جامعة أم القرى
UMM AL-QURA UNIVERSITY

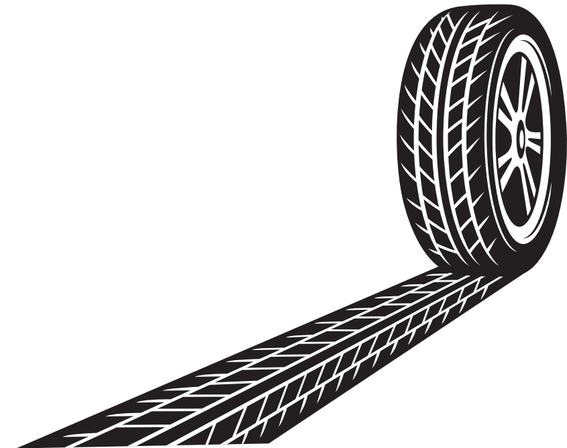
TAXI MANAGEMENT SYSTEM

Our Team :

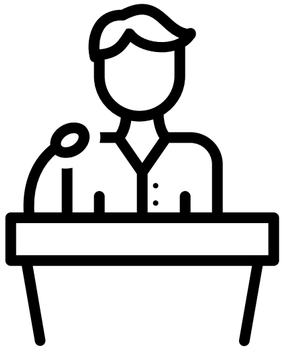
Layan Alsayed 444001570
Ghala alqarni 444000585
Aya Babkoor 444002180
Rema Alghamdi 444001279

Supervised by:

Eiman Alharby

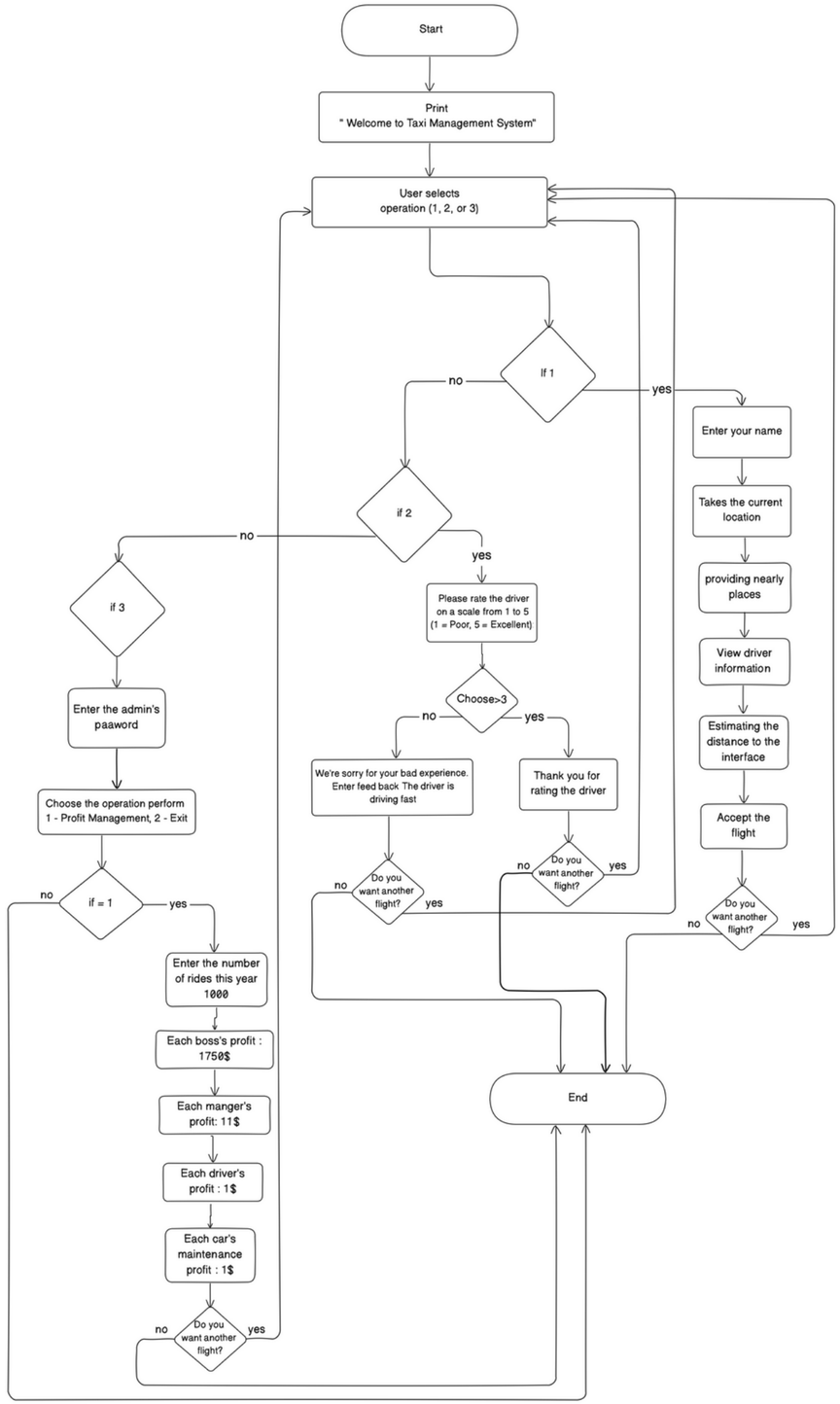


INTRODUCTION



MEET THE TAXI MANAGEMENT SYSTEM (TMS): YOUR GO-TO SOLUTION FOR A SMARTER, MORE EFFICIENT TAXI EXPERIENCE. TMS SIMPLIFIES OPERATIONS FOR MANGERS, AND ENSURES CUSTOMERS ENJOY A SEAMLESS, HASSLE-FREE RIDE WITH USER-FRIENDLY FEATURES.

FLOWCHART


Start
Print
" Welcome to Taxi Management System"
User selects
operation (1, 2, or 3)
If 1
no
yes
Enter your name
Takes the current location
providing nearly places
View driver information
Estimating the distance to the interface
Accept the flight
Do you want another flight?
no
yes
if 2
no
yes
Please rate the driver on a scale from 1 to 5 (1 = Poor, 5 = Excellent)
Choose>3
no
yes
We're sorry for your bad experience. Enter feed back The driver is driving fast
Thank you for rating the driver
Do you want another flight?
no
yes
Do you want another flight?
no
yes
if 3
Enter the admin's paaword
Choose the operation perform 1 - Profit Management, 2 - Exit
if = 1
no
yes
Enter the number of rides this year 1000
Each boss's profit : 1750$
Each manger's profit: 11$
Each driver's profit : 1$
Each car's maintenance profit : 1$
Do you want another flight?
no
yes
End

LET'S WALK THROUGH THE INTERACTION WITH THE TAXI MANAGEMENT SYSTEM STEP BY STEP:

THE USER IS WELCOMED TO THE TAXI MANAGEMENT SYSTEM AND PROMPTED TO SELECT AN ACTION:

- ENTER 1 TO REQUEST A RIDE.

- ENTER 2 TO RATE A DRIVER.

- ENTER 3 IF THE USER IS A MANAGER.

IF THE USER CHOOSES TO REQUEST A RIDE (OPTION 1)

TCP FOR SERVICE AND CLIENT

```python
import socket
import threading
import requests
import googlemaps
```

SERVER

- socket: Provides low-level networking operations, it is also in the client .
- threading: Enables the creation and management of threads for parallel execution.
- requests: Used for making HTTP requests.
- googlemaps: A Python client library for the Google Maps API.

```python
def send_message(client_socket, message):
    try:
        client_socket.send(message.encode('utf-8'))
    except (BrokenPipeError, ConnectionResetError):
        print("Client disconnected unexpectedly.")
        raise
```

- This function is responsible for sending a message from the server to the connected client.

```python
def receive_message(client_socket):
    return client_socket.recv(1024).decode('utf-8')
```

- This function is responsible for receiving a message from the connected client.

```python
def get_location(api_key):
    gmaps = googlemaps.Client(key=api_key)

    try:
        # Use Google Maps Geolocation API to get the client's location
        url = f"https://www.googleapis.com/geolocation/v1/geolocate?key={api_key}"
```

- This function It allows access to the Google Maps Geolocation API and Google Places API

```python
# Use Google Places API to get nearby locations
places_result = gmaps.places_nearby(location=latlng, radius=5000, type='point_of_interest')

# Extract the names of the nearby places
nearby_locations = [place['name'] for place in places_result['results']]
```

