Controls:

Push off / Go forward - LShift

Break - S

Steer Left / Right - A / D

Jump - Space

Thought process:

I received the task and I decided to spend time proofreading and noting the requirements to make sure I fully understand what exactly is asked of me and what are the priorities in order to make proper time estimates.

I've spent some time thinking about the project as a whole, thinking of what is the MVP that would be fun and what are all the things that could take time before making estimates and dividing the remaining time. I don't want to start working for the sake of working without the proper preparation and certainty that I'm making the most out of my time.

Noted down the bases of the design and of the implementation plan. I will use this instead of the task document which should improve the workflow.

Now that I have a clear understanding of the project, I'm creating and estimating tasks. Decided on using the game jam model of :

- Day 1 for ideas, planning and setting up the project
- Day 2 for building most of the mechanics and systems
- Day 3 as a cushion for bugs, polish and building

While searching for assets I realized that I would want snappy jump and turn controls to fit the quick gameplay loop. Realistic assets would make that look like an "asset flip" in today's climate, so I decided to use more stylized stuff to make it feel more indy, while still trying to get a "retro" feel, like a modern Jet Set Radio

Organized the folder structure based on how I think the project will look. If it was my own 48h long project I would have kept everything in the same folder, but I want to try and make it extendable as I will be evaluated.

I'm investing some extra time into exposing the right movement parameters so I can get the feel correct quickly once I start testing the level.

I’m starting to see myself having to make shortcuts in clean code in order to make a fun product in time. My understanding is that in a 2 day project the decision will have to be made. I’m willing to make tradeoffs in clean architecture because I value the product more and I will live with consequences.

Tasks / Time management:

Day 1

Task	Description	Time estimated	Time spent
Task analysis			1h
Design	Includes ideating solutions, spotting potential blockers and time sinks…		2h
Plannig	Tasks / Estimates	2h	1h
Initial setup	Blank project, git, folder structure…	2h	3h
Assets search and creation		2 - 3h	0.3h

Day 2

Task	Description	Time estimated	Time spent
Movement	Basic bp editable move, push, break and jump	3h	4h
Obstacles	I will never be able to explain this bug	1h	5h
Scoring system		2h	0.3h
Animations		2h	4h

UI		2h	2h
LD and playtesting		3h	3h

Day 3

Task	Description	Time estimated	Time spent
Bugfixing	Any remaining bugs	2h	2h
Polish	Anims, LD, skipped /features…	2h	?
SFX and VFX	Background music and ride / land sounds, sparks	1h	/
Build & Ship		2h	1h
Margin of error		4h	

These are not exact metrics, so somewhere around 25 hours

Initial Design:

Movement:

Push:
Movement forward will add velocity to the capsule. Max speed, turn rate and jump height will be blueprint editable for easier design and testing.
W + Shift will be the push input
Releasing ***W*** will make the character lose speed more quickly then when holding it

Turning:
A and ***D*** - snappy turning

Breaking:
Holding ***S*** will make the player lose a lot of speed quickly (if no mixamo anim, I will try to use half of the reversed push off animation)

Jumping:

Space button to jump. Space can be held to increase jump height - this is very important for adding variety in level design and allowing freedom of movement and expression (even if the animations don't end up show the "loading" of the jump in the polish phase)

Obstacles:

Trigger above them to add points for jumping over
Collision on top to allow riding (also adds points)

Scoring and UI:

Jumping over an obstacle increases the score

Hitting an obstacle mid-jump / landing on it increases score and adds a multiplier
Riding on an obstacle increases score per second and increases the multiplier

Hitting 0 Velocity or not doing any tricks for x(3?) seconds adds the current score to the total and resets the multiplier

Having more velocity doesn't add to the score, but it can help in keeping the multiplier going by being able to get to the next obstacle faster. This makes it so that the game doesn't push the player to gain and maintain speed, but makes it a tool to allow for even more player expression, just like holding the jump button.

time total score

active score x multiplier

SPEED

Animations:

Basic push-off animation

Flashy jump animation - It has to be flashy as the jump is the only trick mechanic so it shouldn't look bland. If it was a simple jump it would create dissonance as the player is doing cool and hard stuff balancing while the character is just simply jumping.

**(polish stage only)* Lean sideways steering animation
**(polish stage only)* Breaking with the tail of the board

Level Design:

The player has x(1?) minute(s) to get the highest possible score.

The park is concise (as per requirements).
Obstacles are grouped in some places, while few and far between in others to make some places easier to start combos, but also make it a challenge to keep.
The player spawns in a part of the skate park with fewer obstacles, facing a sole ramp facing sideways, prompting him to jump it and then lead onto forward facing ramps / boxes prompting him to slide on them which should teach him all the basics and from then on out he is in a sandbox created for flow and quick decisions.

Questions / Concerns :

UI -

- How do you make a UI fully through C++?
- Too simple UI/UX/SFX/VFX will take away from the experience.

LD-

- How do I introduce verticality without making it difficult to go uphill?

Jump-

- There is a potential problem of a jump ending before the animation ends so that edge case needs to be solved
- How is holding jump while flying handled?
- Coyote time?
- Do I make the jump off of the obstacle itself give points?
- Does jumping back and forth on the same obstacle give points?
  -Could be solved in LD

- Turning while jumping?
- Direction change on jump press?

System explanation:

Movement

I decided on going with a more physics-like solution in order to simulate the dragging motion and feel of skateboarding. I first made sure I had a good push-off feel before doing anything else. Adding impulse makes the snappiest result so I went with that.

I utilized a timer to force a player to hold push for a moment to give more weight to the mechanic.

The result of relying on that made me have to make some hacks to get turning and jumping to sync correctly, but since I knew those mechanics would be simple I solved them easily by introducing checks for outlier behavior.

So it's a combination of some pieces of the movement component with bases in physics-style motion.

Animations

Sped up and cut them to make it snappy. Used a delegate to let the Anim BP know if he should be playing the push off animation. Used casting as for AnimBP it won't matter if it has a reference to the player character.

Scoring

Created a manager to handle all the scoring mechanics. Made it a child to game mode as I thought it would be easy to access everywhere, but that introduced problems. Another mistake here was that it was AActor instead of UActorComponent.

This script was a big mistake in development.

Obstacles

This was the biggest time saver in development. I realized that the only time a point is given is when the player jumps over something, and it can be any obstacle. On such a small map, it took me no more than a few minutes to spread the triggers on top of all the jumpable obstacles and it required almost no coding.

UI

Most simple representation of what is inside the scoring manager. This is where lack of time hit me and in the end I didn’t even finish the game mode reset.

Personal assessment:

Gameplay: 4/10

While I did have fun playing the game, I can see that the mechanics are not juicy enough for a game heavily based in movement. Taking into account the short time frame however, I think I have successfully shown a glimpse of my vision and game’s potential. I can definitely say I had fun while testing.

Code Architecture: 3/10

While starting on pace, I’ve dropped the quality of the code and adherence to good practices as time went on. I decided to put less emphasis on code structure as within such a short time frame something has to take a hit and I knew it’s not going to cost me too much since it’s a 48h project.

Planning and Design 7/10

I think the game can already be fun, but I think with more time it has the potential to be really solid. I’m happy with the vision and how it slowly came to be. I think I used my short time okay in order to make what I think would be good, but there were features that should have made it into the game over some of the playtesting and movement tweaking.

Overall I’d say a 5/10. If I take into consideration the time frame, I can’t be too mad at myself, it was a solid effort.