Requested Dev Log

After reading the challenge, first things that come to mind are:
- We will need something responsible for creating and managing anchors
- We will need something that will create the room
- We need to handle the start of the app with the intended configuration

Main

For starting the app, and considering the simplicity of the project, we can add a "Main" gameobject to the scene that has the needed references to setup in the Unity inspector. This includes configuration parameters as well as references to prefabs. We could fetch the config parameters from something external, like a json file or a scriptable object, but I think that would be unnecessary for the purposes of the challenge. Same thing with the assets, they could be loaded externally but a simple reference in the inspector is enough.
The responsibility of this Main component is to create instances of the classes responsible for building the room and for the anchor system, while providing them with the necessary parameters.

Room

The class responsible for building the room needs to receive it's dimensions, the center of the room in world position, as well as a prefab representing a wall. It will create 6 walls and scale/rotate them appropriately around the center. It also should provide a method for knowing if a world position is inside the room bounds (since the challenge specifies that no processing should be done outside).

Anchor System

The anchor system class will need to know the anchor range, a reference for the room (to check the out of bounds condition), a reference for the user controller (for it's current position/rotation), and a reference for the anchor prefab.
For the user controller, I use an interface that exposes the methods needed for the anchor system:
- events for the position and rotation changes:
    - this will allow the system to only process when there is an actual change instead of every frame or frames
- the current position
    - this is for any calculation requiring the position of the user, like creating new anchors
- the current look vector
    - for checking if the user is looking at an anchor

This way we can uncouple the controller from the system, as long as it implements what we need.

The anchor creation flow is as follows:

1 - If the user is not currently anchored, create a new anchor at the user position and anchor it, saving the new anchor to a list and saving the index of the new anchor as the nearest anchor to the user
2 - Every time the user moves, check the distance from the nearest anchor, if it's bigger than half the anchor range
    - If so, go through the list of anchors we created and check if there is an anchor closer to the user
        - If there is a nearer anchor, save it's index and set it as the new nearest anchor
3 - If the distance to the nearest anchor is bigger than the anchor range, create a new anchor and set it as the nearest

Instead of iterating a list of anchors to find the nearest, I could do instead a sphere cast from the position of the user. But considering we are working with spatial anchors and real physical rooms, the number of anchors we will create shouldn't be too big. Accessing the Physics engine to do sphere casts will probably be more expensive.
I could also optimise the anchor search by for example separating the anchors by grids and only checking the nearest grids, but again I don't think the impact would be too big considering the number of anchors we would have in a real situation.

To activate the UI of the nearest anchor, I can check every time the user rotates the camera, do a raycast to find the nearest anchor (limiting the distance of the ray with the anchor range). I leave the visual changes to the anchor itself, so I just signal it that the user is currently focused on it.

Anchor

Each anchor needs a reference of the prefab to instantiate and the position. The anchor model will have all the data and the anchor view will be responsible for the visuals. This architecture is more MVP than MVC, since the view will not respond to model changes, but be completely passive. The Anchor class will act as the "presenter", having references to both model and view, and updating both as needed. The Anchor class will also be the only one to have communication with other classes in the system.
Whenever the user is attached to an anchor, I save a reference to the user so I can provide information related to the distance of the user in the UI. I also ask the anchor view to highlight the anchor if the user is anchored and to show the UI if the user is focused on it.

Final considerations

I used interfaces whenever referencing objects that could be interesting to have multiple different implementations (like IRoom, IAnchorView, IAnchorTargetable). The config parameters are directly in the inspector of the Main component because of the simplicity of the project, not because there weren't any other ways to do it. I didn't want to make things too complex, and I wasn't sure about the expectations from the interviewers.
Getting the parameters to absurd values will of course have a big impact on the performance, since the anchor search used simply iterates a list of created anchors. As previously mentioned, I considered the real use case of spatial anchors and didn't think it

would be necessary to optimise this search too much since there isn't a real need to have absurd numbers of anchors.