Appendix

Codes Used:

import sys

from PyQt4 import QtGui, uic, QtCore

from PyQt4.QtGui import \*

from PyQt4.QtCore import \*

from firebase import firebase

import datetime

import time

import RPi.GPIO as GPIO

import threading

def runProgram():

try:

GPIO.setmode(GPIO.BOARD)

GPIO.setwarnings(False)

timeToOff = 120 #120 SECONDS

PIN\_TRIGGER = 7

PIN\_ECHO = 11

PIN\_RELAY = 16

GPIO.setup(PIN\_TRIGGER, GPIO.OUT)

GPIO.setup(PIN\_ECHO, GPIO.IN)

GPIO.setup(PIN\_RELAY, GPIO.OUT)

GPIO.output(PIN\_TRIGGER, GPIO.LOW)

time.sleep(2)

while 1:

GPIO.output(PIN\_TRIGGER, GPIO.HIGH)

time.sleep(0.00001)

GPIO.output(PIN\_TRIGGER, GPIO.LOW)

while GPIO.input(PIN\_ECHO) == 0:

pulse\_start\_time = time.time()

while GPIO.input(PIN\_ECHO) == 1:

pulse\_end\_time = time.time()

pulse\_duration = pulse\_end\_time - pulse\_start\_time

distance = round(pulse\_duration \* 17150 \* 0.393701, 2)

if distance <= 24: #24in Distance (2 RULERS)

GPIO.output(PIN\_RELAY, GPIO.HIGH)

time.sleep(timeToOff)

else:

GPIO.output(PIN\_RELAY, GPIO.LOW)

time.sleep(0.001)

except KeyboardInterrupt:

print("Program exits.")

finally:

GPIO.cleanup()

th = threading.Thread(target=runProgram)

th.daemon = True

th.start()

try:

GPIO.setmode(GPIO.BOARD)

GPIO.setwarnings(False)

PINBUZZER = 18

GPIO.setup(PINBUZZER, GPIO.OUT)

database = firebase.FirebaseApplication('https://mapuafacultymanagingsystem.firebaseio.com/', authentication=None)

directory = '/home/pi/ECE110D/'

class DoThreading(QThread):

def \_\_init\_\_(self, \_func):

self.func = \_func

QThread.\_\_init\_\_(self)

def \_\_del\_\_(self):

self.wait()

def run(self):

self.func()

class MyWindow(QtGui.QMainWindow):

def \_\_init\_\_(self):

try:self.close()

except:pass

res = ((database.get("", None)))

super(MyWindow, self).\_\_init\_\_()

uic.loadUi(directory + 'AttendanceAnnouncement.ui', self)

self.setFixedSize(self.size())

self.setWindowFlags(QtCore.Qt.FramelessWindowHint)

self.center()

self.show()

self.pushButtonF.clicked.connect(lambda : self.stackedWidget.setCurrentIndex(self.stackedWidget.currentIndex() + 1))

self.pushButtonB.clicked.connect(lambda : self.stackedWidget.setCurrentIndex(self.stackedWidget.currentIndex() - 1))

self.pushButtonStudent.clicked.connect(self.openStudentUI)

self.pushButtonFaculty.clicked.connect(lambda : self.openLoginUI("Faculty", self.openFacultyUI, self.\_\_init\_\_))

self.pushButtonDatabase.clicked.connect(lambda : self.openLoginUI("Database", self.openDatabaseUI, self.\_\_init\_\_))

self.pushButtonAnnouncement.clicked.connect(lambda : self.openLoginUI("Database", self.showSecretearyAnnouncementUI, self.\_\_init\_\_))

date = str(datetime.datetime.now().strftime('%b-%d-%Y'))

#date = "Aug-15-2018"

self.listWidgetEE.clear()

self.listWidgetECE.clear()

self.listWidgetCPE.clear()

self.labelDate.setText(date)

if (res != None):

if (("Attendance" in res) and

(date in res["Attendance"])):

for k, v in res["Attendance"][date].items():

if k == "EE":

for k1, v1 in v.items():

if ("In" in v1 and not "Out" in v1):

if ("FacultyMembers" in res and

k in res["FacultyMembers"] and

k1 in res["FacultyMembers"][k]):

#print(res["FacultyMembers"][k][k1]["Name"])

self.listWidgetEE.addItem(QListWidgetItem(str(res["FacultyMembers"][k][k1]["Name"])+"\n-"+v1["In"]))

if k == "ECE":

for k1, v1 in v.items():

if ("In" in v1 and not "Out" in v1):

if ("FacultyMembers" in res and

k in res["FacultyMembers"] and

k1 in res["FacultyMembers"][k]):

