.handle_switch_to_decide_position)
        self.btn_goback_image.clicked.connect(self.handle_switch_to_choose_image)
        self.btn_apply_position.clicked.connect(self.handle_switch_to_input_text)

self.btn_goback_position.clicked.connect(self.handle_switch_to_decide_position)

        self.edit_x.setReadOnly(True)
        self.edit_y.setReadOnly(True)
        self.edit_w.setReadOnly(True)
        self.edit_h.setReadOnly(True)

        self.setWindowTitle(config.app_name_cn)

        self.btn_apply_text.clicked.connect(self.handle_display_text)
        self.btn_save_image.clicked.connect(self.handle_save_image)
        self.btn_check.setCheckState(QtCore.Qt.Checked)
        self.btn_check.clicked.connect(self.handle_toggle_rect)

    def handle_choose_a_image(self):
        ld = self.get_last_directory()
        fp, _etx = QtWidgets.QFileDialog.getOpenFileName(
            parent=self, caption="Choose An Image", directory=ld, filter="Image
Files(*.jpg *.jpeg *.png)"
```

```
60          )
61          if not fp:
62              return
63          fp = os.path.abspath(fp)
64          self.current_image = fp
65          self.current_pix = QtGui.QPixmap(fp)
66          # print(self.current_pix.size())
67          self.save_last_directory(dir_path=os.path.dirname(fp))
68          self.graphics_viewer.display_pix(pix=self.current_pix)
69
70      def handle_switch_to_decide_position(self):
71          if not self.current_image:
72              return self.show_warning_message(
73                  message="Please Choose An Image", title="Please Choose Image",
    parent=self, only_yes=True
74              )
75          self.__hide_all()
76          self.group_decide_position.show()
77          self.graphics_viewer.setAllowDrawRect(True)
78          self.graphics_viewer.clear_text()
79          self.graphics_viewer.display_rect()
80
81      def handle_switch_to_choose_image(self):
82          self.__hide_all()
83          self.group_choose_image.show()
84          self.graphics_viewer.setAllowDrawRect(False)
85          self.graphics_viewer.hide_rect()
86
87      def __hide_all(self):
88          for i in [self.group_choose_image, self.group_decide_position,
    self.group_input_text]:
89              i.hide()
90
91      def handle_switch_to_input_text(self):
92          if self.text_pos == (0, 0, 0, 0):
93              return self.show_warning_message(
94                  message="Please choose an area!", parent=self, title="Warning",
    only_yes=True
95              )
96
97          self.__hide_all()
98          self.group_input_text.show()
99          self.graphics_viewer.setAllowDrawRect(False)
100         self.handle_toggle_rect()
101
102     def handle_user_pick_position(self, text_pos):
103         # save and update the chosen area
104         x, y, w, h = text_pos
105         self.edit_x.setText(f"{x:.0f}")
106         self.edit_y.setText(f"{y:.0f}")
107         self.edit_w.setText(f"{w:.0f}")
108         self.edit_h.setText(f"{h:.0f}")
109
110         self.text_pos = text_pos
111
112     def handle_display_text(self):
113         txt = self.edit_txt.toPlainText()
114         # txt.setWordWrapMode(QtGui.QTextOption.WrapMode)
115         if not txt:
116             return self.show_warning_message(
117                 message="Please input some text", parent=self, title="Warning",
    only_yes=True
```

```
            )

        font = self.font_chooser.currentFont()
        size = self.size_chooser.value()
        font.setPixelSize(size)
        metrics = QtGui.QFontMetrics(font)
        x, y, w, h = self.text_pos

        r = metrics.boundingRect(int(x), int(y), int(w), int(h),
QtCore.Qt.AlignLeft, txt)

        if r.width() * r.height() > w * h:
            resp = self.show_warning_message(
                message="The text is too long for the area being chosen. " "Try
break long lines?",
                title="Warning",
                parent=self,
                only_yes=False,
            )
            if resp is False:
                return self.graphics_viewer.display_text(txt=txt, font=font)

        def break_line(line: str, width: int) -> typing.List[str]:
            length = len(line)
            ls = [0]
            while True:
                last = ls[-1]
                t = line[last:]
                rect = metrics.boundingRect(0, 0, 0, 0, QtCore.Qt.AlignLeft, t)
                s = rect.width()
                if s <= width:
                    break
                for i in range(1, length):
                    t = line[last : last + i]
                    rect = metrics.boundingRect(0, 0, 0, 0, QtCore.Qt.AlignLeft, t)
                    s = rect.width()

                    if s == width:
                        ls.append(last + i)
                        break
                    elif s > width:
                        ls.append(last + i - 1)
                        break

            ls.append(length)
            results = []
            for i in range(len(ls) - 1):
                results.append(line[ls[i] : ls[i + 1]])
            return results

        lines = txt.split("\n")  # break line
        pieces = []

        for line in lines:
            pieces.extend(break_line(line=line, width=w))

        r = metrics.boundingRect(int(x), int(y), int(w), int(h),
QtCore.Qt.AlignLeft, "\n".join(pieces))

        if r.width() * r.height() > w * h:
            resp = self.show_warning_message(
                message="The area your chosen can not hold that much txt your typed
```

```
even after line break. "
                "Hit Yes, the txt on the image will be cleared",
                title="Warning",
                parent=self,
                only_yes=False,
            )
            if resp:
                return self.graphics_viewer.clear_text()
        self.graphics_viewer.display_text("\n".join(pieces), font=font)

    def handle_save_image(self):
        """save modified image"""
        ld = self.get_last_directory()
        fp, _ext = QtWidgets.QFileDialog.getSaveFileName(
            parent=self, caption="Save image", directory=ld, filter="Png
Image(*.png)"
        )
        if not fp:
            return
        fp = os.path.abspath(fp)
        self.save_last_directory(dir_path=os.path.dirname(fp))
        pix = self.graphics_viewer.save_image()
        pix.save(fp, "PNG")

    def handle_toggle_rect(self):
        if self.btn_check.checkState() == QtCore.Qt.Checked:
            self.graphics_viewer.display_rect()
        else:
            self.graphics_viewer.hide_rect()


# if __name__ == '__main__':
#     import sys
#     app = QtWidgets.QApplication(sys.argv)
#     m = WidgetMain()
#     m.show()
#     sys.exit(app.exec_())

```

Bibliography

[1] Wikipedia - GUI, https://en.wikipedia.org/wiki/Graphical_user_interface

[2] Wikipedia - Qt (software),
https://en.wikipedia.org/wiki/Qt_(software)

[3] Wikipedia - Qt Creator,
https://en.wikipedia.org/wiki/Qt_Creator

[4] Python - Introduction to PyQt5,
https://www.geeksforgeeks.org/python-introduction-to-pyqt5/

[5] Wikipedia - Visual Studio Code,
https://en.wikipedia.org/wiki/Visual_Studio_Code

[6] Wikipedia - Computer Vision,
https://en.wikipedia.org/wiki/Computer_vision

[7] OpenCV-Python,
https://www.geeksforgeeks.org/python-opencv-cv2-imread-method/

[8] Computer Vision for Beginners: Part 4,
https:
//towardsdatascience.com/computer-vision-for-beginners-part-4-64a8d9856208

[9] opencv function docs,
https://docs.opencv.org/4.1.2/index.html