لتحقيق مشروع تطبيق حجز رحلات الطيران باستخدام Tkinter وSQLite، سنتبع نهجًا عمليًا وخطوة بخطوة. هذا الدليل مصمم للمبتدئين، مع التركيز على تنظيم الكود وتسهيل الفهم.

1. الإعدادات الأولية

قبل البدء في كتابة الكود، ستحتاج إلى إعداد بيئة العمل.

1. إنشاء مجلد للمشروع:

أنشئ مجلدًا جديدًا على جهازك باسم flight\_reservation\_app وهذا هو المجلد الرئيسي للمشروع.

1. تثبيت المكتبات المطلوبة:

ستحتاج إلى Tkinter (والتي تأتي مثبتة بشكل افتراضي مع بايثون) وSQLite (مدمجة أيضًا في بايثون). لكن، لتسهيل تتبع المتطلبات، أنشئ ملف requirements.txt في مجلد المشروع واكتب فيه:

# لا تحتاج إلى أي مكتبات إضافية، Tkinter و SQLite مدمجتان.

1. هيكل الملفات:

داخل مجلد flight\_reservation\_app، أنشئ الملفات التالية:

* + main.py
  + database.py
  + home.py
  + booking.py
  + reservations.py
  + edit\_reservation.py
  + README.md
  + requirements.txt

2. إعداد قاعدة البيانات (database.py)

هذا هو الجزء الأساسي الذي سيتعامل مع تخزين واسترجاع البيانات.

* افتح ملف database.py واكتب الكود التالي. هذا الكود سيتأكد من وجود قاعدة البيانات والجداول عند تشغيل التطبيق.
* استخدم sqlite3 وهو مكتبة بايثون المدمجة للتفاعل مع قواعد بيانات SQLite.

Python

import sqlite3

DATABASE\_NAME = "flights.db"

def create\_table():

#يتصل بقاعدة البيانات وينشئ جدول 'reservations' إذا لم يكن موجودًا.

conn = sqlite3.connect(DATABASE\_NAME)

cursor = conn.cursor()

cursor.execute('''

CREATE TABLE IF NOT EXISTS reservations (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

flight\_number TEXT NOT NULL,

departure TEXT NOT NULL,

destination TEXT NOT NULL,

date TEXT NOT NULL,

seat\_number TEXT NOT NULL

)

''')

conn.commit()

conn.close()

def get\_all\_reservations():

"""يسترجع جميع الحجوزات من قاعدة البيانات."""

conn = sqlite3.connect(DATABASE\_NAME)

cursor = conn.cursor()

cursor.execute("SELECT \* FROM reservations")

reservations = cursor.fetchall()

conn.close()

return reservations

def add\_reservation(name, flight\_number, departure, destination, date, seat\_number):

"""يضيف حجزًا جديدًا إلى قاعدة البيانات."""

conn = sqlite3.connect(DATABASE\_NAME)

cursor = conn.cursor()

cursor.execute('''

INSERT INTO reservations (name, flight\_number, departure, destination, date, seat\_number)

VALUES (?, ?, ?, ?, ?, ?)

''', (name, flight\_number, departure, destination, date, seat\_number))

conn.commit()

conn.close()

def update\_reservation(reservation\_id, name, flight\_number, departure, destination, date, seat\_number):

"""يحدث حجزًا موجودًا بناءً على الـ ID."""

conn = sqlite3.connect(DATABASE\_NAME)

cursor = conn.cursor()

cursor.execute('''

UPDATE reservations

SET name=?, flight\_number=?, departure=?, destination=?, date=?, seat\_number=?

WHERE id=?

''', (name, flight\_number, departure, destination, date, seat\_number, reservation\_id))

conn.commit()

conn.close()

def delete\_reservation(reservation\_id):

"""يحذف حجزًا بناءً على الـ ID."""

conn = sqlite3.connect(DATABASE\_NAME)

cursor = conn.cursor()

cursor.execute("DELETE FROM reservations WHERE id=?", (reservation\_id,))

conn.commit()

conn.close()

3. تصميم واجهة المستخدم (GUI)

سنقوم بإنشاء صفحات مختلفة للتطبيق.

