```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Jan 22 11:10:18 2021

@author: fmipro.com
"""
import sys

from PyQt5.QtCore import (QSettings, pyqtSlot,  Qt)

from ui_dialog import aboutDialog, prefsDialog, errorDialog

from class_menu import Ui_MainMenu

from class_form import form_make

from class_table import TableWidget

from class_spell_lineEdit import spelling

from class_PDF import designPDF

from PyQt5.QtWidgets import (QMainWindow, QApplication, QVBoxLayout, QWidget,
                             QPushButton,  QHBoxLayout, QTabWidget,
                             QTextEdit, QFileDialog)


def loadDict():
    counter = 0
    i = 0
    key = []
    a_dict = {}

    with open("dict.txt") as f:
        for line in f:
            if line == "\n" and line != int() and line != str():
                counter += 1
            else:
                x = line.startswith('<>')
                if x:
                    (symbol, nowKey) = line.split()
                    key = nowKey
                    url = ("{}-url".format (nowKey))
                else:
                    try:
                        (item1, item2) = line.split(", ")
```

```
                        if key in a_dict:
                            a_dict[key].append(item1.strip())
                        if url in a_dict:
                            a_dict[url].append(item2.strip())
                        else:
                            a_dict[key] = [item1.strip()]
                            a_dict[url] = [item2.strip()]
                    except:
                        item = line.strip()
                        if key in a_dict:
                            a_dict[key].append(item.strip())
                        else:
                            a_dict[key] = [item.strip()]

    print("counter counted  this many empty line", counter)
    for key, value in a_dict.items():
        print ("key ", counter, " is ",   key)
        i += 1



    return a_dict


class MainWindow(QMainWindow):
    def __init__(self, parent=None):
        super(MainWindow, self).__init__(parent)
        widget = QWidget()
        self.setCentralWidget(widget)
        self.saveFlag = 0
        self.resize(1600,600)
        self.move(100,100)
        self.formData = {}
        self.same_as = False
        self.flag = 0
        self.filename =['']
        self.TabNumber = 0
        self.t_data = {}
        self.rc = []
        self.setFocus(Qt.MouseFocusReason)
        self.spell = spelling(self)#(self.edit, self.text_box)
        self.PDF = designPDF()


        data_dict = loadDict()

        Hlabel = data_dict['Header_data2']
```

```
Hlabel2 = data_dict['Header_data1']

#init tables
table = TableWidget(self)
self.table = table
table.init_table("table", Hlabel, data_dict)
table2 = TableWidget(self)
table2.init_table("table2", Hlabel2, data_dict)


####---------------- Settings initialization ---------------####
self.settings = QSettings("fmipro", "cve_app")

try:
    self.resize(self.settings.value('window size'))
    self.move(self.settings.value('window possition'))
except:
    pass



        # string value
title = "Job Site Quote Application"

# set the title
self.setWindowTitle(title)

self.list = ["Quote Info", "Billing Same", "Billing Country",
             "Billing Name", "Billing Contact", "Billing Phone",
             "Billing Email","Billing Address", "Billing State",
        "Company Billing", "Company Country", "Company Name",
        "Company Contact","Company Phone","Company Email",
        "Company Address", "Company State", "Site Country",
        "Site Name", "Site Contact", "Site Phone", "Site Email",
        "Site Address", "Site State", "Site Room","Site Info",
        "Engineer Country", "Engineer Name", "Engineer Lead",
        "Engineer Contact", "Engineer Phone", "Engineer Email",
        "Engineer Address", "Engineer State", "Engineer Date",
        "Engineer Info"]


#1a3e6e is CVE Blue

app.setStyleSheet('''
    QHeaderView::section {
        background-color: rgb(26, 62, 110, 150);
        border: 1px solid #C4C4C3;
```

```
        color: rgb(255, 255, 255)
                        }

    QTabBar::tab {
        background: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1,
        stop: 0 #E1E1E1, stop: 0.4 #DDDDDD,
        stop: 0.5 #D8D8D8, stop: 1.0 #D3D3D3);
        color: #1a3e6e;
        border: 2px solid #C4C4C3;
        border-bottom-color: #C2C7CB; /* same as the pane color */
        border-top-left-radius: 4px;
        border-top-right-radius: 4px;
        min-width: 8ex;
        padding: 2px;
                }

     /* make use of negative margins for overlapping tabs */

     QTabBar::tab:selected {