- In the same function This part of the code aims to find information about places of interest near a specific geographical location.

```
def start_server(api_key):
    host = '127.0.0.1'
    port = 5566

    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((host, port))
    server_socket.listen()
```

- **This function initializes a basic server, listens for incoming connections, and prints accepted connection information.**

```
def main():
    # Replace the placeholder with your actual API key
    api_key = "AIzaSyD5AXYAJfBWC4CQLS2dN_0yyWyFjqhbfjI"
    start_server(api_key)
```

```
if __name__ == "__main__":
    main()
```

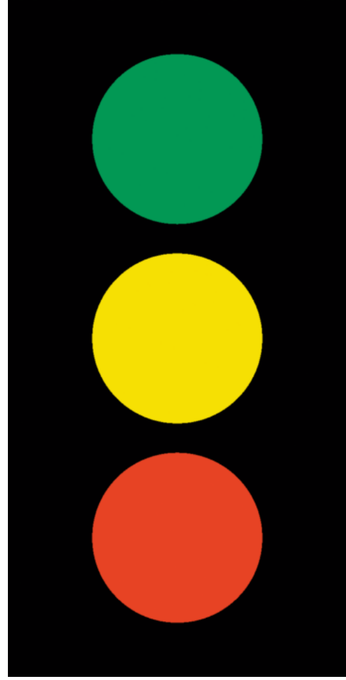
- **The `main` function in the provided code initializes a server by calling the `start_server` function with a specified API key. When the script is executed as the main program (`__name__ == "__main__"`), it runs the `main` function, initiating the server with the provided API key.**

```python
def main0():
    host = '127.0.0.1'
    port = 5566

    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((host, port))
    send_message(client_socket,"1")
    try:
        while True:
            response = receive_message(client_socket)
            print(response)

            if "Please enter your name" in response:
                name = input("Your name: ")
                send_message(client_socket, name)
            elif "Choose a destination" in response:
                destination = input("Choose a destination (e.g., 1, 2, 3, 4): ")
                send_message(client_socket, destination)
            elif "Do you accept the ride?" in response:
                user_input = input("Do you accept the ride? (yes/no): ")
                send_message(client_socket, user_input)
                if user_input.lower() == 'no':
                    print("Thank you. Goodbye!")
                    break  # Break out of the loop if the ride is declined
            elif "Continue the ride?" in response:
                user_input = input("Continue the ride? (yes/no): ")
                send_message(client_socket, user_input)
            elif "Do you want to request another ride?" in response:
                user_input = input("Do you want to request another ride? (yes/no): ")
                send_message(client_socket, user_input)
                if user_input.lower() == 'no':
                    print("Goodbye!")
                    break
            else:
                print("Invalid response from the server.")
    except Exception as e:
        print(f"An error occurred: {e}")
```

- `main0` simulates client-side interaction with a ride-request system. It establishes a connection, sends responses to server prompts, and handles user input for actions like naming, destination selection, ride acceptance, continuation, and ride requests, with exception handling for errors.



THE OUTPUT

```
(base) ghalaalqarni@Ghalas-MacBook-Pro ~ % cd /Users/ghalaalqarni/Documents/Distributed\ programming\ project
(base) ghalaalqarni@Ghalas-MacBook-Pro Distributed programming project % python /Users/ghalaalqarni/Documents/Distributed\ programming\
project/clint\ 2.py
~~ Welcome to Taxi Management System ~~
Select 1 if you want to request a ride
Select 2 if you want to rate a driver
Select 3 if you're a Manager
Enter your choice: 1
Requesting a ride...
Please enter your name:
Your name: ghala
Your current location: 21.4237184,39.7606912
Invalid response from the server.
Nearby locations: 1 - Dar Raies HotelNearby locations: 2 - Rawhaa ResortNearby locations: 3 - Prince Abdul Majeed Secondary SchoolNearby
locations: 4 - skyinnChoose a destination from the nearby locations (e.g., 1, 2, 3, 4):
Choose a destination (e.g., 1, 2, 3, 4): 1
Your driver is John. Car: Toyota Prius, License Plate: ABC123
Invalid response from the server.
Estimated distance to your destination: 10.0 milesDo you accept the ride? (yes/no):
Do you accept the ride? (yes/no): yes
Simulating ride...
Invalid response from the server.
Your ride is complete. Here is your bill: $10.00Do you want to request another ride? (yes/no):
Do you want to request another ride? (yes/no): yes
Your current location: 21.4237184,39.7606912
Invalid response from the server.
Nearby locations: 1 - Dar Raies HotelNearby locations: 2 - Rawhaa ResortNearby locations: 3 - Prince Abdul Majeed Secondary SchoolNearby
locations: 4 - skyinnChoose a destination from the nearby locations (e.g., 1, 2, 3, 4):
Choose a destination (e.g., 1, 2, 3, 4): 3
Your driver is John. Car: Toyota Prius, License Plate: ABC123
Invalid response from the server.
Estimated distance to your destination: 10.0 milesDo you accept the ride? (yes/no):
Do you accept the ride? (yes/no): no
Thank you. Goodbye!
Do you want to continue? (yes/no): yes
Select 1 if you want to request a ride
Select 2 if you want to rate a driver
Select 3 if you're a Manager
Enter your choice: 2
```

