<TransitMode>

A Term Project

Presented to Mr. Stephen Ruiz

In Partial Fulfillment of the

Requirements for the Course Programming Logic and Design (PROLOGI)

---

by

<Sarmiento>, <Juan Paolo> <L.> - JPS

<Sy>, <Wencarll Cynric> <Q.> - WQS

<Uy>, <Nathan Kyle> <T.> - NKU

<EQ3>
<Schedule>

April 20, 2023

Table of Contents

List of Figures

List Tables

I. Introduction

It can be said that transportation is the backbone of a country and its economy. From transporting goods, materials, and services that are utilized for manufacturing to be either used internally, or exported, to moving around people, bringing them to their jobs, schools, or wherever they may need to be. It is paramount that safe, effective, and comfortable means of transportation can be developed for the benefit of a country and its citizens. As such, one of the ways the group has thought of making this into a possibility is by creating a simple program that would assist commuters/travellers in their daily commute to different destinations in Metro Manila.

A. Background of the Study

At the height of the pandemic in 2020, there were clamors from various cycling and commuter groups which were also joined by politicians such as then Antique Representative Loren Legarda to "move people not cars", claiming that current policies and infrastructure were catering more to the 12 percent of Filipino households in Metro Manila who own cars as opposed to the 88 percent who do not, who then rely on other modes of mobility such as cycling and taking public transportation.

Indeed, the majority of Filipinos who do not have the fortune of owning their own vehicle and opt to take public transportation face many problems on an almost daily basis. In fact, the result of the 2022 Urban Mobile Readiness Index which ranks different cities around the world based on their "urban mobility readiness" which is based on factors such as quality of its public transit system, commute speed, affordability and others concluded that Metro Manila's public transportation system has a way to go compared to other cities. It stated issues such as poor road quality and its limited connectivity to other regions. Additionally, it also saw problems with the speed, connectivity, station density, and waiting times in Metro Manila's transportation system.

Furthermore, it does not necessarily take a study to become aware of the problems faced by the commuting public. Long lines in transport terminals; jam-packed trains, jeepneys, and buses; very small to almost non existent sidewalks; vehicles that are in poor condition; and many more.

In order to aid the commuting public in their travels amid the current woes in public transportation in the Philippines, various platforms, applications, and websites have been created. Services such as Waze and Google maps are helpful to those who are driving to see the estimated time it would take to reach their destination. Meanwhile, Angkas, Grab, Joyride, and other Transportation Network Vehicle Services (TNVS) are used by commuters as alternative forms of transportation.

In order to contribute and ease the burden faced by commuters, the group has developed a simple program called TransitMode whose specifics are discussed in the succeeding sections of the paper.

B. Problem Statement

The public transportation system in the Philippines, particularly in Metro Manila remains largely inefficient, burdening the majority of Filipinos who do not own a car - wasting time, effort, and resources. This research/project sought to make the commuting experience of the public much easier through the features of the program developed.

C. Objectives

C.1 General Objective

- Assist the commuting public in their travels.

C.2 Specific Objectives

- Lessen the stress/anxiety experienced by commuters.
- Help commuters plan their commute.

D. Significance of the Project

This project aims to provide benefits to the following:

- Local government agencies - Since the program would show the estimated time and fare that specific modes of transportation would take on the user, they may use this as a basis to improve on certain things within their jurisdiction such as traffic policies, public transportation infrastructure, etc. to lessen the time it takes to travel from one place to another and to improve the experience of commuters.

- Commuters - The main beneficiaries of our program, commuters may get a general idea of the time and money it would take to reach their destination, hopefully lessening stress since they may use it to plan ahead.

II. Review of Related Literature

a. Evaluation of Transit Mobile Applications: Case study of Transit App in Toronto.

There has been a rise in the use of mobile applications that ease the way people live. One such example would be the simplification of public transportation. The Transit App is one of the most popular transit apps in Toronto, Canada, and has gained widespread use by commuters.

The study concluded that the most valuable part of a transit app would be the ease of use and accuracy.

b. Key Elements of Mobility Apps for Improving Urban Travel Patterns: A Literature Review

With the popularity of smartphones, it has been identified that it could be used as a way to encourage the public to depend less on cars and patronize taking public transportation through mobility apps, particularly Mobility as a service schemes. This literature review by Castro et al. discussed aspects of a mobility app that could make it have a wider reach to the general public. They have found that eco feedback and a reward system are generally well received by the users. Additionally, functions such as displaying real time information on the state of traffic, comfort (crowd density on public transport), and health (calories burned, etc.) are considered useful to have on an application as this may entice more users to use it, thus making it easier for them to use public transportation, lessening car dependency, which in turn reduces traffic congestions.