3.1. الصفحة الرئيسية (home.py)

* هذه هي الصفحة الأولى التي ستظهر للمستخدم.
* تحتوي على أزرار للتنقل إلى صفحة حجز جديد وعرض الحجوزات.

Python

import tkinter as tk

from tkinter import ttk

class HomePage(tk.Frame):

def \_\_init\_\_(self, parent, controller):

tk.Frame.\_\_init\_\_(self, parent)

self.controller = controller

label = ttk.Label(self, text=" Flight Booking Page", font=("Arial", 20, "bold")) label.pack(pady=20)

book\_button = ttk.Button(self, text=" Book a new flight", command=lambda: controller.show\_frame("BookingPage")) book\_button.pack(pady=10, ipadx=10, ipady=5)

view\_button = ttk.Button(self, text=" View reservations", command=lambda: controller.show\_frame("ReservationsPage")) view\_button.pack(pady=10, ipadx=10, ipady=5)

3.2. صفحة حجز جديد (booking.py)

* تحتوي على حقول لإدخال معلومات الحجز وزر للحفظ.
* تستخدم الدالة add\_reservation من ملف database.py.

Python

import tkinter as tk

from tkinter import ttk, messagebox

import database

class BookingPage(tk.Frame):

def \_\_init\_\_(self, parent, controller):

tk.Frame.\_\_init\_\_(self, parent)

self.controller = controller

label = ttk.Label(self, text="حجز رحلة جديدة", font=("Arial", 16, "bold"))

label.pack(pady=10)

# حقول الإدخال

fields = ["اسم الراكب:", "رقم الرحلة:", "المغادرة:", "الوجهة:", "التاريخ:", "رقم المقعد:"]

self.entries = {}

for i, field in enumerate(fields):

row = tk.Frame(self)

lbl = ttk.Label(row, text=field, width=15)

ent = ttk.Entry(row, width=30)

row.pack(pady=5)

lbl.pack(side="left", padx=5)

ent.pack(side="right", padx=5)

self.entries[field] = ent

save\_button = ttk.Button(self, text="حفظ الحجز", command=self.save\_booking)

save\_button.pack(pady=10)

back\_button = ttk.Button(self, text="العودة للصفحة الرئيسية", command=lambda: controller.show\_frame("HomePage"))

back\_button.pack(pady=5)

def save\_booking(self):

try:

name = self.entries["اسم الراكب:"].get()

flight\_number = self.entries["رقم الرحلة:"].get()

departure = self.entries["المغادرة:"].get()

destination = self.entries["الوجهة:"].get()

date = self.entries["التاريخ:"].get()

seat\_number = self.entries["رقم المقعد:"].get()

if not all([name, flight\_number, departure, destination, date, seat\_number]):

messagebox.showerror("خطأ", "يجب ملء جميع الحقول.")

return

database.add\_reservation(name, flight\_number, departure, destination, date, seat\_number)

messagebox.showinfo("نجاح", "تم حجز الرحلة بنجاح!")

self.controller.show\_frame("ReservationsPage")

except Exception as e:

messagebox.showerror("خطأ", f"حدث خطأ: {e}")

3.3. صفحة عرض الحجوزات (reservations.py)

* تستخدم ttk.Treeview لعرض البيانات في جدول.
* تستخدم الدالة get\_all\_reservations من ملف database.py.

Python

import tkinter as tk

from tkinter import ttk, messagebox

import database

class ReservationsPage(tk.Frame):

def \_\_init\_\_(self, parent, controller):

tk.Frame.\_\_init\_\_(self, parent)

self.controller = controller

label = ttk.Label(self, text="قائمة الحجوزات", font=("Arial", 16, "bold"))

label.pack(pady=10)

self.tree = ttk.Treeview(self, columns=("ID", "Name", "Flight", "Departure", "Destination", "Date", "Seat"), show="headings")

self.tree.heading("ID", text="ID")

self.tree.heading("Name", text="اسم الراكب")

self.tree.heading("Flight", text="رقم الرحلة")

self.tree.heading("Departure", text="المغادرة")

self.tree.heading("Destination", text="الوجهة")

self.tree.heading("Date", text="التاريخ")

self.tree.heading("Seat", text="رقم المقعد")

self.tree.pack(fill="both", expand=True)

self.tree.bind("<Double-1>", self.on\_double\_click)

refresh\_button = ttk.Button(self, text="تحديث القائمة", command=self.refresh\_list)

refresh\_button.pack(pady=10)

back\_button = ttk.Button(self, text="العودة للصفحة الرئيسية", command=lambda: controller.show\_frame("HomePage"))

back\_button.pack(pady=5)

self.refresh\_list()

def refresh\_list(self):

