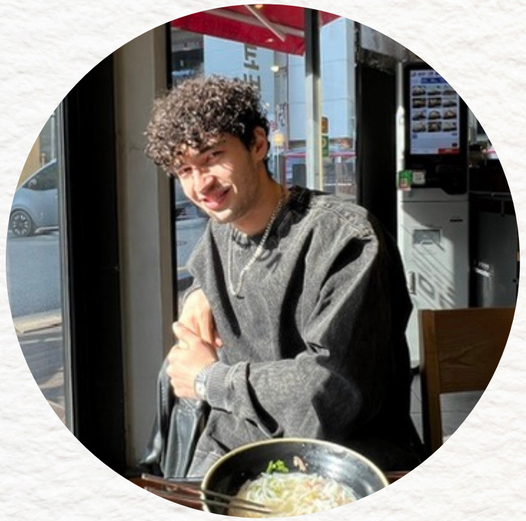
CODE WALKTHROUGH

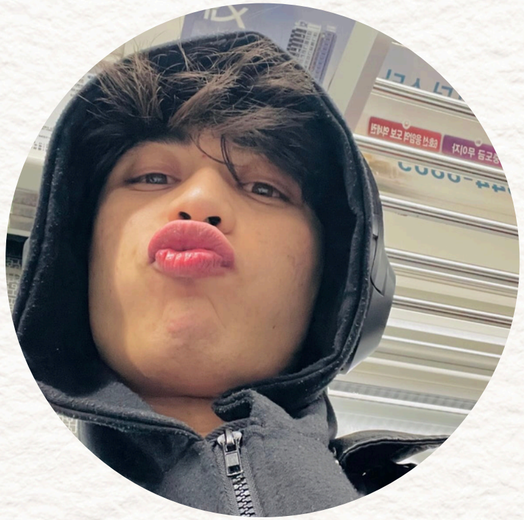
Presented By : Puzzles

- **Geocoding Function: Converts addresses to coordinates.**
- **Route Calculation: Retrieves route data from the GraphHopper API.**
- **User Input: How the program takes input and displays results.**

