Implementation

- I tried to build the game as modular as possible. Each content slide (called DanceClip) consists of a visual element, with a background set at startup, and a CharacterRenderTexture as a child (created and set at runtime).
- We have a ‘Number of Dance Clips’ value, customizable from the inspector, which tells how many cameras, characters and DanceClips we need to spawn. We have one array for each of the following: RenderTexture (this is where the character is rendered), DanceClip background (Image type), CharacterPrefabs (GameObject type), layer names (string type).
- When the game starts, we spawn a character, a camera (positioned to look at the character), a Visual Element (DanceClip) and another Visual Element (CharacterRenderTexture, set as a child of DanceClip). We set the proper values for each of these - the layer on each character, the same layer on the character’s corresponding camera and the RenderTexture corresponding to the character on both the camera and the CharacterRenderTexture Visual Element. All of this happens in the DanceClipSpawner class.
- We also have a script for smooth scrolling and snapping (DanceClipMovement.cs) in which we handle the Input events and making sure the user can smoothly scroll through the Dance Clips and the snapping happens.
- The last script which I made is DanceClipHeightAdjuster.cs. Since the DanceClip is a child of a ScrollView, it automatically scales with ‘unity-content-container’, not with the ‘unity-content-viewport’. This makes sure that the DanceClip properly scales to the whole screen size.
- The UI Hierarchy for UI Toolkit looks like this:
  We have a ParentContainer (Visual Element), holding a ContentContainer (Scroll View) and a BannerContainer (Visual Element).
  BannerContainer is a fixed banner at the bottom edge of the screen, taking 10% of the screen size, containing three buttons (same as the demo in the video).
  ContentContainer is a ScrollView containing the content slides (DanceClips), which are spawned at startup.
- P.S: I tried to add as many comments as possible (where it made sense) and tried to give some details about the ‘why’ or the ‘what’ of a line of code.

Troubles

- Like always, it’s annoying to use Unity’s default scroll view, you have to write basic functionality yourself - I consider smooth scrolling & snapping to be basic functionality.
- In the majority of projects I’ve used uGUI instead of UI Toolkit, so some things weren’t as natural for me to do as someone who mostly uses UI Toolkit.
  I didn’t find a better way to have a Visual Element scaled to the full size of the content-viewport of my ScrollView other than writing code.
- From my research, it seems there are some things not implemented yet: you don’t have button text scaling with %, you either use pixels or depend on the ScaleMode from Panel Settings asset.
- I also didn’t want to use external assets. I know there are some assets which improve ScrollRect/ ScrollView a lot and I’ve used them in the past, but I wasn’t sure if that is

the point of this technical test. I also think it's usually better (not always, of course, it's always a compromise) to write your own code rather than depend on a lot of different assets.
- It's not as easy to debug the UI Toolkit compared to uGUI, you rely a lot on the Debug.Logs. Sometimes it's tricky to see if it's a sizing issue or a rendering issue.

Improvements

- Move all the string variables containing visual elements names into a dedicated ScriptableObjects data holder, or somewhere else.
- Implement object pooling for the DanceClips and characters/ cameras, it's really inefficient to spawn dozens of them.
- See if it's really needed to use a coroutine for adjusting the height of the dance clips or not.

Final thoughts

- This wasn't really in the 'Submit your solution' step, but I wanted to add a couple of things:
- First, I really enjoyed doing this task - I always like challenges, and since I didn't use a lot of UI Toolkit before it was really fun solving the problems it brings. I know I complained a bit about it, but hey, that's what new features are for, haha.
- Second, congrats to the game designer - I've taken a few interview tasks in my six years of Unity and they are boring usually, this one was great!
- P.S: Forgive my really ugly characters and backgrounds I used, I don't like them either, but hey - first come first serve.