"""يحذف البيانات القديمة ويقوم بتحميل البيانات الجديدة من قاعدة البيانات."""

for row in self.tree.get\_children():

self.tree.delete(row)

reservations = database.get\_all\_reservations()

for reservation in reservations:

self.tree.insert("", "end", values=reservation)

def on\_double\_click(self, event):

"""عند النقر المزدوج على صف، يتم عرض صفحة التعديل."""

selected\_item = self.tree.focus()

if not selected\_item:

return

item\_values = self.tree.item(selected\_item, "values")

reservation\_id = item\_values[0]

self.controller.show\_frame("EditReservationPage", reservation\_id=reservation\_id)

3.4. صفحة تعديل وحذف الحجوزات (edit\_reservation.py)

* هذه الصفحة تسمح بتعديل أو حذف حجز محدد.
* تستخدم الدالتين update\_reservation و delete\_reservation من database.py.

Python

import tkinter as tk

from tkinter import ttk, messagebox

import database

class EditReservationPage(tk.Frame):

def \_\_init\_\_(self, parent, controller):

tk.Frame.\_\_init\_\_(self, parent)

self.controller = controller

self.reservation\_id = None

self.entries = {}

label = ttk.Label(self, text="تعديل الحجز", font=("Arial", 16, "bold"))

label.pack(pady=10)

fields = ["ID:", "اسم الراكب:", "رقم الرحلة:", "المغادرة:", "الوجهة:", "التاريخ:", "رقم المقعد:"]

for i, field in enumerate(fields):

row = tk.Frame(self)

lbl = ttk.Label(row, text=field, width=15)

ent = ttk.Entry(row, width=30)

row.pack(pady=5)

lbl.pack(side="left", padx=5)

ent.pack(side="right", padx=5)

self.entries[field] = ent

self.entries["ID:"].config(state="readonly")

update\_button = ttk.Button(self, text="تحديث الحجز", command=self.update\_booking)

update\_button.pack(pady=10)

delete\_button = ttk.Button(self, text="حذف الحجز", command=self.delete\_booking)

delete\_button.pack(pady=5)

back\_button = ttk.Button(self, text="العودة لقائمة الحجوزات", command=lambda: controller.show\_frame("ReservationsPage"))

back\_button.pack(pady=5)

def load\_reservation\_data(self, reservation\_id):

"""يملأ الحقول ببيانات الحجز المحددة."""

self.reservation\_id = reservation\_id

conn = sqlite3.connect(database.DATABASE\_NAME)

cursor = conn.cursor()

cursor.execute("SELECT \* FROM reservations WHERE id=?", (reservation\_id,))

record = cursor.fetchone()

conn.close()

if record:

self.entries["ID:"].config(state="normal")

self.entries["ID:"].delete(0, tk.END)

self.entries["ID:"].insert(0, record[0])

self.entries["ID:"].config(state="readonly")

self.entries["اسم الراكب:"].delete(0, tk.END)

self.entries["اسم الراكب:"].insert(0, record[1])

self.entries["رقم الرحلة:"].delete(0, tk.END)

self.entries["رقم الرحلة:"].insert(0, record[2])

self.entries["المغادرة:"].delete(0, tk.END)

self.entries["المغادرة:"].insert(0, record[3])

self.entries["الوجهة:"].delete(0, tk.END)

self.entries["الوجهة:"].insert(0, record[4])

self.entries["التاريخ:"].delete(0, tk.END)

self.entries["التاريخ:"].insert(0, record[5])

self.entries["رقم المقعد:"].delete(0, tk.END)

self.entries["رقم المقعد:"].insert(0, record[6])

def update\_booking(self):

try:

name = self.entries["اسم الراكب:"].get()

flight\_number = self.entries["رقم الرحلة:"].get()

departure = self.entries["المغادرة:"].get()

destination = self.entries["الوجهة:"].get()

date = self.entries["التاريخ:"].get()

seat\_number = self.entries["رقم المقعد:"].get()

if not all([name, flight\_number, departure, destination, date, seat\_number]):

messagebox.showerror("خطأ", "يجب ملء جميع الحقول.")

