IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Formula 1 Race Data Visualization and Performance Analysis Using FastF1

GUNTUPALLI BHUVASYA[1], KODIPYAKA LIKITH[2], LAKKAKULA VAMSHI RAJ[3], AUSHALA ARVIND[4], GADDAM JHANSI RANI[5]

[1,2,3,4]Department of Information Technology, J.B. Institute of Engineering & Technology, Hyderabad

[5]Assistant Professor, Department of Information Technology, J.B. Institute of Engineering & Technology, Hyderabad

***Abstract*—**Formula 1 (F1) motorsport operates at the intersec- tion of high-speed engineering and data-driven decision making, where fractions of a second determine competitive outcomes. Despite the existence of publicly accessible telemetry libraries such as FastF1 and structured APIs like the Ergast Developer API, the raw data they expose remains largely inaccessible to students, independent analysts, and motorsport enthusiasts due to the technical complexity involved in extraction and interpretation. This paper presents the design and development of an interactive web-based platform that consolidates F1 telemetry and race data into a structured analytical dashboard. The system integrates data pipelines built on FastF1 and Ergast, performs preprocessing to normalize lap times, telemetry channels, and sector timings, and exposes processed results through a responsive frontend built with ReactJS. Core analytical features include lap-time differential computation, multi-driver telemetry overlays, speed- versus-distance profiles, sector performance comparisons, and consistency metrics derived from statistical measures of lap- time variance. The architecture follows a modular layered design encompassing an API controller, data processor, database storage layer, and visualization module. Evaluation confirms that the platform accurately renders historical race sessions and provides meaningful performance insights in an intuitive visual format. The work demonstrates practical application of data engineering, dynamic visualization, and web application development within the domain of real-world motorsport analytics.

***Index Terms*—**Formula 1, FastF1, Telemetry Analysis, Race Data Visualization, Lap Time Comparison, Driver Performance, Ergast API, Web Dashboard, Data Preprocessing, Interactive Charts

I. INTRODUCTION

Formula One racing stands apart from virtually every other sport in the world due to the sheer volume and granularity of data generated during each competitive event. Every car on the grid produces continuous streams of telemetry covering speed, throttle position, brake pressure, gear selection, tyre compound state, fuel load variation, aerodynamic forces, and GPS-based positional tracking [5]. Modern F1 teams employ dedicated data science departments that process these signals in near real-time to advise on race strategy, car setup changes be- tween sessions, and driver coaching. The competitive margins involved are so tight that decisions informed by even minor statistical improvements in lap time or tyre management can decide championship outcomes.

However, this level of analytical capability has historically been confined to the constructor teams themselves. While broadcasting platforms provide surface-level statistics such as race position and fastest lap times, they offer virtually no access to the underlying telemetry that engineers use. For engineering students, data science researchers, and motivated motorsport fans, this creates a significant gap. The datasets do exist in the public domain through community-driven efforts. The Ergast Developer API provides structured access to historical race results, standings, lap time records, and pit stop timings dating back decades. The FastF1 Python library extends this further by offering lap-level telemetry data including detailed speed traces, DRS activation windows, and sector-by-sector breakdowns for sessions from recent seasons [6].

The challenge is that consuming these data sources mean- ingfully requires writing custom Python scripts, understanding data alignment issues across telemetry channels sampled at dif- ferent frequencies, and then constructing visualizations from scratch. This technical barrier discourages a large potential audience from engaging with F1 data at a deeper level. A unified, user-friendly platform that handles data retrieval, preprocessing, and visualization transparently would serve a meaningful educational and analytical purpose [9].

This paper describes the development of exactly such a plat- form. The system presented here is a web-based F1 analytics application that integrates the Ergast API and FastF1 library into a single backend pipeline, processes the incoming data to compute performance metrics, and presents results through interactive charts rendered in a ReactJS-based frontend. Users can select any race session from supported seasons, choose drivers of interest, and compare their performance through multiple visualization modes including lap-time trend charts, speed-versus-distance overlays, throttle and brake comparison traces, and sector timing breakdowns.

The technical contributions of this work include a modular backend architecture combining Flask-based API routing with a Python data processing layer, a structured database schema linking driver, team, race, lap, and telemetry entities, a set of mathematical performance metrics derived from raw teleme- try, and a responsive frontend visualization suite. Section II reviews related work and establishes the context for this project. Section III defines the problem this system addresses. Section IV states the objectives. Section V describes existing systems and their limitations. Section VI presents the proposed

system and its design. Section VII details the methodology including mathematical formulations. Section VIII covers implementation specifics. Section IX presents results and discussion. Section X concludes and outlines future directions.