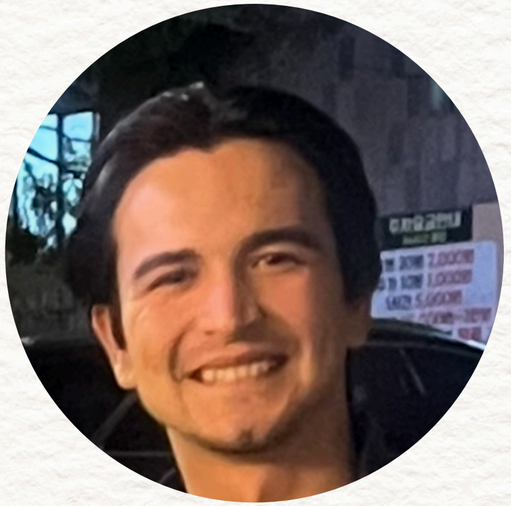
MEET THE GROUP



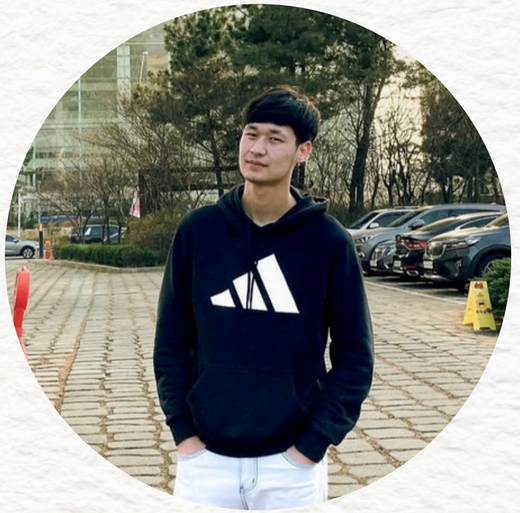
Adkham



Bekmurod



Jasurbek



Nozimjon

SAMPLE OUTPUT

```
Last login: Fri May  9 06:09:21 on ttys000
apple@APPLEs-MacBook-Pro ~ % cd ~/labs/devnet-src/graphhopper

[apple@APPLEs-MacBook-Pro graphhopper % python3 graphhopper_parse-json_7.py



Starting Location: Washington, d.c
Geocoding API URL for Washington, District of Columbia, United States (Location Type: city)
https://graphhopper.com/api/1/geocode?q=Washington%2C+d.c&limit=1&key=bfccea8a-5217-4539-a2e0-679412c9dd86
Destination: baltimore, maryland
Geocoding API URL for Baltimore, Maryland, United States (Location Type: city)
https://graphhopper.com/api/1/geocode?q=baltimore%2C+maryland&limit=1&key=bfccea8a-5217-4539-a2e0-679412c9dd86
=============================================
Routing API Status: 200
Routing API URL:
https://graphhopper.com/api/1/route?key=bfccea8a-5217-4539-a2e0-679412c9dd86&point=38.8950368%2C-77.0365427&point=39.2908816%2C-76.610759
=============================================
Directions from 38.8950368,-77.0365427 to 39.2908816,-76.610759
=============================================
Distance Traveled: 39.2 miles / 63.1 km
Trip Duration: 00:51:52
=============================================
Continue (0.07 km / 0.04 miles)
Turn sharp left onto South Drive (0.05 km / 0.03 miles)
Turn left onto West Executive Avenue Northwest (0.01 km / 0.01 miles)
Turn right onto State Place Northwest (0.12 km / 0.08 miles)
Turn right onto 17th Street Northwest (0.68 km / 0.42 miles)
Turn right onto K Street Northwest (1.07 km / 0.67 miles)
Turn left onto 11th Street Northwest (0.14 km / 0.08 miles)
Turn right onto L Street Northwest (0.06 km / 0.03 miles)
Turn slight right onto Massachusetts Avenue Northwest (5.23 km / 3.25 miles)
Continue onto New York Avenue Northeast (2.82 km / 1.75 miles)
Keep left onto Baltimore-Washington Parkway and drive toward Baltimore (51.63 km / 32.07 miles)
Turn right onto West Baltimore Street (0.62 km / 0.38 miles)
Turn left onto North Charles Street (0.18 km / 0.11 miles)
Turn right onto East Lexington Street (0.36 km / 0.22 miles)
Turn right onto Guilford Avenue (0.04 km / 0.03 miles)
Arrive at destination (0.00 km / 0.00 miles)
=============================================
Starting Location:
```

BENEFITS OF USING GRAPHHOPPER API:

- **Provides accurate and efficient routing solutions.**
- **Supports multiple transport types (driving, walking, biking).**
- **Reliable and scalable API with real-time data.**

FUTURE WORK

- Integrate additional features like traffic data.
- Add support for more transport types or transportation modes.
- Improve user interface (GUI) for easier interaction.

CONTRIBUTIONS:

ADKHAM 12214746 TEAM LEADER

- Led the project and coordinated the development process.
- Designed the architecture and implemented the entire codebase.
- Created the presentation and PPT file.

JASURBEK

- Participated in early team meetings and contributed ideas during initial planning stages.
- Provided feedback on presentation structure.

NOZIMJON

- Assisted in brainstorming transport mode features.
- Reviewed output samples and suggested small improvements.

BEKMUROD

- Supported with basic layout suggestions for the slides.
- Helped in checking the grammar and flow of the final PPT.

CHALLENGES

1. Using Two Different APIs Together:
One of the most complex parts was getting two separate GraphHopper services — geocoding and routing — to work smoothly together. We had to first convert the user's input into coordinates, then use those coordinates to calculate a route. This required careful handling of how data flowed between the two parts.

