Take Home Assessment

For

Senior Full Stack Engineer

Md Mostafizur Rahman

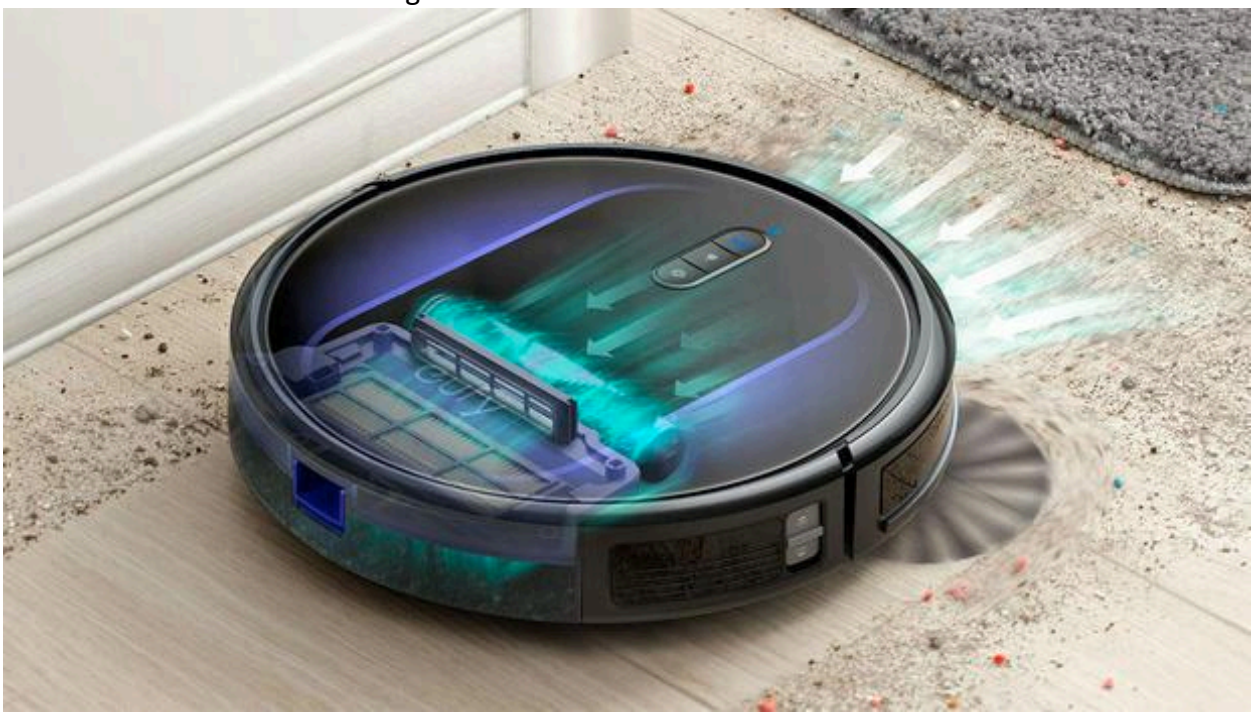
csemrm@gmail.com

Date: September 4, 2023

Table of Contents

Project Title: 3
Project Requirements: 4
Architecture: 5
Methodology Description: 6
Source Code: 7
Assumptions: 9
Possibilities of Improvement: 10
Demonstration of Results: 11

Project Title:
Robot Vacuum Cleaner Challenge



Project Requirements:

1. Starting from Room 1, there is an endless sequence of rooms. Each subsequent room is located to the right of the previous one.
2. The vacuum cleaner can either move to the left or to the right, going in between rooms. (We are assuming that the rooms are numbered)
3. The vacuum cleaner won't move unless it receives a cleaning command.
4. The vacuum cleaner starts in Room 1.
5. The vacuum receives its cleaning instructions as an array of arrays (cleaning batches).