II. Literature Survey

The analytical transformation of Formula One from a purely mechanical sport to a data-intensive competitive environment has been well-documented across industry commentary, academic research, and open-source software development. This section reviews the key works that directly inform the design and motivation of the present system.

Industry analysts at Catapult Sports [5] describe how modern F1 operations use telemetry pipelines to convert continuous sensor streams into actionable strategic inputs. Parameters such as tyre degradation rates, fuel consumption curves, and sector-level lap time deltas are processed during races to advise on pit stop windows and driver pacing instructions. This industry perspective situates the current project within an established real-world trend, demonstrating that telemetry-driven analysis genuinely determines competitive outcomes at the highest level.

Academic engagement with F1 telemetry is illustrated in Struthers' study [9], which examined telemetry analysis techniques specifically within an educational context. The work showed that F1 datasets are well-suited for teaching signal processing concepts, data cleaning procedures, and performance interpretation to engineering students. The pedagogical utility identified by Struthers directly supports the motivation of this project, which targets students and analysts who lack access to professional-grade tools.

The FastF1 library [6], maintained by Oehrly on GitHub and distributed via PyPI [2], provides the primary technical foundation for this project. FastF1 abstracts the complexity of accessing official F1 timing and telemetry data, offering Python functions for loading session objects, retrieving per-lap telemetry channels, and aligning data across drivers. Its reusable session loaders and telemetry parsers make it feasible to build higher-level web applications without implementing low-level data format handling from scratch.

Singh and Patel [8] examined real-time telemetry data analysis in F1, addressing the technical constraints of streaming data ingestion, feature extraction under latency requirements, and noise removal from high-frequency sensor channels. Their work highlighted specific engineering challenges that any analytics platform must address, including synchronization of telemetry channels recorded at different sampling rates and the handling of corrupted or missing data points during transmission. These considerations directly shaped the preprocessing design of the current system.

Tavares et al. [10] contributed to the broader field of real-time motorsport analytics by outlining algorithmic approaches to processing live race data streams. Their IEEE-published work emphasized the importance of low-latency data pipelines and structured feature representations for enabling meaningful comparisons across drivers and laps. The architectural principles they describe are reflected in the backend design adopted in this project.

Community analytical platforms such as Tracing Insights [4] demonstrate public appetite for interactive F1 data exploration. These platforms aggregate lap time tables, telemetry snippets, and race statistics, yet typically offer static pre-computed outputs rather than dynamic, user-driven exploration. This limitation reinforces the need for a platform that allows users to interactively select sessions, drivers, and metrics on demand.

Research published in IJRASET [3] explored reinforcement learning approaches to F1 strategy prediction, using historical race data to train models that recommend pit stop timing and tyre selection. While the current project does not implement predictive modelling, this work establishes a forward path for future enhancement of the described platform, where the visualization layer could eventually surface ML-generated strategic recommendations alongside historical telemetry.

Wright [7] provided a broad engineering perspective on F1 data analysis and race strategy understanding. This work underscored the value of combining historical race records with telemetry overlays to reveal patterns in driver behaviour across circuits, a capability the current platform specifically implements through its multi-driver comparison and lap-time consistency analysis features.

Taken together, the reviewed literature confirms that while data sources, computational tools, and analytical methods for F1 are available in fragmented form, a unified interactive web platform that brings data retrieval, preprocessing, visualization, and performance comparison into a single accessible system represents a meaningful and practical contribution.

III. Problem Statement

Formula One generates substantial volumes of telemetry, timing, and positional data across every practice session, qualifying run, and race weekend. While public sources including the Ergast API and FastF1 library make portions of this data accessible, the information remains scattered across multiple endpoints, requires technical expertise to extract reliably, and is presented in raw formats that are difficult to interpret for users without specialized programming skills. Existing commercial dashboards provide only summary-level insights, and advanced analytics platforms used by teams are entirely proprietary. The result is that students, independent analysts, and motorsport enthusiasts have no access to a consolidated, interactive system for exploring driver performance, comparing lap strategies, or understanding telemetry patterns. This project addresses the absence of such a platform by building a web application that integrates public F1 data sources, applies structured preprocessing and performance metric computation, and delivers results through clear interactive visualizations.