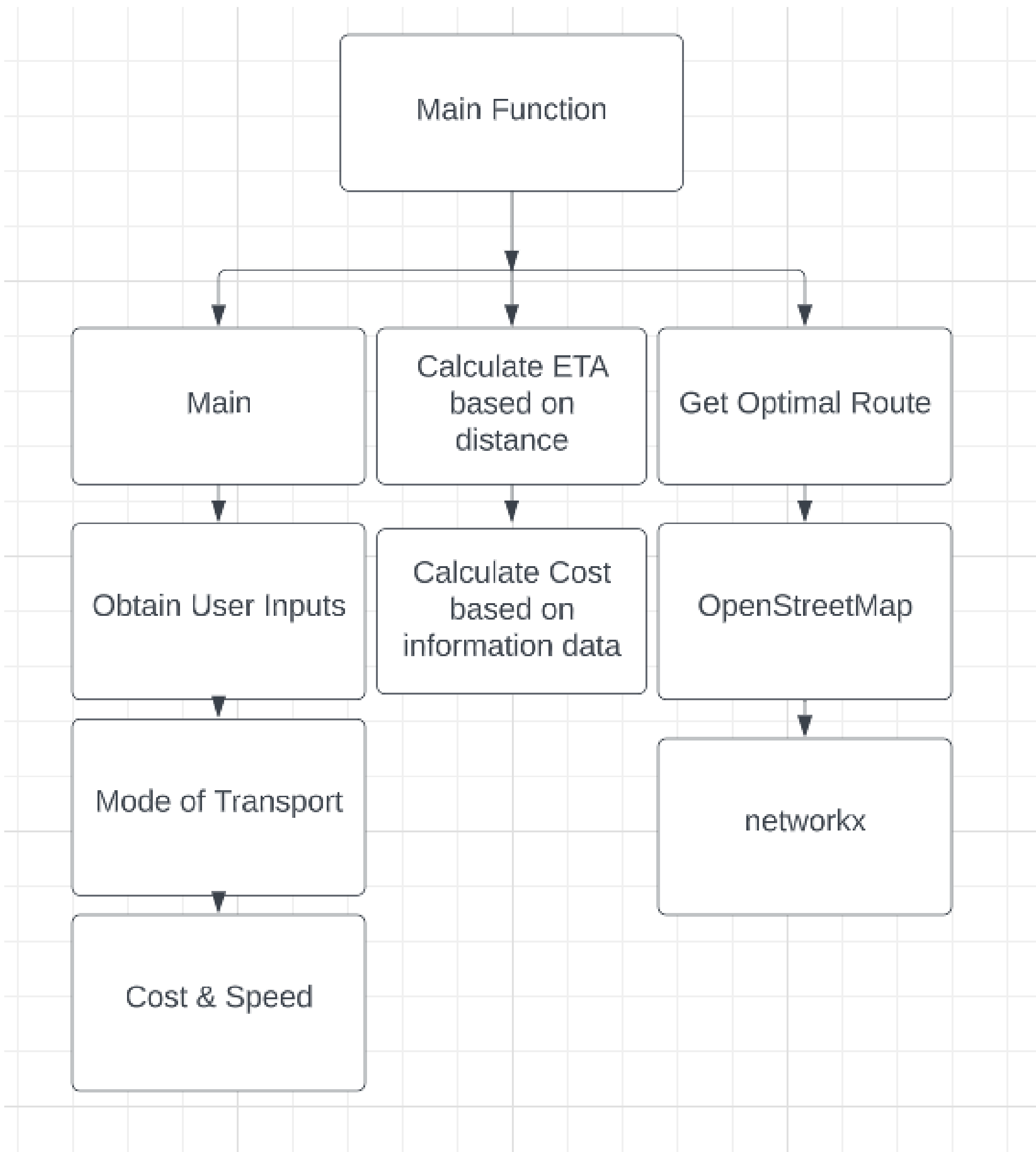
III. Methodology

The purpose of this methodology is to provide a framework for the development of the program, to outline the conceptual framework and to provide the complete process that was involved in the making of this program. This will go into the detail of the design of the program as well.