2. Dealing with API Responses:
The data we got back from the API wasn't always simple. We had to go deep into the response structure to get exactly what we needed — like location names, distances, and durations — while also making sure the program didn't crash if any values were missing or unexpected.

```python
import requests
import urllib.parse

route_url = "https://graphhopper.com/api/1/route?"
key = "bfccea8a-5217-4539-a2e0-679412c9dd86"

def geocoding(location, key):
    while not location:
        location = input("Enter the location again: ")

    geocode_url = "https://graphhopper.com/api/1/geocode?"
    url = geocode_url + urllib.parse.urlencode({"q": location, "limit": "1", "key": key})
    try:
        replydata = requests.get(url)
        json_status = replydata.status_code
        json_data = replydata.json()

        if json_status == 200 and json_data["hits"]:
            lat = json_data["hits"][0]["point"]["lat"]
            lng = json_data["hits"][0]["point"]["lng"]
            name = json_data["hits"][0]["name"]
            value = json_data["hits"][0]["osm_value"]
            country = json_data["hits"][0].get("country", "")
            state = json_data["hits"][0].get("state", "")

            if state and country:
                new_loc = f"{name}, {state}, {country}"
            elif state:
                new_loc = f"{name}, {state}"
            elif country:
                new_loc = f"{name}, {country}"
            else:
                new_loc = name

            print(f"Geocoding API URL for {new_loc} (Location Type: {value})\n{url}")
            return json_status, lat, lng, new_loc
        else:
            print(f"Geocode API status: {json_status}\nError message: {json_data.get('message', 'Unknown error')}")
            return json_status, None, None, location
    except requests.RequestException as e:
        print(f"Error making geocoding request: {e}")
        return None, None, None, location

def get_route(orig_lat, orig_lng, dest_lat, dest_lng):
    op = f"&point={orig_lat}%2C{orig_lng}"
    dp = f"&point={dest_lat}%2C{dest_lng}"
    paths_url = route_url + urllib.parse.urlencode({"key": key}) + op + dp
    try:
        response = requests.get(paths_url)
        paths_status = response.status_code
        paths_data = response.json()
```

```
        if paths_status == 200:
            print(f"Routing API Status: {paths_status}")
            print(f"Routing API URL:\n{paths_url}")
            print("================================================")
            print(f"Directions from {orig_lat},{orig_lng} to {dest_lat},{dest_lng}")
            print("================================================")

            distance = paths_data["paths"][0]["distance"]
            duration = paths_data["paths"][0]["time"] // 1000

            hr = duration // 3600
            min = (duration % 3600) // 60
            sec = duration % 60

            print(f"Distance Traveled: {distance / 1609.34:.1f} miles / {distance / 1000:.1f} km")
            print(f"Trip Duration: {hr:02d}:{min:02d}:{sec:02d}")
            print("================================================")

            for each in range(len(paths_data["paths"][0]["instructions"])):
                path = paths_data["paths"][0]["instructions"][each]["text"]
                distance = paths_data["paths"][0]["instructions"][each]["distance"]
                print(f"{path} ({distance / 1000:.2f} km / {distance / 1000 / 1.61:.2f} miles)")

            print("================================================")
        else:
            print(f"Routing API error with status: {paths_status}")
    except requests.RequestException as e:
        print(f"Error making routing request: {e}")

def main():
    while True:
        try:
            loc1 = input("Starting Location: ")
            if loc1.lower() in ["q", "quit"]:
                break
            orig = geocoding(loc1, key)

            if orig[0] == 200 and orig[1] and orig[2]:
                loc2 = input("Destination: ")
                if loc2.lower() in ["q", "quit"]:
                    break
                dest = geocoding(loc2, key)

                if dest[0] == 200 and dest[1] and dest[2]:
                    print("================================================")
                    get_route(orig[1], orig[2], dest[1], dest[2])
                else:
                    print(f"Invalid destination: {dest[3]}")
            else:
                print(f"Invalid starting location: {orig[3]}")

```

```python
def main():
    while True:
        try:
            loc1 = input("Starting Location: ")
            if loc1.lower() in ["q", "quit"]:
                break
            orig = geocoding(loc1, key)

            if orig[0] == 200 and orig[1] and orig[2]:
                loc2 = input("Destination: ")
                if loc2.lower() in ["q", "quit"]:
                    break
                dest = geocoding(loc2, key)

                if dest[0] == 200 and dest[1] and dest[2]:
                    print("=============================================")
                    get_route(orig[1], orig[2], dest[1], dest[2])
                else:
                    print(f"Invalid destination: {dest[3]}")
            else:
                print(f"Invalid starting location: {orig[3]}")

        except KeyboardInterrupt:
            print("\nProgram terminated.")
            break

if __name__ == "__main__":
    main()

```