# 🚀 The Great Galactic Delivery Race – Backend Challenge (C#)

**Welcome to SpaceTruckers Inc.**, the fastest growing interplanetary delivery service in the galaxy. Our fleet moves cargo between planets, moons, and space stations. Each delivery involves skilled drivers, unique vehicles, and complex routes across dangerous space lanes.

We'd like you to design and build a backend system that helps us run "The Great Galactic Delivery Race."

## 🪩 What's Going On?

Every delivery in our universe has a few key ingredients:

- 👨‍🚀 **Driver** – the fearless pilot in charge of the vehicle
- 🚚 **Vehicle** – anything from a HoverTruck to a RocketVan or SpaceCycle
- 🗺️ **Route** – a journey from an origin to a destination, often with checkpoints along the way
- ⚡ **Events** – things that happen during the trip: starting the route, reaching checkpoints, completing deliveries, or dealing with unexpected incidents

Space is not a calm place. Asteroid fields, cosmic storms, emergency maintenance, and other surprises can happen at any time during a delivery. Operations needs a clear, reliable way to see what's going on and how each trip is progressing.

## 🎯 Your Mission

Create a backend application in C# that can:

- Manage basic information about drivers, vehicles, and routes
- Create and track deliveries (trips) as they move through their routes
- Record events that occur during a delivery
- Produce a useful summary once a trip is complete

The system should support multiple deliveries happening at the same time and give us a clear sense of what happened during each trip, including any "interesting" moments along the way.

💡 **Note:** We're intentionally not defining every detail for you. A key part of this exercise is how you interpret the problem, shape the domain, and design the solution.

## 💎 Hints About What We Care About

We're not asking you to use any specific framework or library, but as senior backend engineers, we value solutions that:

- 🔄 **Use event-driven ideas** to handle and react to trip events
- 🏗️ **Reflect domain-driven thinking** in how the core concepts (drivers, vehicles, routes, events, trips) are modeled
- 🧪 **Are supported by tests** (ideally written in a test-first or test-friendly way)
- 🏛️ **Follow clean architecture principles** that separate domain logic from technical details
- ✨ **Apply clean code and SOLID principles** so the code is readable, maintainable, and easy to reason about
- 🔗 **Are loosely coupled**, making it easier to extend or swap out parts over time
- ⚡ **Show awareness of concurrency** when multiple trips are active at once

## 📦 Deliverables and How to Share Your Work

Please upload your solution to a **public GitHub repository** that we can clone and review.

Include all required dependencies so that we can build and run the solution on our side.

### 📋 Add a README that explains:

- How to set up and build the solution
- How to run the application
- How to run tests (if you've added them)
- Any assumptions you've made
- Any key design decisions you think are worth calling out

## ⏰ Timing

We don't want you to rush, and we understand you may have other commitments. However, we do ask that you submit your completed solution within **1 week** from the time you receive this challenge.

## 🔍 What Happens Next

We will:

- **Clone and run** your solution
- **Review** the code, structure, and tests
- **Use your implementation** as the basis for a technical conversation in the interview, where we'll discuss your design choices, trade-offs, and how you might evolve the system further

## 🌟 Your Challenge

**Build a backend system that SpaceTruckers Inc. would be confident using to run our operations — solid, flexible, and ready for the hazards of deep space.**

🚀 *Good luck, and may the force be with your code!*