Conectivity advisor

Semer Aissami

Project Proposal: NYC Connectivity Advisor

- Many New York City residents and visitors need quick access to the internet, but connectivity quality varies widely across neighborhoods, so the goal is to create a **live connectivity map of New York City** that helps users identify the best areas for fast and reliable wireless internet. The idea is based on analyzing **LinkNYC kiosk data**, but the program is designed to scale and work with **any wireless connectivity dataset** in the future.

Problem Statement

- Why users need to know neighborhood connectivity?
- Why LinkNYC data matters?
- What motivates the tool?

Project Description

- The program will store and utilize a dataset containing:
- NYC neighborhoods
- Wireless connectivity quality
- Estimated internet speed (Mbps)
- Number of public LinkNYC kiosks or access points

How are we going to use c++?

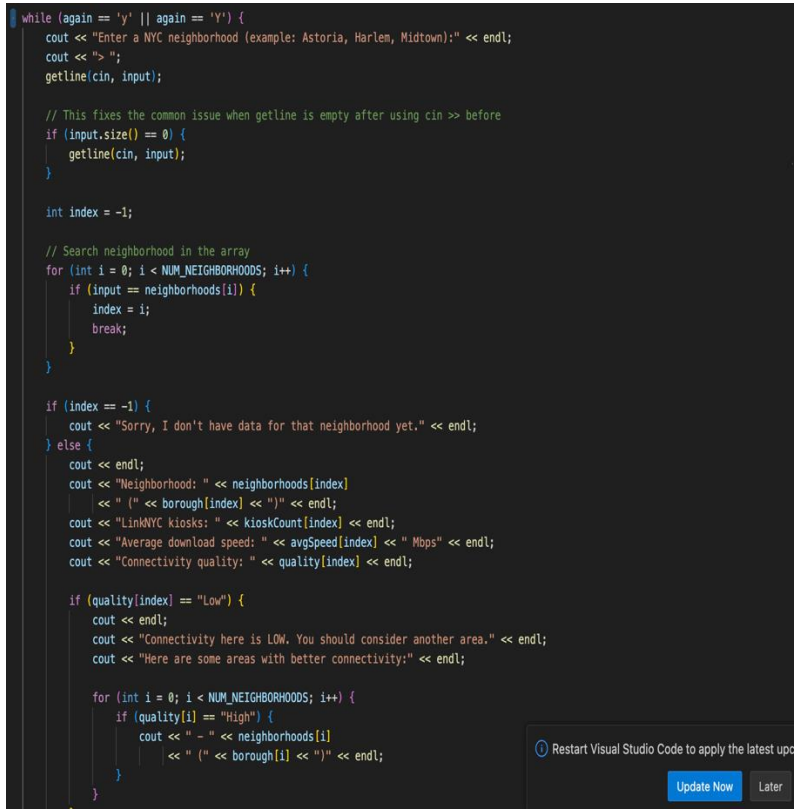
- This program uses beginner-friendly C++ tools to analyze NYC wireless connectivity by storing neighborhood data in parallel arrays for names, boroughs, kiosk counts, speeds, and quality ratings. It reads user input with getline() and uses a sentinel-controlled while loop that repeats as long as the user enters “y” or “Y,” allowing multiple searches. A simple linear search algorithm scans the arrays to find the neighborhood the user entered, and if/else statements determine what information to display—showing connectivity speed and quality or recommending a better area if the result is low. Through arrays, loops, conditional logic, and formatted output, the program creates an interactive C++ tool that helps users identify the best connectivity areas in NYC.

Components:

- getline() → reads full neighborhood names, including spaces.
- Parallel arrays → store neighborhoods, boroughs, speeds, kiosks, and quality.
- While loop (sentinel loop) → repeats the program while the user enters y/Y.
- Linear search → checks each array item to find the matching neighborhood.
- If/else statements → decide whether connectivity is high, medium, or low and what message to show.
- Formatted output → displays kiosk count, speed, quality, and recommends better areas if needed.

Sample Code (not final)