return

database.update\_reservation(self.reservation\_id, name, flight\_number, departure, destination, date, seat\_number)

messagebox.showinfo("نجاح", "تم تحديث الحجز بنجاح!")

self.controller.show\_frame("ReservationsPage")

except Exception as e:

messagebox.showerror("خطأ", f"حدث خطأ: {e}")

def delete\_booking(self):

if messagebox.askyesno("تأكيد الحذف", "هل أنت متأكد من رغبتك في حذف هذا الحجز؟"):

database.delete\_reservation(self.reservation\_id)

messagebox.showinfo("نجاح", "تم حذف الحجز بنجاح!")

self.controller.show\_frame("ReservationsPage")

4. ملف التطبيق الرئيسي (main.py)

هذا الملف هو نقطة البداية. يقوم بإنشاء النافذة الرئيسية وإدارة التنقل بين الصفحات.

Python

import tkinter as tk

import database

from home import HomePage

from booking import BookingPage

from reservation import ReservationsPage

from edit\_reservation import EditReservationPage

class FlightReservationApp(tk.Tk):

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

tk.Tk.\_\_init\_\_(self)

self.title("تطبيق حجز رحلات الطيران")

self.geometry("800x600")

database.create\_table() # التأكد من إنشاء الجدول عند تشغيل التطبيق

container = tk.Frame(self)

container.pack(side="top", fill="both", expand=True)

container.grid\_rowconfigure(0, weight=1)

container.grid\_columnconfigure(0, weight=1)

self.frames = {}

for F in (HomePage, BookingPage, ReservationsPage, EditReservationPage):

page\_name = F.\_\_name\_\_

frame = F(parent=container, controller=self)

self.frames[page\_name] = frame

frame.grid(row=0, column=0, sticky="nsew")

self.show\_frame("HomePage")

def show\_frame(self, page\_name, \*\*kwargs):

"""يعرض صفحة معينة."""

frame = self.frames[page\_name]

if page\_name == "EditReservationPage" and "reservation\_id" in kwargs:

frame.load\_reservation\_data(kwargs["reservation\_id"])

frame.tkraise()

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

app = FlightReservationApp()

app.mainloop()

5. رفع المشروع على GitHub

1. إنشاء حساب على GitHub: إذا لم يكن لديك، قم بإنشاء واحد.
2. إنشاء مستودع جديد:
   * اذهب إلى صفحتك الرئيسية على GitHub.
   * انقر على زر "New repository".
   * أعطه اسمًا مثل Flight-Reservation-App واجعله عامًا (Public).
3. فتح سطر الأوامر (Command Prompt أو Terminal):
   * انتقل إلى مجلد مشروعك cd flight\_reservation\_app.
   * نفذ الأوامر التالية خطوة بخطوة:

Bash

git init

git add .

git commit -m "Initial commit of the project"

git branch -M main

git remote add origin https://github.com/[moonlightcalmnight-prog](https://github.com/moonlightcalmnight-prog/Flight_Reservation_App/commits?author=moonlightcalmnight-prog)/[Flight\_Reservation\_App](https://github.com/moonlightcalmnight-prog/Flight_Reservation_App).git

git push -u origin main

(تأكد من استبدال YourGitHubUsername باسم المستخدم الخاص بك).

6. تصدير التطبيق إلى ملف تنفيذي (.exe)

هذه الخطوة تتيح تشغيل التطبيق على أي جهاز يعمل بنظام Windows دون الحاجة إلى تثبيت بايثون.

1. تثبيت PyInstaller:

افتح سطر الأوامر ونفذ:

Bash

pip install pyinstaller

1. تصدير التطبيق:
   * تأكد من وجود جميع ملفات المشروع في مجلد واحد.
   * افتح سطر الأوامر داخل مجلد المشروع.
   * نفذ الأمر التالي:

Bash

pyinstaller --onefile --windowed main.py

* + --onefile: يجمع كل الملفات في ملف تنفيذي واحد.
  + --windowed: يمنع ظهور نافذة الأوامر السوداء عند تشغيل التطبيق.