IV. Objectives

The primary objectives of this work are as follows. First, to develop an integrated data acquisition pipeline that retrieves telemetry and timing data from both the Ergast Developer API

and the FastF1 library. Second, to implement a preprocessing layer that cleans, normalizes, and aligns multi-source data into structured analytical formats suitable for comparative analysis. Third, to compute performance metrics including lap-time differentials, speed profiles, and consistency scores, supported by mathematical formulations. Fourth, to design and implement an interactive web-based dashboard enabling multi-driver comparison across sessions. Fifth, to visualize telemetry parameters including speed, throttle position, brake pressure, gear selection, RPM, and DRS activation state in an intuitive and accessible manner.

V. Existing System

Several tools and platforms currently exist that provide partial access to Formula 1 race data and performance statistics. Understanding their capabilities and limitations establishes the motivation for the system proposed in this work.

The official Formula 1 website and its companion mobile application offer live timing, lap count, gap intervals, and simplified sector indicators during race broadcasts. However, these interfaces are designed for entertainment consumption rather than technical analysis. They do not expose raw telemetry channels, do not support driver-to-driver comparison at the data level, and provide no mechanism for historical session exploration beyond curated highlight statistics.

The Ergast Developer API is a widely used community resource that provides structured historical race data in JSON format, including race results, constructor standings, lap time records, pit stop timings, and qualifying results dating back to the 1950 Formula One season. While Ergast provides excellent coverage of structured race metadata, it does not include telemetry-level data such as speed traces, throttle inputs, brake pressure, gear selections, or DRS activation states. Users must write custom scripts to query and process its endpoints, and no visualization layer is provided.

The FastF1 Python library [6] addresses the telemetry gap by offering programmatic access to official F1 timing data and session telemetry for recent seasons. FastF1 provides session loaders, lap selectors, and telemetry channel accessors through a Python API. While powerful for developers, the library has no graphical interface and requires significant programming knowledge to use effectively. Each analysis must be scripted from scratch, including data cleaning, channel alignment, and plot construction.

Community-developed Jupyter notebooks, published on platforms such as GitHub and Kaggle, demonstrate various F1 data analysis techniques using FastF1 and Ergast. These notebooks typically cover single-session analyses or specific comparison scenarios, but they are static documents with fixed driver and session parameters. They cannot be interactively queried by a non-technical user, and they do not persist processed results across sessions.

Commercial motorsport analytics platforms such as those used internally by F1 constructor teams offer comprehensive real-time telemetry dashboards, predictive modelling tools, and strategy simulation environments. However, these systems are entirely proprietary, inaccessible to external users, and not designed for educational or research purposes.

Tracing Insights [4] and similar community websites aggregate selected F1 statistics and publish pre-generated charts covering lap time progressions, tyre stint comparisons, and race pace analyses. While these resources are visually accessible, they offer only fixed, pre-computed outputs. Users cannot select arbitrary sessions, change driver pairs, or adjust metric parameters interactively. The underlying data and methodology are also not transparent or reproducible.

A. Limitations of Existing Systems

The existing landscape of F1 data tools shares several recurring limitations that the proposed system is designed to overcome:

- **Data Fragmentation:** No existing open platform consolidates Ergast race records and FastF1 telemetry into a single queryable system. Users must combine multiple sources manually.
- **High Technical Barrier:** Accessing meaningful telemetry requires Python programming skills, knowledge of data alignment techniques, and familiarity with visualization libraries.
- **Static Outputs:** Community notebooks and pre-generated charts do not support interactive, on-demand exploration. Session and driver parameters are fixed at generation time.
- **No Formal Metric Definitions:** Existing tools apply inconsistent strategies for missing lap handling, telemetry interpolation, and consistency scoring, making results difficult to reproduce or compare.
- **No Integrated Storage:** No existing open platform caches processed session results, meaning repeated analyses require repeated API calls and full reprocessing.
- **Limited Comparative Depth:** Available platforms surface aggregate statistics such as fastest lap or total race time, which conceal the intra-race performance variations that are analytically significant.

VI. Proposed System

The proposed system is a web-based Formula 1 race data visualization and performance analysis platform designed to address each of the limitations identified in the existing system review. The platform consolidates data from the FastF1 Python library and the Ergast Developer API into a unified backend pipeline, applies structured preprocessing and mathematical metric computation, and delivers results through an interactive ReactJS frontend accessible to users without any programming knowledge.

A. System Overview

The proposed platform follows a modular four-layer architecture. The *Data Integration Layer* handles all communication with external F1 data sources, merging Ergast race records and FastF1 telemetry into a unified session dataset using shared driver code and lap number identifiers. The *Data*

Processing Layer applies cleaning, normalization, resampling, and mathematical metric computation to produce structured analytical outputs. The *Storage Layer* persists processed results in a hybrid database comprising MySQL for structured race and lap records and MongoDB for variable-length telemetry documents, enabling fast retrieval for repeat queries. The *Presentation Layer* provides a ReactJS web interface with dynamic, interactive chart components that update in response to user selections without full page reloads.

