In [51]:
```
import requests
import numpy as np
import config
import math
import json
from collections import Counter
```

In [52]:
```
def get_coordinates(place):
    # Google Maps Geocoding API endpoint
    geocoding_endpoint = "https://maps.googleapis.com/maps/api/geocode/js

    params = {
        "address": place,
        "key": config.GOOGLE_MAPS_API_KEY
    }

    response = requests.get(geocoding_endpoint, params=params)
    data = response.json()

    if data["status"] == "OK":
        # Extract latitude and longitude from the response
        location = data["results"][0]["geometry"]["location"]
        return location["lat"], location["lng"]
    else:
        return None



def get_place_name(latitude, longitude):
    # Define the Google Maps Geocoding API endpoint
    geocoding_url = "https://maps.googleapis.com/maps/api/geocode/json?"

    # Construct the request URL with latitude, longitude, and API key
    request_url = f"{geocoding_url}latlng={latitude},{longitude}&key={con

    try:
        # Send the HTTP GET request
        response = requests.get(request_url)
        # Parse the JSON response
        data = json.loads(response.text)

        # Check if the response contains any results
        if "results" in data and len(data["results"]) > 0:
            # Extract the name of the place from the first result
            place_name = data["results"][0]["formatted_address"]
            return place_name
        else:
            return "Unknown"
    except Exception as e:
        print("An error occurred:", e)
        return "Error"



def get_elevation(lat, lng):
    # Google Maps Elevation API endpoint
    elevation_endpoint = "https://maps.googleapis.com/maps/api/elevation/
```

```
    params = {
        "locations": f"{lat},{lng}",
        "key": config.GOOGLE_MAPS_API_KEY
    }

    response = requests.get(elevation_endpoint, params=params)
    data = response.json()

    if data["status"] == "OK":
        return data["results"][0]["elevation"]
    else:
        print("Failed to obtain elevation data. API response:")
        print(data)
        return None
```

In [53]:
```
def create_elevation_data(center_lat, center_lng):
    # Define the spatial extent around the center coordinate (1 km x 1 km
    # This will vary depending on your application and level of detail re
    extent = 0.008983  # Approximately 1 km in degrees

    # Generate latitude and longitude coordinates within the spatial exte
    num_points = 4 # Number of points in each dimension
    lats = np.linspace(center_lat - extent/2, center_lat + extent/2, num_
    lngs = np.linspace(center_lng - extent/2, center_lng + extent/2, num_
    grid_lats, grid_lngs = np.meshgrid(lats, lngs)

    # Query elevation data for each point in the grid
    elevation_data = np.zeros_like(grid_lats)  # Initialize elevation dat
    for i in range(num_points):
        for j in range(num_points):
            lat, lng = grid_lats[i, j], grid_lngs[i, j]
            elevation = get_elevation(lat, lng)  # Function to query elev
            elevation_data[i, j] = elevation

    return elevation_data
```

In [85]:
```
def calculate_slope(elevation_data):
    dx, dy = 1, 1  # Assuming unit distance between each point
    dz_dx = np.gradient(elevation_data, axis=0) / dx
    dz_dy = np.gradient(elevation_data, axis=1) / dy
    slope = np.arctan(np.sqrt(dz_dx**2 + dz_dy**2))
    return np.degrees(slope)
```

In [86]:
```
def get_downslope_direction(slope_matrix):
    # Calculate the gradient vector using numpy's gradient function
    grad_x, grad_y = np.gradient(slope_matrix)

    # Determine the direction of steepest descent (downward slope)
    downward_slope_directions = []
    for i in range(slope_matrix.shape[0]):
        for j in range(slope_matrix.shape[1]):
            # Calculate the angle of the gradient vector (use arctan2 to
            angle_rad = np.arctan2(grad_y[i, j], grad_x[i, j])
            angle_deg = np.degrees(angle_rad)
            downward_slope_directions.append(angle_deg)

    # Count occurrences of each direction
    direction_counts = Counter(downward_slope_directions)
```

```
    # Determine the most common direction (downslope direction)
    angle_deg = direction_counts.most_common(1)[0][0]

        # Ensure angle is positive
    if angle_deg < 0:
        angle_deg += 360

    # Determine the direction string based on the angle
    if 22.5 <= angle_deg < 67.5:
        direction = "northeast"
    elif 67.5 <= angle_deg < 112.5:
        direction = "east"
    elif 112.5 <= angle_deg < 157.5:
        direction = "southeast"
    elif 157.5 <= angle_deg < 202.5:
        direction = "south"
    elif 202.5 <= angle_deg < 247.5:
        direction = "southwest"
    elif 247.5 <= angle_deg < 292.5:
        direction = "west"
    elif 292.5 <= angle_deg < 337.5:
        direction = "northwest"
    else:
        direction = "north"
    print("Dominant direction of downward slope:", direction)

    return angle_deg
```