1. الناتج:
   * ستجد ملف .exe داخل مجلد يسمى dist.
   * يمكنك نسخ هذا الملف وتشغيله مباشرة.

7. الخطوة الأخيرة

* تأكد من كتابة ملف README.md جيد يشرح كيفية تشغيل التطبيق، ما هي مميزاته، وكيف يمكن للمستخدمين المساهمة فيه.
* قم بإنشاء ملف مضغوط (.zip) يحتوي على كل الملفات التي تم ذكرها في قائمة التسليمات النهائية.

سأقوم بإعادة كتابة جميع الأكواد لك، مع استبدال كل الكلمات العربية باللغة الإنجليزية، لتكون الأكواد أكثر احترافية وقابلة للتطبيق عالميًا.

تم استبدال التعليقات وأسماء المتغيرات بكلمات إنجليزية.

1. إعداد قاعدة البيانات (database.py)

Python

import sqlite3

DATABASE\_NAME = "flights.db"

def create\_table():

"""

Connects to the database and creates the 'reservations' table if it doesn't exist.

"""

conn = sqlite3.connect(DATABASE\_NAME)

cursor = conn.cursor()

cursor.execute('''

CREATE TABLE IF NOT EXISTS reservations (

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

flight\_number TEXT NOT NULL,

departure TEXT NOT NULL,

destination TEXT NOT NULL,

date TEXT NOT NULL,

seat\_number TEXT NOT NULL

)

''')

conn.commit()

conn.close()

def get\_all\_reservations():

"""Retrieves all reservations from the database."""

conn = sqlite3.connect(DATABASE\_NAME)

cursor = conn.cursor()

cursor.execute("SELECT \* FROM reservations")

reservations = cursor.fetchall()

conn.close()

return reservations

def add\_reservation(name, flight\_number, departure, destination, date, seat\_number):

"""Adds a new reservation to the database."""

conn = sqlite3.connect(DATABASE\_NAME)

cursor = conn.cursor()

cursor.execute('''

INSERT INTO reservations (name, flight\_number, departure, destination, date, seat\_number)

VALUES (?, ?, ?, ?, ?, ?)

''', (name, flight\_number, departure, destination, date, seat\_number))

conn.commit()

conn.close()

def update\_reservation(reservation\_id, name, flight\_number, departure, destination, date, seat\_number):

"""Updates an existing reservation based on its ID."""

conn = sqlite3.connect(DATABASE\_NAME)

cursor = conn.cursor()

cursor.execute('''

UPDATE reservations

SET name=?, flight\_number=?, departure=?, destination=?, date=?, seat\_number=?

WHERE id=?

''', (name, flight\_number, departure, destination, date, seat\_number, reservation\_id))

conn.commit()

conn.close()

def delete\_reservation(reservation\_id):

"""Deletes a reservation based on its ID."""

conn = sqlite3.connect(DATABASE\_NAME)

cursor = conn.cursor()

cursor.execute("DELETE FROM reservations WHERE id=?", (reservation\_id,))

conn.commit()

conn.close()

2. تصميم واجهة المستخدم (GUI)

تم تغيير النصوص المعروضة للمستخدم لتكون باللغة الإنجليزية.

2.1. الصفحة الرئيسية (home.py)

Python

import tkinter as tk

from tkinter import ttk

class HomePage(tk.Frame):

def \_\_init\_\_(self, parent, controller):

tk.Frame.\_\_init\_\_(self, parent)

self.controller = controller

label = ttk.Label(self, text="Flight Reservation System", font=("Arial", 20, "bold"))

label.pack(pady=20)

book\_button = ttk.Button(self, text="Book a New Flight", command=lambda: controller.show\_frame("BookingPage"))

book\_button.pack(pady=10, ipadx=10, ipady=5)

view\_button = ttk.Button(self, text="View Reservations", command=lambda: controller.show\_frame("ReservationsPage"))

view\_button.pack(pady=10, ipadx=10, ipady=5)

2.2. صفحة حجز جديد (booking.py)

Python

import tkinter as tk

from tkinter import ttk, messagebox

import database

class BookingPage(tk.Frame):

def \_\_init\_\_(self, parent, controller):

tk.Frame.\_\_init\_\_(self, parent)

self.controller = controller

label = ttk.Label(self, text="Book a New Flight", font=("Arial", 16, "bold"))

label.pack(pady=10)

