Robert Sanchez

Jonathan Schultz

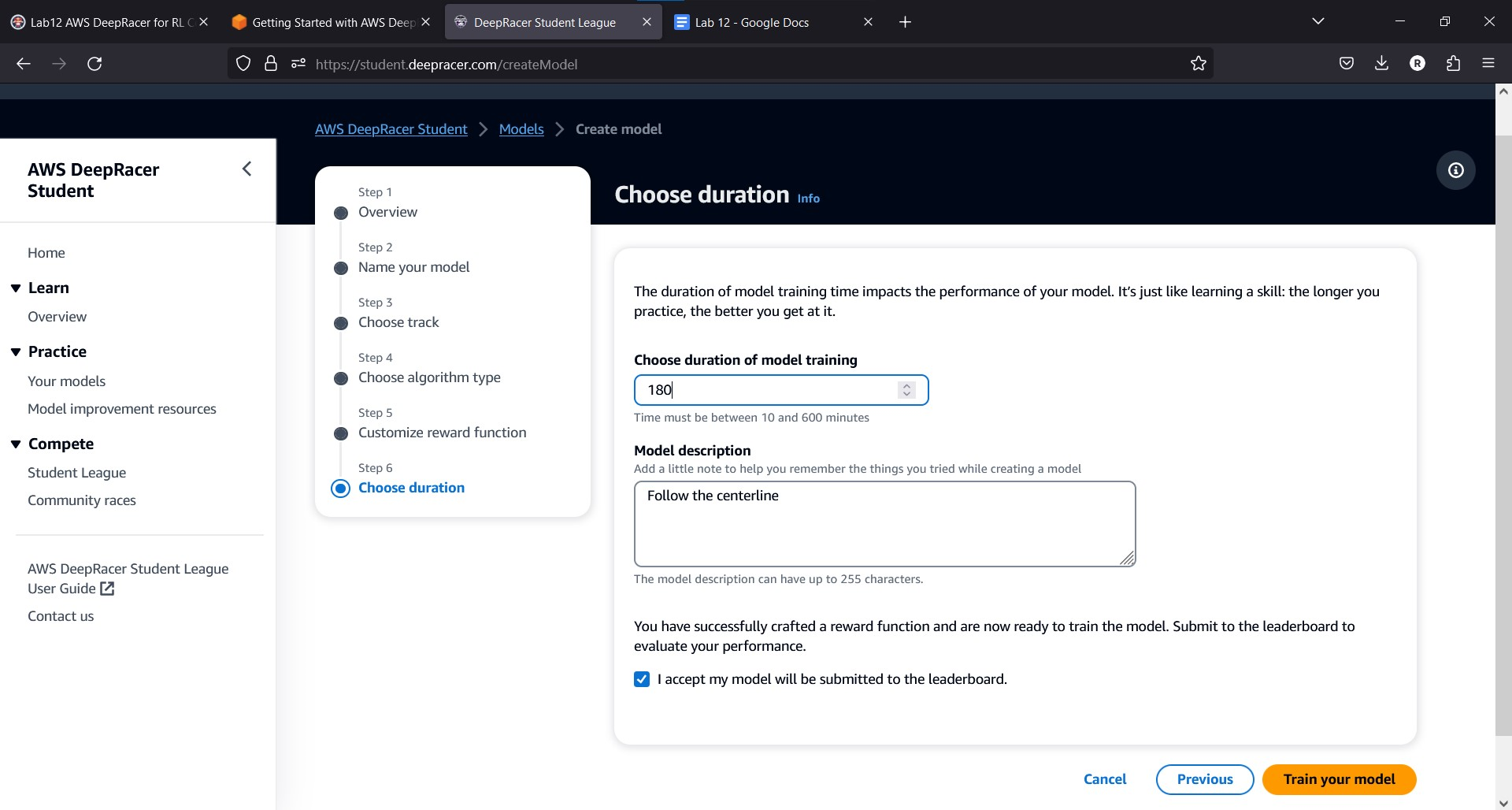
IS 160

Dr. Stephen Choi

Lab 12 AWS DeepRacer for RL Concept