#print(res["FacultyMembers"][k][k1]["Name"])

self.listWidgetECE.addItem(QListWidgetItem(str(res["FacultyMembers"][k][k1]["Name"])+"\n-"+v1["In"]))

if k == "CPE":

for k1, v1 in v.items():

if ("In" in v1 and not "Out" in v1):

if ("FacultyMembers" in res and

k in res["FacultyMembers"] and

k1 in res["FacultyMembers"][k]):

#print(res["FacultyMembers"][k][k1]["Name"])

self.listWidgetCPE.addItem(QListWidgetItem(str(res["FacultyMembers"][k][k1]["Name"])+"\n-"+v1["In"]))

if ("Remarks" in res and

date in res["Remarks"] and

"SECRETARY" in res["Remarks"][date]):

self.plainTextEditA.setPlainText(res["Remarks"][date]["SECRETARY"]["Remarks"])

elif (not ("Remarks" in res and

date in res["Remarks"] and

"SECRETARY" in res["Remarks"][date])):

self.plainTextEditA.setPlainText("No Announcements.")

def showSecretearyAnnouncementUI(self):

res = ((database.get("Remarks/"+str(datetime.datetime.now().strftime('%b-%d-%Y'))+"/SECRETARY", None)))

self.close()

super(MyWindow, self).\_\_init\_\_()

uic.loadUi(directory + 'SecretaryAnnouncement.ui', self)

self.setFixedSize(self.size())

self.setWindowFlags(QtCore.Qt.FramelessWindowHint)

self.center()

self.show()

if (res != None):

if ("Remarks" in res):

self.plainTextEditAnnounce.setPlainText(str(res["Remarks"]))

self.labelDateToday.setText(str(datetime.datetime.now().strftime('%b-%d-%Y')))

self.pushButtonUpdate.clicked.connect(self.SAUIUpdate)

self.pushButtonBack.clicked.connect(self.\_\_init\_\_)

def SAUIUpdate(self):

database.patch("Remarks/"+str(datetime.datetime.now().strftime('%b-%d-%Y'))+"/SECRETARY",

{"Remarks":self.plainTextEditAnnounce.toPlainText()})

self.doMessage("Announcement updated.", "Information")

def center(self):

frameGm = self.frameGeometry()

screen = QtGui.QApplication.desktop().screenNumber(QtGui.QApplication.desktop().cursor().pos())

centerPoint = QtGui.QApplication.desktop().screenGeometry(screen).center()

frameGm.moveCenter(centerPoint)

self.move(frameGm.topLeft())

def doMessage(self, text, caption):

msg = QMessageBox(self)

msg.setText(text)

msg.setWindowTitle(caption)

msg.show()

return msg

def doQuestion(self, text, caption):

return QMessageBox.question(w, caption, text, QMessageBox.Yes | QMessageBox.No, QMessageBox.No)

def openStudentUI(self):

self.close()

super(MyWindow, self).\_\_init\_\_()

uic.loadUi(directory + 'Student.ui', self)

self.setFixedSize(self.size())

self.setWindowFlags(QtCore.Qt.FramelessWindowHint)

self.center()

self.show()

self.pushButtonBack.clicked.connect(self.\_\_init\_\_)

self.pushButtonRefresh.clicked.connect(self.cBDCIC)

self.comboBoxDepartment.currentIndexChanged.connect(self.cBDCIC)

self.cBDCIC()

def doReset(self):

self.labelID.setText("-")

self.plainTextEditRemarks.setPlainText("")

def cBDCIC(self):

self.comboBoxDate.clear()

self.doReset()

try:self.comboBoxDate.currentIndexChanged.disconnect()

except:pass

res = ((database.get("", None)))

if (res != None):

if ("Remarks" in res):

for k, v in res["Remarks"].items():

if (self.comboBoxDepartment.currentText() in res["Remarks"][k]):

self.comboBoxDate.addItem(k)

self.comboBoxDate.setCurrentIndex(-1)

self.comboBoxDate.currentIndexChanged.connect(lambda: self.cBDateCIC(res))

def cBDateCIC(self, res):

self.comboBoxFacultyMember.clear()

self.labelID.setText("-")

self.plainTextEditRemarks.setPlainText("")

self.comboBoxCourseCode.clear()

try:self.comboBoxFacultyMember.currentIndexChanged.disconnect()

except:pass

if (res != None):

if ("Remarks" in res and self.comboBoxDate.currentText() in res["Remarks"] and

self.comboBoxDepartment.currentText() in res["Remarks"][self.comboBoxDate.currentText()]):

for k,v in res["Remarks"][self.comboBoxDate.currentText()][self.comboBoxDepartment.currentText()].items():

if ("FacultyMembers" in res and self.comboBoxDepartment.currentText() in res["FacultyMembers"] and