In [97]:

```
def generate_points(input_lat, input_lng, angle_deg,min_distance,max_dist
    # Convert angle to radians
    angle_rad = math.radians(angle_deg)

    # Generate points spaced from 10 km to 100 km
    distances = range(min_distance, max_distance, freq)
    map = {}

    for distance in distances:
        # Calculate coordinates of the point at the specified distance an
        lat_offset = math.sin(angle_rad) * distance / 111.32  # 1 degree
        lng_offset = math.cos(angle_rad) * distance / (111.32 * math.cos(
        point_lat = input_lat + lat_offset
        point_lng = input_lng + lng_offset
        map[(point_lat, point_lng)]=get_elevation(point_lat, point_lng)
    return map
```

In [98]:

```
def calculate_distance(coord1, coord2):
    # Calculate Euclidean distance between two coordinates
    return np.sqrt((coord1[0] - coord2[0])**2 + (coord1[1] - coord2[1])**

def is_peak(coord, elevation, coordinates_dict):
    # Check if the coordinate is a peak (higher than all its neighbors)
    for neighbor_coord, neighbor_elevation in coordinates_dict.items():
        if neighbor_elevation > elevation and calculate_distance(coord, n
            return False
    return True

def find_nearest_peak(input_coord, input_elevation, coordinates_dict):
    nearest_peak = None
    min_distance = float('inf')
```

```
    max_elevation =0

    for coord, elevation in coordinates_dict.items():
        # Calculate distance between input coordinate and current coordin
        distance = calculate_distance(input_coord, coord)
        max_elevation = max(elevation,max_elevation)

        # Check if current coordinate is higher than input elevation and
        if elevation > input_elevation and is_peak(coord, elevation, coor
            min_distance = distance
            nearest_peak = coord

    if max_elevation < input_elevation:
        print(f"Max Elevation at current range is{max_elevation}<{input_e
        print("Increase your Search space")
    if
    return nearest_peak
```

