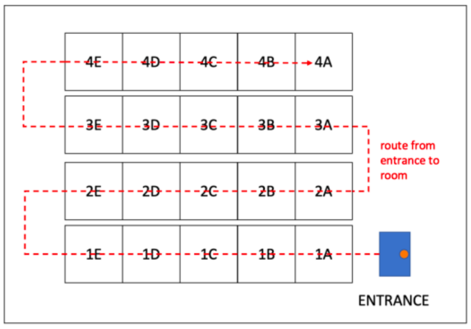
Backend Engineer Assignment

Submitted by Ajay Kumar Maurya

Question 1: Part 1 (Software Design and Implementation):

An owner of a boutique hotel asked you to develop a program to help him assign available rooms to his guests. Your program must find the nearest available room measured by the route taken from the entrance to the room (see diagram) and assign it to the guest.

- The hotel has M floors and there are N rooms on each floor
- Rooms are numbered by the floor as prefix followed by a single alphabet
  suffix A, B, C, D or E where A is the first room (relative to the entrance) until E for the last room.



Hotel Layout

$$\begin{tikzcd} \text{Available} \arrow[r, "\text{check-in}"] & \text{Occupied} \end{tikzcd}$$

1. Only **Available** rooms can be assigned to guests which will then become **Occupied**.

$$\begin{tikzcd} \text{Occupied} \arrow[r, "\text{check-out}"] & \text{Vacant} \end{tikzcd}$$

2. After the guest checks out of the room, the room becomes **Vacant**.

$$\begin{tikzcd} \text{Vacant} \arrow[r, "\text{clean}"] & \text{Available} \end{tikzcd}$$

3. If housekeeping cleans a **Vacant** room, it becomes **Available**.

$$\begin{tikzcd} \text{Vacant} \arrow[r, "\text{out of service}"] & \text{Repair} \end{tikzcd}$$

4. Housekeeping may take a **Vacant** room for repairs by marking the room as **Repair**.

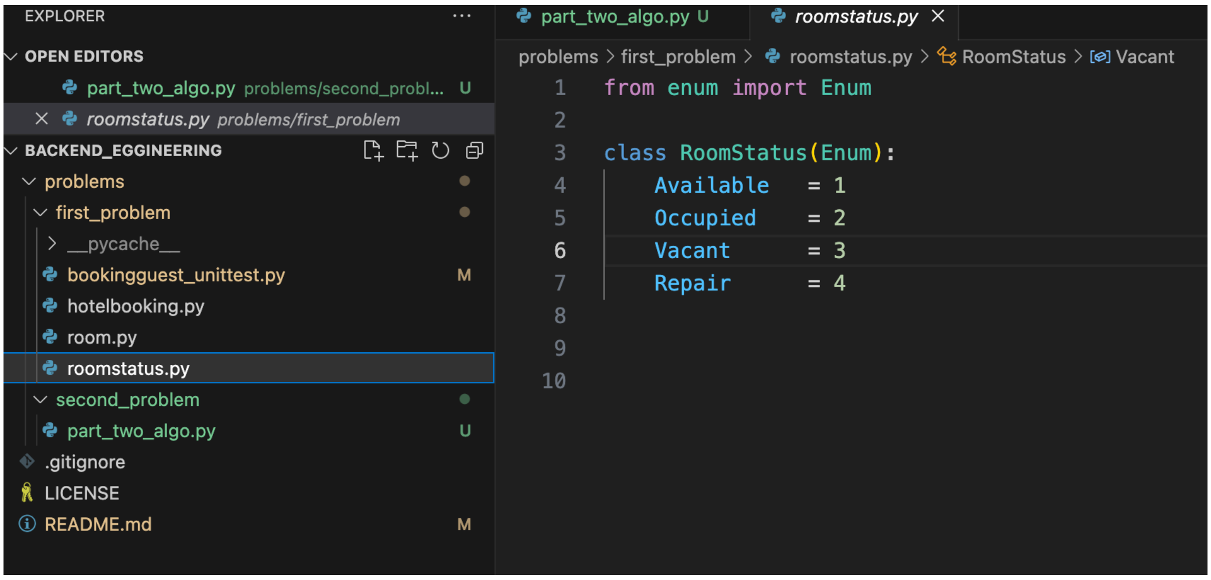
$$\begin{tikzcd} \text{Repair} \arrow[r, "\text{repaired}"] & \text{Vacant} \end{tikzcd}$$

5 .Once a room is repaired, it will become **Vacant** again for cleaning.