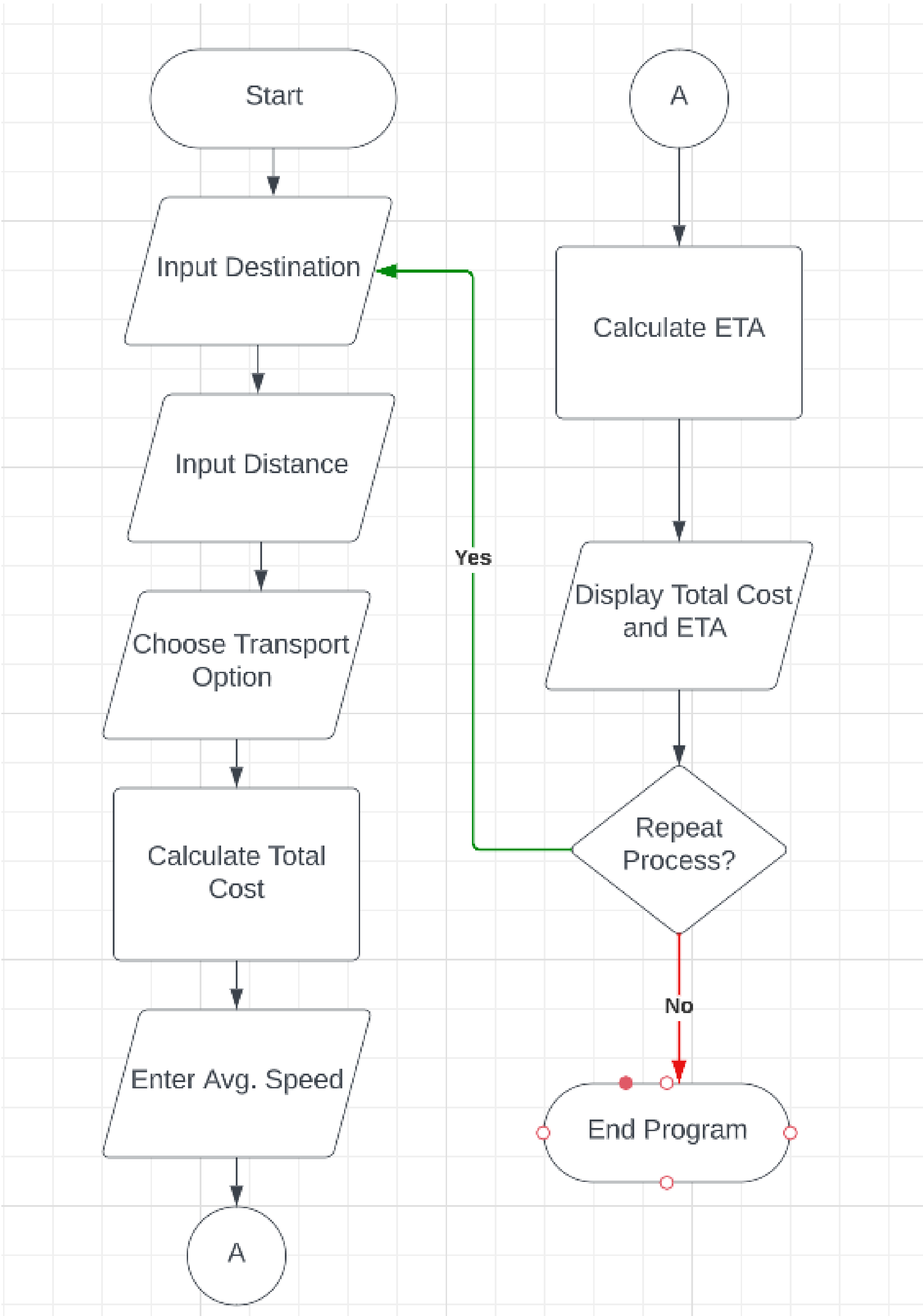
A. Conceptual Framework – IPO Chart (Input-Process-Output-Chart)

Input	Process	Output
Destination Distance Transportation of Choice Avg Speed	Calculate Total Cost (Find Avg Cost of Transportation) Calculate ETA (Distance/Avg Speed)	Total Cost ETA

