البيانات:

import sqlite3

def create\_connection():

conn = None

try:

conn = sqlite3.connect('saffarlak.db') # قم بإنشاء أو الاتصال بقاعدة البيانات

return conn

except sqlite3.Error as e:

print(e)

return conn

def execute\_query(query, params=()):

conn = create\_connection()

cur = conn.cursor()

cur.execute(query, params)

result = cur.fetchall()

conn.close()

return result

def execute\_update(query, params=()):

conn = create\_connection()

cur = conn.cursor()

cur.execute(query, params)

conn.commit()

conn.close()

2. تعديل دالة البحث:

الآن سنقوم بربط دالة البحث مع قاعدة البيانات. سيتم استدعاء الدالة search\_trips من واجهة المستخدم.

دالة البحث عن الرحلات:

def search\_trips(origin, destination, date):

query = "SELECT \* FROM trips WHERE origin = ? AND destination = ? AND departure\_time LIKE ?"

result = execute\_query(query, (origin, destination, f"{date}%"))

return result

تعديل دالة البحث في الواجهة:

في تطبيق PyQt، سنقوم بتعديل دالة search\_trips لعرض النتائج المستخرجة من قاعدة البيانات.

تحديث دالة search\_trips في الواجهة:

def search\_trips(self):

origin = self.origin\_input.text()

destination = self.destination\_input.text()

# البحث عن الرحلات في قاعدة البيانات

trips = search\_trips(origin, destination, "")

# إذا كانت هناك نتائج، نعرضها في نافذة النتائج

if trips:

formatted\_trips = [{'route': f'{trip[1]} إلى {trip[2]}', 'time': trip[3], 'price': trip[5]} for trip in trips]

self.results\_window = ResultsWindow(formatted\_trips)

self.results\_window.show()

else:

print("لا توجد رحلات متاحة")

3. إضافة دالة تأكيد الحجز:

بعد اختيار المستخدم لرحلة معينة، سنحتاج إلى حجز المقاعد وتحديث قاعدة البيانات.

دالة تأكيد الحجز:

def confirm\_booking(trip\_id, user\_name, user\_phone, user\_email, seats\_booked):

# التحقق من المقاعد المتاحة

query = "SELECT seats\_available FROM trips WHERE id = ?"

seats = execute\_query(query, (trip\_id, ))[0][0]

if seats >= seats\_booked:

# تأكيد الحجز

update\_query = "UPDATE trips SET seats\_available = seats\_available - ? WHERE id = ?"

execute\_update(update\_query, (seats\_booked, trip\_id))

# إدخال بيانات الحجز في جدول الحجوزات

insert\_query = "INSERT INTO bookings (trip\_id, user\_name, user\_phone, user\_email, seats\_booked) VALUES (?, ?, ?, ?, ?)"

execute\_update(insert\_query, (trip\_id, user\_name, user\_phone, user\_email, seats\_booked))

return True

return False

إضافة واجهة الحجز:

بعد أن يقوم المستخدم باختيار الرحلة، سنقوم بفتح نافذة جديدة تحتوي على حقول لإدخال بياناته الشخصية وتأكيد الحجز.

نافذة تأكيد الحجز:

from PyQt5.QtWidgets import QLineEdit, QSpinBox, QMessageBox

class BookingWindow(QWidget):

def \_\_init\_\_(self, trip\_id):

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

self.trip\_id = trip\_id

self.initUI()

def initUI(self):

self.setWindowTitle('تأكيد الحجز')

self.name\_label = QLabel('الاسم:')

self.name\_input = QLineEdit(self)

self.phone\_label = QLabel('الهاتف:')

self.phone\_input = QLineEdit(self)

self.email\_label = QLabel('البريد الإلكتروني:')

self.email\_input = QLineEdit(self)

self.seats\_label = QLabel('عدد المقاعد:')

self.seats\_input = QSpinBox(self)

self.seats\_input.setMinimum(1)

self.book\_button = QPushButton('تأكيد الحجز', self)

self.book\_button.clicked.connect(self.book\_trip)

layout = QVBoxLayout()

