Mauricio Martinez
MAC101 - Intro to Computer Science
CS 101 Technical Design Doc
Professor Hessvacio Hassan

CleanLoo NYC: A Smart Public Bathroom Finder

The Problem:

Finding a clean, open public bathroom in New York City is extremely difficult, especially in crowded areas like Midtown Manhattan, Times Square, and downtown Brooklyn. Many bathrooms are locked, poorly maintained, or lack updated information about their availability. As a result, residents and tourists waste time searching for restrooms or are forced to settle for unsanitary options. The problem becomes worse during peak hours when the foot traffic is high and the need for clean facilities increases but the information available remains limited and unreliable.

Who It Affects:

This issue affects a wide range of people including NYC residents, daily commuters, delivery drivers, gig workers, tourists unfamiliar with the city, parents with young children, and elderly or disabled individuals who may urgently need access to restrooms. These groups rely on public facilities but often cannot quickly determine which restrooms are clean, safe, or open, creating discomfort, stress, and potential health risks.

Solution Summary:

CleanLoo NYC is a simple, mobile-friendly, system that helps users quickly locate nearby public restrooms that are open and meet a minimum cleanliness rating based on users reports and public data.

Inputs:

CleanLoo NYC will combine user-generated information and NYC Open Data:

- **From NYC Open Data (Public Restroom Dataset):**
  - Restroom name/location
  - Latitude & Longitude
  - Hours of Operation
  - Accessibility features
  - Facility type (park restroom, comfort station, library restroom, etc)
- **From the User:**
  - Current location
  - Minimum cleanliness or usability rating (1-5)

    - Maximum distance they are willing to walk
    - Optional: require accessibility (Yes/No)

Outputs:

- List of nearby restroom filtered by:
    - Open status (based on hours & current time)
    - Minimum cleanliness rating
    - Maximum distance
- Top recommended restroom (closest + highest rating)
- Warning messages if no restroom fit the criteria
- Optional: show accessible restrooms only

C++ Variables

Restroom Related Variables

```
string restroomName;
string restroomLocation;
int cleanlinessRating;        // user generated
bool isOpen;                  // based on hours + time of day
bool isAccessible;            // from NYC Open Data
double restroomLatittude;
double restroomLongitude;
```

User input variables:

```
double userLatittude;
double userLatittude;
int userMinCleanliness;
int userMaxDistance;
bool requireAccessibility;
```

Logic Control Variable:

```
bool restroomFound;
```

Data Structure:

```
restroomList;          // vector containing restroom objects loaded from NYC Open Data
```

Loop Variable:

```
int i;           // index for iterating through restroomList in a loop
```

PSEUDOCODE

```
START

// STEP 1: Load NYC Open data into a list
LOAD restroomList from NYC Public Restroom dataset

// STEP 2: Get user preferences and location
INPUT userLatitude;
INPUT userLongtitude;
INPUT userMinCleanliness;
INPUT userMaxDistance;
INPUT requireAccessibility; // true if user needs accessible
restroom, false  otherwise

SET RestroomFound = false;

// Step 3: Loop through each restroom
FOR each restroom in restroomList

     // Get restrooms coordinates from the record
     READ restroomLatitude from restroom
     READ restroomLongtitude from restroom

     // Calculate the distance between the user and this
restroom
     CALCULATE distance between (userLatitude, userLongtitude)
                         and (restroomLatitude, restroomLongtitude)

     // Determine if the restroom is open (using its hours and
current time)
     CHECK restrooms hours of operation
     SET isOpen based on whether current time is within open
hours

     // Apply filters: open status, distance, cleanliness,
accessibility
```

```
     IF isOpen == true AND distance <= userMaxDistance AND
restroom.cleanlinessRating >= userMinCleanliness AND
(requireAccessibility == false OR restroom.isAccessible == true)
     THEN
          DISPLAY restroom.name, restroom.location, distance,
restroom.cleanlinessRating, restroom.isAccessible

          SET restroomFoud = true
     END IF

END FOR

// Step 4: If no matching restroom is found, inform the user
IF restroomFound = false THEN
     DISPLAY “No suitable restrooms found with your
preferences.”
END IF

END
```

C++ Connection:

1. Variables and Data Types:

- userMinCleanliness, userMaxDistance, and cleanlinessRating would be int variables in C++.
- userLatitude, userLongitude, restroomLatitude, and restroomLongitude would be double variables.
- isOpen, isAccessible, restroomFound, and requireAccessibility would be bool variables.
- restroomName and restroomLocation would be string variables that store text.

2. Main Data Structure - The Restroom List

- In C++, the restroomList could be implemented as a vector<Restroom> where Restroom is a struct or class containing fields like name, location, latitude, longitude, isAccessible, and cleanlinessRating.

- The line in pseudocode:
  LOAD restroomList from NYC Public Restrooms dataset

corresponds in C++ to reading each record from a file or API and pushing it into the vector using something like restroomList.push_back(newRestroom);.

3. For Loop

- The pseudocode:
  FOR each restroom in restroomList
  would become a for loop in C++, such as:

```
     for (int i = 0; i < restroomList.size(); i++) {

     // access  restroomList[i]

     }
```

4. IF Conditions

- The pseudocode condition:

IF isOpen == true AND distance <= userMaxDistance AND restroom.cleanlinessRating >= userMinCleanliness AND (requireAccessibility == false OR restroom.isAccessible == true)

```
          if (isOpen && distance <= userMaxDistance &&

          cleanlinessRating >= userMinCleanliness &&

          (!requireAccessibility || isAccessible)) {

               // display restroom information

          }
```

5. Display Output:

- The pseudocode DISPLAY lines would map C++ output using cout:

```
     cout << restroomName << " at " << restroomLocation <<
" is "  << distance << " units away" << endl;
```

The pseudocode uses the same logical building blocks as C++: variables, loops, and conditional statements