6. Rooms under **Repair** can only be made **Vacant**.

7. **Available** and **Occupied** rooms cannot be repaired.

>> Solution:



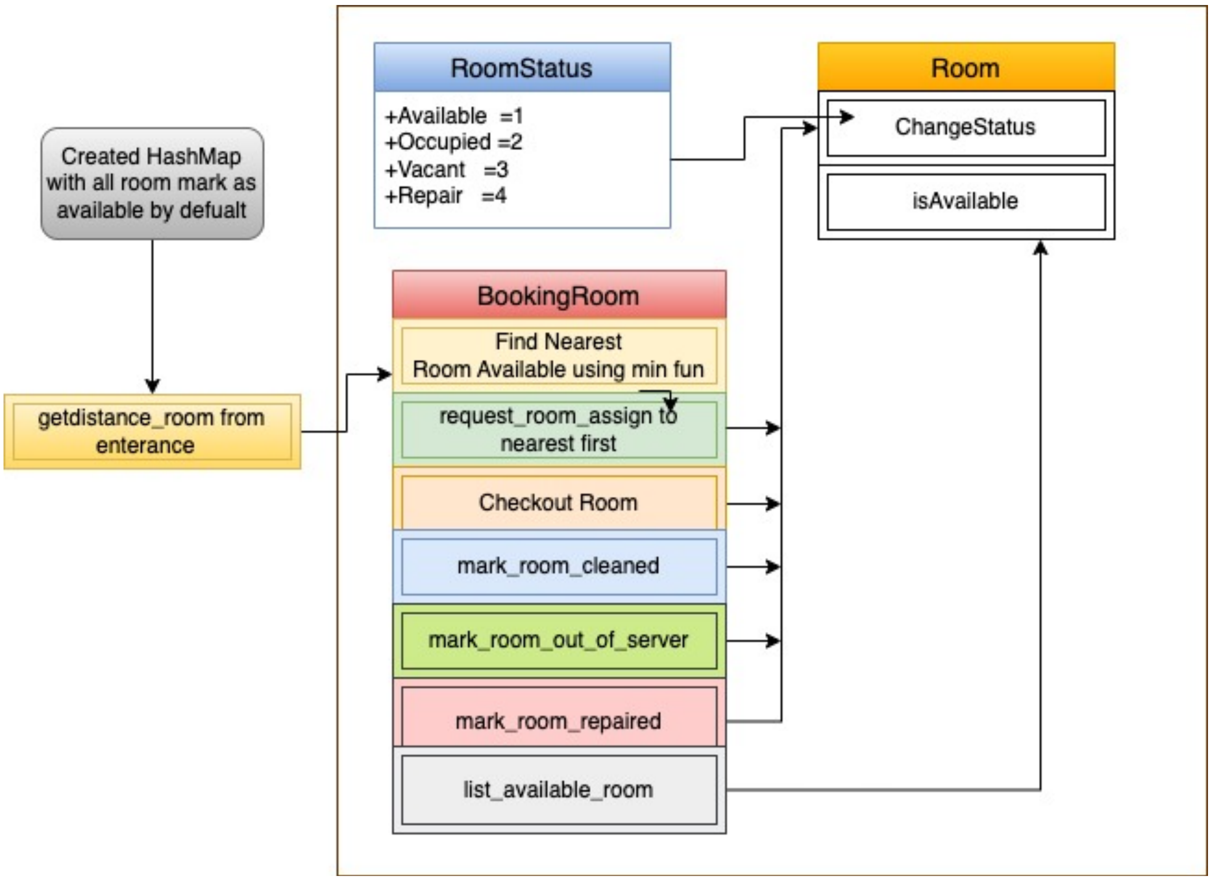
```
EXPLORER
OPEN EDITORS
    part_two_algo.py problems/second_probl...   U
  × roomstatus.py problems/first_problem
BACKEND_EGGINEERING
  problems
    first_problem
      > __pycache__
      bookingguest_unittest.py                  M
      hotelbooking.py
      room.py
      roomstatus.py
    second_problem
      part_two_algo.py                          U
  .gitignore
  LICENSE
  README.md                                     M
```

```
part_two_algo.py U    roomstatus.py ×
problems > first_problem > roomstatus.py > RoomStatus > Vacant
```

```python
from enum import Enum

class RoomStatus(Enum):
    Available   = 1
    Occupied    = 2
    Vacant      = 3
    Repair      = 4
```