B. Hierarchy Chart (Refer to the lecture on modules)



C. Flowchart



Start
Input Destination
Input Distance
Choose Transport Option
Calculate Total Cost
Enter Avg. Speed
A
A
Calculate ETA
Display Total Cost and ETA
Repeat Process?
Yes
No
End Program

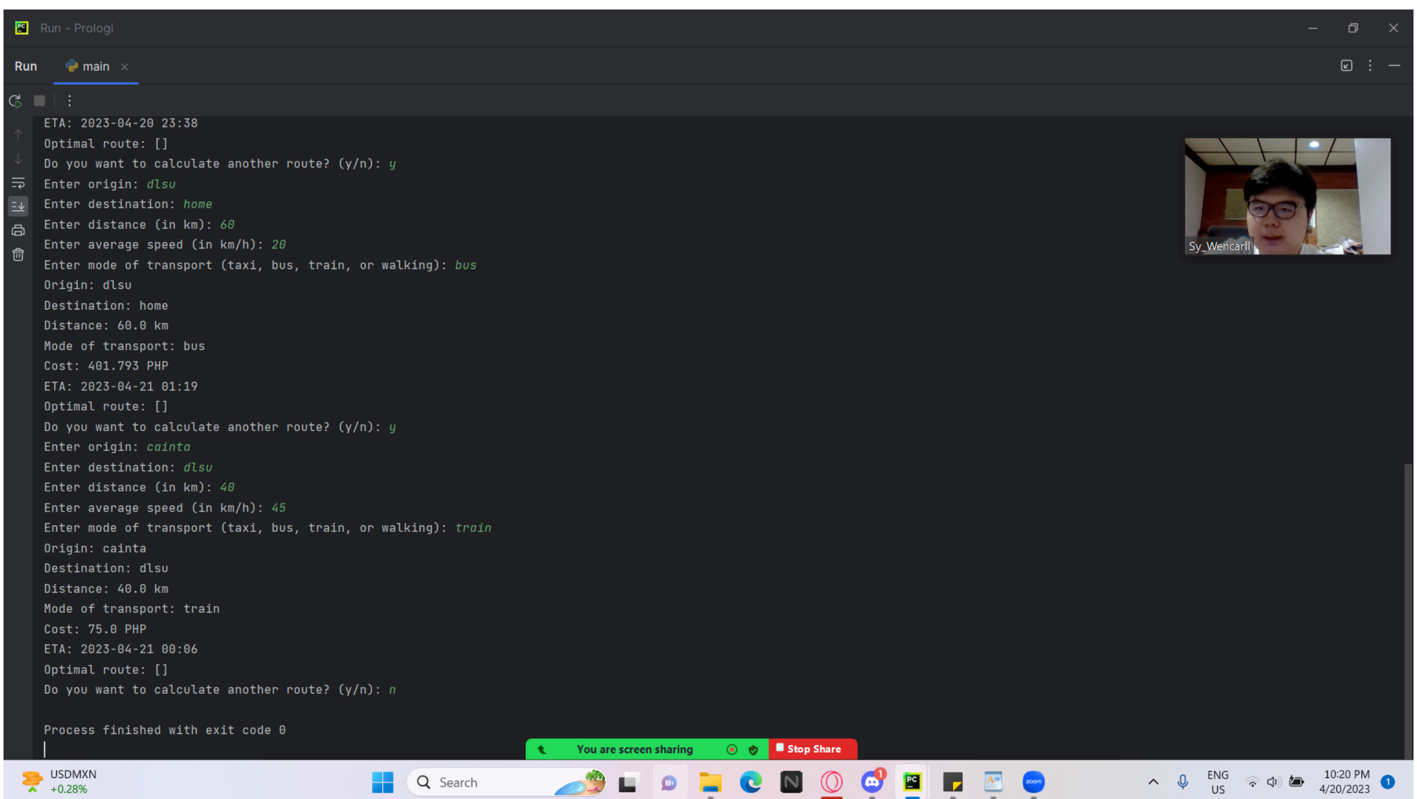
D. Pseudocode

- Import necessary libraries
- Define function calculate_eta with distance, speed, and rush_hour as parameters:
    - If rush_hour is True:
        - Multiply speed by 0.5
    - Calculate time_in_hours by dividing distance by speed
    - Calculate time_in_minutes by multiplying time_in_hours by 60
    - Calculate eta by adding time_in_minutes to the current time
    - Return eta in the format of "YYYY-MM-DD HH:MM"
- Define function calculate_cost with distance and cost_per_km as parameters:
    - Multiply distance by cost_per_km
    - Return the result
- Define function get_optimal_route with origin, destination, and mode_of_transport as parameters:
    - Retrieve OpenStreetMap data for the area around origin and destination
    - Merge the graphs for the origin and destination areas
    - If there are no nodes in the graph, raise a ValueError
    - Define source and target nodes as the first node in the graph
    - Calculate the shortest path between origin and destination using mode of transport
    - Convert optimal route to a list of distance and duration dictionaries
    - Return the list
- Define function handle_input_error with message as a parameter:
    - While True:
        - Print the message
        - Ask the user if they want to continue
        - If the user answers "y", return True
        - If the user answers "n", return False
        - If the user enters an invalid input, print an error message and repeat
- While True:
    - Get user inputs for origin, destination, distance, speed, and mode_of_transport
    - If any of the inputs are empty, print an error message and repeat
    - Convert distance and speed to float
    - If the input for distance or speed is not a number, print an error message and repeat
    - If the mode_of_transport is not one of the valid options, print an error message and repeat

- Set cost_per_km and rush_hour based on mode_of_transport
- Call calculate_cost and calculate_eta functions with appropriate parameters
- Call get_optimal_route function with appropriate parameters
- Print the results
- Ask the user if they want to continue
- If the user answers "n", break out of the loop.

IV. Results


Run - Prologi
Run
main
