```
import requests

from typing import List, Dict, Any




def fetch_flights_in_area(

    latitude: float, longitude: float, radius: float = 0.5

) -> List[Dict[str, Any]]:

    """

    Fetch and summarize flight data for a given area using the OpenSky Network API.


    Args:

        latitude (float): The latitude of the center point.

        longitude (float): The longitude of the center point.

        radius (float): The radius around the center point to search for flights, in degrees. Default is 0.5.


    Returns:

        List[Dict[str, Any]]: A list of summarized flight data in the specified area.


    Raises:

        Exception: If the request fails or the response is invalid.

    """

    url = "https://opensky-network.org/api/states/all"

    params = {

        "lamin": latitude - radius,

        "lamax": latitude + radius,

        "lomin": longitude - radius,
    "lomax": longitude + radius,
}

try:
    response = requests.get(url, params=params)
    response.raise_for_status()
    data = response.json()
    flights = data.get("states", [])

    summarized_flights = []
    for flight in flights:
        if (
            flight[1]
            and flight[5]
            and flight[6]
            and flight[7] is not None
        ):  # Ensure essential data is available
            summarized_flights.append(
                {
                    "callsign": flight[1].strip(),
                    "origin_country": flight[2],
                    "last_position": f"Lat: {flight[5]}, Lon: {flight[6]}",
                    "altitude_meters": flight[7],
                }
            )
        return summarized_flights
    except requests.RequestException as e:
        raise Exception(f"Failed to fetch flight data: {e}")
    except ValueError:
        raise Exception("Invalid response format.")


# Example usage
latitude = 28.3922  # Latitude for Cape Canaveral, FL
longitude = -80.6077  # Longitude for Cape Canaveral, FL
radius = 0.5  # 0.5 degrees (~55 km)

try:
    flights = fetch_flights_in_area(latitude, longitude, radius)
    for flight in flights:
        print(
            f"Callsign: {flight['callsign']}, Origin: {flight['origin_country']}, "
            f"Position: {flight['last_position']}, Altitude: {flight['altitude_meters']} meters"
        )
except Exception as e:
    print(e)
```