    i) Each array represents a batch of rooms to clean.
    ii) The numbers represent the rooms the vacuum cleaner has been assigned in a batch
    iii) Each array of array represents the number of cleaning batches
    iv) We'll assume that a new room is not inserted into the batch once the vacuum starts cleaning.
    v) For example, these are 3 cleaning batches described as an array of arrays (Ruby): [[3,2,4],[2,8,4],[4,6,4,9]]
    vi) Describing the cleaning batch jobs above:
    vii) Vacuum starts at room 1 and moves right to room 3, crossing room 2.
    viii) Finishes its cleaning and then moves left to room 2, and finally moves right crossing room 3 to room 4. The vacuum is now in room 4.
    ix) The vacuum now starts in room 4, crosses room 3 and moves to room 2, then room 8 and back to room 4. The vacuum is now in room 4.
    x) The vacuum starts in room 4, moves to room 6, then to room 4, then to room 9. The vacuum is now in room 9.

6. The vacuum also receives another set of commands called ‘Priority Rooms’

    i) It receives the command as a single array
    ii) Example: [7,14, 1]
    iii) If a priority room is in the cleaning batch, the vacuum should first attend to these rooms before cleaning others.

7. The vacuum furthermore has to prioritize optimizing its room traversal such that it visits the least amount of rooms
8. At the end of processing an array of array of cleaning commands, the system should output:

    i) The actual path the vacuum took as arrays of rooms (An array of arrays showcasing its actual traversal of rooms)
    ii) How many rooms it cleaned in total
    iii) How many batches it processed
    iv) How many rooms it passed without cleaning, either going left or right
    v) The final (current) room the vacuum cleaner is in

Architecture:

1. The program should be set up as a microservice. Think about how you can segregate each functionality as a service.
2. For the purposes of this challenge, I will implement a few primary endpoints.
3. Implement a client script or use tools like Postman to demonstrate sending commands between service(s) and getting output(s)

Methodology Description:

Based on Our project requirements and architecture, I will use containerization architecture to isolate and self-service microservice with the Python tech stacks with the following:

1. Server Environment:
    - Docker
    - Python 3.9
    - Python Virtual environment
    - Alpine Linux 3.18.0
2. Programming Environment:
    - Django (v:4.2.1)
    - Django Rest Framework (v:3.14.0)

In this project, I will develop a REST API-based microservice, where I will send two list data via post method, cleaning batches and Priority Rooms. My app will accept data and process with Django Rest Framework. After receiving data, my vacuum cleaner function will process data and return our processed output data.

Source Code:
In my code, I have two major components.
Full source code link: https://github.com/csemrm/vacuum-cleaner

```
def vacuum_cleaner(self, cleaning_batches, priority_rooms):
    """
            Simulate a vacuum cleaner cleaning algorithm.

            Args:
        request (HttpRequest): The HTTP request object.
        In request object cleaning_batches, priority_rooms data
        will available for process implementation.

    Returns:
        An response with cleaning results with 5
        outputs like:
        traversal_path: The actual path the vacuum took as arrays of
        rooms (An array of arrays showcasing its actual traversal of rooms)
        total_cleaned: # of rooms it cleaned in total
        total_batches: # of batches it processed
        total_unvisited: # of rooms it passed without cleaning either
        going left or right
        final_room: The final (current) room the vacuum cleaner is in
    """
    current_room = 1
    cleaned_rooms = set()
    traversed_rooms = []
    total_cleaned = 0
    total_batches = 0
    total_unvisited = 0

    for batch in cleaning_batches:
        total_batches += 1
        print('batch', batch)
        priority_rooms_in_batch = set(priority_rooms) & set(batch)
        print('priority_room', priority_rooms_in_batch)
        # Visit priority rooms first
        for room in priority_rooms_in_batch:
            print('priority_room: ', room, current_room)
            # room = 7 < current_room =1
            # checking cleaner moving directions for priority rooms
            if room < current_room:
                rooms = range(current_room - 1, room, -1)
                total_unvisited += len(rooms)
                traversed_rooms.extend(rooms)
            else:
                rooms = range(current_room + 1, room)
                total_unvisited += len(rooms)
                traversed_rooms.extend(rooms)
            current_room = room
            cleaned_rooms.add(current_room)
            total_cleaned += 1
            traversed_rooms.append(current_room)

        # Visit Non-priority rooms in the batch
        print('Visit rooms in the batch: ', batch)
```

```
        for room in batch:
            print('Visit other room in the batch: ', room)
            if room != current_room and room not in priority_rooms:
                print('Visit room and current_room in the batch: ', room,
current_room)

                # checking cleaner moving directions for non-priority rooms

                if room < current_room:
                    rooms = range(current_room - 1, room, -1)
                    total_unvisited += len(rooms)
                    traversed_rooms.extend(rooms)
                else:
                    rooms = range(current_room + 1, room)
                    total_unvisited += len(rooms)
                    traversed_rooms.extend(rooms)

                current_room = room
                cleaned_rooms.add(current_room)
                total_cleaned += 1
                traversed_rooms.append(current_room)

            elif room == current_room and room not in priority_rooms:
                cleaned_rooms.add(room)
                total_cleaned += 1

    return {
        "traversal_path": traversed_rooms,
        "total_cleaned": total_cleaned,
        "total_batches": total_batches,
        "total_unvisited": total_unvisited,
        "final_room": current_room,
    }
```