C:\Users\User\PycharmProjects\Prologi\venv\Scripts\python.exe C:\Users\User\PycharmProjects\Prologi\main.py
Enter origin: DLSU
Enter destination: Home
Enter distance (in km): 30
Enter average speed (in km/h): 40
Enter mode of transport (taxi, bus, train, or walking): taxi
Origin: DLSU
Destination: Home
Distance: 30.0 km
Mode of transport: taxi
Cost: 429.0 PHP
ETA: 2023-04-20 18:01
Optimal route: []
Do you want to calculate another route? (y/n): y
Enter origin: DLSu
Enter destination: home
Enter distance (in km): 30
Enter average speed (in km/h): 20
Enter mode of transport (taxi, bus, train, or walking): bus
Origin: DLSu
Destination: home
Distance: 30.0 km
Mode of transport: bus
Cost: 200.8965 PHP
ETA: 2023-04-20 18:02
Optimal route: []
Do you want to calculate another route? (y/n): n
Process finished with exit code 0


Run - Prologi
Run
main
ETA: 2023-04-20 23:38
Optimal route: []
Do you want to calculate another route? (y/n): y
Enter origin: dlsu
Enter destination: home
Enter distance (in km): 60
Enter average speed (in km/h): 20
Enter mode of transport (taxi, bus, train, or walking): bus
Origin: dlsu
Destination: home
Distance: 60.0 km
Mode of transport: bus
Cost: 401.793 PHP
ETA: 2023-04-21 01:19
Optimal route: []
Do you want to calculate another route? (y/n): y
Enter origin: cainta
Enter destination: dlsu
Enter distance (in km): 40
Enter average speed (in km/h): 45
Enter mode of transport (taxi, bus, train, or walking): train
Origin: cainta
Destination: dlsu
Distance: 40.0 km
Mode of transport: train
Cost: 75.0 PHP
ETA: 2023-04-21 00:06
Optimal route: []
Do you want to calculate another route? (y/n): n
Process finished with exit code 0
Sy_Wencarl
You are screen sharing
Stop Share
USDMXN
+0.28%
Search
ENG
US
10:20 PM
4/20/2023

V. Discussion of Results

It is a Python application that asks the user to enter information regarding a trip they want to take (such as the start and end points, the mode of transportation, the distance, and the average speed). The projected arrival time, the cost, and the most efficient route to take are among the many details regarding the voyage that are then calculated and displayed. Calculating these details makes use of a number of the program's functionalities. Utilizing the distance traveled and the speed of the vehicle, the calculate_eta() method determines the projected arrival time while accounting for rush hour traffic. The distance and cost per kilometer of the selected mode of transportation are inputted into the calculate_cost() function, which then determines the overall cost of the trip.