In [124…

```
def main():
    place = input("Enter place name or coordinates (latitude,longitude):
    lat, lng = get_coordinates(place)
    original_coordinates = [lat, lng]
    elevation = get_elevation(lat, lng)
    print(f"\nElevation at {place}: \033[1m{elevation} meters\033[0m")

    # Create elevation data around the specified coordinates
    elevation_data = create_elevation_data(lat, lng)
    print("\nElevation data matrix around", place, ":")
    print(np.array(elevation_data), "in meters")

    # Calculate slope from the elevation data
    slope = calculate_slope(elevation_data)
    print("\nSlope matrix around", place, ":")
    print(np.array(slope), "in degrees\n")

    # Determine the dominant direction of downward slope
    downslope_direction = get_downslope_direction(slope)

    # Define parameters for peak search
    min_distance = 2  # Minimum distance in kilometers
    max_distance = 10  # Maximum distance in kilometers
    freq = 10 # repeition of points in a direction
    input_elevation = elevation  # Elevation at input coordinates

    # Generate points in the direction of line of sight
    Map = generate_points(lat, lng, downslope_direction, min_distance, ma
    print("\nMapping of coordinates with heights in search space:")
    print(Map)

    # Search for peaks along the dominant downslope direction
    peaks = find_nearest_peak((lat,lng), elevation, Map)
    print(f"\nNearest peak coordinate:\033[1m{peaks}\033[0m")
    ans = get_place_name(peaks[0],peaks[1])