Created various classes to fulfill the requirement, I will provide a Python program with the following classed and method by following the OOPs Solid principal programing language in Python.

1. **"hotelbooking.py":** A class "**HotelRoomBooking**" to implement the business logic that handles the room assignment, checkout and room status changes.
2. **"Room"** class **in "room.py" file** represent each room with their status
3. **"roomstatus.py"** enum class to represent the possible room status



Note* Please clone the project and perform the instruction to execute this program.

Question 1: Part 2 (Algorithm)

The input to the program is a string with the following format:

```
M N
<hotel matrix>
```

For example, given the following input and infection sequence:

```
3 5
2 1 0 2 1       2 2 0 2 2      2 2 0 2 2
1 1 1 1 1  ->   2 1 1 2 1  ->  2 2 2 2 2
1 0 0 2 1       1 0 0 2 2      2 0 0 2 2
```

The output would be 2 units of time.

The size of the input matrix is at most 1000 x 1000.

Deliverables:

- The solution can be developed using the language of your choice complete with unit tests:
    - The program must output the answer as a string
    - You are free to design your own models, inputs and outputs
- Please describe in a few sentences your solution
- The code will be judged on its efficiency (time complexity) and readability
- Please provide instructions on how to compile, run and use your code
- Code must include frequent Git commits

Each room in the matrix or rooms can have a value of 0, 1 or 2 with the following meaning:

- The room is empty (0)
- The room has only uninfected guest(s) (1)
- The room has infected guest(s) (2)

Solution: # The infection can spread from an infected room to an uninfected one only **up, down, left and right** in the matrix of rooms during one unit of time.

Approaches: if Node is infected i.e **2 then look the adjacent of that node Matrix [row][col] by adding +1 and minus -1 with respect to the column. Algorithm is mentioned below in python.**

```
# Step:1 create algorithm to find the adjacent up, down, left and right

def adjacent_rooms(hotel, row, col):

  adjacent = []

  # Move Left

  if row > 0:

    adjacent.append((row - 1, col))

  # Move Right

  if row < len(hotel) - 1:

    adjacent.append((row + 1, col))

  # Move Up

  if col > 0:

    adjacent.append((row, col - 1))

  # Move Down
  if col < len(hotel[0]) - 1:
    adjacent.append((row, col + 1))
  return adjacent
```

By applying **Breadth first Search algorithm**, we will perform **BFS starting from the initially infected** rooms and propagate the infection to its adjacent uninfected.