k in res["FacultyMembers"][self.comboBoxDepartment.currentText()]):

self.comboBoxFacultyMember.addItem(res["FacultyMembers"][self.comboBoxDepartment.currentText()][k]["Name"])

self.comboBoxFacultyMember.setCurrentIndex(-1)

self.comboBoxFacultyMember.currentIndexChanged.connect(lambda: self.cBFacMCIC(res))

def cBFacMCIC(self, res):

self.comboBoxCourseCode.clear()

self.plainTextEditRemarks.setPlainText("")

try:self.comboBoxCourseCode.currentIndexChanged.disconnect()

except:pass

if (res != None):

if ("FacultyMembers" in res and self.comboBoxDepartment.currentText() in res["FacultyMembers"]):

for k,v in res["FacultyMembers"][self.comboBoxDepartment.currentText()].items():

if (v["Name"] == self.comboBoxFacultyMember.currentText()):

self.labelID.setText(str(k))

if ("Remarks" in res and self.comboBoxDate.currentText() in res["Remarks"] and

self.comboBoxDepartment.currentText() in res["Remarks"][self.comboBoxDate.currentText()] and

self.labelID.text() in res["Remarks"][self.comboBoxDate.currentText()][self.comboBoxDepartment.currentText()]):

for k,v in res["Remarks"][self.comboBoxDate.currentText()][self.comboBoxDepartment.currentText()][self.labelID.text()].items():

self.comboBoxCourseCode.addItem(k)

self.comboBoxCourseCode.setCurrentIndex(-1)

self.comboBoxCourseCode.currentIndexChanged.connect(lambda: self.cBCCIC(res))

def cBCCIC(self, res):

if (res != None):

if ("Remarks" in res and self.comboBoxDate.currentText() in res["Remarks"] and

self.comboBoxDepartment.currentText() in res["Remarks"][self.comboBoxDate.currentText()] and

self.labelID.text() in res["Remarks"][self.comboBoxDate.currentText()][self.comboBoxDepartment.currentText()] and self.comboBoxCourseCode.currentText() in res["Remarks"][self.comboBoxDate.currentText()]

[self.comboBoxDepartment.currentText()][self.labelID.text()]):

self.plainTextEditRemarks.setPlainText(res["Remarks"][self.comboBoxDate.currentText()]

[self.comboBoxDepartment.currentText()][self.labelID.text()][self.comboBoxCourseCode.currentText()]["Remarks"].replace("\"","").replace("\\n","\n"))

def openFacultyUI(self):

self.close()

super(MyWindow, self).\_\_init\_\_()

uic.loadUi(directory + 'Faculty.ui', self)

self.setFixedSize(self.size())

self.setWindowFlags(QtCore.Qt.FramelessWindowHint)

self.center()

self.show()

self.pushButtonBack.clicked.connect(self.\_\_init\_\_)

self.pushButtonTimeIn.clicked.connect(self.timeInClicked)

self.pushButtonTimeOut.clicked.connect(self.timeOutClicked)

self.comboBoxDepartment.currentIndexChanged.connect(self.currentIndexChanged)

self.pushButtonSubmit.clicked.connect(self.submitButtonClicked)

self.currentIndexChanged()

def doLoop():

while True:

self.labelTime.setText(str(datetime.datetime.now().strftime('%I:%M:%S %p')).upper())

self.labelDate.setText(str(datetime.datetime.now().strftime('%b-%d-%Y')))

time.sleep(1)

self.th = DoThreading(doLoop)

self.th.start()

def openLoginUI(self, database, ifTrue, ifFalse):

self.close()

super(MyWindow, self).\_\_init\_\_()

uic.loadUi(directory + 'Login.ui', self)

self.setFixedSize(self.size())

self.setWindowFlags(QtCore.Qt.FramelessWindowHint)

self.center()

self.show()

self.pushButtonLogin.clicked.connect(lambda : self.authenticateLogin(database, ifTrue, ifFalse))

self.pushButtonBack.clicked.connect(ifFalse)

def authenticateLogin(self, data, ifTrue, ifFalse):

getPassword = ((database.get("/Authentication", None)))

if (getPassword != None):

if (data == "Database" and "Database" in getPassword and "Username" in getPassword["Database"] and

"Password" in getPassword["Database"]):

if (str(self.lineEditUsername.text()) == str(getPassword["Database"]["Username"]) and

str(self.lineEditPassword.text()) == str(getPassword["Database"]["Password"])):

ifTrue()

else:

self.doMessage("Wrong username & password.", "Information")

ifFalse()

elif (data == "Faculty" and "Faculty" in getPassword and "Username" in getPassword["Faculty"] and

