Problem statement -

Design a Parking lot management service. Which needs to be deployed on the cloud and all shopping complexes and office spaces are on this platform.

On day one we need to launch with about 1200 parking spaces; each having 3 floors, where each floor has - 100 Small, 100 Medium, 100 Large, and 100 XL slots.

Different parking spaces will have a different number of slots, but we can take the above break-up as the standard for this problem statement.

User Base and Scale -

The service will scale to 10 million parking spaces {each having about 1000+ parking spaces} in the future with about 100 spaces getting on board each week, so the design and implementation need to be scalable.

In Scope - The service should take care of the following -

1. On Board a new Parking lot to the system -
    a. The number of Parking Slots.
    b. Size of slots { Small, Large, Medium, X-Large }
        i. Number of slots for each size
    c. Parking Floors, number of Slots on each floor with size.
2. When a car arrives at the gate –
    a. We need to allocate it the slot number which should be printed on the parking ticket
        i. Slot Number Template – [FloorID:Bay ID]
    b. If a Small car arrives and –
        i. If we have a small slot then we should allocate a Small Slot
        ii. If No Small slot if free then Medium slot
        iii. If No Medium is free then Large slot
        iv. If no Large is free then XLarge slot
        v. If no XLarge is available then don't print the slot, print no SLOT FOUND
    c. If a Large car arrives then we start with Large then Xlarge
    d. If a Medium car arrives, we start with Medium, then large then XLarge
    e. If a Large car arrives then Large, followed by Xlarge If no large is free
    f. If an Xlarge car arrives then it needs to be Xlarge slot.

    The goal is to get the smallest slot that can fit the car.
3. When the car leaves the gate; marked the slot as free to be made available for the next car.

Assumptions you can make –

1. Car will be scanned automatically; when the customer presses the button at the gate to get the slot following Api will get invoked; with the parking space identifier and car size.
    a. http://parkingservice.com/getslot/<Parking_lot_id>/<Size>
    b. Response –
        i. { Slot : <Floor>-<Bay-ID> }
    c. When a car leaves the parking bay- http://parkingservice.com/releaseslot/<Parking_lot_id>/<Slot-ID>
        i. This endpoint will make the parking bay available for another car.
2. We don't have to allocate parking bays in a sequence, having said that there is a constraint if you want to do that feel free.

WARNING – Please note that we have ways to figure out if you have copied the solution from the internet. And copying the whole code/design from the internet could lead to rejection and block your candidature in the near and distant future. Copying a small section of code is okay. You need to treat it as how you will work on this on a job; you will have access to the internet and can browse and refer to sites; but don't copy and paste the code.

Out of Scope –

1. Billing, invoicing, charging the customer, taking payments, and timing the car for parking duration.
    a. All the above items are out of scope, for this, however, if you want to include any one or some of them in your solution, please feel free to surprise us with your imagination on the same.
2. Each floor will have different order of the parking bay sizes
    a. Some may have 50 Small, followed by 10 Large, some may have in random order
    b. You need not worry about the arrangement of the parking slots.
    c. **All you need to take care of is** - Parking Lot → Floor → Bay Size → Number of Slots

Basic Expectation –

1. Design UI for
    a. Allocation of a slot to a car
    b. Deallocation of a slot from the car
2. Design for the services -
    a. Request response payload
    b. Back end design
    c. DB Schema
        i. Type of DB
        ii. Table Schema
3. Implementation of at least, but not limited to, the following two services -
    a. GetSlot
    b. FreeSlot
4. Solution needs to be –
    a. Scalable – Should not get slow as more and more spaces get onboarded.
    b. Traceable – trace service execution, and support debugging.
    c. Maintainable
        i. The code needs to be simple, and readable – No java docs
        ii. Adequate unit test – at least 85%-line coverage, 100% class coverage
    d. Thread-safe –
        i. Under no circumstances should the solution allocate one slot to multiple cars for a given parking lot.
        ii. No synchronized code blocks as they will kill the scale and performance of the solution.
        iii. All Services need to be thread safe.
5. The service will be deployed on Cloud so try not to have an on-prem kind of design/Solution, keep it for cloud-based deployments.

Programming Language -

Our preferred tech stack for this is – Flutter { Mandatory] for UI, Typescript, React, and Node/Java (backend). You are free to choose any version of these technologies.

WARNING – Please note that we have ways to figure out if you have copied the solution from the internet. And copying the whole code/design from the internet could lead to rejection and block your candidature in the near and distant future. Copying a small section of code is okay. You need to treat it as how you will work on this on a job; you will have access to the internet and can browse and refer to sites; but don't copy and paste the code.

WARNING – Please note that we have ways to figure out if you have copied the solution from the internet. And copying the whole code/design from the internet could lead to rejection and block your candidature in the near and distant future. Copying a small section of code is okay. You need to treat it as how you will work on this on a job; you will have access to the internet and can browse and refer to sites; but don't copy and paste the code.