The get_optimal_route() function retrieves OpenStreetMap information for the region surrounding the origin and destination using the OSMnx library, combines the graphs for this region, and returns the results. The shortest path between the origin and destination is then determined based on the selected mode of transportation using the shortest_path() function from the NetworkX library. When a user enters incorrect information, such as an invalid mode of transportation or leaves a mandatory field empty, the handle_input_error() function is used to handle the error. The program then publishes the outcomes to the console, including the start and end points, the distance traveled, the form of transportation used, the associated cost, the anticipated time of arrival, and the best course of action. Additionally, it offers the user the choice to end the program or move on to another journey.

The paid api keys for a genuine good route system manager were restrictive, and the constant error of the optimal route system was when the use of longitude and latitude was attempted it kept on making it such that the routes never existed or "No path between." As a result, the output doesn't really output the optimal route.

VI. Analysis, Conclusion and Future Directives (Paraphrase)

The presented code is a Python script that asks for information from the user regarding the origin, destination, distance, average speed, and mode of transportation before calculating the cost, estimated time of arrival (ETA), and best route based on the data. The script extracts the road network surrounding the origin and destination addresses using the OSMnx library and OpenStreetMap data, merges those addresses, and then uses the NetworkX library to determine the shortest path between them. Travel time for taxis and duration for walking, taking the bus, and taking the train are determined by the mode of transportation, which also influences the weight parameter used in the shortest path algorithm.

The script uses a while loop and the handle_input_error function to handle input errors, asking the user whether to continue using the application or enter the proper inputs. The calculate_eta function accepts as inputs the distance, average speed, and rush hour and outputs the expected time of arrival as a formatted text. If set to True, rush hour slows down the pace by 50%. The calculate_cost function takes the distance and cost per kilometer of the mode of transportation as inputs and returns the trip cost as a float. The NetworkX library is used to get the road network and determine the shortest path between the inputs of the get_optimal_route function: the origin, destination, and mode of transportation. The best route is then transformed into a list of distance and time dictionaries by the function, which then returns it. The script outputs the origin, destination, distance, mode of transportation, cost, ETA, and best route using f-strings.

In conclusion, the script is a helpful tool for figuring out the price, estimated time of arrival (ETA), and best route for a given origin, destination, distance, average speed, and mode of transportation. However, it could be still improved by:

1. database for storing and retrieving user inputs and results
2. implementing a user authentication system
3. adding support for additional modes of transport such as cycling, driving, and ride-sharing
4. include real-time traffic updates and alternative routes for better accuracy and flexibility.
5. get an actual api key to better improve the optimal route process

References

1. Terzidis, K. (2020, September 23). Top Python concepts for Data Science. FreeCodeCamp. https://www.freecodecamp.org/news/top-python-concepts-for-data-science/#strings-in-python
2. Statista. (2021). Philippines: minimum fare by type of public transport (in Philippine pesos) as of 2021. https://www.statista.com/statistics/1329307/philippines-minimum-fare-by-type-of-public-transport/
3. Philippine News Agency. (2021, January 1). Transport groups raise minimum jeepney fare in Metro Manila. https://www.pna.gov.ph/articles/1183909
4. Light Rail Manila Corporation. (n.d.). Fare matrix. https://lrmc.ph/our-business-featured/fare-matrix/
5. Philippine Star. (2023, February 2). Fare hike for LRT-1 and LRT-2 looms. Retrieved from https://www.philstar.com/nation/2023/02/02/2241863/fare-hike-lrt-1-lrt-2-looms-

6. *"liberate the 88%!": Rep. Legarda echoes experts' calls to prioritize Metro Manila majority of non-car owners*. ICSC. (2020, June 11). Retrieved April 15, 2023, from https://icsc.ngo/liberate-the-88

7. Luna, F. (2022, November 28). *Metro Manila public transportation among the worst in the world: Study*. Philstar.com. Retrieved April 15, 2023, from https://www.philstar.com/headlines/2022/11/25/2226400/metro-manila-public-transportation-among-worst-world-study

8. Abad, M. (2019, October 10). *Fast facts: State of Metro Manila's public transport system*. RAPPLER. Retrieved April 15, 2023, from https://www.rappler.com/newsbreak/iq/242244-things-to-know-about-metro-manila-public-transport-system/

9. Manocha, N., & Miller, E. J. (2018). Evaluation of transit mobile applications: Case study of Transit App in Toronto. Transportation Research Record, 2672(26), 1-9. doi: 10.1177/0361198118803327

10. Casquero, D., Monzon, A., García, M. R., & Martínez, O. S. (2022). Key Elements of Mobility Apps for Improving Urban Travel Patterns: A Literature Review. *Future Transportation*, *2*(1), 1–23. https://doi.org/10.3390/futuretransp2010001

Appendices

A. User's Manual

The provided code is a Python script that allows users to calculate the optimal route, estimated time of arrival, and cost of a trip between two locations using various modes of transport. The user will be prompted to input the origin, destination, distance, average speed, and mode of transport.

