**Faculty of Engineering**
**Department of Industrial Engineering**
**King Abdulaziz University**

## Computer Applications in Industrial Engineering (IE-322)

# Dynamic Trip Pricing App



| NAME | ID |
|---|---|
| Omar Almahyawi | 2239270 |
| Abdulrahman Aladhadh | 2338137 |
| Yazan Ismail | 2243640 |
| Hazem Bafail | 2237006 |
| Yousef Baeshen | 2340306 |



**Instructor: Dr. Muhammad Atif Shahzad**
**Section : HA**
**Team : 12**
**Submitted on 3 / 5 / 2025**

# Table of contents

## Introduction

The Dynamic Trip Pricing App is a Python program that calculates the cost of a trip between two locations. It uses:

- ❖ Geopy to find distance between two places.
- ❖ Pytz and datetime to get the current time in Saudi Arabia.
- ❖ Tkinter to create a simple window for users.

This app is useful for transportation services like ride-hailing apps (Uber, Kareem) to calculate trip prices based on distance and time of day.

## Required Libraries

| Library | Purpose |
|---|---|
| tkinter | To build the app window (GUI) |
| geopy | To calculate distance |
| pytz | To get Saudi time zone |
| datetime | To get current hour |

```
import tkinter as tk
from tkinter import messagebox
from geopy.geocoders import Nominatim
from geopy.distance import geodesic
from datetime import datetime
import pytz
```

## Code Explanation

### Confirm Location Function

This function is used to make sure the address found is correct before continuing.

```
def confirm_location(address, label):
    return messagebox.askyesno("Confirm Location", f"{label}:\n{address}\n\nIs this the correct location?")
```

What does this function do?

- It opens a Yes/No popup.
- It shows the full address for the location that was found.
- If the user clicks Yes, the function returns True.
- If the user clicks No, it returns False.

This is helpful to avoid using a wrong location by mistake.

### Price Calculation Function

This is the main function in our program. It handles:

- Getting the user's input
- Finding locations
- Calculating distance
- Calculating base price
- Adjusting price based on the time of day

**1. Get User Inputs**

```
def calculate_price():
    start_input = entry_start.get()
    end_input = entry_end.get()
```

Gets the text entered by the user in the “Start” and “End” boxes.

**2. Find Location Coordinates**

```
geolocator = Nominatim(user_agent="dynamic_pricing_gui")

try:
    loc1 = geolocator.geocode(start_input)
    if not loc1:
        messagebox.showerror("Error", "Starting location not found.")
        return

    if not confirm_location(loc1.address, "Starting location"):
        return
```

- Uses a geolocator to search for the start location.
- If not found, it shows an error.
- If found, it asks the user to confirm the location.

The same process is repeated for the destination:

```
    loc2 = geolocator.geocode(end_input)
    if not loc2:
        messagebox.showerror("Error", "Destination location not found.")
        return

    if not confirm_location(loc2.address, "Destination location"):
        return
```

**3. Calculate Distance and Base Price**

```
coord1 = (loc1.latitude, loc1.longitude)
coord2 = (loc2.latitude, loc2.longitude)

distance_km = geodesic(coord1, coord2).kilometers
base_price = round(distance_km * 2.0, 2)  # 2 SAR per km
```

coord1 and coord2 are the GPS positions.

geodesic() finds the distance between them.

Base price is 2 SAR per kilometer.

**4. Time-Based Price Increase**

```
sa_time = datetime.now(pytz.timezone("Asia/Riyadh"))
hour = sa_time.hour

if 6 <= hour < 12:
    multiplier = 1.1
elif 12 <= hour < 18:
    multiplier = 1.2
elif 18 <= hour < 23:
    multiplier = 1.3
else:
    multiplier = 1.05
```

Get the current hour in Saudi Arabia.