    elev= get_elevation(peaks[0],peaks[1])
    print(f"Peak location \033[1m{ans}\033[0m with elevation \033[1m{elev

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

Elevation at Udhampur: **713.9642944335938 meters**

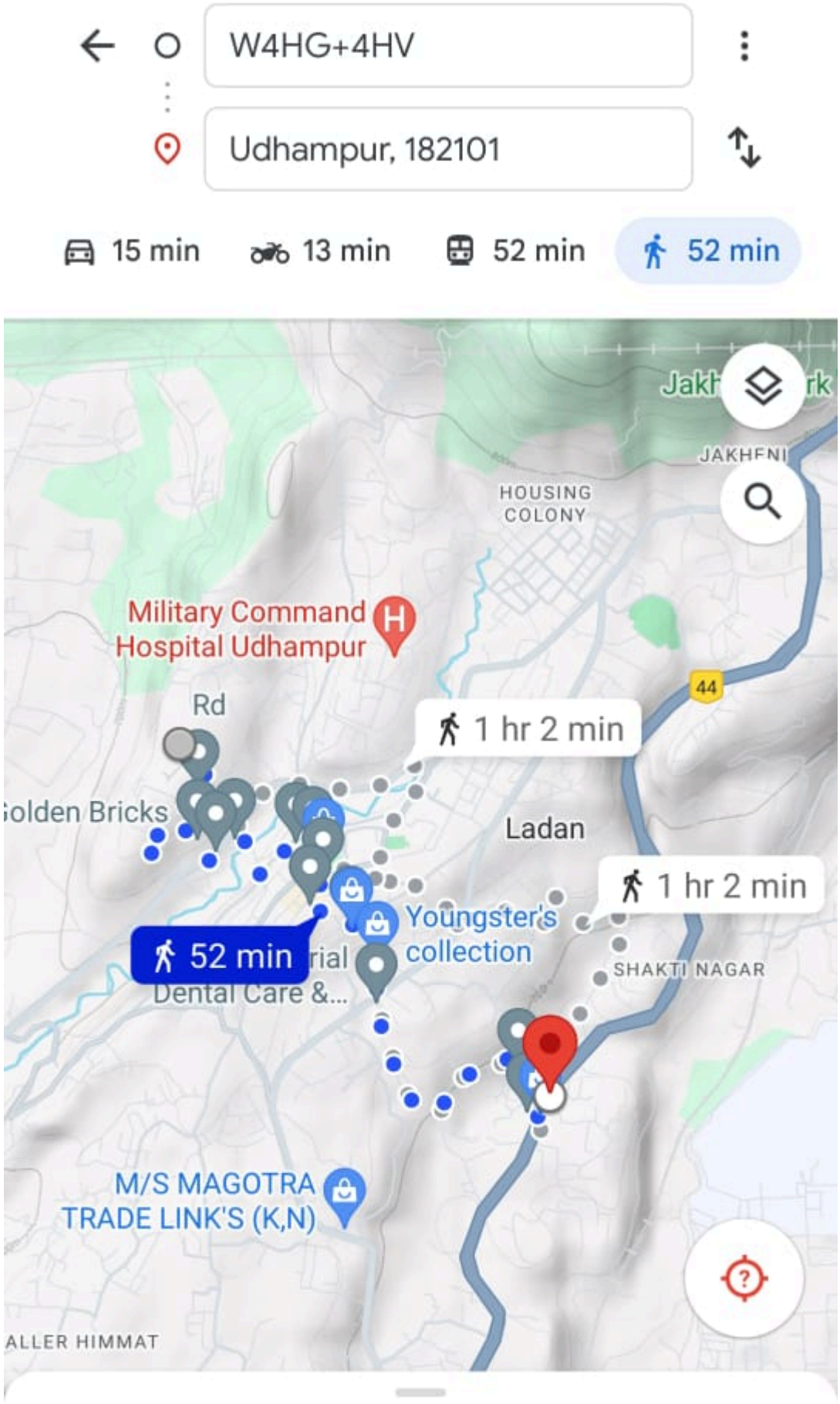
Elevation data matrix around Udhampur :
[[712.74383545 722.92858887 755.27203369 768.58721924]
 [716.21557617 720.27825928 727.79589844 755.86767578]
 [701.29101562 721.53857422 700.87359619 716.640625]
 [672.00073242 681.59759521 687.25891113 725.60791016]] in meters

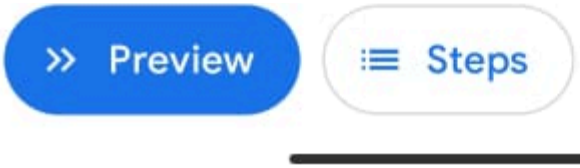
Slope matrix around Udhampur :
[[84.69046981 87.32814263 88.39651754 86.89154527]
 [81.89411227 80.2697878 88.23777499 88.50219446]
 [88.0894701 87.04030546 87.19581301 87.37984281]
 [88.14175129 88.59124653 87.78688902 88.54549467]] in degrees

Dominant direction of downward slope: southeast

Mapping of coordinates with heights in search space:
{(32.9283124995673, 75.12604857523549): 781.6180419921875}

Nearest peak coordinate:**(32.9283124995673, 75.12604857523549)**
Peak location **W4HG+4HV, Leray, 182101** with elevation **781.6180419921875 meters**

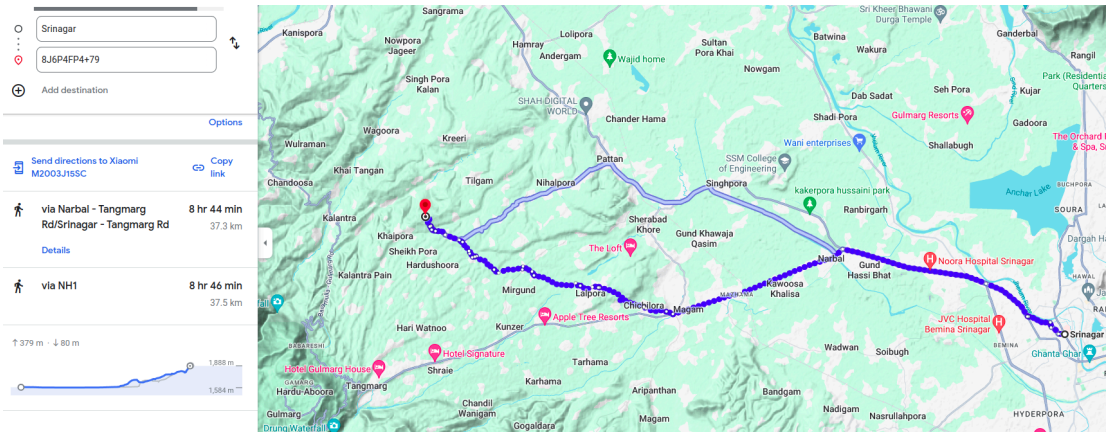

W4HG+4HV
Udhampur, 182101
15 min
13 min
52 min
52 min
HOUSING COLONY
Military Command Hospital Udhampur
Rd
1 hr 2 min
Golden Bricks
Ladan
1 hr 2 min
Youngster's collection
52 min
Dental Care &...
SHAKTI NAGAR
M/S MAGOTRA TRADE LINK'S (K,N)
52 min (3.8 km) Mostly flat
via Devika Ln



In [132…