"Password" in getPassword["Faculty"]):

if (str(self.lineEditUsername.text()) == str(getPassword["Faculty"]["Username"]) and

str(self.lineEditPassword.text()) == str(getPassword["Faculty"]["Password"])):

ifTrue()

else:

self.doMessage("Wrong username & password.", "Information")

ifFalse()

def buzzThis(self):

GPIO.output(PINBUZZER, GPIO.HIGH)

time.sleep(1)

GPIO.output(PINBUZZER, GPIO.LOW)

def timeInClicked(self):

self.th = DoThreading(self.buzzThis)

self.th.start()

if (str(self.labelTimeIn.text()) == "-"):

res = ((database.get("/FacultyMembers/"+self.comboBoxDepartment.currentText()+"/"+self.labelID.text(), None)))

if (res != None):

if (str(self.lineEditPassword.text()) == res["Password"]):

self.labelTimeIn.setText(str(self.labelTime.text()));

database.patch("/Attendance/"+str(self.labelDate.text())+"/"+str(self.labelID.text()).split("-")[0]+"/"+

str(self.labelID.text()),{"In":str(self.labelTimeIn.text())})

else:

self.doMessage("Wrong password.", "Information")

else:

self.doMessage("You have already timed in for this day.", "Information")

self.lineEditPassword.clear()

#RemarksSubmit

def submitButtonClicked(self):

if (self.lineEditRemarksCC.text() != "" and

self.plainTextEditRemarks.toPlainText() != ""):

res = ((database.get("/FacultyMembers/"+self.comboBoxDepartment.currentText()+"/"+self.labelID.text(), None)))

if (res != None):

if (str(self.lineEditPassword.text()) == res["Password"]):

database.patch("/Remarks/"+self.labelDate.text()+"/"+self.comboBoxDepartment.currentText()+"/"+self.labelID.text()+"/"+self.lineEditRemarksCC.text(),{

"Remarks":'\"'+str(self.plainTextEditRemarks.toPlainText())+'\"'})

self.doMessage("Remarks submitted.", "Complete")

else:

self.doMessage("Password field is empty or Wrong password.", "Information")

else:

self.doMessage("Some remarks field are empty.", "Information")

self.lineEditPassword.clear()

def timeOutClicked(self):

self.th = DoThreading(self.buzzThis)

self.th.start()

if (str(self.labelTimeIn.text()) != "-"):

if (str(self.labelTimeOut.text()) == "-"):

res = ((database.get("/FacultyMembers/"+self.comboBoxDepartment.currentText()+"/"+self.labelID.text(), None)))

if (res != None):

if (str(self.lineEditPassword.text()) == res["Password"]):

self.labelTimeOut.setText(str(self.labelTime.text()));

database.patch("/Attendance/"+str(self.labelDate.text())+"/"+str(self.labelID.text()).split("-")[0]+"/"+

str(self.labelID.text()),{"Out":str(self.labelTimeOut.text())})

else:

self.doMessage("Wrong password.", "Information")

else:

self.doMessage("You have already timed out for this day.", "Information")

else:

self.doMessage("You have to time in first.", "Information")

self.lineEditPassword.clear()

def currentIndexChanged(self):

self.labelTimeIn.setText("-")

self.labelTimeOut.setText("-")

self.labelID.setText("-")

self.comboBoxFacultyMember.clear()

try:self.comboBoxFacultyMember.currentIndexChanged.disconnect()

except:pass

res = ((database.get("", None)))

if (res != None):

if ("FacultyMembers" in res and self.comboBoxDepartment.currentText() in res["FacultyMembers"]):

for k1, v1 in res["FacultyMembers"][self.comboBoxDepartment.currentText()].items():

if ("Name" in v1):

self.comboBoxFacultyMember.addItem(v1["Name"])

self.plainTextEditRemarks.setDisabled(True)

self.lineEditRemarksCC.setDisabled(True)

self.comboBoxFacultyMember.setCurrentIndex(-1)

self.comboBoxFacultyMember.currentIndexChanged.connect(lambda: self.currentIndexChangedFac(res))

def currentIndexChangedFac(self, res):

self.labelTimeIn.setText("-")

self.labelTimeOut.setText("-")

self.labelID.setText("-")

res = ((database.get("", None)))

if ("FacultyMembers" in res and self.comboBoxDepartment.currentText() in res["FacultyMembers"]):

for k1, v1 in res["FacultyMembers"][self.comboBoxDepartment.currentText()].items():

if ("Name" in v1):

if (str(v1["Name"]) == str(self.comboBoxFacultyMember.currentText())):

self.labelID.setText(str(k1))

if ("Attendance" in res and self.labelDate.text() in res["Attendance"]

and str(k1).split("-")[0] in res["Attendance"][self.labelDate.text()]

