You are now working on a game called Car Driving Online. It's a car driving simulation game, where player can freely drive his car around, collecting stuffs, making stunts, and joining online with other friends (using Photon Unity Networking - PUN)

### Physics

1. You are about to implement the buoyancy effect for a special car type, which can move freely above the water level. How can you solve that problem using PhysicsX?

### Architecture Design questions

1. You are about to implement the collection system, which handles 3 things : allows the designer to manually place tokens around the map, allows the player to collect those tokens by colliding with them, and allows the player to unlock new character skins / vehicle skins / etc when collecting enough tokens. How do you design that system, in such a way that least-dependant on other gameplay elements, because our current codebase is quite chaotic.

### Networking

1. The game uses PUN, each player updates his location to others. There is a reported exploit that allows cheaters to cheat by using Cheat Engine to speed up the game. How is that possible ? And what can you do to minimize impacts from that exploit?
2. You are about to design the synchronization mechanism for the game. Each player controls a vehicle, which needs to sync properties below. How do you synchronize those properties?
   1. Position of the vehicle
   2. Rotation of the vehicle.
   3. Wiper State: 3 state - on , low, high
   4. Rear light State: 3 state: off , on , blink
   5. Sidelight State: 4 state: off, on , blink-slow, blink-fast
   6. Horn: player can press and release the horn button, to start and end the horn sound ( he sound continuously plays when player holds the button)

**Answers**

### Physics

We can approach this buoyancy effect for a special car type by implementing these steps on the car object:

* 1. Collider
  2. Rigidbody
  3. Buoyancy Script

We can implement a simple buoyancy script like:

Sebuah gambar berisi teks, cuplikan layar, software, Font

Deskripsi dibuat secara otomatis

### Architecture Design questions

To design a manageable codebase system for this case, there are some steps that needs to be implemented:

* + - 1. Collection System Overview: The collection system can be broken down into three main functionalities: manual token placement, token collection through collision, and unlocking rewards. This system should be designed to be independent of other gameplay elements to reduce dependencies and improve code organization.
      2. TokenManager Class: Begin by creating a TokenManager class that will be responsible for managing the placement, collection, and unlocking functionalities. This class should have methods for placing tokens in the game world, detecting collisions with the player, and tracking the number of collected tokens.
      3. Token Class: Implement a Token class to represent individual tokens. Each token can store information such as position, collision detection, and reward data.
      4. RewardManager Class: Create a RewardManager class to handle the unlocking of character skins, vehicle skins, etc. This class should have methods for tracking the number of collected tokens and determining when enough tokens have been collected to unlock a reward.
      5. Data-driven Approach: Use a data-driven approach to store the reward information. This could be a JSON file or a database. This makes it easy to add or modify rewards without changing the code.
      6. UI System: Implement a UI system to display the progress of token collection and the unlocked rewards to the player. The UI system should be separate from the gameplay code to maintain modularity and reduce dependencies.

### Networking

1. This exploit may be possible due to the lack of proper validation or security measures for the game's location updates. When each player updates their location to others, it's essential to have some form of validation to ensure the information being shared is accurate and legitimate. For the client-side protection, we can encrypt every variable that being sent to other players to avoid manipulation and for such as player’s data, we can re-validate through the serve
2. Here are the synchronization using PUN:
   1. Position & Rotation of the vehicle: using PhotonTransformView component on the vehicle GameObject.
   2. Wiper, Rear light, sidelight State: can be achieved by using PunRPC attribute like:

Sebuah gambar berisi teks, software, Software multimedia, cuplikan layar

Deskripsi dibuat secara otomatis

* 1. Horn: can be achieved by using Photon's built-in events such as OnPhotonSerializeView to handle continuous actions like holding the horn button:

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Deskripsi dibuat secara otomatis