     /* expand/overlap to the left and right by 4px */
         margin-left: -4px;
         margin-right: -4px;
         }

     QTabBar::tab:first:selected {
         margin-left: 0;
/* the first selected tab has nothing to overlap with on the left */
                        }

     QTabBar::tab:last:selected {
         margin-right: 0;
/* the last selected tab has nothing to overlap with on the right */
                        }

     QTabBar::tab:selected {
         border-color: #1a3e6e; /*#9B9B9B;*/
         border-bottom-color: #C2C7CB; /* same as pane color */
                        }

     QTabBar::tab:!selected {
         margin-top: 4px; /* make non-selected tabs look smaller */
                        }

     QLineEdit {
         background-color: white;
         border-style: outset;
         border-width: 2px;
```

```
        border-color: rgb(26,62,110);
        }

                    QCheckBox {
                        width: 200px;
                        height: 24px;
                        }
                        ''')


list = ["Billing Country", "Company Country",  "Site Country",
        "Engineer Country"]
for items in list:
    key = items
    value = "USA"
    self.formData[key]=(value)

mainLayout = QHBoxLayout()

#Init the tabs
tabs = QTabWidget()
tabs.setDocumentMode(True)
tabs.setTabPosition(QTabWidget.North)
tabs.setMovable(True)
tab1 = QWidget()
tab2 = QWidget()
tab3 = QWidget()



# Create first tab
L1 = QHBoxLayout()
LV = QVBoxLayout()

L1V = QVBoxLayout()
L1V2 = QVBoxLayout()
L1V3 = QVBoxLayout()


####------------ make tab 1 --------#######

self.bubbalogo = form_make(self, "Clogo")
self.bubbaQinfo = form_make(self,"Qinfo")
self.bubba1 = form_make(self,"1up")
self.bubba2 =form_make(self,"1down")
```

```
self.bubba3 = form_make(self,"2up")
self.bubba4 =form_make(self,"2down")
self.bubba5 = form_make(self,"3up")
self.bubba6 =form_make(self,"3down")
self.set_prefs()

L1V.addWidget(self.bubba1)
L1V.addWidget(self.bubba2)

L1V2.addWidget(self.bubba3)
L1V2.addWidget(self.bubba4)

L1V3.addWidget(self.bubba5)
L1V3.addWidget(self.bubba6)

LV.addWidget(self.bubbalogo)
LV.addWidget(self.bubbaQinfo)

L1.addLayout(LV)
L1.addLayout(L1V)
L1.addLayout(L1V2)
L1.addLayout(L1V3)

tab1.setLayout(L1)


####------------ make tab 1 --------#######
L2 = QHBoxLayout()

buttonLayout = QVBoxLayout()

button_new = QPushButton('New')
button_new.clicked.connect(table.Automate_input)
self.rc = button_new.clicked.connect(lambda: table.rowCount())
buttonLayout.addWidget(button_new)

button_remove = QPushButton('Remove')
button_remove.clicked.connect(table._removeRow)
buttonLayout.addWidget(button_remove)

button_next = QPushButton('next')
button_next.clicked.connect(lambda: table2.set_table_data(2,
                                                          self.t_data))
button_next.clicked.connect(lambda: tabs.setCurrentIndex(2))
buttonLayout.addWidget(button_next)

L2.addLayout(buttonLayout)
```

```
        L2.addWidget(table)
        tab2.setLayout(L2)

        # Create third tab
        L3 = QHBoxLayout()
        L3.addWidget(table2)

        buttonLayout1 = QVBoxLayout()
        button_set1 = QPushButton('set data')
        button_set1.clicked.connect(lambda: table2.set_table_data(self.t_data))

        buttonLayout1.addWidget(button_set1)
        button_new1 = QPushButton('populate')
        button_new1.clicked.connect(table2.populate)

        buttonLayout1.addWidget(button_new1)
        button_remove1 = QPushButton('save_db')
        button_remove1.clicked.connect(table2.save_db)
        buttonLayout1.addWidget(button_remove1)

        button_save1 = QPushButton('save_pdf')
        button_save1.clicked.connect(table2.save_pdf)
        buttonLayout1.addWidget(button_save1)

        L3.addLayout(buttonLayout1)
        tab3.setLayout(L3)


        tabs.addTab(tab1, "Job Site Information")
        tabs.addTab(tab2, "Materials and Labor")
        tabs.addTab(tab3, "Job Quote and Export")


        self.handle_tabbar_clicked(0)
        tabs.tabBarClicked.connect(self.handle_tabbar_clicked)

        self.ui = Ui_MainMenu(self)
        self.ui.setupUi(self)
        mainLayout.addWidget(tabs)
        widget.setLayout(mainLayout)


    def mousePressEvent(self, QMouseEvent):
        focused = QApplication.focusWidget()
        if QMouseEvent.button() == Qt.RightButton:
            if isinstance(focused, QTextEdit):
                self.spell.mousePressEvent(QMouseEvent)
```

```
def set_prefs(self):
    prefs = self.settings.value('configure_pres_1')
    data3 = {}
    data1 = {}
    data2 = {}
    for key, value in prefs.items():
        (first, second) = key.split()
        if first == "Engineer":