layout.addWidget(self.name\_label)

layout.addWidget(self.name\_input)

layout.addWidget(self.phone\_label)

layout.addWidget(self.phone\_input)

layout.addWidget(self.email\_label)

layout.addWidget(self.email\_input)

layout.addWidget(self.seats\_label)

layout.addWidget(self.seats\_input)

layout.addWidget(self.book\_button)

self.setLayout(layout)

def book\_trip(self):

user\_name = self.name\_input.text()

user\_phone = self.phone\_input.text()

user\_email = self.email\_input.text()

seats\_booked = self.seats\_input.value()

if confirm\_booking(self.trip\_id, user\_name, user\_phone, user\_email, seats\_booked):

QMessageBox.information(self, "نجاح", "تم تأكيد الحجز بنجاح!")

else:

QMessageBox.warning(self, "خطأ", "عدد المقاعد المتاحة غير كافٍ.")

1. لغة البرمجة المستخدمة:

اللغة المستخدمة في هذا المشروع هي Python، لأنها تدعم مكتبات قوية لبناء واجهات المستخدم (PyQt5) وقواعد البيانات (sqlite3).

2. إطار العمل (Framework):

لم يتم استخدام إطار عمل متكامل مثل Django أو Flask لأن المشروع يعتمد فقط على PyQt5 لواجهة المستخدم وSQLite كقاعدة بيانات. هذا يجعل المشروع خفيفًا وسهل التنفيذ دون الحاجة إلى بيئة خادم متقدمة.

3. بيئة التطوير المتكاملة (IDE):

يمكن استخدام أي بيئة تطوير مثل:

Visual Studio Code: مناسب جدًا لتطوير التطبيقات الصغيرة والمشاريع المتوسطة.

PyCharm: بيئة قوية ومناسبة لتطوير تطبيقات Python.

4. هيكل المشروع:

الهيكل الأساسي للمشروع يمكن أن يكون بسيطًا ويشمل الملفات التالية:

saffarlak\_project/

│

├── main.py # ملف Python الرئيسي لتشغيل التطبيق

├── database.py # ملف يحتوي على الدوال التي تتعامل مع قاعدة البيانات

├── ui/ # مجلد يحتوي على ملفات واجهة المستخدم

│ ├── search\_window.py # نافذة البحث عن الرحلات

│ ├── results\_window.py # نافذة عرض نتائج البحث

│ ├── booking\_window.py # نافذة تأكيد الحجز

├── saffarlak.db # ملف قاعدة البيانات SQLite

└── README.md # ملف يحتوي على تعليمات التشغيل

5. الكود الكامل:

سأشاركك الهيكل العام للكود الذي تم كتابته:

main.py: الملف الرئيسي الذي يبدأ التطبيق.

import sys

from PyQt5.QtWidgets import QApplication

from ui.search\_window import SaffarlakApp

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

app = QApplication(sys.argv)

window = SaffarlakApp()

window.show()

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

database.py: يحتوي على الوظائف المتعلقة بقاعدة البيانات مثل execute\_query وexecute\_update.

import sqlite3

def create\_connection():

conn = sqlite3.connect('saffarlak.db')

return conn

def execute\_query(query, params=()):

conn = create\_connection()

cur = conn.cursor()

cur.execute(query, params)

result = cur.fetchall()

conn.close()

return result

def execute\_update(query, params=()):

conn = create\_connection()

cur = conn.cursor()

cur.execute(query, params)

conn.commit()

conn.close()

ui/search\_window.py: نافذة البحث عن الرحلات.

from PyQt5.QtWidgets import QWidget, QLabel, QLineEdit, QPushButton, QVBoxLayout, QHBoxLayout

from ui.results\_window import ResultsWindow

from database import execute\_query

class SaffarlakApp(QWidget):