CONCLUSION

TCP ensures reliable, robust, and ordered client-server communication with effective flow control and error detection, emphasizing its widespread adoption for dependable data exchange.

IF THE USER WOULD LIKE TO RATE A DRIVER (OPTION 2)

TCP CHAT ROOM FOR MANAGEMENT SUGGESTIONS

```
def start_client2(host, port):
    host = '127.0.0.1'
    port = 5566
```

```
def main3(host, port):
    start_client2(host, port)
```

- `start_client2` connects to a server, initiates driver rating, collects feedback, and concludes; `main3` invokes this for simulating driver rating in a ride-sharing system.

```
def start_server_rating():
    host = '127.0.0.1'
    port = 5567  # Use a different port for the rating server
```

- start_server_rating initializes a server on a different port, accepts client connections, and spawns threads for handling driver ratings in a concurrent manner.

```
def handle_client2(client_socket, port):
    try:
        if "Please rate the driver" in receive_message(client_socket):
            send_message(client_socket, "We're sorry to hear that your experience was less than satisfactory.")
            # Directly ask for feedback and suggestions for improvement
            send_message(client_socket, "\nPlease share additional details about your experience and any sugges
            feedback = receive_message(client_socket)
            print(f"Received feedback: {feedback}")
            # Send a confirmation message to the client
            send_message(client_socket, "Thank you for providing feedback. We have received your suggestions ar

    except (BrokenPipeError, ConnectionResetError):
        print("Client disconnected unexpectedly.")
    finally:
        # Close the connection
        client_socket.close()
        print("Connection closed.")
```

- `handle_client2` manages low driver ratings, requests and acknowledges feedback from clients, and handles unexpected disconnections.

```
def main():
    start_server_rating()


if __name__ == "__main__":
    main()
```

- The `main` function initiates the server for handling driver ratings when the script is executed as the main program.

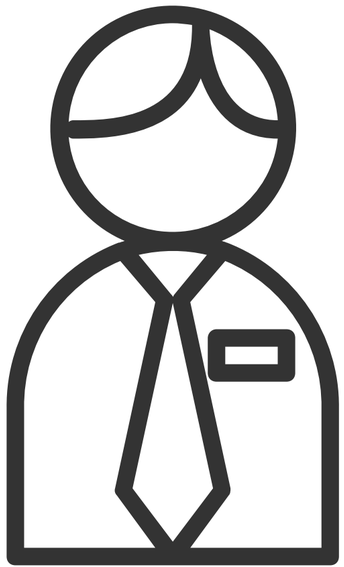
THE OUTPUT

```
Do you want to continue? (yes/no): yes
Select 1 if you want to request a ride
Select 2 if you want to rate a driver
Select 3 if you're a Manager
Enter your choice: 2
Please rate the driver on a scale from 1 to 5 (1 = Poor, 5 = Excellent): 5
Thank you for rating the driver 5 out of 5!
Do you want to continue? (yes/no): yes
Select 1 if you want to request a ride
Select 2 if you want to rate a driver
Select 3 if you're a Manager
Enter your choice: 2
Please rate the driver on a scale from 1 to 5 (1 = Poor, 5 = Excellent): 1
We're sorry for your bad experience.
Please wait a moment until a manager gets connected with you.
We're sorry to hear that your experience was less than satisfactory.
Please share additional details about your experience and any suggestions for improvement:
Enter feed back The driver is driving fast
Thank you for providing feedback. We have received your suggestions and will work on improving our service.
Do you want to continue? (yes/no): yes
```

CONCLUSION

Utilizing TCP for dependable client-server communication, this system ensures robust data exchange with efficient error detection. Threading supports scalability, fostering faster issue resolution, proactive problem-solving, and improved communication among customers, managers, and drivers.