B. Key Features of the Proposed System

- **Unified Data Integration:** The system integrates FastF1 telemetry and Ergast race records through a single backend API, eliminating the need for users to interact with either source directly.
- **Interactive Session Selection:** Users can select any supported Formula 1 season year, race round, and session type (Practice, Qualifying, or Race) through a browser-based interface.
- **Multi-Driver Comparison:** The platform supports simultaneous comparison of two or more drivers through lap-time differential charts, speed-versus-distance overlays, and sector time bar charts.
- **Mathematical Performance Metrics:** Three formally defined metrics are computed: lap-time differential ΔT_n, consistency score σ, and distance-normalized speed profile d_i, as described in Section VII.
- **Telemetry Visualization:** Speed, throttle, brake, gear, RPM, and DRS channels are rendered as interactive time-series charts with hover tooltips and synchronized axes.
- **Database Caching:** Processed session data is stored after first retrieval, ensuring that subsequent requests for the same session return results significantly faster than uncached requests.
- **No Programming Required:** The entire analytical workflow from session selection to chart rendering is accessible through graphical controls, making the platform usable by students, analysts, and enthusiasts without coding skills.

C. Comparison with Existing Systems

Table I summarizes the capabilities of the proposed system against existing tools across the key dimensions identified in the limitations analysis.

D. Expected Outcomes

The proposed system is expected to produce the following outcomes upon successful implementation. First, a functional web-based dashboard that renders accurate, interactive visualizations of driver performance across any supported historical F1 session. Second, computed performance metrics including lap-time differentials, consistency scores, and speed profiles that match reference values derivable from raw FastF1 and Ergast data. Third, a responsive interface that allows non-technical users to explore and compare F1 telemetry data without writing any code. Fourth, a modular backend architecture that supports future extension with additional data sources, new metric types, or predictive modelling components. Fifth, a documented preprocessing pipeline with formally defined metric formulations that can be reproduced independently by other researchers working with the same data sources.

TABLE I
COMPARISON OF PROPOSED SYSTEM WITH EXISTING TOOLS

Feature	F1 App	Ergast API	FastF1 Notebooks	Proposed System
Telemetry Access	No	No	Yes	Yes
Race Records	Limited	Yes	Partial	Yes
Interactive UI	Limited	No	No	Yes
No Coding Required	Yes	No	No	Yes
Multi-Driver Compare	No	No	Manual	Yes
Database Caching	No	No	No	Yes
Formal Metrics	No	No	Varies	Yes
Historical Sessions	Limited	Yes	Yes	Yes

VII. METHODOLOGY

The methodology follows a sequential pipeline from data acquisition through preprocessing, metric computation, backend integration, and frontend visualization. Each stage is described below.

A. Data Acquisition

Race session data is collected from two complementary sources. The Ergast Developer API supplies structured JSON responses containing race results, driver standings, historical lap times, and pit stop records. The FastF1 Python library supplements this with high-resolution telemetry for recent sessions, including per-sample speed, throttle, brake, gear, RPM, and DRS channel values. A session loader module queries both sources in parallel and merges the results using shared identifiers such as driver code, race round number, and lap number.

B. Data Preprocessing

Raw data from both sources is heterogeneous in format and sampling frequency. The preprocessing stage addresses this through the following operations. Missing lap entries are flagged and excluded from comparative analysis. Timestamps are converted to a uniform elapsed-time format measured in seconds from session start. Telemetry channels sampled at different rates are resampled to a common frequency using linear interpolation, ensuring alignment across drivers for overlay comparisons. Driver identifiers from Ergast and FastF1 are reconciled using a mapping table. The output of preprocessing is a set of normalized time-series DataFrames, one per driver per session, ready for metric computation.

C. Performance Metric Computation

Three primary mathematical metrics are derived from the preprocessed data.

1) Lap-Time Differential: For a given lap n and two drivers A and B, the lap-time differential is:

$$\Delta T_n = T_n^A - T_n^B \quad (1)$$

where T_n^A and T_n^B represent the recorded lap times in seconds for drivers A and B on lap n respectively. A positive ΔT_n indicates driver A was slower on that lap.