```python
def main():
    place = input("Enter place name or coordinates (latitude,longitude):
    lat, lng = get_coordinates(place)
    original_coordinates = [lat, lng]
    elevation = get_elevation(lat, lng)
    print(f"\nElevation at {place}: \033[1m{elevation} meters\033[0m")

    # Create elevation data around the specified coordinates
    elevation_data = create_elevation_data(lat, lng)
    print("\nElevation data matrix around", place, ":")
    print(np.array(elevation_data), "in meters")

    # Calculate slope from the elevation data
    slope = calculate_slope(elevation_data)
    print("\nSlope matrix around", place, ":")
    print(np.array(slope), "in degrees\n")

    # Determine the dominant direction of downward slope
    downslope_direction = get_downslope_direction(slope)

    # Define parameters for peak search
    min_distance = 2  # Minimum distance in kilometers
    max_distance = 40  # Maximum distance in kilometers
    freq = 10 # repeition of points in a direction
    input_elevation = elevation  # Elevation at input coordinates

    # Generate points in the direction of line of sight
    Map = generate_points(lat, lng, downslope_direction, min_distance, ma
    print("\nMapping of coordinates with heights in search space:")
    print(Map)

    # Search for peaks along the dominant downslope direction
    peaks = find_nearest_peak((lat,lng), elevation, Map)
    print(f"\nNearest peak coordinate:\033[1m{peaks}\033[0m")
    ans = get_place_name(peaks[0],peaks[1])
    elev= get_elevation(peaks[0],peaks[1])
    print(f"Peak location \033[1m{ans}\033[0m with elevation \033[1m{elev

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

```
Elevation at Srinagar: 1607.78466796875 meters

Elevation data matrix around Srinagar :
[[1605.1072998  1602.27941895 1607.91394043 1608.66162109]
 [1607.13781738 1607.26135254 1607.72583008 1606.50744629]
 [1607.18676758 1605.54382324 1604.8092041  1604.23278809]
 [1607.09265137 1606.55200195 1608.13012695 1605.72399902]] in meters

Slope matrix around Srinagar :
[[73.97362986 79.06481362 72.62883482 66.32013259]
 [46.31659016 58.91148139 57.95400416 68.41363634]
 [58.6750613  51.12821215 34.44930707 34.87355961]
 [28.7571059  48.58815513 73.36342623 70.54357455]] in degrees

Dominant direction of downward slope: south

Mapping of coordinates with heights in search space:
{(34.08692342094681, 74.77594838567589): 1604.158935546875, (34.10318652
568086, 74.66927781405538): 1599.3740234375, (34.119449630414906, 74.562
60724243485): 1661.689819335938, (34.13571273514896, 74.45593667081434):
1892.469482421875}

Nearest peak coordinate:(34.13571273514896, 74.45593667081434)
Peak location 8J6P4FP4+79 with elevation 1892.469482421875 meters
```

