National Codathon

Project Title: Three.js Car Collision Game (One-Day Challenge)
Duration- 1 day (8-10 hrs)

1. Objective

You will build a 3D car runner game in a web browser using HTML + JavaScript + Three.js. The game should be playable: a player controls a car that drives forward on a road while avoiding oncoming cars.

Gameplay Flow

1. The player's car starts at the bottom of the road.
2. The car can steer left/right (lane switching) and control speed (accelerate/brake).
3. Enemy cars appear from the top of the road and move toward the player.
4. The player must dodge enemies by changing lanes.
5. If the player collides with an enemy → Game Over.
6. A score is shown based on survival time (bonus: add speed factor).
7. After Game Over, the game can be restarted (bonus).

You will:

1. **Render basics:** set up scene, camera, renderer, and simple shapes (cylinder/square).
2. **Model a car:** build a simple car from boxes (body/cabin) and cylinders (wheels).
3. **Build the world:** road plane, green grass/"garden", optional dashed lane texture
4. **Control the car:** keyboard steering (← →) and speed (↑ accelerate, ↓ brake/reverse).
5. **Chase camera:** smooth camera that follows behind the player car.
6. **Enemies & spawning:** add oncoming traffic with lane-based spawning and forward motion.
7. **Collisions:** detect hits using **Box3 (AABB)**; show Game Over.
8. **HUD:** score (survival/time) and speed meter.
9. **Difficulty:** ramp spawn rate/speed over time
10. **Polish:** restart button, headlights (emissive), and optional wheel animation/sound.

No prior CAD knowledge required. **Vanilla HTML + JS + Three.js only.** (You may use lil-gui for simple debug sliders if desired.)

2. Technology Stack & References

1. HTML + Javascript
2. Three js - Docs: https://threejs.org/docs ,
3. Three js Examples Gallery: https://threejs.org/examples, examples/js/controls/OrbitControls.js (optional for free-look testing)
4. IDE: Visual Studio Code (or any editor)
5. Local server: VS Code Live Server (or any simple HTTP server)

3. Task Ladder & Points (Total 100)

1. **Level 1 - Setup (10 pts)**
   a. Run the starter project and render a simple scene with a background color.
   b. Add two shapes (e.g., a cylinder and a square) with MeshBasicMaterial to verify rendering works.
2. **Level 2 - Build the Player Car (15 pts)**
   a. Create a Group to hold the player's car parts. Add the following:
      i. Body: BoxGeometry (red).
      ii. Cabin: smaller box (dark gray/blue).
      iii. Wheels: CylinderGeometry, rotated correctly on X-axis, colored black.
   b. Position all parts properly so the car sits just above the road.
3. **Level 3 - Environment (15 pts)**
   a. Add a road using PlaneGeometry (dark gray).
   b. Add two grass planes (green) on either side.
   c. *Optional:* Create a dashed center line using a CanvasTexture (no external files needed).
   d. Ensure the road can receive shadows.
4. **Level 4 - Camera Movement (10 pts)**
   a. Implement a chase camera that follows behind the car smoothly (use lerp).
   b. The camera should always look slightly ahead of the car's direction.
5. **Level 5 - Car Controls (10 pts)**
   a. Add keyboard controls:
      i. ← / → to steer left and right.
      ii. ↑ to accelerate, ↓ to brake or reverse.
   b. Clamp the car's X-position so it cannot leave the road.
   c. Implement basic physics: acceleration, friction, and min/max speed.
6. **Level 6 - Enemy Cars (15 pts)**
   a. Write a function makeEnemyCar() that builds an enemy car.
   b. Face it toward the player (rotateY(Math.PI)).
   c. Spawn enemies randomly in three lanes (left / center / right).
   d. Give each enemy a random non-red color.
   e. Move enemies forward each frame toward the player.
7. **Level 7 - Collision Detection (10 pts)**

   a. Use THREE.Box3 (AABB bounding boxes) for both the player car and enemy cars.
   b. If they intersect → trigger Game Over (pause the game and show an overlay or alert).
8. **Level 8 - HUD: Score & Speed (10 pts)**
   a. Create a simple HUD (HTML overlay) showing:
      i. Score → based on survival time (bonus: multiply by speed).
      ii. Speed → current velocity.
   b. Update HUD values every frame.
9. **Level 9 - Difficulty & Spawning (10 pts)**
   a. Increase spawn frequency or enemy speed as score/time increases.
   b. *Optional:* Give some enemies a slight lane drift.
   c. Ensure speeds remain within a reasonable min/max range.
10. **Level 10 - Final Polish (5 pts + Bonus)**
   a. Add a Game Over screen with the final score and a Restart button.
   b. Give the player's car headlights (emissive material).
   c. Bonus (+5 pts): Animate wheels to rotate while moving.
   d. Bonus (+5 pts): Add basic sound effects (engine hum, collision sound).

4. Judging Criteria

- **Functionality (40%)** – Features completed per the task ladder.
- **Code quality (20%)** – Clear structure (groups/functions), readable, no major bugs.
- **UI/UX (20%)** – Intuitive controls, legible HUD, smooth camera, basic pause/restart flow.
- **Creativity (20%)** – Custom car styling, varied enemy behaviors, extra polish (particles, day/night tint, etc.).

5. Submission

- A zip or GitHub repo with your code.
- Include a short README:
  - Controls.
  - Which features work.
  - Any extras you added.

5. Expected Output Video

- Expected output car game three js.mp4