Assumptions:
Let's consider a scenario where both parameters are expected to be included exclusively in an API call utilizing the POST method.

Another assumption is that rooms will not sorted. For a sorted room, it will be one-directional moving.

Possibilities of Improvement:

By sorting our cleaning batch into ascending order, we can save time and avoid the cost and time of back-and-forth moving between random rooms. This will help to make the process more efficient.

Demonstration of Results:

API URL: http://localhost:8000/api/vacuum
Content-Type: application/json

_____

Parameters 1:

```
cleaning_batches = [[3, 6, 7]]
priority_rooms= [7, 4]
```

CURL Request 1:

```
curl -X POST -H "Content-Type: application/json" -d '{
  "cleaning_batches": [[3, 2, 4], [2, 8, 4], [4, 6, 4, 9]],
  "priority_rooms": [7, 14, 1]
}' http://localhost:8000/api/vacuum
```

Result 1:

{
"traversal_path": [2, 3, 4, 5, 6, 7, 6, 5, 4, 3, 4, 5, 6],
"total_cleaned": 3,
"total_batches": 1,
"total_unvisited": 10,
"final_room": 6
}

Traversal Path: [2, 3, 4, 5, 6, 7, 6, 5, 4, 3, 4, 5, 6]

Total Cleaned: 3

Total Batches: 1

Total Unvisited: 10

Final Room: 6

_____

Parameters 2:

```
cleaning_batches =  [[3, 2, 4], [2, 1, 4], [4, 5]]
priority_rooms= [7, 14, 1]
```

CURL Request 2:

```
curl -X POST -H "Content-Type: application/json" -d '{
  "cleaning_batches":  [[3, 2, 4], [2, 1, 4], [4, 5]],
  "priority_rooms": [7, 14, 1]
}' http://localhost:8000/api/vacuum
```

Result 2:

{
"traversal_path":[2,3,2,3,4,3,2,1,2,3,4,5],
"total_cleaned":8,
"total_batches":3,
"total_unvisited":5,
"final_room":5

}

Traversal Path: [2, 3, 2, 3, 4, 3, 2, 3, 4, 5, 6, 7, 8, 7, 6, 5, 4, 5, 6, 5, 4, 5, 6, 7, 8, 9]
Total Cleaned: 8
Total Batches: 3
Total Unvisited: 5
Final Room: 5

2,3,4,5,6,7 →

CR #6

Cleaned Room #{7}

cleaning batch = [[3,6,7]]

Priority rooms = [7,4]

7,3

R#6, 2 ⟹ 6,5,4,3.

5+3

8

4, 6

4,5,6

2,3,4,5,6,7, 6,5,4,

CR#3

Traversal Path = 2,3,4,5,6,7,6,5,4,3,4,5,6

Total cleaned = 3

Batch(B) = 1

Unvisited(UV) = 10 (5+3+2)

Final room = 6