# Input fields

fields = ["Passenger Name:", "Flight Number:", "Departure:", "Destination:", "Date:", "Seat Number:"]

self.entries = {}

for i, field in enumerate(fields):

row = tk.Frame(self)

lbl = ttk.Label(row, text=field, width=15)

ent = ttk.Entry(row, width=30)

row.pack(pady=5)

lbl.pack(side="left", padx=5)

ent.pack(side="right", padx=5)

self.entries[field] = ent

save\_button = ttk.Button(self, text="Save Reservation", command=self.save\_booking)

save\_button.pack(pady=10)

back\_button = ttk.Button(self, text="Back to Home", command=lambda: controller.show\_frame("HomePage"))

back\_button.pack(pady=5)

def save\_booking(self):

try:

name = self.entries["Passenger Name:"].get()

flight\_number = self.entries["Flight Number:"].get()

departure = self.entries["Departure:"].get()

destination = self.entries["Destination:"].get()

date = self.entries["Date:"].get()

seat\_number = self.entries["Seat Number:"].get()

if not all([name, flight\_number, departure, destination, date, seat\_number]):

messagebox.showerror("Error", "All fields must be filled.")

return

database.add\_reservation(name, flight\_number, departure, destination, date, seat\_number)

messagebox.showinfo("Success", "Flight reserved successfully!")

self.controller.show\_frame("ReservationsPage")

except Exception as e:

messagebox.showerror("Error", f"An error occurred: {e}")

2.3. صفحة عرض الحجوزات (reservations.py)

Python

import tkinter as tk

from tkinter import ttk, messagebox

import database

class ReservationsPage(tk.Frame):

def \_\_init\_\_(self, parent, controller):

tk.Frame.\_\_init\_\_(self, parent)

self.controller = controller

label = ttk.Label(self, text="List of Reservations", font=("Arial", 16, "bold"))

label.pack(pady=10)

self.tree = ttk.Treeview(self, columns=("ID", "Name", "Flight", "Departure", "Destination", "Date", "Seat"), show="headings")

self.tree.heading("ID", text="ID")

self.tree.heading("Name", text="Passenger Name")

self.tree.heading("Flight", text="Flight Number")

self.tree.heading("Departure", text="Departure")

self.tree.heading("Destination", text="Destination")

self.tree.heading("Date", text="Date")

self.tree.heading("Seat", text="Seat Number")

self.tree.pack(fill="both", expand=True)

self.tree.bind("<Double-1>", self.on\_double\_click)

refresh\_button = ttk.Button(self, text="Refresh List", command=self.refresh\_list)

refresh\_button.pack(pady=10)

back\_button = ttk.Button(self, text="Back to Home", command=lambda: controller.show\_frame("HomePage"))

back\_button.pack(pady=5)

self.refresh\_list()

def refresh\_list(self):

"""Clears old data and loads new data from the database."""

for row in self.tree.get\_children():

self.tree.delete(row)

reservations = database.get\_all\_reservations()

for reservation in reservations:

self.tree.insert("", "end", values=reservation)

def on\_double\_click(self, event):

"""On double-click, shows the edit page for the selected item."""

selected\_item = self.tree.focus()

if not selected\_item:

return

item\_values = self.tree.item(selected\_item, "values")

reservation\_id = item\_values[0]

self.controller.show\_frame("EditReservationPage", reservation\_id=reservation\_id)

2.4. صفحة تعديل وحذف الحجوزات (edit\_reservation.py)

Python

import tkinter as tk

from tkinter import ttk, messagebox

import database

import sqlite3

class EditReservationPage(tk.Frame):

def \_\_init\_\_(self, parent, controller):

tk.Frame.\_\_init\_\_(self, parent)

self.controller = controller

self.reservation\_id = None

self.entries = {}

label = ttk.Label(self, text="Edit Reservation", font=("Arial", 16, "bold"))

label.pack(pady=10)

fields = ["ID:", "Passenger Name:", "Flight Number:", "Departure:", "Destination:", "Date:", "Seat Number:"]

for i, field in enumerate(fields):

row = tk.Frame(self)

lbl = ttk.Label(row, text=field, width=15)