            data2[key]= value
        if first == "Font":
            data1[key]=value
        if first == "Export":
            data3[key]=value

    self.fill_form(data2)

def set_self_data(self, data):
    self.t_data = data

    return self.t_data

def get_self_data(self):

    return self.t_data


def Clicked(self, b):
    if b.text() == "Billing Same":
        if b.isChecked() == True and self.flag == 0:
            self.flag = 1
            self.bubba2.hide()
            self.same_as = True
            self.Save_form("Company Billing", "True")
            if self.same_as == True:

                la = ["Company Country","Company Name", "Company Contact",
                      "Company Phone","Company Email","Company Address",
                      "Company State"]
                lb = ["Billing Country", "Billing Name", "Billing Contact",
                      "Billing Phone", "Billing Email","Billing Address",
                      "Billing State"]
                i = 0
                for key in la:
                    value = self.formData[key]
                    self.formData[lb[i]]=(value)
                    i+=1
```

```
        elif b.isChecked() == False and self.flag == 1:
            self.flag = 0
            self.Save_form("Company Billing", "False")
            self.bubba2.show()
        else:
            print("error in hide billing error error")

    def Save_form(self, name, name1):

        if name != "Company Billing":
            try:
                name1 = name1.text()
            except:
                try:
                    name1 = name1.toPlainText()
                except:
                    name1 = name1.currentText()

        else:
            name = name

        for item in self.list:
            if item == name:
                key = name
                value = name1
                self.formData[key]=(value)

    def file_new(self):

        default_dtat2 ={'Quote Info': '', 'Billing Same': '',
                        'Billing Country': 'USA', 'Billing Name': '',
                        'Billing Contact': '', 'Billing Phone': '',
                        'Billing Email': '', 'Billing Address': '',
                        'Billing State': '', 'Company Billing': 'False',
                         'Company Country': 'USA', 'Company Name': '',
                        'Company Contact': '', 'Company Phone': '',
                        'Company Email': '', 'Company Address': '',
                        'Company State': '', 'Site Country': 'USA',
                        'Site Name': '', 'Site Contact': '', 'Site Phone': '',
                        'Site Email': '', 'Site Address': '', 'Site State': '',
                        'Site Room': '', 'Site Info': '',
                        'Engineer Country': 'USA', 'Engineer Name': '',
                        'Engineer Lead': '', 'Engineer Contact': '',
                        'Engineer Phone': '', 'Engineer Email': '',
                        'Engineer Address': '', 'Engineer State': '',
                        'Engineer Date': '', 'Engineer Info': ''}
```

```
        self.filename = ['']
        self.table.remove_all_rows()
        self.fill_form(default_dtat2)

    def file_just_save(self):
        if self.filename != ['']:

            with open(self.filename, 'w') as f:
                f.write("<> Form_Data \n")
                for item in self.list:
                    if item in self.formData.keys():
                        value = self.formData[item]
                        try:
                            value = value.replace("\n", "\n"+item+" >:< ")
                        except:
                            value = value
                        data = (item+" >:< "+str(value)+"\n")
                    else:
                        if item == "Company Billing":
                            data = (item+" >:< "+str('False')+"\n")
                        else:
                            data = (item+" >:< "+str('')+"\n")

                    f.write(data)
        else:
            self.file_save

    def file_save(self):
        seggestedFileName = "CVE_customer"
        options = QFileDialog.Options()

        fileName, _ = QFileDialog.getSaveFileName(
                    self, "QFileDialog.getSaveFileName()", seggestedFileName,
                    "CVE Files (*.cve);;All Files (*)",  options=options)

        if fileName:
            print(fileName)

        self.filename = fileName

        with open(self.filename, 'w') as f:
            f.write("<> Form_Data \n")
            for item in self.list:
                if item in self.formData.keys():
                    value = self.formData[item]
                    try:
                        value = value.replace("\n", "\n"+item+" >:< ")
```

```
                except:
                    value = value
                data = (item+" >:< "+str(value)+"\n")

            else:
                if item == "Company Billing":
                    data = (item+" >:< "+str('False')+"\n")
                else:
                    data = (item+" >:< "+str('')+"\n")
            f.write(data)

        df_list = []
        data = self.t_data

        for key, value in data.items():
            df_list.append("<r> "+str(key))
            for item in value:
                df_list.append(item)

        f.write("<> Table_Data \n")
        for item in df_list:
            data = (item+"\n")

            f.write(data)

def file_load(self):

    options = QFileDialog.Options()

    fileName, _ = QFileDialog.getOpenFileName(
        self,"QFileDialog.getOpenFileName()", "",
        "CVE Files (*.cve);;All Files (*)",  options=options)

    if fileName:
        print("load file named ---",fileName)

    self.filename = fileName
    counter = 0
    i = 0
    ignore = 0
    key = []
    data = {}
    data2 = {}

    with open(self.filename) as f:
        value = []
        data_type = []
```

```
    for line in f:
        if line == "\n" and line != int() and line != str():
            counter += 1
        else:
            x = line.startswith('<>')
            if x:
                (symbol, data_type) = line.split()
                #print("data_type ------ ", data_type)

            else:

                if data_type == "Form_Data":
                    (key, value) = line.split(" >:< ")
                    value = value.strip()
                    if key in data2:
                        data2[key]+=("\n"+value.strip())
                    else:
                       data2[key] = value
                if data_type == "Table_Data":
                    if line.startswith('<>'):
                        ignore += 1
                    elif line.startswith('<r>'):
                        (symbol, key) = line.split()
                    else:
                        value = line.strip()
                        if key in data:
                            data[key].append(value)
                        else:
                            data[key] = [value]



    if data_type == "Form_Data":
        for key, value in data2.items():
            print (key, " ; ", value)
            i += 1

    if data_type == "Table_Data":
        for key, value in data.items():
            print ("key ", counter, " is ",   key)
            i += 1

self.fill_form(data2)
self.formData = data2
self.table.fill_form(data)
```

```
def fill_form(self, data2):
    data_l2 = {}
    data_b1 = {}
    data_b2 = {}
    data_b3 = {}
    data_b4 = {}
    data_b5 = {}
    data_b6 = {}

    for key, value in data2.items():
        (pre, post) = key.split()
        if pre == "Company":
            data_b1[key] = value
        if pre == "Billing":
            data_b2[key] = value
        if pre == "Site" and post != "Info":
            data_b3[key] = value
        if pre == "Site" and post == "Info":
            data_b4[key] = value
        if pre == "Quote":
            data_l2[key] = value
            #print("\nQuote has been set \n", data_l2, "\n")
        if pre == "Engineer" and post != "Info":
            data_b5[key] = value
        if pre == "Engineer" and post == "Info":
            data_b6[key] = value
    if data_l2:
        self.bubbaQinfo.fill_form(data_l2)
    if data_b1:
        self.bubba1.fill_form(data_b1)
    if data_b2:
        self.bubba2.fill_form(data_b2)
    if data_b3:
        self.bubba3.fill_form(data_b3)
    if data_b4:
        self.bubba4.fill_form(data_b4)
    if data_b5:
        self.bubba5.fill_form(data_b5)
    if data_b6:
        self.bubba6.fill_form(data_b6)


def open_form(self, name, name1):
    n = 1


def handle_tabbar_clicked(self, index):
```

```
        if index == 1:
            for key, value in self.formData.items():
                print (key, " : ", value)

        if index == 2:
            empty = []

        print("tab index is", index)

    def closeEvent(self, event):
        self.settings.setValue('window size', self.size())
        self.settings.setValue('window position', self.pos())
        print(self.settings.fileName())


    def window_pos(self):
        print("MainWindow Possition is ",self.pos().x(), self.pos().y())
        return self.pos().x(), self.pos().y()

    def errorCode(string):
        error = errorDialog("please use saveas")
        error.show()
        if error.exec_():
            print('ok error has been seen \n')

        else:
            print("Cancel!")