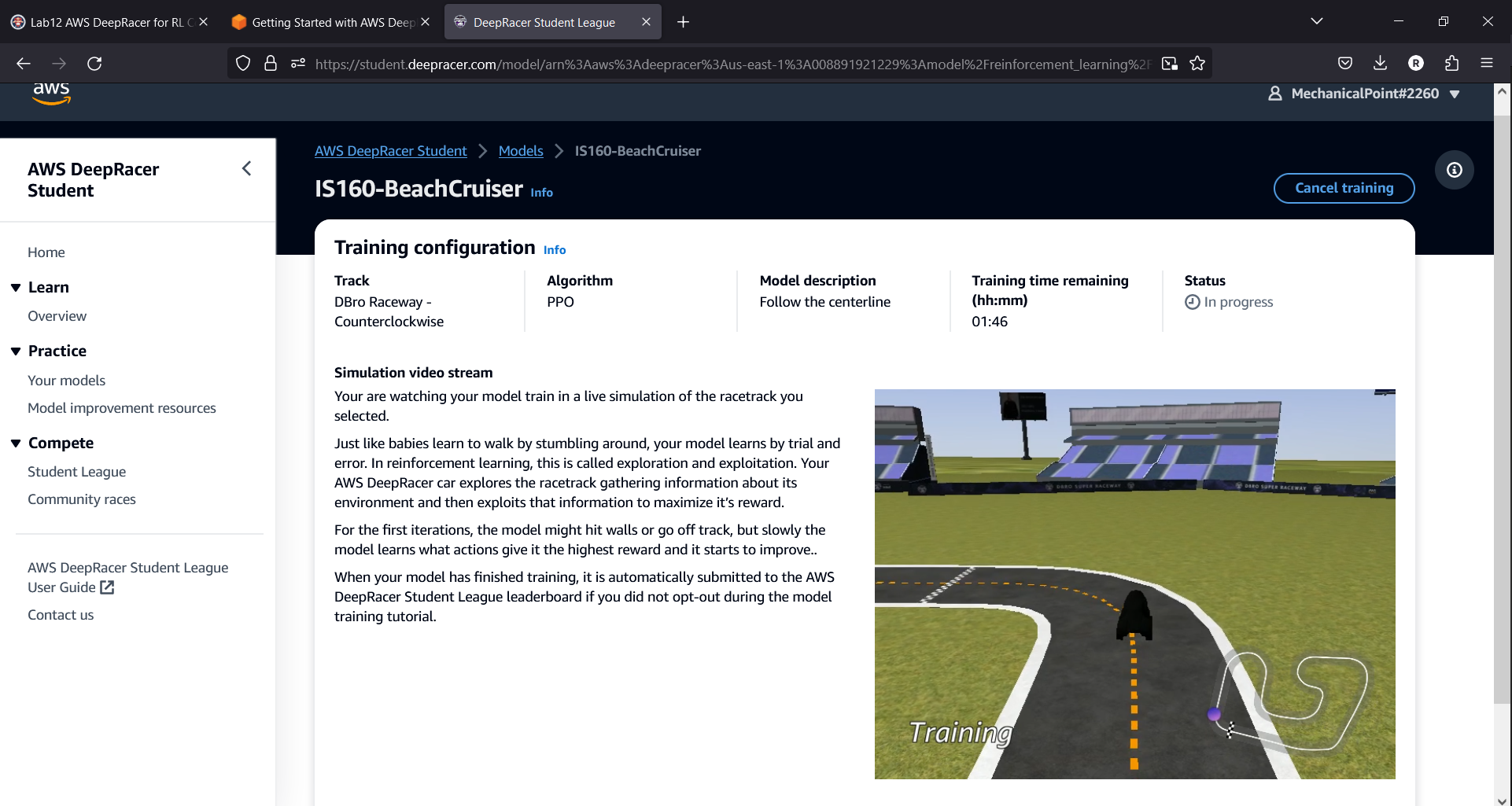
**Model being created**

Choosing the name and amount of time training the model