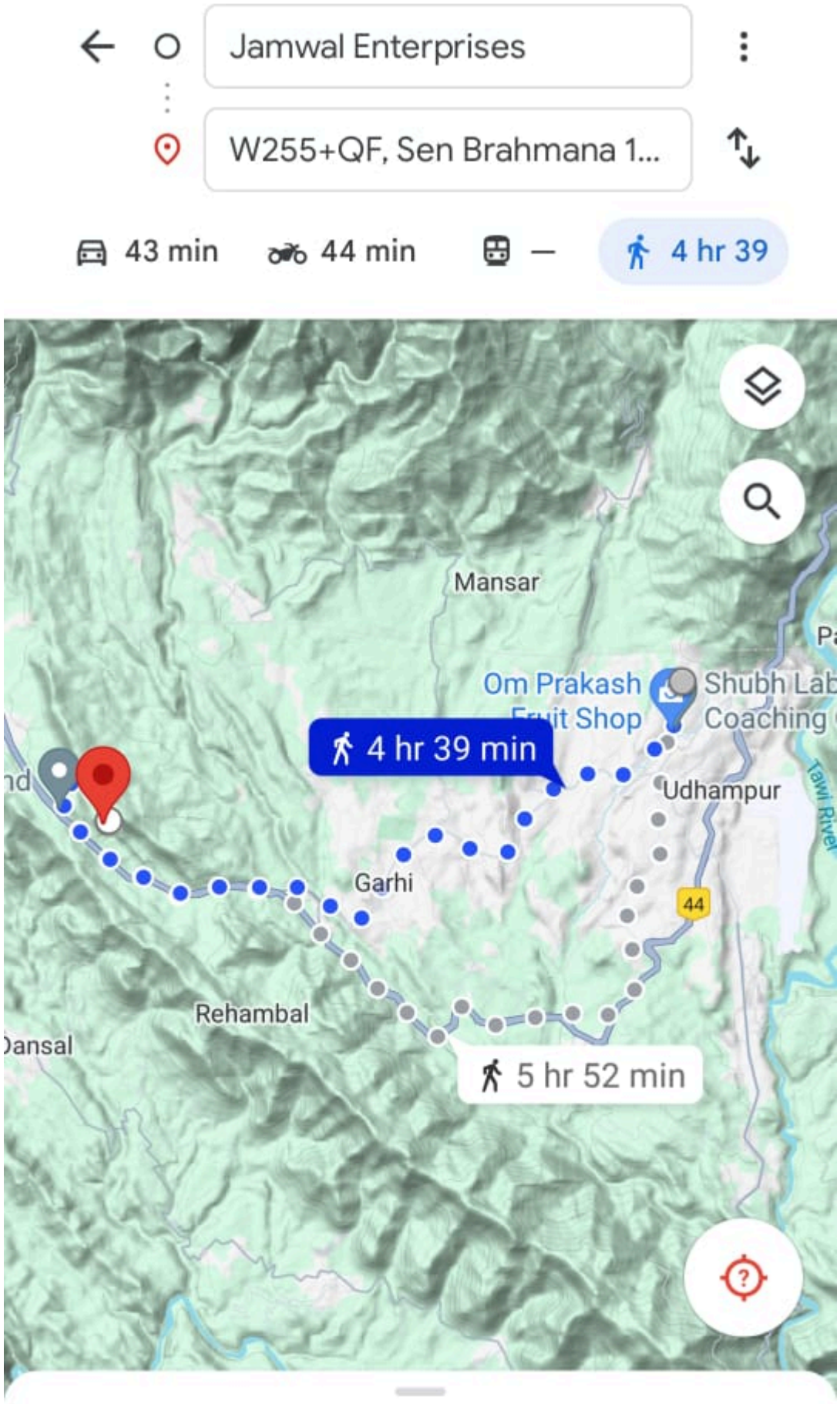


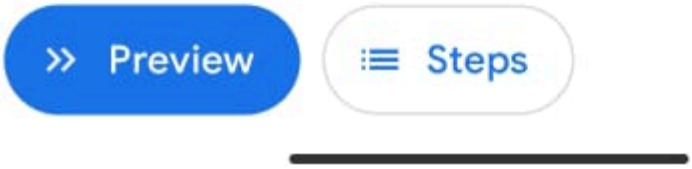
In [121…

```python
def main():
    place = input("Enter place name or coordinates (latitude,longitude):
    lat, lng = get_coordinates(place)
    original_coordinates = [lat, lng]
    elevation = get_elevation(lat, lng)
    print(f"\nElevation at {place}: \033[1m{elevation} meters\033[0m")

    # Create elevation data around the specified coordinates
    elevation_data = create_elevation_data(lat, lng)
    print("\nElevation data matrix around", place, ":")
    print(np.array(elevation_data), "in meters")

    # Calculate slope from the elevation data
    slope = calculate_slope(elevation_data)
    print("\nSlope matrix around", place, ":")
    print(np.array(slope), "in degrees\n")

    # Determine the dominant direction of downward slope
    downslope_direction = get_downslope_direction(slope)

    # Define parameters for peak search
```

```
    min_distance = 2  # Minimum distance in kilometers
    max_distance = 50  # Maximum distance in kilometers
    freq = 10 # repeition of points in a direction
    input_elevation = elevation  # Elevation at input coordinates

    # Generate points in the direction of line of sight
    Map = generate_points(lat, lng, downslope_direction, min_distance, ma
    print("\nMapping of coordinates with heights in search space:")
    print(Map)

    # Search for peaks along the dominant downslope direction
    peaks = find_nearest_peak((lat,lng), elevation, Map)
    print(f"\nNearest peak coordinate:\033[1m{peaks}\033[0m")
    ans = get_place_name(peaks[0],peaks[1])
    elev= get_elevation(peaks[0],peaks[1])
    print(f"Peak location \033[1m{ans}\033[0m with elevation \033[1m{elev

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

```
Elevation at 32.935720803983344, 75.13342656158873: 790.513916015625 met
ers

Elevation data matrix around 32.935720803983344, 75.13342656158873 :
[[788.30328369 815.7833252  790.02032471 767.47454834]
 [772.86322021 789.6619873  815.16412354 809.98724365]
 [794.68609619 801.27856445 808.2822876  839.24743652]
 [777.02825928 792.02575684 809.4979248  842.83062744]] in meters

