🏎️ F1 Interactive Telemetry & Data Lab

I built this project to dive deep into Formula 1 data analysis, moving beyond standard TV broadcasts to understand driver performance through data-driven insights[cite: 4]. This lab has evolved from a collection of static scripts into a **fully interactive Web Application** designed for high-performance telemetry visualization[cite: 12].

🌟 Interactive Features

The lab is built using **Streamlit**, providing a reactive interface where data updates instantly based on your selections.

1. Master Telemetry Dashboard (5-Panel)

The primary analysis tool. It synchronizes the **Time Gap** with every driver input (Speed, Throttle, Brake, and Gear) and geolocates them using turn markers across the entire lap[cite: 12, 29].

- **Real-time Gap Calculation**: Visualizes exactly where a car gains or loses time using custom delta calculations[cite: 10, 12].
- **Dynamic Lap Selection**: In Race mode, users can select specific laps to compare performance consistency or specific incidents[cite: 28, 93].

2. Technical Deep Dive (Interactive Zoom)

Allows for "surgical" analysis of specific sectors[cite: 40].

- **Distance Sliders**: Use the interactive sidebar to focus on specific track segments (e.g., 1500m - 3500m) to see "micro" differences in throttle application and braking points[cite: 41, 42, 60].

3. Track Performance Heatmaps

Projects telemetry data directly onto circuit coordinates (X, Y)[cite: 13, 14].

- **Speed Delta Maps**: Uses divergent colormaps to highlight where one driver is faster (Red) vs. their rival (Blue) along the physical track layout[cite: 14, 73, 74].
- **Absolute Speed Heatmap**: Identifies apex speeds and heavy braking zones[cite: 13, 76].

4. Race Pace & Tyre Strategy

A race pace analyzer that filters data to visualize thermal degradation and lap time consistency[cite: 9, 78].

- **Automated Cleaning**: Uses `pick_quicklaps` to remove pit stops and non-representative laps (Safety Car, Out laps) for a clean "performance cliff" visualization[cite: 18, 93].

🛠️ Technical Problems Solved

1. **Data Inconsistency & Sync**: Sensor sampling rates often differ. I implemented **Linear Interpolation (NumPy)** to create a common distance grid for a 1:1 "apples to apples" comparison[cite: 16, 17].

2. **Dynamic Circuit Mapping**: Automated corner coordinate fetching and labeling (T1, T2...) via the FastF1 API ensures the lab works for any Grand Prix on the calendar[cite: 22, 29].
3. **Optimized Performance**: Implemented a **Local Caching System** to handle massive telemetry datasets and reduce reload times significantly[cite: 21].
4. **Professional Branding**: Automated official Team Color extraction via API for consistent, high-quality visuals[cite: 20].

🚀 How to Run

Tech Stack

- **Language**: Python 3.12 [cite: 108]
- **Core Libraries**: Streamlit, FastF1, Pandas, NumPy, Matplotlib [cite: 109]

Installation

1. Clone the repository:

```
git clone [https://github.com/your-username/f1-telemetry-lab.git]
(https://github.com/your-username/f1-telemetry-lab.git)
```