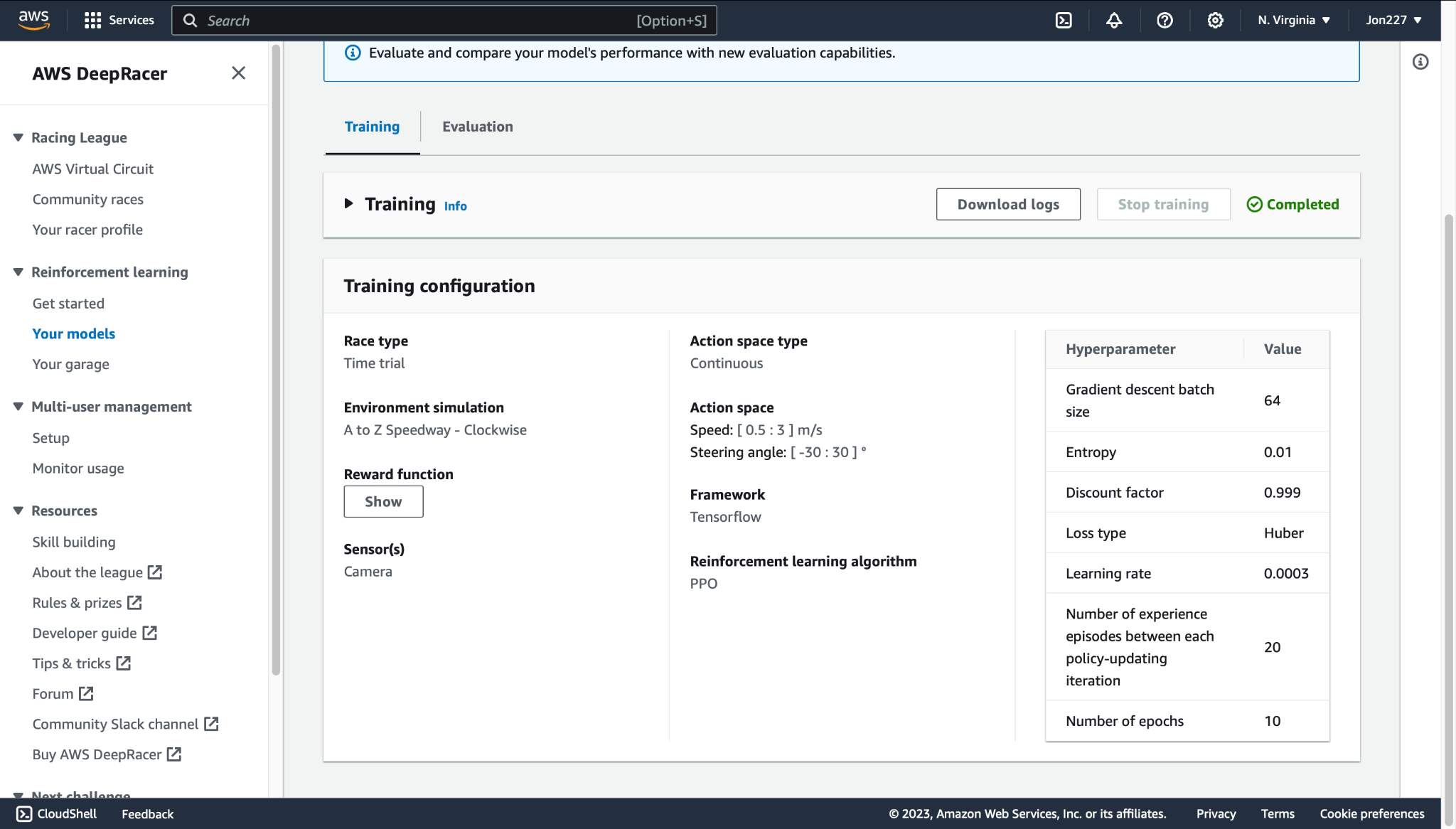
**Model being trained**

Trained model for 3 hours on Dbro Raceway



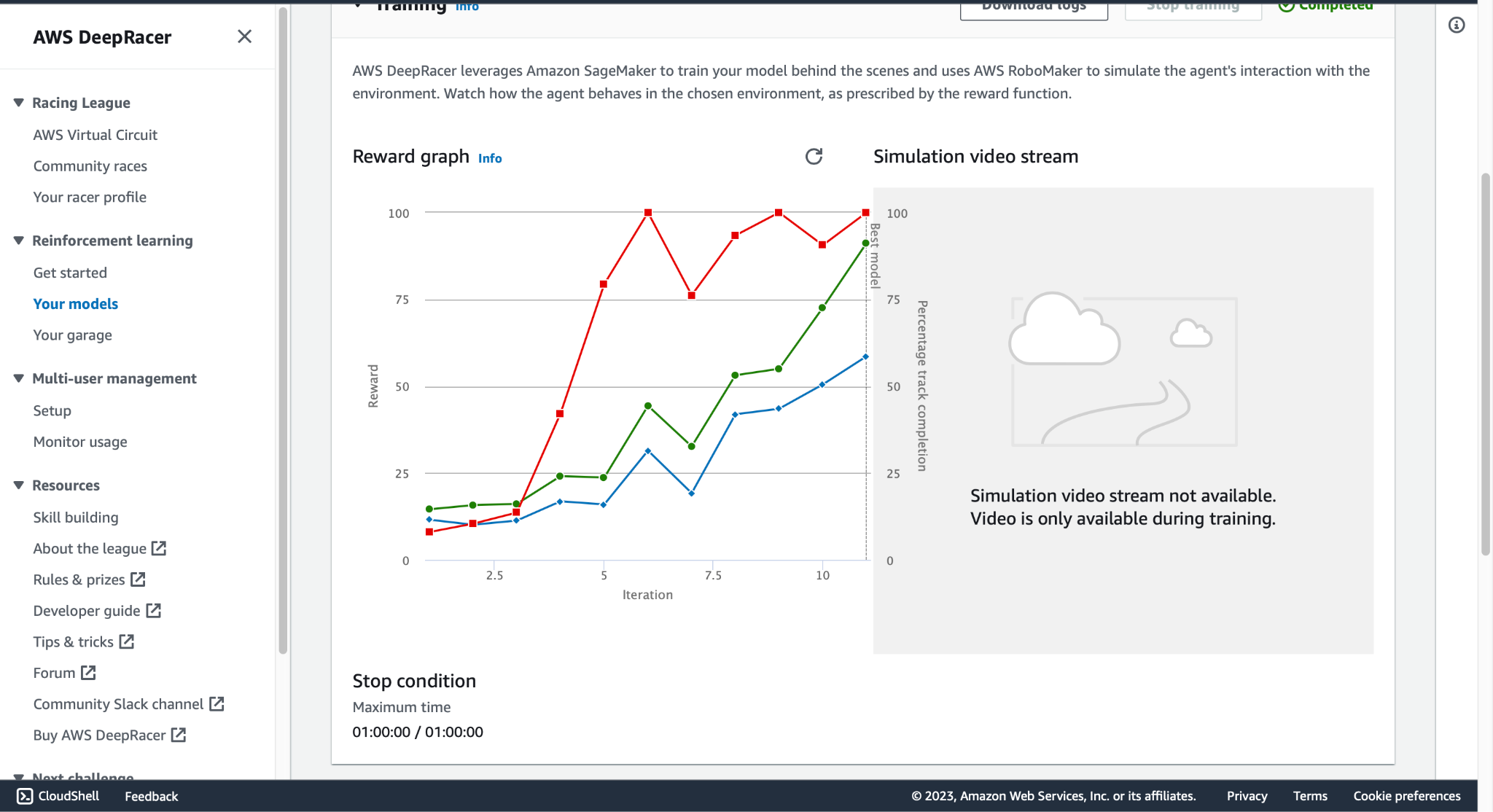
**Model Parameters selected**

Increased speed threshold to 3 from a maximum of 1 while also increasing the steering angle