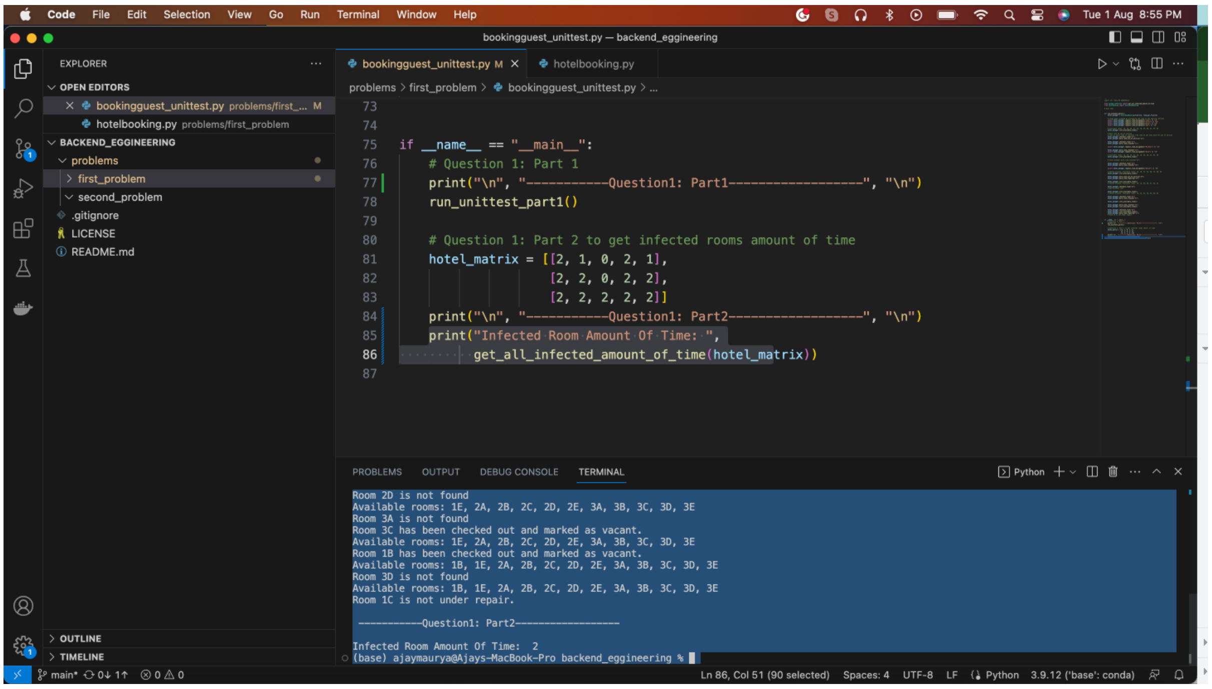
Now the step to create the final function to iterate the matrix and replace with 2 i.e as infected if the room was uninfected.

```
# Find all infected vertices if node value is 2
def get_initial_infected_rooms(hotel):
  # Adding here all vertices as tuple in list (row, col) format which is going to be stored into the infected_rooms
  infectedrooms = []
  for row in range(len(hotel)):
    for col in range(len(hotel[0])):
      if hotel[row][col] == 2:
        infectedrooms.append((row, col))
  return infectedrooms
def get_all_infected_amount_of_time(hotel):
  # Check if empty guests, then the infect all is 0
  if not hotel or not hotel[0]:
    return "0"

  # Step:2 to fetch all infected room to perform BFS to traverse each verticies
  infected_rooms = get_initial_infected_rooms(hotel)

  # No initial infection, so all guests cannot be infected
  if not infected_rooms:
    return "-1"

  time = 0
  # Traversing each vertices anf override with updated uninfected vertices if the guest exist.
  while infected_rooms:
    new_infected_rooms = []
    for row, col in infected_rooms:
      for adj_row, adj_col in adjacent_rooms(hotel, row, col):
        if hotel[adj_row][adj_col] == 1:
          hotel[adj_row][adj_col] = 2
          # update for all verticess if hotel[adj_row][adj_col] == 1 run again by getting new affected vertices.
          new_infected_rooms.append((adj_row, adj_col))
    # assing new queue of vertices which got effected for next iterations

    infected_rooms = new_infected_rooms

    time += 1

  if any(1 in row for row in hotel):

    return "-1"

  return str(time)
```

How to execute this program:

1. Clone the project from GitHub repository [git@github.com:ajay9889/backend_eggineering.git]
2. Make sure you have Python installed on local system
3. Run the below command.
4. > **python** bookingguest_unittest.py