and str(k1) in res["Attendance"][self.labelDate.text()][str(k1).split("-")[0]]

and "In" in res["Attendance"][self.labelDate.text()][str(k1).split("-")[0]][str(k1)]):

self.labelTimeIn.setText(res["Attendance"][self.labelDate.text()][str(k1).split("-")[0]][str(k1)]["In"])

if ("Attendance" in res and self.labelDate.text() in res["Attendance"]

and str(k1).split("-")[0] in res["Attendance"][self.labelDate.text()]

and str(k1) in res["Attendance"][self.labelDate.text()][str(k1).split("-")[0]]

and "Out" in res["Attendance"][self.labelDate.text()][str(k1).split("-")[0]][str(k1)]):

self.labelTimeOut.setText(res["Attendance"][self.labelDate.text()][str(k1).split("-")[0]][str(k1)]["Out"])

self.plainTextEditRemarks.setDisabled(False)

self.lineEditRemarksCC.setDisabled(False)

def openDatabaseUI(self, number=None):

self.close()

super(MyWindow, self).\_\_init\_\_()

uic.loadUi(directory + 'Database.ui', self)

self.setFixedSize(self.size())

self.setWindowFlags(QtCore.Qt.FramelessWindowHint)

if (number != None):

self.tabWidget.setCurrentIndex(number)

self.center()

self.show()

self.pushButtonRefreshR.clicked.connect(self.tabWidgetTabChanged)

self.pushButtonRefreshA.clicked.connect(self.tabWidgetTabChanged)

self.pushButtonRefreshF.clicked.connect(self.tabWidgetTabChanged)

self.pushButtonBackF.clicked.connect(self.\_\_init\_\_)

self.pushButtonRegisterF.clicked.connect(self.openFRegisterUI)

self.pushButtonEditF.clicked.connect(self.openFEditUI)

self.pushButtonDeleteF.clicked.connect(self.deleteFMember)

self.pushButtonBackA.clicked.connect(self.\_\_init\_\_)

self.pushButtonEditA.clicked.connect(self.openAEditMainMenuUI)

self.tabWidget.currentChanged.connect(self.tabWidgetTabChanged)

self.comboBoxDepartmentF.currentIndexChanged.connect(self.tabWidgetTabChanged)

self.comboBoxDepartmentA.currentIndexChanged.connect(self.tabWidgetTabChanged)

self.comboBoxDepartmentR.currentIndexChanged.connect(self.tabWidgetTabChanged)

self.tableWidgetF.setColumnWidth(0, 200)

self.tableWidgetF.setColumnWidth(1, 600)

self.tableWidgetR.setColumnWidth(0, 200)

self.tableWidgetR.setColumnWidth(1, 600)

self.tableWidgetA.setColumnWidth(0, 200)

self.tableWidgetA.setColumnWidth(1, 600)

self.tableWidgetA.setColumnWidth(2, 200)

self.tabWidgetTabChanged()

self.pushButtonBackS.clicked.connect(self.\_\_init\_\_)

self.pushButtonSave.clicked.connect(self.savePasswords)

self.pushButtonBackR.clicked.connect(self.\_\_init\_\_)

self.pushButtonDeleteR.clicked.connect(self.deleteRMember)

self.pushButtonDeleteA.clicked.connect(self.deleteAMember)

def deleteRMember(self):

if (self.tableWidgetR.currentRow() != -1):

res = database.delete("/Remarks/"+self.comboBoxDateR.currentText()+"/"+self.tableWidgetR.item(self.tableWidgetR.currentRow(),

0).text().split("-")[0]+"/"+

self.tableWidgetR.item(self.tableWidgetR.currentRow(), 0).text(), None)

self.doMessage("Remarks data deleted.", "Information")

self.tabWidgetTabChanged()

else:

self.doMessage("Select a remarks data in the table view.", "Information")

def deleteAMember(self):

if (self.tableWidgetA.currentRow() != -1):

res = database.delete("/Attendance/"+self.comboBoxDateA.currentText()+"/"+self.tableWidgetA.item(self.tableWidgetA.currentRow(),