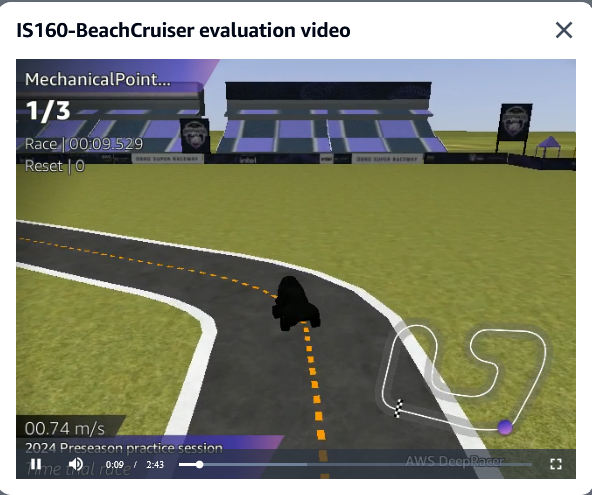
****

**Evaluation of our selected reward function**

It seems our reward function is not very limited and experienced benchmarks quickly but then halted in progression

****

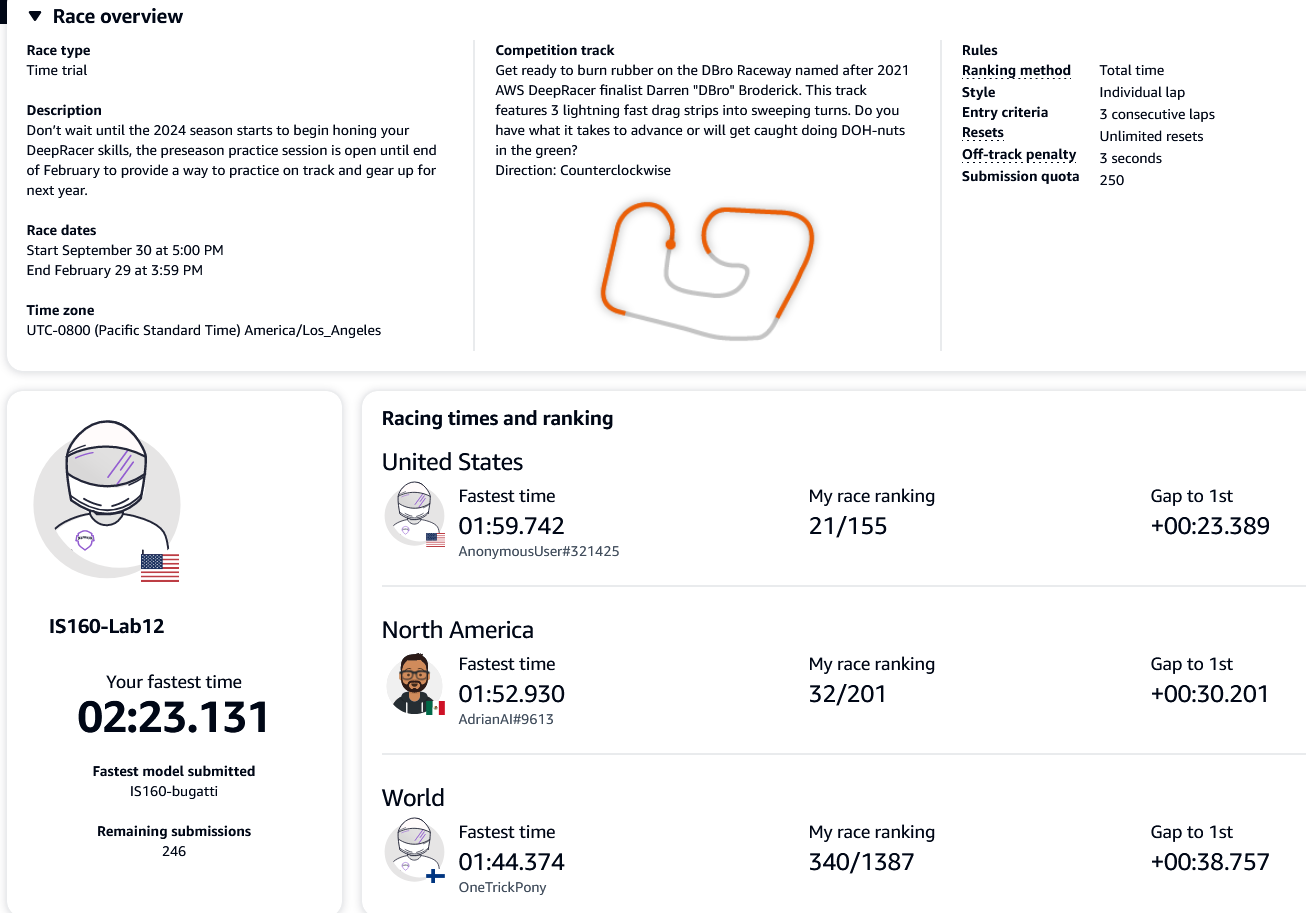
**Model competing in online race**

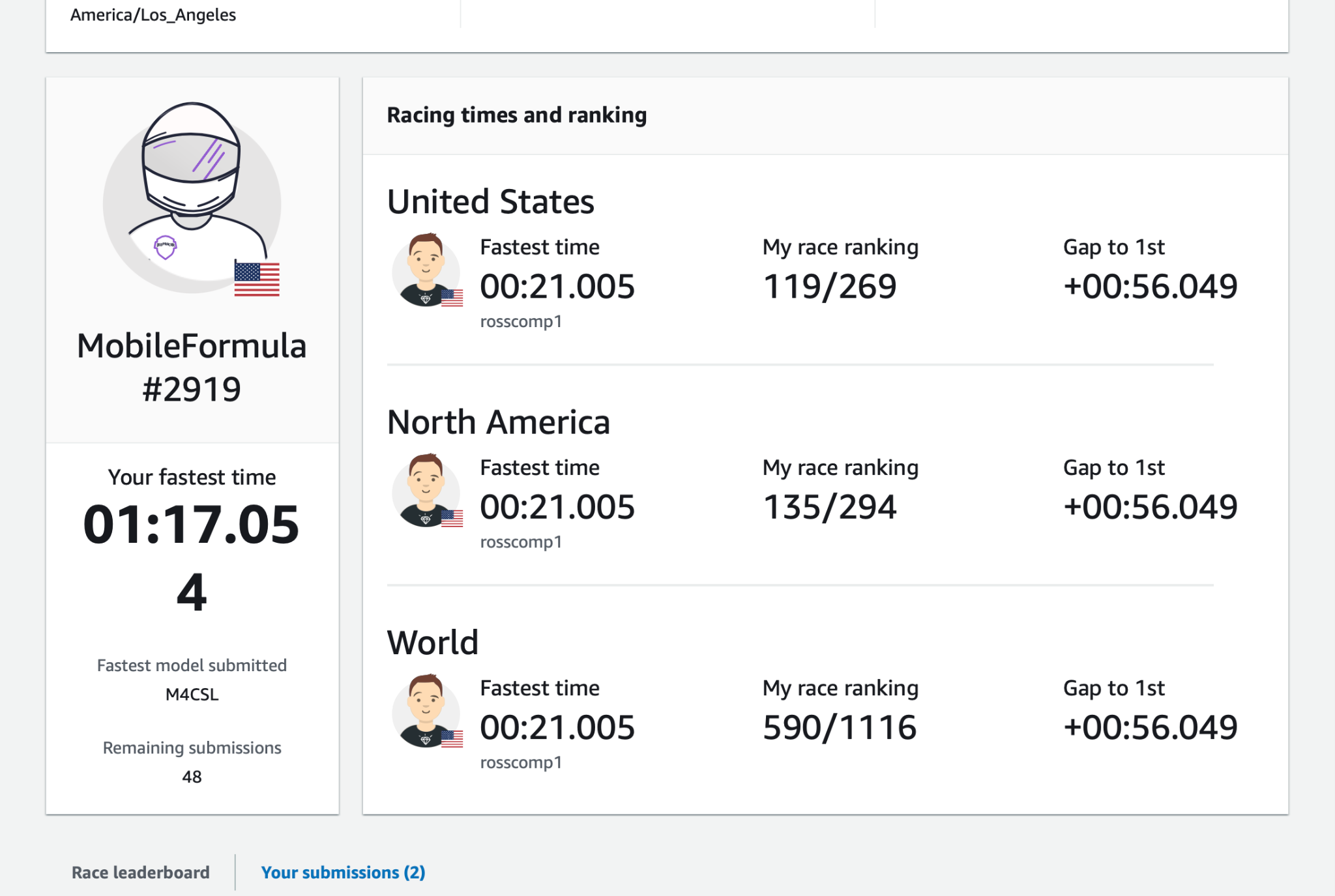
Our model participating in the race on Dbro Raceway  