Time Complexity: O(M*N)



```python
if __name__ == "__main__":
    # Question 1: Part 1
    print("\n", "-----------Question1: Part1-----------------", "\n")
    run_unittest_part1()

    # Question 1: Part 2 to get infected rooms amount of time
    hotel_matrix = [[2, 1, 0, 2, 1],
                    [2, 2, 0, 2, 2],
                    [2, 2, 2, 2, 2]]
    print("\n", "-----------Question1: Part2-----------------", "\n")
    print("Infected Room Amount Of Time: ",
          get_all_infected_amount_of_time(hotel_matrix))
```

```
Room 2D is not found
Available rooms: 1E, 2A, 2B, 2C, 2D, 2E, 3A, 3B, 3C, 3D, 3E
Room 3A is not found
Room 3C has been checked out and marked as vacant.
Available rooms: 1E, 2A, 2B, 2C, 2D, 2E, 3A, 3B, 3C, 3D, 3E
Room 1B has been checked out and marked as vacant.
Available rooms: 1B, 1E, 2A, 2B, 2C, 2D, 2E, 3A, 3B, 3C, 3D, 3E
Room 3D is not found
Available rooms: 1B, 1E, 2A, 2B, 2C, 2D, 2E, 3A, 3B, 3C, 3D, 3E
Room 1C is not under repair.

 -----------Question1: Part2-----------------

Infected Room Amount Of Time:  2
(base) ajaymaurya@Ajays-MacBook-Pro backend_eggineering %
```

In order to perform the testing we need to execute the file "bookingguest_unittest.py"
Command: **python** bookingguest_unittest.py

Output:

-----------Question1: Part1-----------------

Room 1A assigned to John
Room 1B assigned to Allen
Room 1C assigned to Alex
Room 1D assigned to Sam
Available rooms: 1E, 2A, 2B, 2C, 2D, 2E, 3A, 3B, 3C, 3D, 3E
Room 1A has been marked for out of service and under repair.
Room 1C has been checked out and marked as vacant.
Room 1C assigned to Michael

```
Room 1A has been checked out and marked as vacant.
Room 1A assigned to Sara
Available rooms: 1E, 2A, 2B, 2C, 2D, 2E, 3A, 3B, 3C, 3D, 3E
Room 1B has been checked out and marked as vacant.
Room 1B assigned to David
Available rooms: 1E, 2A, 2B, 2C, 2D, 2E, 3A, 3B, 3C, 3D, 3E
Room 2B not found.
Room 2B is not under repair.
Available rooms: 1E, 2A, 2B, 2C, 2D, 2E, 3A, 3B, 3C, 3D, 3E
Room 2D is not found
Available rooms: 1E, 2A, 2B, 2C, 2D, 2E, 3A, 3B, 3C, 3D, 3E
Room 3A is not found
Room 3C has been checked out and marked as vacant.
Available rooms: 1E, 2A, 2B, 2C, 2D, 2E, 3A, 3B, 3C, 3D, 3E
Room 1B has been checked out and marked as vacant.
Available rooms: 1B, 1E, 2A, 2B, 2C, 2D, 2E, 3A, 3B, 3C, 3D, 3E
Room 3D is not found
Available rooms: 1B, 1E, 2A, 2B, 2C, 2D, 2E, 3A, 3B, 3C, 3D, 3E
Room 1C is not under repair.

 -----------Question1: Part2------------------

Infected Room Amount Of Time:  2
(base) ajaymaurya@Ajays-MacBook-Pro backend_eggineering %
```