Increases the price depending on the time:

- Morning = +10%
- Afternoon = +20%
- Evening = +30%
- Late night = +5%

**5. Final Price + Show Result**

```
final_price = round(base_price * multiplier, 2)
increase_percent = int((multiplier - 1.0) * 100)  # Example: 1.3 -> 30%

result_label.config(
    text=f"From: {start_input}\n"
         f"To: {end_input}\n"
         f"Distance: {round(distance_km, 2)} km\n"
         f"Saudi Time: {sa_time.strftime('%H:%M')}\n\n"
         f"Base Price: {base_price} SAR\n"
         f"Time-based Increase: {increase_percent}%\n"
         f"Final Price: {final_price} SAR"
)
```

- ❖ Calculates the final price with a multiplier.
- ❖ Shows all results in the app screen.

## Designing the GUI (Graphical User Interface)

We used Tkinter to build a simple and clean window for the user to interact with.
This is what the GUI includes:

| Component | Purpose |
|---|---|
| Label (Title) | Show the title of the app |
| Entry (Textbox) | For user to enter start and end |
| Button (Calculate) | To run the price calculation |
| Label (Results) | To display the output |
| Button (Exit) | To close the app |

**Code for GUI Design**

```
# GUI setup
window = tk.Tk()
window.title("Trip Price")
window.geometry("520x450")
window.config(bg="#f7f7f7")
```

Creates the main window.

Sets the title to "Trip Price".

Sets size and background color.

**Title Label**

```
tk.Label(window, text="Your trip price", font=("Arial", 18, "bold"), bg="#f7f7f7", fg="#333").pack(pady=10)
```

Shows the app title at the top in big font.

**Entry Boxes for Locations**

```
entry_start = tk.Entry(window, width=50)
entry_start.insert(0, "Enter Location, City")
entry_start.pack(pady=5)

entry_end = tk.Entry(window, width=50)
entry_end.insert(0, "Enter destination, City")
entry_end.pack(pady=5)
```

Two text boxes: one for start, one for destination.

We use .insert() to put a placeholder text inside each box.

**Calculate Button**

```
tk.Button(window, text="Calculate Price", command=calculate_price, bg="#4CAF50", fg="white", width=20).pack(pady=10)
```

This red button runs the calculate_price() function when clicked.

The color and size make it easy to see.

**Result Label**

```
result_label = tk.Label(window, text="", font=("Arial", 11), bg="#f7f7f7", fg="#333", justify="left")
result_label.pack(pady=10)
```

- ❖ This label will show the results (distance, price, time).
- ❖ Starts empty and is updated later.

**Exit Button**

```
tk.Button(window, text="Exit", command=window.quit, bg="#d9534f", fg="white", width=10).pack(pady=5)
```

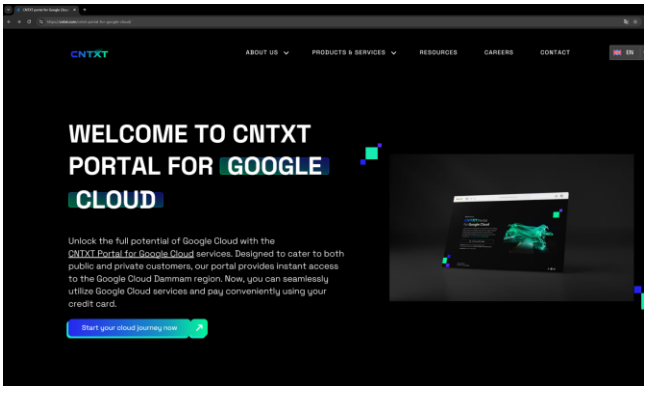
Red button that closes the window when clicked.

