SQUADSTACK

Problem Statement Analysis :-

The problem states that a Parking ticketing system should be implemented.

The system can allow ‘n’ cars at a time.

There are ‘n’ slots. Slot number starts from 1 to ‘n’.



When a car enters,

we note down :-   
 -> driver age

-> Vehicle number

We assign a slot (if available)

-> to car which is closest to the entry

When car leaves,  
 we get command “leave <x> “

-> if car exists in that slot then we mark it as vacant

To be able to give results for :-

-> given age, find vehicle reg nos. of cars parked (currently)

-> slot no of a car, given reg. no

-> given age, find slot nos. of cars parked (currently)

Edge cases :-

1. Cars > n come, then deny parking

“Parking full”

1. Given age, no cars then -> null (if asked for slots or vehicle no)
2. Vacate a slot but that is not occupied, just print ->

“Slot <x> vacated”

ASSUMPTIONS :-

1. Commands have no spell error, if they have spell error, then I print “Incorrect command and read the next command”
2. Car reg. no format is fixed
3. Only 1 create\_parking\_lot command at the start
4. Cars have distinct registration no
5. If commands have case different, then I ignore case
6. File provided is of format .txt and readable
7. Output displayed in terminal
8. N < 1000 cars
9. When we get vehicle nos. of all cars whose drivers are of Age = ‘x’, we return a Comma separated list of vehicle nos.
10. History is not saved

TECH STACK :-

Python, with framework to interface with DB