Question 2: Graph Design

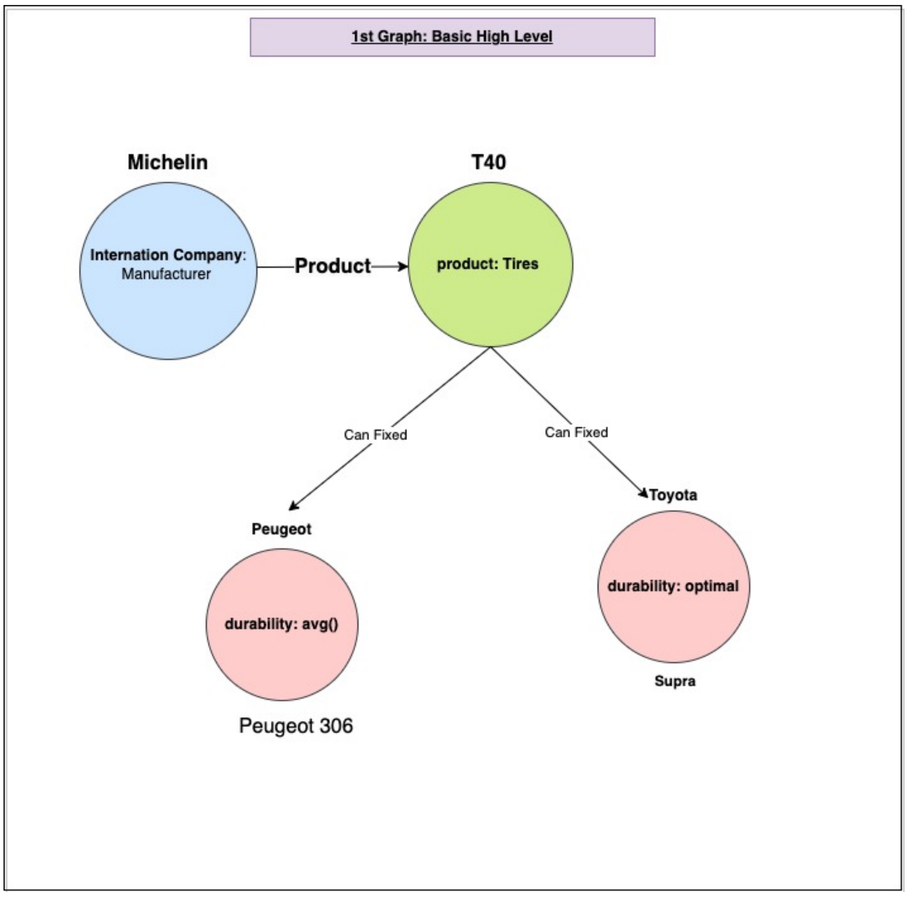
Given the following statements:

“Michelin is an international manufacturer of tires. The company has a model of tires T40 that can be fixed on Peugeot and Toyota’s products. However, the durability of this model is average for the Peugeot 306 cars, but optimal for Toyota Supra cars.”

1st Graph: Basic Design:

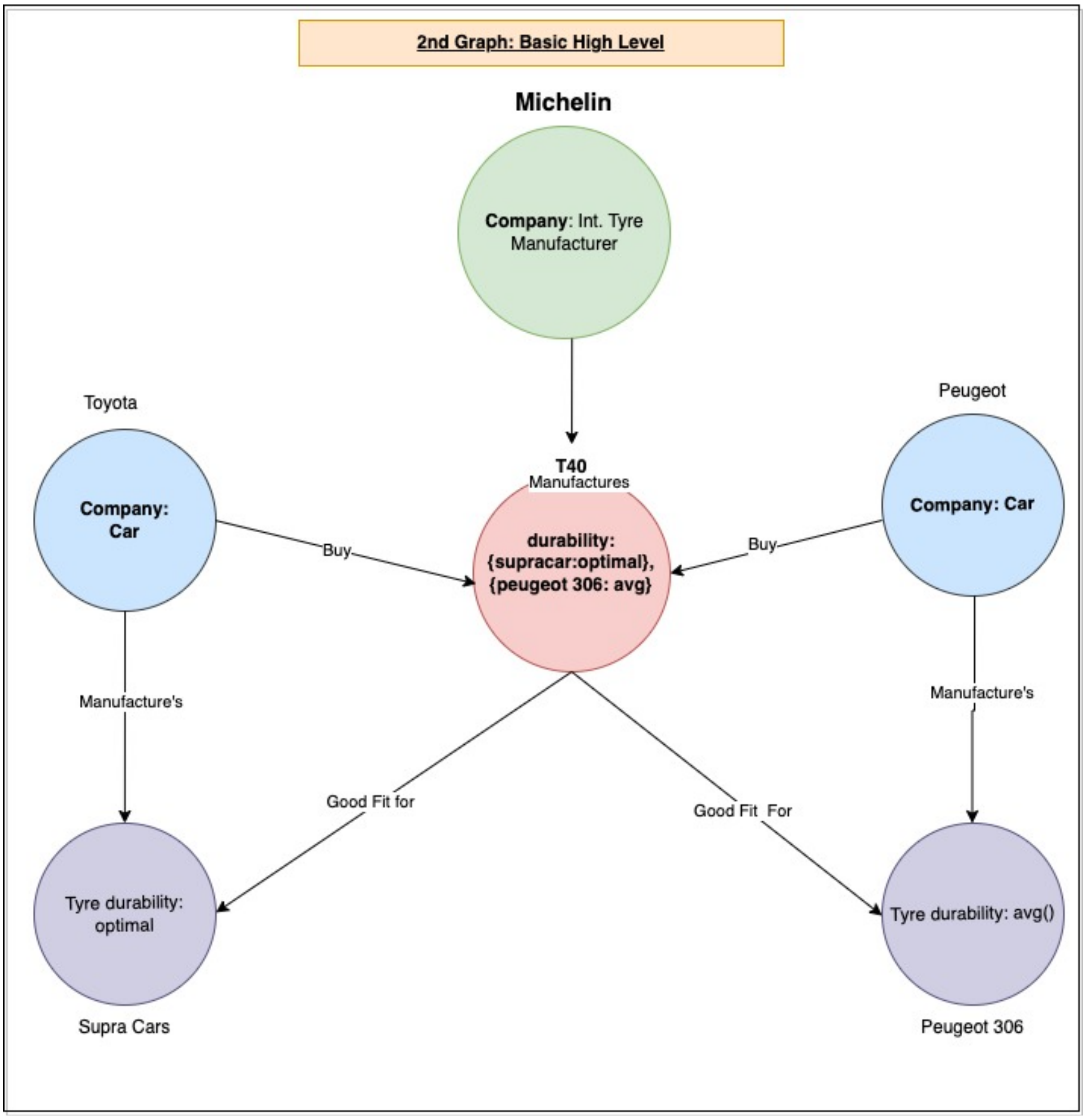
Representing on the graph the relationships and characteristics:

1. "Michelin" to "T40" to show that Michelin manufactures the T40 tire model.
2. "T40" to "Peugeot" and "T40" to "Toyota" to indicate that the T40 model can be fixed on both Peugeot and Toyota products.
3. "Average" from "Peugeot" to "306 Cars" to represent that the durability of the T40 model is average for Peugeot 306 cars.
4. "Optimal" from "Toyota" to "Supra Cars" to represent that the durability of the T40 model is optimal for Toyota Supra cars.



In 2nd approach into:
Representing on the graph the relationships and characteristics:

1. "Michelin" to "T40" to show that Michelin manufactures the T40 tire model.
2. Car company Toyata and Peugeot must Buy T40 Tires from manufactures
3. "T40" tires is good fit for "Peugeot"
4. Similarly, "T40" tires also good fit for "Toyota" cars product.
5. "Average" from "Peugeot" to "306 Cars" to represent that the durability of the T40 model is average for Peugeot 306 cars.
6. "Optimal" from "Toyota" to "Supra Cars" to represent that the durability of the T40 model is optimal for Toyota Supra cars.


2nd Graph: Basic High Level
Michelin
Company: Int. Tyre Manufacturer
T40
Manufactures
durability: {supracar:optimal}, {peugeot 306: avg}
Toyota
Company: Car
Buy
Peugeot
Company: Car
Buy
Manufacture's
Manufacture's
Good Fit for
Good Fit For
Tyre durability: optimal
Supra Cars
Tyre durability: avg()
Peugeot 306