0).text().split("-")[0]+"/"+

self.tableWidgetA.item(self.tableWidgetA.currentRow(), 0).text(), None)

self.doMessage("Data deleted.", "Information")

self.tabWidgetTabChanged()

else:

self.doMessage("Select a data in the table view.", "Information")

def cmbdAIndexChanged(self, res):

self.comboBoxDateA.setDisabled(True)

self.comboBoxDepartmentA.setDisabled(True)

while (self.tableWidgetA.rowCount() > 0):

self.tableWidgetA.removeRow(0)

if (res != None):

if ("Attendance" in res and str(self.comboBoxDateA.currentText()) in res["Attendance"] and

self.comboBoxDepartmentA.currentText() in res["Attendance"][self.comboBoxDateA.currentText()]):

for k1, v1 in res["Attendance"][self.comboBoxDateA.currentText()][self.comboBoxDepartmentA.currentText()].items():

rowPosition = self.tableWidgetA.rowCount()

self.tableWidgetA.insertRow(rowPosition)

self.tableWidgetA.setItem(rowPosition , 0, QtGui.QTableWidgetItem(str(k1)))

if ("FacultyMembers" in res and str(k1).split("-")[0] in res["FacultyMembers"] and

str(k1) in res["FacultyMembers"][str(k1).split("-")[0]]

and "Name" in res["FacultyMembers"][str(k1).split("-")[0]][str(k1)]):

self.tableWidgetA.setItem(rowPosition , 1,

QtGui.QTableWidgetItem(res["FacultyMembers"][str(k1).split("-")[0]][str(k1)]["Name"]))

if ("In" in v1):

self.tableWidgetA.setItem(rowPosition , 2, QtGui.QTableWidgetItem(str(v1["In"])))

if ("Out" in v1):

self.tableWidgetA.setItem(rowPosition , 3, QtGui.QTableWidgetItem(str(v1["Out"])))

self.comboBoxDateA.setDisabled(False)

self.comboBoxDepartmentA.setDisabled(False)

def deleteFMember(self):

if (self.tableWidgetF.currentRow() != -1):

res = database.delete("/FacultyMembers/"+str(self.comboBoxDepartmentF.currentText())+"/"+

self.tableWidgetF.item(self.tableWidgetF.currentRow(), 0).text(), None)

self.doMessage("Faculty member deleted.", "Information")

self.tabWidgetTabChanged()

else:

self.doMessage("Select a faculty member in the table view.", "Information")

def savePasswords(self):

if (str(self.lineEditUsernameF.text()) != "" and str(self.lineEditPasswordF.text()) != ""

and str(self.lineEditUsernameD.text()) != "" and str(self.lineEditPasswordD.text()) != ""):

database.patch('Authentication', {'Faculty':{'Username':str(self.lineEditUsernameF.text()),

'Password':str(self.lineEditPasswordF.text())},

'Database':{'Username':str(self.lineEditUsernameD.text()),

'Password':str(self.lineEditPasswordD.text())}})

self.doMessage("Password updated succesfully.", "Information")

else:

self.doMessage("Some fields are empty.", "Information")

def cmbdRIndexChanged(self, res):

self.comboBoxDateR.setDisabled(True)

self.comboBoxDepartmentR.setDisabled(True)

while (self.tableWidgetR.rowCount() > 0):

self.tableWidgetR.removeRow(0)

if (res != None):

if ("Remarks" in res and str(self.comboBoxDateR.currentText()) in res["Remarks"] and

self.comboBoxDepartmentR.currentText() in res["Remarks"][self.comboBoxDateR.currentText()]):

for k1, v1 in res["Remarks"][self.comboBoxDateR.currentText()][self.comboBoxDepartmentR.currentText()].items():

rowPosition = self.tableWidgetR.rowCount()

self.tableWidgetR.insertRow(rowPosition)

self.tableWidgetR.setItem(rowPosition , 0, QtGui.QTableWidgetItem(str(k1)))

if ("FacultyMembers" in res and str(k1).split("-")[0] in res["FacultyMembers"] and

str(k1) in res["FacultyMembers"][str(k1).split("-")[0]]

and "Name" in res["FacultyMembers"][str(k1).split("-")[0]][str(k1)]):

self.tableWidgetR.setItem(rowPosition , 1,

QtGui.QTableWidgetItem(res["FacultyMembers"][str(k1).split("-")[0]][str(k1)]["Name"]))

if (str(k1) in res["Remarks"][self.comboBoxDateR.currentText()][self.comboBoxDepartmentR.currentText()]):

self.tableWidgetR.setItem(rowPosition , 2, QtGui.QTableWidgetItem("Yes"))

else:

self.tableWidgetR.setItem(rowPosition , 2, QtGui.QTableWidgetItem("No"))

self.comboBoxDateR.setDisabled(False)

self.comboBoxDepartmentR.setDisabled(False)

def tabWidgetTabChanged(self):

if (self.tabWidget.tabText(self.tabWidget.currentIndex()) == "REMARKS"):

self.comboBoxDateR.setDisabled(True)

self.comboBoxDateR.clear()

res = ((database.get("", None)))

self.comboBoxDateR.currentIndexChanged.connect(lambda:self.cmbdRIndexChanged(res))

if (res != None):

if ("Remarks" in res):