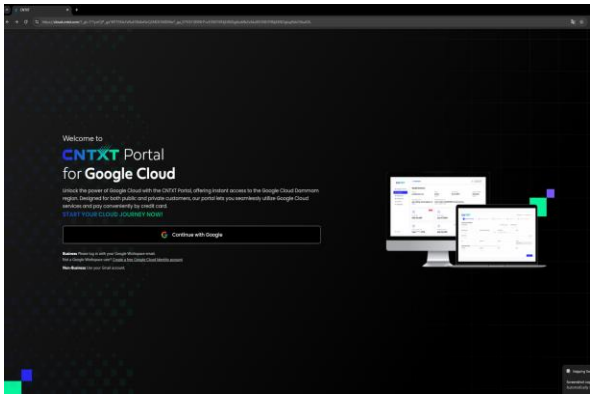
**Start the GUI Loop**

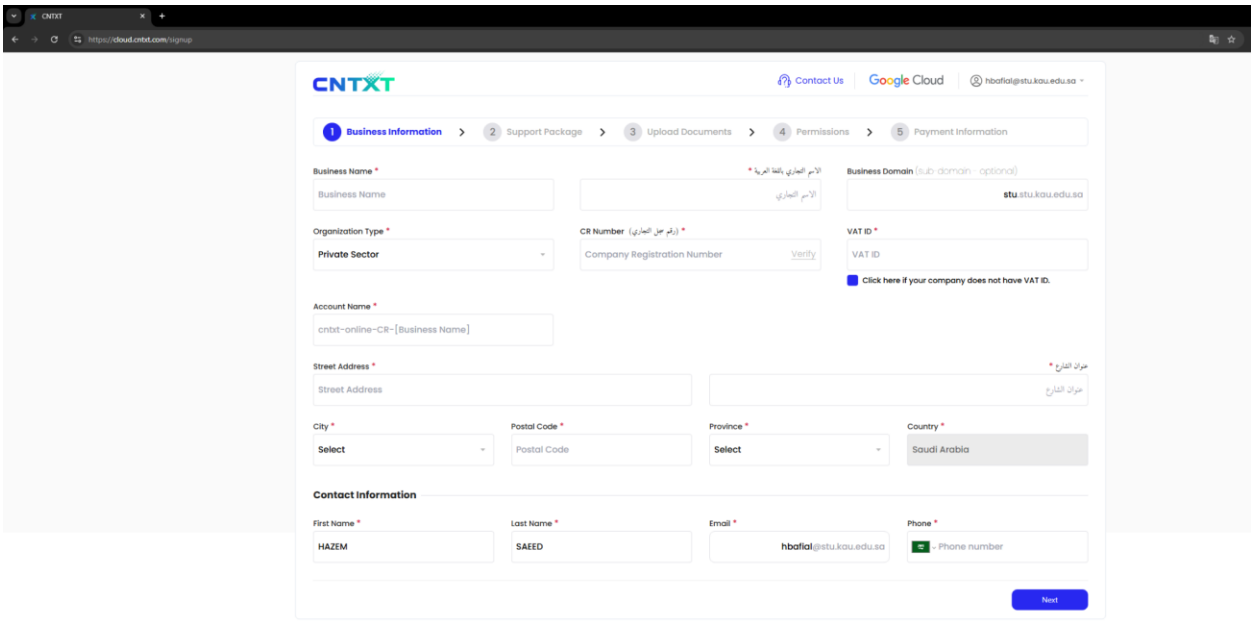
```
window.mainloop()
```

This keeps the window open and running until the user closes it.

Unfortunately, we were unable to activate Google Maps APIs in Saudi Arabia using either a personal or a university email. This is due to regional restrictions enforced by the official Google Cloud reseller in Saudi Arabia, CNTXT, a joint venture between Aramco and Cognite. CNTXT has exclusive control over Google Cloud services in the Kingdom, and in order to enable billing and access to APIs such as Google Maps, they require users to have a valid commercial registration (CR) and operate as a registered business entity. Individual users, students, and academic accounts are not eligible to activate these services, even with a valid credit card. As a result, we were unable to proceed with Google Maps API and instead used the free `geopy` library with OpenStreetMap as a suitable alternative.






## Conclusion

This project shows how Python can be used to solve real-life problems with a simple user interface. The Dynamic Trip Pricing App helps users calculate trip costs based on:

- Distance between two places
- Time of day (morning, afternoon, evening, or night)

We combined several Python tools to build this project:

- Tkinter for the window and buttons
- Geopy for calculating distances
- Datetime and Pytz for using Saudi time
- Visual Studio to write and run the code easily

The app is simple, but powerful. It shows how programmers can build tools for transportation, delivery services, and other businesses that depend on trip pricing