Progress Report

This is my first time working with VR and multiplayer, so it's like rediscovering video games all over again, much like when I was a child.
I started by testing simple tasks, like synchronizing a cube's position, rotation, and destruction across players to understand the basics of Normcore.
Since I was constantly checking if the object had an owner in the Update method, this caused lag. Then I realized that by subscribing to events like realtime.didConnectToRoom += DidConnectToRoom and realtime.ownerIDSelfDidChange += OnOwnerIDSelfDidChange, I could handle ownership changes without constantly checking in Update. For example, when the owner disconnects, the ownership ID is set to -1. I then use it to detect when ownership changes, eliminating the need to check every frame in Update.
Since it's a multiplayer game, I began by creating different spawning points for each player. I overcame this challenge by using each player's unique ID to ensure they don't start at the same point. For example, assuming the starting points are in an array, using startingPoints[myID] would ensure that each player has a different starting point.
For the AI, I decided to use a NavMeshAgent. To ensure position synchronization across all players, the owner sets the destination using agent.SetDestination(destination), while non-owners use the Warp() method to align with the owner's position. This approach works because the agent has the RealtimeTransform component, and by calling Warp() with transform.position, the agent is instantly 'teleported' to the correct position on each player's client.
To sync the animation of the enemy, I learned the concept of RealtimeModel. I added a property to the model, like this: [RealtimeProperty(3, true, true)] private string _currentAnimation;, which automatically synchronizes the animation state across all players.
The owner of the enemy object is responsible for setting the animation (e.g., SetAnimation("run")). All other players (non-owners) receive the updated animation through the currentAnimationDidChange event. This event is triggered whenever the animation changes on the owner’s side, allowing non-owners to update their local animator to reflect the current animation state of the owner.
I used the same concept for the flashlight. When a player places their hand over their head and presses the grip button, it synchronizes the flashlight state across all players, allowing everyone to see whether the flashlight is on or off.
In regards to map creation, I struggled a lot since I'm more of a programmer. I used Probuilder to create a basic map.
Since I lacked experience with multiplayer and Normcore, I spent a lot of time on small details, which made it slower and more difficult to focus on the horror aspect, which is my favorite part.
One example of a challenge that took longer than expected was parenting the key to the player’s 'pocket.' Since the key’s scale was being resized locally when parenting, but the other players weren’t parenting it, the key’s scale would change incorrectly on their end, causing it to appear oversized. The solution was simply to use Realtime Transform and untick the scale sync option, which fixed the issue. If I started now, I could resolve this and other challenges much more efficiently, but the learning curve was slow.
I believe I would have been able to complete the project more quickly if I had previous experience with Normcore and networked multiplayer systems, but now I have a foundation on VR and multiplayer.

Notes:
I was excited that the project was horror-related, especially since I had recently been recreating the AI from some old *Silent Hill* games. I managed to achieve something quite similar to what the project proposed in just four days, if you compare it to the previous video, but I struggled with this project due to the learning curve involved with multiplayer and VR. Because of that I left the horror aspect until the very end. https://www.youtube.com/watch?v=lQP460nJaA4