import sys

import csv

import threading

from PyQt5.QtWidgets import QApplication, QMainWindow, QFileDialog, QTreeWidget, QTreeWidgetItem, QVBoxLayout, QWidget, QPushButton, QLineEdit, QLabel

from PyQt5.QtCore import Qt

class GradeCalculator(QMainWindow):

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

super().\_\_init\_\_()

self.setWindowTitle("Grade Calculator by Ncokazi Zukisa")

self.setGeometry(100, 100, 800, 600)

# Create widgets:

#Labels help the user navigate the features.

#I embedded definitions within buttons so that they can execute my desired instructions ON CLICK.

#I wanted a feature that is monitor-like to desplay stored data. So I opted to use a binary tree, it helps even if we are reading sql files.

print("Paste your file path on the slot below if you have the adress")

self.file\_path\_label = QLabel("PASTE CSV file path and press enter:")

self.file\_path\_input = QLineEdit()

self.browse\_button = QPushButton(" Manually BROWSE FOR CSV File")

self.calculate\_button = QPushButton("Calculate Grades of chosen file")

self.display\_button = QPushButton("Display Calculated Grades Here")

self.tree\_widget = QTreeWidget()

self.tree\_widget.setColumnCount(6)

self.tree\_widget.setHeaderLabels(["Module", "Quiz", "Project", "Final Exam", "Practical Exam", "Overall Grade"])

# Here I create layout and add widgets that are needed for this interface.

# I just arranged the features in the order I wished the user would start, vertically going down.

central\_widget = QWidget()

layout = QVBoxLayout()

layout.addWidget(self.file\_path\_label)

layout.addWidget(self.file\_path\_input)

layout.addWidget(self.browse\_button)

layout.addWidget(self.calculate\_button)

layout.addWidget(self.display\_button)

layout.addWidget(self.tree\_widget)

central\_widget.setLayout(layout)

self.setCentralWidget(central\_widget)

self.browse\_button.clicked.connect(self.browse\_file)

self.calculate\_button.clicked.connect(self.calculate\_grades)

self.display\_button.clicked.connect(self.display\_grades)

self.grades = []

#The following functions are invoked when the user interacts with the relevant component of the interface.

def browse\_file(self):

file\_path, \_ = QFileDialog.getOpenFileName(self, "Select CSV File", "", "CSV Files (\*.csv)")

if file\_path:

self.file\_path\_input.setText(file\_path)

self.read\_csv\_file(file\_path)

def read\_csv\_file(self, file\_path):

self.grades = []

try:

with open(file\_path, 'r') as file:

reader = csv.reader(file)

for row in reader:

if all(item.replace('.', '', 1).isdigit() for item in row[1:]):

self.grades.append(row)

else:

print(f"We apologize, we seem to think there is an nvalid data in row: {row}")

except Exception as e:

print(f"Error reading CSV file: {e}")

#The definition below allow for multithreading and management of those multiple threads.

def calculate\_grades(self):

threads = []

for i in range(len(self.grades)):

thread = threading.Thread(target=self.calculate\_grade, args=(i,))

threads.append(thread)

thread.start()

for thread in threads:

thread.join()

#this one is the most important one because it contains fomulars that convert each oof the scores in respect to their weights.

#Without this definition, the binary tree would have trouble displaying , because calculations are needed. And the scores would'nt be the same had it been any different.

def calculate\_grade(self, index):

row = self.grades[index]

module = row[0]

quiz = float(row[1]) \* 0.1

project = float(row[2]) \* 0.2

final\_exam = float(row[3]) \* 0.5

practical\_exam = float(row[4]) \* 0.2

overall\_grade = quiz + project + final\_exam + practical\_exam

self.grades[index].append(f"{overall\_grade:.2f}")

#This function is invoked through a button of same name to display calculated scores in the tree below.

def display\_grades(self):

self.tree\_widget.clear()

for row in self.grades:

item = QTreeWidgetItem(row)

self.tree\_widget.addTopLevelItem(item)

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

app = QApplication(sys.argv)

calculator = GradeCalculator()

calculator.show()

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