for k1, v1 in res["Remarks"].items():

self.comboBoxDateR.addItem(k1)

self.comboBoxDateR.setDisabled(False)

if (self.tabWidget.tabText(self.tabWidget.currentIndex()) == "ATTENDANCE"):

self.comboBoxDateA.setDisabled(True)

self.comboBoxDateA.clear()

res = ((database.get("", None)))

self.comboBoxDateA.currentIndexChanged.connect(lambda:self.cmbdAIndexChanged(res))

if (res != None):

if ("Attendance" in res):

for k1, v1 in res["Attendance"].items():

self.comboBoxDateA.addItem(k1)

self.comboBoxDateA.setDisabled(False)

if (self.tabWidget.tabText(self.tabWidget.currentIndex()) == "FACULTY"):

self.comboBoxDepartmentF.setDisabled(True)

while (self.tableWidgetF.rowCount() > 0):

self.tableWidgetF.removeRow(0)

res = ((database.get("/FacultyMembers", None)))

if (res != None):

if (self.comboBoxDepartmentF.currentText() in res):

for k1, v1 in res[self.comboBoxDepartmentF.currentText()].items():

rowPosition = self.tableWidgetF.rowCount()

self.tableWidgetF.insertRow(rowPosition)

for k2, v2 in v1.items():

self.tableWidgetF.setItem(rowPosition , 0, QtGui.QTableWidgetItem(str(k1)))

if ("Name" in k2):

self.tableWidgetF.setItem(rowPosition , 1, QtGui.QTableWidgetItem(str(v2)))

if ("Date" in k2):

self.tableWidgetF.setItem(rowPosition , 2, QtGui.QTableWidgetItem(str(v2)))

self.comboBoxDepartmentF.setDisabled(False)

if (self.tabWidget.tabText(self.tabWidget.currentIndex()) == "SETTINGS"):

self.lineEditUsernameF.setDisabled(True)

self.lineEditPasswordF.setDisabled(True)

self.lineEditUsernameD.setDisabled(True)

self.lineEditPasswordD.setDisabled(True)

self.pushButtonSave.setDisabled(True)

res = ((database.get("Authentication", None)))

if (res != None):

self.lineEditUsernameF.setText(res["Faculty"]["Username"])

self.lineEditPasswordF.setText(res["Faculty"]["Password"])

self.lineEditUsernameD.setText(res["Database"]["Username"])

self.lineEditPasswordD.setText(res["Database"]["Password"])

self.lineEditUsernameF.setDisabled(False)

self.lineEditPasswordF.setDisabled(False)

self.lineEditUsernameD.setDisabled(False)

self.lineEditPasswordD.setDisabled(False)

self.pushButtonSave.setDisabled(False)

def convert24(self, str1):

# Checking if last two elements of time

# is AM and first two elements are 12

if str1[-2:] == "AM" and str1[:2] == "12":

return "00" + str1[2:-2]

# remove the AM

elif str1[-2:] == "AM":

return str1[:-2]

# Checking if last two elements of time

# is PM and first two elements are 12

elif str1[-2:] == "PM" and str1[:2] == "12":

return str1[:-2]

else:

# add 12 to hours and remove PM

return str(int(str1[:2]) + 12) + str1[2:8]

def openAEditMainMenuUI(self):

if (self.tableWidgetA.currentRow() != -1):

date = self.comboBoxDateA.currentText()

idLbl = self.tableWidgetA.item(self.tableWidgetA.currentRow(), 0).text()

nameLbl = self.tableWidgetA.item(self.tableWidgetA.currentRow(), 1).text()

timeInLbl = self.tableWidgetA.item(self.tableWidgetA.currentRow(), 2).text()

timeOutLbl = ""

if (self.tableWidgetA.item(self.tableWidgetA.currentRow(), 3) != None):

timeOutLbl = self.tableWidgetA.item(self.tableWidgetA.currentRow(), 3).text()

self.close()

super(MyWindow, self).\_\_init\_\_()

uic.loadUi(directory + 'EditTime.ui', self)

self.setFixedSize(self.size())

self.setWindowFlags(QtCore.Qt.FramelessWindowHint)

self.center()

self.show()

self.labelDepartment.setText(idLbl.split("-")[0])

self.labelID.setText(idLbl)

self.labelName.setText(nameLbl)

self.labelDate.setText(date)

if (timeInLbl != ""):

self.timeEditTimeIn.setTime(QTime(int(self.convert24(timeInLbl).split(":")[0]), int(self.convert24(timeInLbl).split(":")[1]),

int((self.convert24(timeInLbl).split(":"))[2].split(" ")[0])))

if (timeOutLbl != ""):

self.timeEditTimeOut.setTime(QTime(int(self.convert24(timeOutLbl).split(":")[0]), int(self.convert24(timeOutLbl).split(":")[1]),

int((self.convert24(timeOutLbl).split(":"))[2].split(" ")[0])))

self.pushButtonBack.clicked.connect(lambda : self.openDatabaseUI(1))

self.pushButtonSave.clicked.connect(self.AEditSave)

else:

self.doMessage("Select a data in the table view.", "Information")

def AEditSave(self):

database.patch("/Attendance/"+self.labelDate.text()+"/"+self.labelID.text().split("-")[0]+"/"+self.labelID.text(),