Slope matrix around 32.935720803983344, 75.13342656158873 :
[[88.18288459 87.80880527 88.35714139 88.80951347]
 [86.65303089 87.43920407 85.81369958 88.42017263]
 [81.76964955 81.75376091 87.017712   88.36576291]
 [87.52840352 86.93675843 87.74821525 88.29144709]] in degrees

Dominant direction of downward slope: south

Mapping of coordinates with heights in search space:
{(32.93079382589567, 75.11260558549732): 715.2036743164062, (32.90945695
5374026, 75.0086360129839): 844.9146118164062, (32.88812008485238, 74.90
466644047049): 501.9024658203125}

Nearest peak coordinate:(32.909456955374026, 75.0086360129839)
Peak location W255+QF Sen Brahmana with elevation 844.9146118164062 mete
rs
```


Jamwal Enterprises
W255+QF, Sen Brahmana 1...
43 min
44 min
—
4 hr 39
Mansar
Om Prakash
Fruit Shop
Shubh Lab
Coaching
4 hr 39 min
Udhampur
Tawi River
Garhi
44
Rehambal
5 hr 52 min
4 hr 39 min (20 km)
341 m
219 m
via NH 244A/NH 44



In [123…

```
def main():
    place = input("Enter place name or coordinates (latitude,longitude):
    lat, lng = get_coordinates(place)
    original_coordinates = [lat, lng]
    elevation = get_elevation(lat, lng)
    print(f"\nElevation at {place}: \033[1m{elevation} meters\033[0m")

    # Create elevation data around the specified coordinates
    elevation_data = create_elevation_data(lat, lng)
    print("\nElevation data matrix around", place, ":")
    print(np.array(elevation_data), "in meters")

    # Calculate slope from the elevation data
    slope = calculate_slope(elevation_data)
    print("\nSlope matrix around", place, ":")
    print(np.array(slope), "in degrees\n")

    # Determine the dominant direction of downward slope
    downslope_direction = get_downslope_direction(slope)

    # Define parameters for peak search
    min_distance = 2  # Minimum distance in kilometers
    max_distance = 20  # Maximum distance in kilometers
    freq = 10 # repeition of points in a direction
    input_elevation = elevation  # Elevation at input coordinates

    # Generate points in the direction of line of sight
    Map = generate_points(lat, lng, downslope_direction, min_distance, ma
    print("\nMapping of coordinates with heights in search space:")
    print(Map)

    # Search for peaks along the dominant downslope direction
    peaks = find_nearest_peak((lat,lng), elevation, Map)
    print(f"\nNearest peak coordinate:\033[1m{peaks}\033[0m")
    ans = get_place_name(peaks[0],peaks[1])
    elev= get_elevation(peaks[0],peaks[1])
    print(f"Peak location \033[1m{ans}\033[0m with elevation \033[1m{elev

