```python
import os
import pandas as pd
import requests
from flask import Flask, request, jsonify

# Initialize Flask App
app = Flask(__name__)

# Constants
MAX_RANGE = 500  # Maximum vehicle range in miles
MPG = 10  # Miles per gallon

# Load fuel prices CSV
FUEL_PRICES_PATH = 'fuel-prices-for-be-assessment.csv'
fuel_prices = pd.read_csv(FUEL_PRICES_PATH)

# Fetch Route from OpenRouteService
def fetch_route(start, finish):
    """
    Fetches route details between start and finish locations using OpenRouteSer
    """
    try:
        routing_api_url = "https://api.openrouteservice.org/v2/directions/drivi
        api_key = "YOUR_API_KEY"  # Replace with your OpenRouteService API key

        # Use geocoding API to get coordinates for start and finish
        geocode_url = "https://api.openrouteservice.org/geocode/search"
        start_coords = requests.get(geocode_url, params={"api_key": api_key, "t
        finish_coords = requests.get(geocode_url, params={"api_key": api_key, "

        # Fetch route details
        route_response = requests.post(
            routing_api_url,
            json={
                "coordinates": [start_coords, finish_coords],
                "instructions": False,
            },
            headers={"Authorization": api_key}
        )
        route_data = route_response.json()
        return route_data
    except Exception as e:
        raise Exception(f"Error fetching route: {e}")

# Calculate Fuel Stops and Costs
def calculate_fuel_stops(route_distance, fuel_prices):
    """
    Calculates fuel stops and total cost.
    """
    current_distance = 0
    stops = []
    total_cost = 0

    # Sort the fuel prices by cost (cheapest first)
    fuel_prices = fuel_prices.sort_values(by='price')

    while current_distance < route_distance:
        # Find next stop within MAX_RANGE
        stop = fuel_prices.iloc[0]  # Assuming the cheapest station
        fuel_cost = (MAX_RANGE / MPG) * stop['price']
        stops.append({
            "station": stop['station_name'],
            "location": stop['location'],
            "cost": round(fuel_cost, 2)
        })
        total_cost += fuel_cost
        current_distance += MAX_RANGE

    return stops, round(total_cost, 2)

# API Endpoint
@app.route('/route', methods=['POST'])
def get_route():
    """
    API to calculate route, fuel stops, and total fuel cost.
    """
    try:
        data = request.get_json()
        start = data.get('start')
        finish = data.get('finish')

        if not start or not finish:
            return jsonify({"error": "Start and finish locations are required."

        # Step 1: Fetch route
        route_data = fetch_route(start, finish)
        route_distance = route_data['features'][0]['properties']['segments'][0]

        # Step 2: Calculate stops and costs
        fuel_stops, total_cost = calculate_fuel_stops(route_distance, fuel_pric

        # Step 3: Format response
        result = {
            "route": route_data['features'][0]['geometry']['coordinates'],
            "fuel_stops": fuel_stops,
            "total_cost": total_cost
        }
        return jsonify(result), 200
```