Required Libraries

Before running this code, make sure the following libraries are installed in your system:

1. datetime: This library is used to work with dates and times.

2. osmnx: This library is used to retrieve OpenStreetMap data for the area around origin and destination.

3. networkx: This library is used to work with complex networks and graphs.

Functions

The code contains four functions that perform specific tasks:

1. calculate_eta(distance, speed, rush_hour): Using the specified distance, average speed, and rush hour status, this function estimates the time of arrival based on the three input parameters of distance (float), speed (float), and rush_hour (boolean). It produces a string of the format "YYYY-MM-DD HH:MM".

2. calculate_cost(distance, cost_per_km): Using the specified distance and cost per kilometer, this function determines the cost of the journey. It accepts two parameters: distance (float) and cost_per_km (float). It gives back the total cost as a float.

3. get_optimal_route(origin, destination, mode_of_transport): This function collects OpenStreetMap data for the region surrounding the origin and destination using three parameters: origin (string), destination (string), and mode_of_transport (string). A list of dictionaries representing the length and duration of each segment of the ideal route is then returned after merging the graphs for the origin and destination areas. This is followed by computing the shortest path between the origin and destination using the specified mode of transportation.

4. handle_input_error(message): When an input error occurs, this function's message (string) parameter, which is the only input required, prompts the user to continue or abort.

How to Use

To use this code, follow these steps:

1. Import the required libraries: datetime, osmnx, and networkx.

2. Copy and paste the code into a Python script file.

3. Run the script.

4. Input the following prompts:

    - Origin: Enter the initial location for the trip.

    - Destination: Enter the final location for the trip.

    - Distance (in km): Enter the distance between the two locations in kilometers.

    - Average speed (in km/h): Enter the average speed for the mode of transport in kilometers per hour.

- Mode of transport: Choose one of the four modes of transport: taxi, bus, train, or walking.

5. Review the output, which includes the following information:
    - Origin: The initial location for the trip.
    - Destination: The final location for the trip.
    - Distance: The distance between the two locations in kilometers.
    - Mode of transport: The selected mode of transport.
    - Cost: The total cost of the trip.
    - ETA: The estimated time of arrival in the format "YYYY-MM-DD HH:MM".
    - Optimal route: A list of dictionaries containing the distance and duration of each segment of the optimal route.
6. Choose whether to continue or quit when prompted.