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

Elevation at chandigarh: **371.1080322265625 meters**

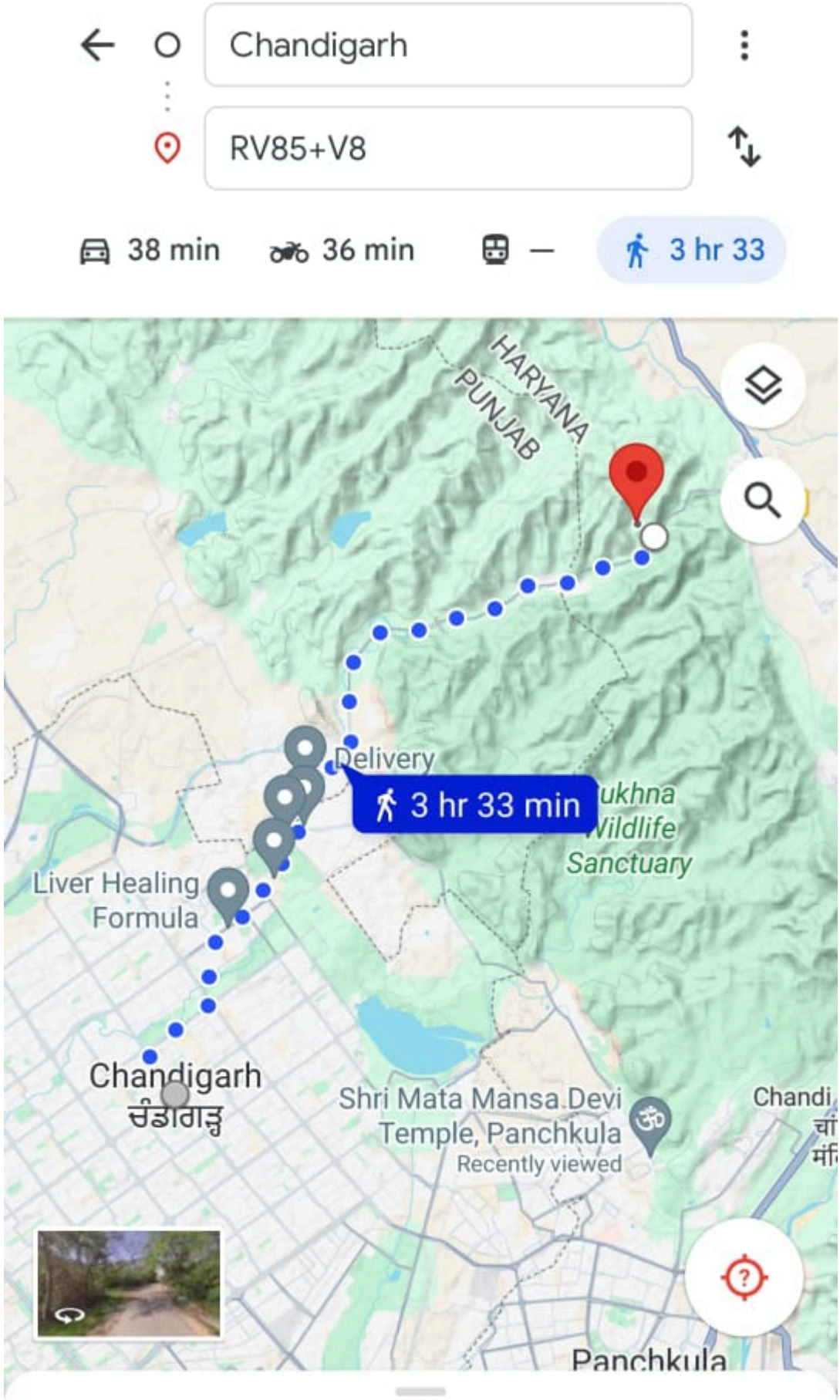
Elevation data matrix around chandigarh :
[[359.46511841 360.64627075 365.91998291 363.3951416]
 [361.68478394 365.91973877 361.94424438 367.85446167]
 [364.31170654 368.58984375 368.60049438 368.54388428]
 [363.33666992 364.26327515 366.64776611 373.62380981]] in meters

Slope matrix around chandigarh :
[[68.31156257 80.81245268 76.62779058 78.95798008]
 [78.41767629 75.87526294 58.82615373 81.1824612]
 [77.07398308 66.49032375 66.9651422 70.88404783]
 [53.37144006 77.81868773 78.84505743 83.39007025]] in degrees

Dominant direction of downward slope: northeast

Mapping of coordinates with heights in search space:
{(30.74728641522902, 76.79255852594612): 377.4261169433594, (30.81714449
1374126, 76.85826165567669): 484.8822937011719}

Nearest peak coordinate:**(30.817144491374126, 76.85826165567669)**
Peak location **RV85+V8 Basawal, Haryana, India** with elevation **484.8822937
011719 meters**


Chandigarh
RV85+V8
38 min
36 min
3 hr 33
HARYANA
PUNJAB
Delivery
3 hr 33 min
Liver Healing Formula
Chandigarh
ਚੰਡੀਗੜ੍ਹ
Shri Mata Mansa Devi Temple, Panchkula
Recently viewed
Panchkula
3 hr 33 min (15 km)
126 m
4 m
via Baddi Rd