ent = ttk.Entry(row, width=30)

row.pack(pady=5)

lbl.pack(side="left", padx=5)

ent.pack(side="right", padx=5)

self.entries[field] = ent

self.entries["ID:"].config(state="readonly")

update\_button = ttk.Button(self, text="Update Reservation", command=self.update\_booking)

update\_button.pack(pady=10)

delete\_button = ttk.Button(self, text="Delete Reservation", command=self.delete\_booking)

delete\_button.pack(pady=5)

back\_button = ttk.Button(self, text="Back to Reservations List", command=lambda: controller.show\_frame("ReservationsPage"))

back\_button.pack(pady=5)

def load\_reservation\_data(self, reservation\_id):

"""Fills the form with the selected reservation's data."""

self.reservation\_id = reservation\_id

conn = sqlite3.connect(database.DATABASE\_NAME)

cursor = conn.cursor()

cursor.execute("SELECT \* FROM reservations WHERE id=?", (reservation\_id,))

record = cursor.fetchone()

conn.close()

if record:

self.entries["ID:"].config(state="normal")

self.entries["ID:"].delete(0, tk.END)

self.entries["ID:"].insert(0, record[0])

self.entries["ID:"].config(state="readonly")

self.entries["Passenger Name:"].delete(0, tk.END)

self.entries["Passenger Name:"].insert(0, record[1])

self.entries["Flight Number:"].delete(0, tk.END)

self.entries["Flight Number:"].insert(0, record[2])

self.entries["Departure:"].delete(0, tk.END)

self.entries["Departure:"].insert(0, record[3])

self.entries["Destination:"].delete(0, tk.END)

self.entries["Destination:"].insert(0, record[4])

self.entries["Date:"].delete(0, tk.END)

self.entries["Date:"].insert(0, record[5])

self.entries["Seat Number:"].delete(0, tk.END)

self.entries["Seat Number:"].insert(0, record[6])

def update\_booking(self):

try:

name = self.entries["Passenger Name:"].get()

flight\_number = self.entries["Flight Number:"].get()

departure = self.entries["Departure:"].get()

destination = self.entries["Destination:"].get()

date = self.entries["Date:"].get()

seat\_number = self.entries["Seat Number:"].get()

if not all([name, flight\_number, departure, destination, date, seat\_number]):

messagebox.showerror("Error", "All fields must be filled.")

return

database.update\_reservation(self.reservation\_id, name, flight\_number, departure, destination, date, seat\_number)

messagebox.showinfo("Success", "Reservation updated successfully!")

self.controller.show\_frame("ReservationsPage")

except Exception as e:

messagebox.showerror("Error", f"An error occurred: {e}")

def delete\_booking(self):

if messagebox.askyesno("Confirm Deletion", "Are you sure you want to delete this reservation?"):

database.delete\_reservation(self.reservation\_id)

messagebox.showinfo("Success", "Reservation deleted successfully!")

self.controller.show\_frame("ReservationsPage")

3. ملف التطبيق الرئيسي (main.py)

Python

import tkinter as tk

import database

from home import HomePage

from booking import BookingPage

from reservation import ReservationsPage

from edit\_reservation import EditReservationPage

class FlightReservationApp(tk.Tk):

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

tk.Tk.\_\_init\_\_(self)

self.title("Flight Reservation App")

self.geometry("800x600")

database.create\_table() # Ensure the table is created on app startup

container = tk.Frame(self)

container.pack(side="top", fill="both", expand=True)

container.grid\_rowconfigure(0, weight=1)

container.grid\_columnconfigure(0, weight=1)

self.frames = {}

for F in (HomePage, BookingPage, ReservationsPage, EditReservationPage):

page\_name = F.\_\_name\_\_

frame = F(parent=container, controller=self)

self.frames[page\_name] = frame

frame.grid(row=0, column=0, sticky="nsew")

self.show\_frame("HomePage")

def show\_frame(self, page\_name, \*\*kwargs):

"""Displays a specific frame."""

frame = self.frames[page\_name]

if page\_name == "EditReservationPage" and "reservation\_id" in kwargs:

frame.load\_reservation\_data(kwargs["reservation\_id"])

frame.tkraise()

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

app = FlightReservationApp()

app.mainloop()