B. Source Code – include comments in your source codes

```
import datetime
import osmnx as ox
import networkx as nx

# Function to calculate ETA
def calculate_eta(distance, speed, rush_hour):
    if rush_hour:
        speed *= 0.5  # Reduce speed by 50% during rush hour
    time_in_hours = distance / speed
    time_in_minutes = time_in_hours * 60
    eta = datetime.datetime.now() + datetime.timedelta(minutes=time_in_minutes)
    return eta.strftime("%Y-%m-%d %H:%M")

# Function to calculate cost
def calculate_cost(distance, cost_per_km):
    return distance * cost_per_km
```

```
# Function to get optimal route
def get_optimal_route(origin, destination, mode_of_transport):
    # Retrieve OpenStreetMap data for the area around origin and destination
    G = ox.graph_from_address(origin, network_type='all')
    H = ox.graph_from_address(destination, network_type='all')

    # Merge the graphs for the origin and destination areas
    G = nx.compose(G, H)

    # Check if there are any nodes in the graph
    if not G:
        raise ValueError("No nodes found in graph")

    # Define source and target nodes
    source = next(iter(G.nodes()))
    target = next(iter(G.nodes()))

    # Calculate the shortest path between origin and destination using mode of transport
    if mode_of_transport == 'taxi':
        optimal_route = nx.shortest_path(G, source=source, target=target, weight='travel_time')
    elif mode_of_transport == 'walking':
        optimal_route = nx.shortest_path(G, source=source, target=target, weight='length')
    elif mode_of_transport == 'bus':
        optimal_route = nx.shortest_path(G, source=source, target=target, weight='length')
    elif mode_of_transport == 'train':
        optimal_route = nx.shortest_path(G, source=source, target=target, weight='length')
    else:
```

```
        raise ValueError("Invalid mode of transport")

    # Convert optimal route to a list of distance and duration dictionaries
    optimal_route_data = []
    for u, v in zip(optimal_route[:-1], optimal_route[1:]):
        data = G.get_edge_data(u, v)[0]
        distance = data['length']
        duration = distance / (data['maxspeed_kph'] / 60)
        optimal_route_data.append({'distance': {'value': distance}, 'duration': {'value': duration}})

    return optimal_route_data

# Function to handle input errors
def handle_input_error(message):
    while True:
        print(message)
        answer = input("Do you want to continue? (y/n): ")
        if answer.lower() == "y":
            return True
        elif answer.lower() == "n":
            return False
        else:
            print("Invalid input. Please enter 'y' or 'n'.")

while True:
    # Get user inputs
    origin = input("Enter origin: ")
    while not origin:
```

```
    print("Origin cannot be empty")
    origin = input("Enter origin: ")
destination = input("Enter destination: ")
while not destination:
    print("Destination cannot be empty")
    destination = input("Enter destination: ")
distance = input("Enter distance (in km): ")
while not distance:
    print("Distance cannot be empty")
    distance = input("Enter distance (in km): ")
try:
    distance = float(distance)
except ValueError:
    print("Distance must be a number")
    continue
speed = input("Enter average speed (in km/h): ")
while not speed:
    print("Speed cannot be empty")
    speed = input("Enter average speed (in km/h): ")
try:
    speed = float(speed)
except ValueError:
    print("Speed must be a number")
    continue
mode_of_transport = input("Enter mode of transport (taxi, bus, train, or walking): ")
while mode_of_transport not in ['taxi', 'bus', 'train', 'walking']:
    print("Invalid mode of transport. Please choose from taxi, bus, train, or walking.")
    mode_of_transport = input("Enter mode of transport (taxi, bus, train, or walking): ")
```

```
current_time = datetime.datetime.now()

# Set cost and speed parameters based on mode of transport
if mode_of_transport == "taxi":
    cost_per_km = 14.3
    rush_hour = True
elif mode_of_transport == "bus":
    cost_per_km = 6.69655
    rush_hour = False
elif mode_of_transport == "train":
    cost_per_km = 1.875
    rush_hour = True
elif mode_of_transport == "walking":
    cost_per_km = 0
    rush_hour = False
    speed = 5  # Set walking speed to 5 km/h

# Calculate cost and ETA
cost = calculate_cost(distance, cost_per_km)
eta = calculate_eta(distance, speed, rush_hour)

# Get optimal route
try:
    optimal_route = get_optimal_route(origin, destination, mode_of_transport)
except Exception as e:
    print(str(e))
    continue
```

```
# Print results
print(f"Origin: {origin}")
print(f"Destination: {destination}")
print(f"Distance: {distance} km")
print(f"Mode of transport: {mode_of_transport}")
print(f"Cost: {cost} PHP")
print(f"ETA: {eta}")
print(f"Optimal route: {optimal_route}")

# Ask user if they want to continue
answer = input("Do you want to calculate another route? (y/n): ")
if answer.lower() != "y":
    break
```

C. **Work breakdown** – itemize the work done by each member of the group.

Student Name	Tasks Assigned	Percentage of the Work Contribution
Wencarll Cynric Sy	Code Project Proposal Appendices Results Discussion of Results Analysis, Conclusion and Future Directives Video Presentation	40%

Nathan Kyle Uy	Video Presentation Introduction Methodology Review of Related Literature Project Proposal Flowchart Poster	30%
Juan Paolo Sarmiento	Video Presentation Introduction Methodology Review of Related Literature Project Proposal	30%

D. Personal Data Sheet

Picture:

Name: Wencarll Cynric Sy

Course: Computer Engineering

Email: wencarll_sy@dlsu.edu.ph

Picture:

Name: Nathan Kyle Uy

Course: Computer Engineering

Email: nathan_uy@dlsu.edu.ph

Picture:

Name: Juan Paolo Sarmiento

Course: Computer Engineering

Email: juan_paolo_sarmiento@dlsu.edu.ph