#!/usr/bin/env python3

from PyQt5 import QtCore, QtGui, QtWidgets

from PyQt5.QtMultimediaWidgets import QVideoWidget, QGraphicsVideoItem

from PyQt5.QtWidgets import QTableWidget,QTableWidgetItem

from generated\_ui import \*

import time

from PyQt5.QtWidgets import QApplication, QWidget, QLabel

from PyQt5.QtGui import QIcon, QPixmap

from PyQt5.QtWidgets import (QApplication, QGraphicsView, QGraphicsScene, QGraphicsItem,

QGridLayout, QVBoxLayout, QHBoxLayout,

QLabel, QLineEdit, QPushButton)

from PyQt5.QtMultimedia import QMediaContent, QMediaPlayer

from PyQt5.QtMultimediaWidgets import QVideoWidget

from PyQt5.QtWidgets import (QMainWindow, QApplication, QWidget, QTableWidget,QVBoxLayout,

QTableWidgetItem, QHBoxLayout,QSplitter,QGroupBox)

from PyQt5 import QtGui, QtCore

from PyQt5.QtCore import QDir, Qt, QUrl

#dynamic pricing function

def pricing\_info():

price = 0.00

time\_obj = time.localtime()

hour\_time = time\_obj.tm\_hour

if hour\_time > 00 and hour\_time <=9:

price = 1.99

elif hour\_time > 9 and hour\_time <=13:

price = 3.99

elif hour\_time >13 and hour\_time <=18:

price = 2.99

elif hour\_time >18 and hour\_time <= 24:

price = 1.99

else:

price = 1.99

ui.price\_label\_15.setText(str(price))

print(price)

return price

#to populate a list with dictionaries

def populate\_db():

db = []

a = {'spot\_no':2,'vehicle\_no':'7agx258','entry\_time':(12,45),'exit\_time':None,'amount\_due':None}

b = {'spot\_no':5,'vehicle\_no':'az87900','entry\_time':(13,45),'exit\_time':(18,30),'amount\_due':None}

c = {'spot\_no':6,'vehicle\_no':'910ino1','entry\_time':(10,45),'exit\_time':(12,10),'amount\_due':4.00}

d = {'spot\_no':4,'vehicle\_no':'6yuh567','entry\_time':(8,45),'exit\_time':None,'amount\_due':None}

db.append(a)

db.append(b)

db.append(c)

db.append(d)

return db

#extract tuple values

def extract\_time(a):

hr = a[0]

mi = a[1]

s = '%d : %d' % (hr,mi)

return s

#return '%s : %s' % (str(hr), str(mi))

#extract hour only

def extract\_hr(a):

hr = a[0]

return hr

#extract minutes only

def extract\_min(a):

mi = a[1]

return mi

#to display number of spots available

def number\_of\_spots\_label(db):

count =0

num = len(db)

for i in range(num):

if db[i]['spot\_no'] >= 1:

count +=1

count = 6 - count

ui.no\_spots\_label.setText(str(count))

#to update the table

def update\_table(db):

cols = 6

if len(db) >= 1:

for i in range(len(db)):

ui.tableWidget.setItem(i,0, QTableWidgetItem(str(db[i]['spot\_no'])))

ui.tableWidget.setItem(i,1, QTableWidgetItem(str(db[i]['vehicle\_no'])))

ui.tableWidget.setItem(i,3, QTableWidgetItem(extract\_time(db[i]['entry\_time'])))

if db[i]['exit\_time'] != None:

ui.tableWidget.setItem(i,4, QTableWidgetItem(extract\_time(db[i]['exit\_time'])))

if db[i]['amount\_due'] != None:

ui.tableWidget.setItem(i,5, QTableWidgetItem(str(db[i]['amount\_due'])))

ui.tableWidget.setItem(i,2, QTableWidgetItem("Full"))

#get spot\_labels and put it in a list

def make\_listOf\_spotLabels():

label\_list=[]

label\_list.append(ui.spot\_no\_1)

label\_list.append(ui.spot\_no\_2)

label\_list.append(ui.spot\_no\_3)

label\_list.append(ui.spot\_no\_4)

label\_list.append(ui.spot\_no\_5)

label\_list.append(ui.spot\_no\_6)

return label\_list

#to update filled spots with images

def update\_filled\_spots(label\_list):

found = False

size = len(label\_list)

for i in range(size):

found = False

for j in range(len(db)):

if db[j]['spot\_no'] == i+1:

found = True

break

if found==True:

grview = label\_list[i]

scene = QGraphicsScene()

scene.addPixmap(QPixmap('red\_car\_img.png'))

grview.setScene(scene)

grview.fitInView(scene.sceneRect())

else:

grview = label\_list[i]

scene = QGraphicsScene()

scene.addPixmap(QPixmap('empty\_car.jpg'))

grview.setScene(scene)

grview.fitInView(scene.sceneRect())

#to calculate amout due

def amount\_due(db, price):

due\_time =0

final\_price =0.00

for i in range(len(db)):

if db[i]['exit\_time'] != None:

hr\_en = extract\_hr(db[i]['entry\_time'])

mi\_en = extract\_min(db[i]['entry\_time'])

hr\_ex = extract\_hr(db[i]['exit\_time'])

mi\_ex = extract\_min(db[i]['exit\_time'])

hr\_en = hr\_en\*60 + mi\_en

hr\_ex = hr\_ex \*60+ mi\_ex

due\_time = hr\_ex - hr\_en

final\_price = due\_time \* (price/60)

final\_price = round(final\_price, 2)

ui.tableWidget.setItem(i,5, QTableWidgetItem(str(final\_price)))

if \_\_name\_\_ == "\_\_main\_\_":

import sys

app = QtWidgets.QApplication(sys.argv)

MainWindow = QtWidgets.QMainWindow()

ui = Ui\_MainWindow()

ui.setupUi(MainWindow)

MainWindow.show()

db = populate\_db()

price = pricing\_info()

number\_of\_spots\_label(db)

update\_table(db)

label\_list = make\_listOf\_spotLabels()

update\_filled\_spots(label\_list)

amount\_due(db, price)

player\_1 = QMediaPlayer(None, QMediaPlayer.VideoSurface)

player\_1.setMedia(QMediaContent(QUrl.fromLocalFile("/Users/uma/Documents/my\_website/sjsuparking1.mp4")))

ui.vid\_1.setAspectRatioMode(1)

player\_1.setVideoOutput(ui.vid\_1)

player\_1.play()

player\_2 = QMediaPlayer(None, QMediaPlayer.VideoSurface)

player\_2.setMedia(QMediaContent(QUrl.fromLocalFile("/Users/uma/Documents/my\_website/sjsuparking2.mp4")))

player\_2.setVideoOutput(ui.vid\_2)

player\_2.play()

player\_3 = QMediaPlayer(None, QMediaPlayer.VideoSurface)

player\_3.setMedia(QMediaContent(QUrl.fromLocalFile("/Users/uma/Documents/my\_website/sjsuparking3.mp4")))

player\_3.setVideoOutput(ui.vid\_3)

player\_3.play()

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