Second race on A to Z raceway



**Race Overview of Model**





AWS Deep Racer Reinforcement Learning Concept

**Agent**: A small autonomous car designed for reinforcement learning. Operates without human intervention. Adapts behavior through trial and error. Uses sensors to analyze its environment

**Action (a)**: The possible moves/actions that the DeepRacer can take various degrees of accelerating, braking, and steering left or right.

**Environment**: A simulated racing track provided by AWS DeepRacer, where the car navigates through a series of checkpoints to complete a lap. This environment also has its own simulated physics.

**State (s)**: The current situation returned by the environment could include the position of the car on the track, its speed, and the proximity to the next checkpoint.

**Reward**: An immediate return sent back from the environment to evaluate the last action by the DeepRacer. For example, the DeepRacer might receive positive rewards for staying on the track, passing through checkpoints, and completing a lap. Negative rewards could be given for going off-track, colliding with obstacles, or taking longer than a certain time to complete a lap. These rewards are given through functions that can be adjusted to encourage certain behaviors.

**Markov Decision Process** : The decision-making process of the robot can be modeled as a Markov Decision Process. At each state, the robot decides on an action based on its current perception, and the environment responds with a new state and a reward.

**Bellman Equation**: In the context of DeepRacer, the Bellman equation captures the agent's objective to maximize the cumulative expected reward over time by making optimal decisions in each state. The agent learns from experience, updating its value estimates for each state based on the immediate rewards and the expected future rewards.

In the models and tracks that we selected, the DeepRacer's goal is to learn a strategy that maximizes the cumulative reward over time. Through trial and error, it explores different actions in various states, learns from the rewards or penalties received, and adjusts its strategy accordingly. Parameters can be used to be adjusted to change the driving behavior of our models. We had to adjust these parameters to create a more balanced reward function that would not choose short term goals over the long term goal of completing the lap. A reward function that prefers short term goals over the long term goal would typically have a worse completion rate, meaning the model was being too risky with its driving and not braking in time or failing to properly reach the correct steering angle. We also discovered how important it is to select a proper amount of training time. We selected an algorithm that allows the car to select its own optimal path than rather allowing it to follow a pre drawn line. This resulted in the model to really learn by itself and take a lot longer to find the optimal race lines to maximize speed through the corners, which would provide a massive advantage compared to the other conventional methods. Doing this really gave us a very well optimized model which we were satisfied with.