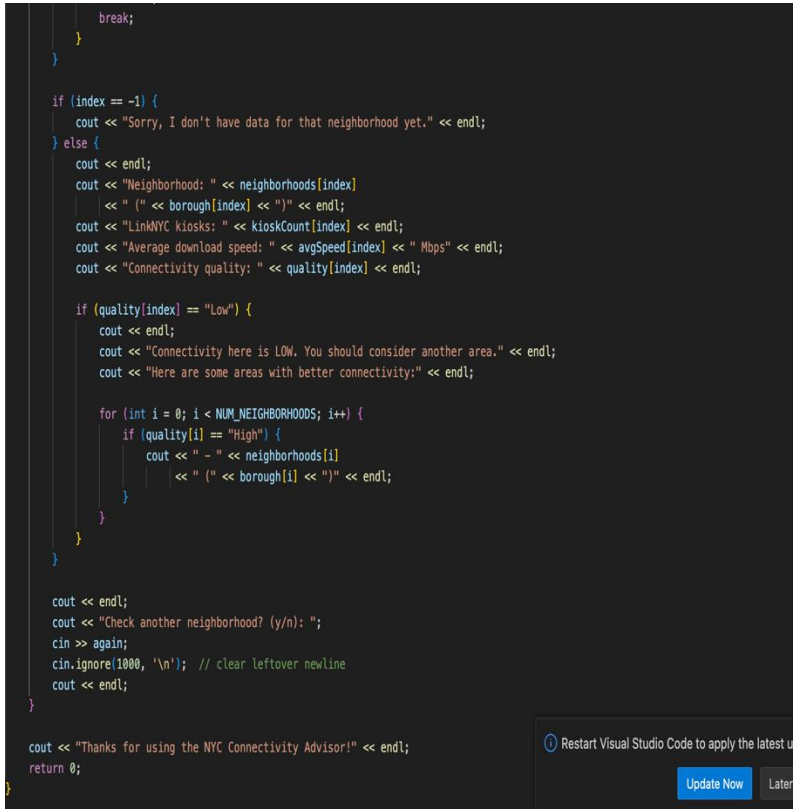
```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
    const int NUM_NEIGHBORHOODS = 8;

    // Basic sample dataset (you can edit names & numbers)
    string neighborhoods[NUM_NEIGHBORHOODS] = {
        "Astoria", "Harlem", "Midtown", "Downtown",
        "Williamsburg", "Jackson Heights",
        "Washington Heights", "Financial District"
    };

    string borough[NUM_NEIGHBORHOODS] = {
        "Queens", "Manhattan", "Manhattan", "Manhattan",
        "Brooklyn", "Queens",
        "Manhattan", "Manhattan"
    };

    int kioskCount[NUM_NEIGHBORHOODS]   = {40, 60, 120, 100, 50, 35, 25, 80};
    double avgSpeed[NUM_NEIGHBORHOODS]  = {150.0, 120.0, 200.0, 180.0,
                                           160.0, 110.0, 90.0, 210.0};
    string quality[NUM_NEIGHBORHOODS]   = {"High", "Medium", "High", "High",
                                           "High", "Medium", "Low", "High"};

    cout << "============================" << endl;
    cout << "  NYC CONNECTIVITY ADVISOR" << endl;
    cout << "============================" << endl << endl;

    string input;
    char again = 'y';

    while (again == 'y' || again == 'Y') {
        cout << "Enter a NYC neighborhood (example: Astoria, Harlem, Midtown):" << endl;
        cout << "> ";
        getline(cin, input);
```



```cpp
while (again == 'y' || again == 'Y') {
    cout << "Enter a NYC neighborhood (example: Astoria, Harlem, Midtown):" << endl;
    cout << "> ";
    getline(cin, input);

    // This fixes the common issue when getline is empty after using cin >> before
    if (input.size() == 0) {
        getline(cin, input);
    }

    int index = -1;

    // Search neighborhood in the array
    for (int i = 0; i < NUM_NEIGHBORHOODS; i++) {
        if (input == neighborhoods[i]) {
            index = i;
            break;
        }
    }

    if (index == -1) {
        cout << "Sorry, I don't have data for that neighborhood yet." << endl;
    } else {
        cout << endl;
        cout << "Neighborhood: " << neighborhoods[index]
             << " (" << borough[index] << ")" << endl;
        cout << "LinkNYC kiosks: " << kioskCount[index] << endl;
        cout << "Average download speed: " << avgSpeed[index] << " Mbps" << endl;
        cout << "Connectivity quality: " << quality[index] << endl;

        if (quality[index] == "Low") {
            cout << endl;
            cout << "Connectivity here is LOW. You should consider another area." << endl;
            cout << "Here are some areas with better connectivity:" << endl;

            for (int i = 0; i < NUM_NEIGHBORHOODS; i++) {
                if (quality[i] == "High") {
                    cout << " - " << neighborhoods[i]
                         << " (" << borough[i] << ")" << endl;
                }
            }
        }
    }
```

Restart Visual Studio Code to apply the latest upd

Update Now | Later



```cpp
            break;
        }
    }

    if (index == -1) {
        cout << "Sorry, I don't have data for that neighborhood yet." << endl;
    } else {
        cout << endl;
        cout << "Neighborhood: " << neighborhoods[index]
             << " (" << borough[index] << ")" << endl;
        cout << "LinkNYC kiosks: " << kioskCount[index] << endl;
        cout << "Average download speed: " << avgSpeed[index] << " Mbps" << endl;
        cout << "Connectivity quality: " << quality[index] << endl;

        if (quality[index] == "Low") {
            cout << endl;
            cout << "Connectivity here is LOW. You should consider another area." << endl;
            cout << "Here are some areas with better connectivity:" << endl;

            for (int i = 0; i < NUM_NEIGHBORHOODS; i++) {
                if (quality[i] == "High") {
                    cout << " - " << neighborhoods[i]
                         << " (" << borough[i] << ")" << endl;
                }
            }
        }
    }

    cout << endl;
    cout << "Check another neighborhood? (y/n): ";
    cin >> again;
    cin.ignore(1000, '\n');  // clear leftover newline
    cout << endl;
}

cout << "Thanks for using the NYC Connectivity Advisor!" << endl;
return 0;
}
```

Restart Visual Studio Code to apply the latest u

Update Now | Later

Future improvements

- **Libraries for Real-Time Data**
- **libcurl** – Fetches live data from APIs (best for real-time updates).
- **Boost.Beast** – Supports HTTP & WebSocket connections for continuous data streams.
- **cpp-httplib** – Lightweight library for simple API requests.
- **How Live Updates Work**
- **Timed update loop** – Program refreshes data automatically every few seconds/minutes.
- **API integration** – Retrieves updated connectivity, speed, and kiosk data.
- **NYC Data Sources**
- LinkNYC Open Data API
- NYC Open Data Portal
- Ookla Speedtest Global Index
- **Visualization Options**
- SFML or SDL2 for a 2D NYC map
- OpenGL for advanced graphics
- Web dashboard connected to C++ backend