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

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

self.initUI()

def initUI(self):

self.setWindowTitle('Saffarlak - حجز تذاكر السفر')

self.origin\_label = QLabel('مدينة الانطلاق:')

self.origin\_input = QLineEdit(self)

self.destination\_label = QLabel('مدينة الوصول:')

self.destination\_input = QLineEdit(self)

self.search\_button = QPushButton('بحث', self)

self.search\_button.clicked.connect(self.search\_trips)

layout = QVBoxLayout()

origin\_layout = QHBoxLayout()

origin\_layout.addWidget(self.origin\_label)

origin\_layout.addWidget(self.origin\_input)

destination\_layout = QHBoxLayout()

destination\_layout.addWidget(self.destination\_label)

destination\_layout.addWidget(self.destination\_input)

layout.addLayout(origin\_layout)

layout.addLayout(destination\_layout)

layout.addWidget(self.search\_button)

self.setLayout(layout)

def search\_trips(self):

origin = self.origin\_input.text()

destination = self.destination\_input.text()

trips = execute\_query("SELECT \* FROM trips WHERE origin = ? AND destination = ?", (origin, destination))

self.results\_window = ResultsWindow(trips)

self.results\_window.show()

ui/results\_window.py: نافذة عرض نتائج البحث.

from PyQt5.QtWidgets import QWidget, QTableWidget, QTableWidgetItem, QVBoxLayout

class ResultsWindow(QWidget):

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

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

self.trips = trips

self.initUI()

def initUI(self):

self.setWindowTitle('نتائج البحث')

self.table = QTableWidget(self)

self.table.setRowCount(len(self.trips))

self.table.setColumnCount(3)

self.table.setHorizontalHeaderLabels(['الرحلة', 'الوقت', 'السعر'])

for row, trip in enumerate(self.trips):

self.table.setItem(row, 0, QTableWidgetItem(f"{trip[1]} إلى {trip[2]}"))

self.table.setItem(row, 1, QTableWidgetItem(trip[3]))

self.table.setItem(row, 2, QTableWidgetItem(str(trip[5])))

layout = QVBoxLayout()

layout.addWidget(self.table)

self.setLayout(layout)

ui/booking\_window.py: نافذة تأكيد الحجز.

from PyQt5.QtWidgets import QWidget, QLabel, QLineEdit, QSpinBox, QPushButton, QVBoxLayout

from database import execute\_update

class BookingWindow(QWidget):

def \_\_init\_\_(self, trip\_id):

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

self.trip\_id = trip\_id

self.initUI()

def initUI(self):

self.setWindowTitle('تأكيد الحجز')

self.name\_label = QLabel('الاسم:')

self.name\_input = QLineEdit(self)

self.phone\_label = QLabel('الهاتف:')

self.phone\_input = QLineEdit(self)

self.email\_label = QLabel('البريد الإلكتروني:')

self.email\_input = QLineEdit(self)

self.seats\_label = QLabel('عدد المقاعد:')

self.seats\_input = QSpinBox(self)

self.seats\_input.setMinimum(1)

self.book\_button = QPushButton('تأكيد الحجز', self)

self.book\_button.clicked.connect(self.book\_trip)

layout = QVBoxLayout()

layout.addWidget(self.name\_label)

layout.addWidget(self.name\_input)

layout.addWidget(self.phone\_label)

layout.addWidget(self.phone\_input)

layout.addWidget(self.email\_label)

layout.addWidget(self.email\_input)

layout.addWidget(self.seats\_label)

layout.addWidget(self.seats\_input)

layout.addWidget(self.book\_button)

self.setLayout(layout)

def book\_trip(self):

user\_name = self.name\_input.text()

user\_phone = self.phone\_input.text()

user\_email = self.email\_input.text()

seats\_booked = self.seats\_input.value()

execute\_update("UPDATE trips SET seats\_available = seats\_available - ? WHERE id = ?", (seats\_booked, self.trip\_id))

6. تعليمات التشغيل:

1. تثبيت المكتبات: تأكد من تثبيت مكتبات PyQt5 و sqlite3. استخدم الأوامر التالية:

pip install pyqt5

2. إنشاء قاعدة البيانات:

قم بتشغيل سكريبت قاعدة البيانات (أو استخدام SQLite) لإنشاء الجداول "trips" و "bookings".

3. تشغيل التطبيق: شغل الملف main.py باستخدام Python:

python main.py