IF THE USER IS A MANAGER (OPTION 3)



MULTIPROCESSING

In our program, we use multiprocessing to make things run faster and smoother. The manager uses it to figure out how much profit each employee makes based on their work per year . We also calculate the maintenance value for each car using this approach. This not only speeds up the program but also makes it easier to manage and track employee and car- related finances.

Profit percentage

-bosses (30%)
- mangers (20%)
- drivers (30%)
- cars (20%)

Number of indvalules

-4 bosses
-180 mangers
-2660 drivers
-1770 cars

STEPS

1-Manger enters the number of rides in a year

```
C:\Users\96650\PycharmProjects\taxiMP\venv\Scripts\
Enter the number of rides this year
50000
```

ofit gets calculated and divided for each group individually

```
def cal_boss(ridesNum):
    for i in range(ridesNum) :
        total=ridesNum*10
        bossShare=total*(100-30)/100  #30%
        eachBoss=int(bossShare/4)

        return eachBoss
```

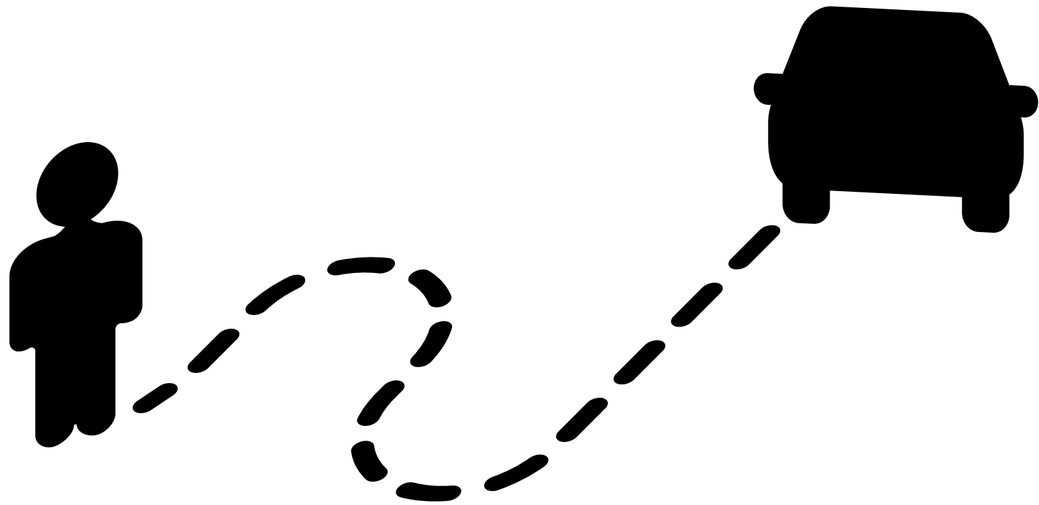
3-First Time is printed without multiprocessing

```
print(f"Each boss's profit: {resultBoss}$\n"
      f"Each Manger's profit: {resultManger}$\n"
      f"Each driver's profit:{resultDriver}$\n"
      f"Each car's maintance profit: {resultCar}$")

print("\nNormal time:",end - start)
```

4-Second time is printed with multiprocessing

```
start = time.time()
p1 = mp.Process(target=cal_boss, args=(rides,))
p2 = mp.Process(target=cal_manger, args=(rides,))
p3 = mp.Process(target=cal_driver, args=(rides,))
p4 = mp.Process(target=cal_car, args=(rides,))
```



Output

```
Enter the number of rides this year
50000
Each boss's profit: 87500$
Each Manger's profit: 555$
Each driver's profit:57$
Each car's maintance profit: 84$

Normal time: 0.006299018859863281
Multiprossing time 0.0313417911529541
```

Conclusion

Multiprocessing took half of the time. It is faster for tasks that can be divided into independent subtasks processed concurrently.

THANK YOU