{"In":str((datetime.datetime.strptime(self.timeEditTimeIn.time().toString(),

'%H:%M:%S')).strftime('%I:%M:%S %p')).upper(),

"Out":str((datetime.datetime.strptime(self.timeEditTimeOut.time().toString(),

'%H:%M:%S')).strftime('%I:%M:%S %p')).upper()})

self.doMessage("Time In and Time Out of the faculty member updated.", "Information")

self.close()

self.openDatabaseUI(1)

def openFRegisterUI(self):

self.close()

super(MyWindow, self).\_\_init\_\_()

uic.loadUi(directory + 'Register.ui', self)

self.setFixedSize(self.size())

self.setWindowFlags(QtCore.Qt.FramelessWindowHint)

self.center()

self.show()

self.pushButtonBack.clicked.connect(lambda : self.openDatabaseUI(2))

self.pushButtonSubmit.clicked.connect(self.submitClicked)

def submitClicked(self):

if (str(self.lineEditName.text()) != "" and str(self.lineEditPassword.text()) != ""):

res = ((database.get("/FacultyMembers", None)))

idNumber = 0

if (res != None):

if (self.comboBoxDepartment.currentText() in res):

numbers = []

for k, v in res[self.comboBoxDepartment.currentText()].items():

numbers.append(int(k.split('-')[1]))

idNumber = max(numbers) + 1

database.patch('/FacultyMembers/'+str(self.comboBoxDepartment.currentText())+'/'+

self.comboBoxDepartment.currentText()+'-'+str(idNumber)+'/', {'Name':str(self.lineEditName.text()).upper(),

'Password':str(self.lineEditPassword.text()),

'Date':str(datetime.datetime.now().strftime('%b-%d-%Y'))})

self.doMessage("Registered a new faculty member in the "+self.comboBoxDepartment.currentText()+" Department.", "Information")

self.lineEditName.clear()

self.lineEditPassword.clear()

else:

self.doMessage("Some required (\*) fields are empty.", "Information")

def openFEditUI(self):

if (self.tableWidgetF.currentRow() != -1):

if (self.tableWidgetF.item(self.tableWidgetF.currentRow(), 0).text() != None):

res = ((database.get("/FacultyMembers/"+str(self.comboBoxDepartmentF.currentText())+"/"+

self.tableWidgetF.item(self.tableWidgetF.currentRow(), 0).text(), None)))

departText = str(self.comboBoxDepartmentF.currentText())

idOfMember = str(self.tableWidgetF.item(self.tableWidgetF.currentRow(), 0).text())

self.close()

super(MyWindow, self).\_\_init\_\_()

uic.loadUi(directory + 'Edit.ui', self)

self.setFixedSize(self.size())

self.setWindowFlags(QtCore.Qt.FramelessWindowHint)

self.center()

self.show()

self.pushButtonBack.clicked.connect(lambda : self.openDatabaseUI(2))

self.pushButtonSave.clicked.connect(self.saveEdit)

if (res != None):

self.comboBoxDepartment.setText(departText)

self.labelID.setText(idOfMember)

if ("Name" in res):

self.lineEditName.setText(res["Name"])

if ("Password" in res):

self.lineEditPasswordOld.setText(res["Password"])

else:

self.doMessage("Select a faculty member in the table view.", "Information")

def saveEdit(self):

if (str(self.lineEditName.text()) != "" and str(self.lineEditPasswordNew.text()) != ""):

res = ((database.get("/FacultyMembers", None)))

database.patch('/FacultyMembers/'+str(self.comboBoxDepartment.text())+'/'+self.labelID.text()+'/',

{'Name':str(self.lineEditName.text()).upper(),

'Password':str(self.lineEditPasswordNew.text())})

self.close()

self.openDatabaseUI(2)

self.doMessage("Updated the faculty member in the "+self.comboBoxDepartment.text()+" Department.", "Information")

else:

self.doMessage("Some required (\*) fields are empty.", "Information")

if \_\_name\_\_ == '\_\_main\_\_':

app = QtGui.QApplication(sys.argv)

window = MyWindow()

sys.exit(app.exec\_())

finally:

GPIO.cleanup()