    @pyqtSlot(bool)
    def on_setPreferencesAction_triggered(self, triggered):
        #settings = self.settings
        self.prefs_dict = {}
        try:
            self.prefs_dict = (self.settings.value('configure_pres_1'))
        except:
            print("no presets saved")

        wp = self.window_pos()
        prefs = prefsDialog()
        prefs.setWP(wp)
        prefs.fill_form(self.prefs_dict)

        prefs.show()
        if prefs.exec_():
            self.settings.setValue('configure_pres_1', prefs.send_prefs())
```

```
        else:
            print("Cancel!")


    @pyqtSlot(bool)
    def on_setAboutAction_triggered(self, triggered):
        info2 = aboutDialog()
        info2.show()

    @pyqtSlot(bool)
    def on_actionSpell_triggered(self, triggered):
        self.spell.spellCheck()


    @pyqtSlot(bool)
    def on_actionNew_triggered(self, triggered):
        self.file_new()
        self.saveFlag = 0

    @pyqtSlot(bool)
    def on_actionOpen_triggered(self, triggered):
        self.file_load()
        self.saveFlag = 1

    @pyqtSlot(bool)
    def on_actionSave_triggered(self, triggered):
        if self.saveFlag == 1:
            self.file_just_save() #file has a name already
        else:
            self.file_save()      #file needs a name

    @pyqtSlot(bool)
    def on_actionSaveas_triggered(self, triggered):
        self.file_save()      #file needs a name
        self.saveFlag = 1

    @pyqtSlot(bool)
    def on_actionPDF_triggered(self, triggered):

        seggestedFileName = "CVE_customer"
        options = QFileDialog.Options()
        fileName, _ = QFileDialog.getSaveFileName(
            self,"QFileDialog.getSaveFileName()",
            seggestedFileName,"PDF Files (*.pdf);;All Files (*)",
            options=options)

        if fileName:
```

```
            print(fileName)

        self.filename = fileName

        data = self.formData

        data2 = self.table.save_table_data()
        self.PDF.makePDF(fileName,data, data2)

    @pyqtSlot(bool)
    def on_actionExport_triggered(self, triggered):
        self.table.save_table_data()
        if self.saveFlag == 1:
            print("exportttttttttt")

        else:
            print("you must save your file first")


    @pyqtSlot(bool)
    def on_actionCut_triggered(self, triggered):
        focused = QApplication.focusWidget()
        if focused != 0:
            focused.cut()

    @pyqtSlot(bool)
    def on_actionCopy_triggered(self, triggered):
        focused = QApplication.focusWidget()
        if focused != 0:
            focused.copy()

    @pyqtSlot(bool)
    def on_actionPaste_triggered(self, triggered):
        focused = QApplication.focusWidget()
        if focused != 0:
            focused.paste()

    @pyqtSlot(bool)
    def on_actionSelectall_triggered(self, triggered):
        focused = QApplication.focusWidget()
        if focused != 0:
            focused.selectAll()

    @pyqtSlot(bool)
    def on_actionUndo_triggered(self, triggered):
        focused = QApplication.focusWidget()
        if focused != 0:
```

```
            focused.undo()

    @pyqtSlot(bool)
    def on_actionRedo_triggered(self, triggered):
        focused = QApplication.focusWidget()
        if focused != 0:
            focused.redo()

if __name__ == '__main__':

    app = QApplication(sys.argv)
    window = MainWindow()
    window.show()
    sys.exit(app.exec_())
```