2) Lap-Time Consistency Score: Driver consistency over a race stint is quantified as the sample standard deviation of all valid lap times recorded during that stint:

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(T_n - \bar{T})^2} \quad (2)$$

where N is the total number of valid laps in the stint, T_n is the lap time for lap n, and $\bar{T}$ is the mean lap time over the stint. A lower σ value indicates more consistent driving.

3) Speed-Distance Profile Normalization: Telemetry speed values are mapped against track distance rather than elapsed time to enable meaningful inter-driver comparison regardless of absolute lap time differences. For each telemetry sample i, the normalized distance is:

$$d_i = \frac{s_i}{s_{\max}} \times L \quad (3)$$

where s_i is the cumulative track distance at sample i, $s_{\max}$ is the total track length, and L is the circuit lap length in metres. Speed values v_i are then plotted against d_i to produce the speed-versus-distance profile.

D. System Architecture and Backend Design

The backend is implemented in Python using the Flask web framework. It exposes RESTful API endpoints that accept session parameters and driver selections from the frontend. The API Controller routes incoming requests to the API Integrator, which calls FastF1 and Ergast. The Data Processor applies the preprocessing and metric computation steps described above. Processed results are serialized as JSON and stored in a MySQL relational database for caching, with a MongoDB collection used for flexible telemetry document storage. Subsequent requests for the same session retrieve data from the database layer, reducing repeated API calls.

E. Frontend Visualization Layer

The frontend is implemented using ReactJS with dynamic chart rendering. API responses are consumed by visualization components that generate lap-time trend charts, dual-driver speed overlays, throttle and brake comparison traces, and sector time bar charts. User selections for session year, race round, session type (practice, qualifying, or race), and driver identifiers trigger new API requests, causing the visualization components to re-render with updated data without full page reloads.

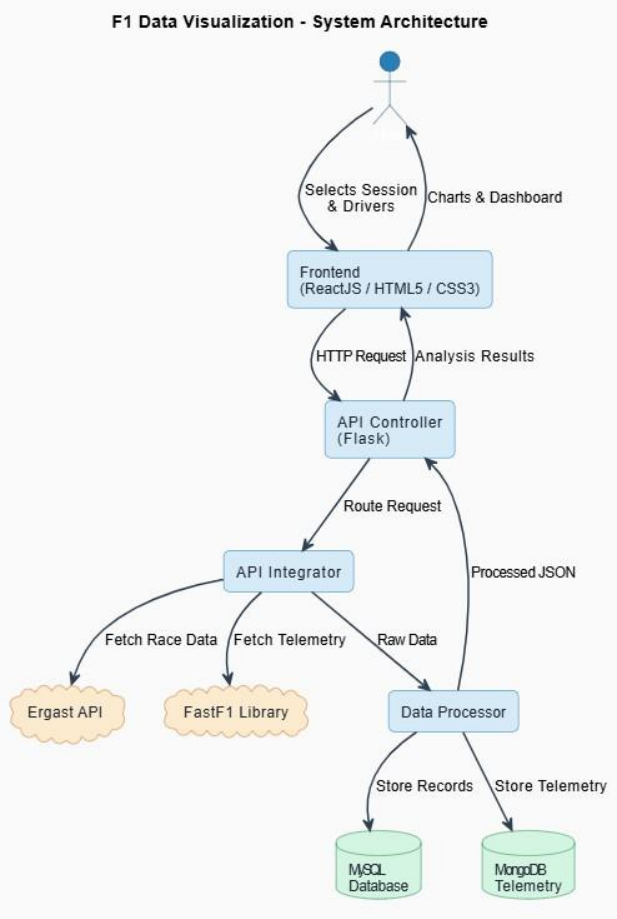

Fig. 1. System Architecture of the F1 Data Visualization Platform

F. Testing and Validation

System testing covers data accuracy verification against known race results, telemetry alignment checks across driver pairs, chart rendering correctness under various session configurations, and response time measurement for cached versus uncached requests. Any discrepancies in telemetry mapping or metric values are corrected in the preprocessing layer before deployment.

VIII. Implementation

The implementation is divided into four interconnected layers: data integration, processing, storage, and presentation.

1) Data Integration Layer: The FastF1 library is used to load session objects by specifying year, race round, and session type. Telemetry for each driver is retrieved using the library's built-in lap selector and telemetry accessor methods. Ergast API calls are made over HTTPS and responses are parsed from JSON into Pandas DataFrames. Both data sources are loaded into memory at session initialization and merged on driver code and lap number to produce a unified session dataset.

2) Processing Layer: A dedicated `DataProcessor` class encapsulates all cleaning and metric computation logic. Methods include `cleanData()`, which removes laps

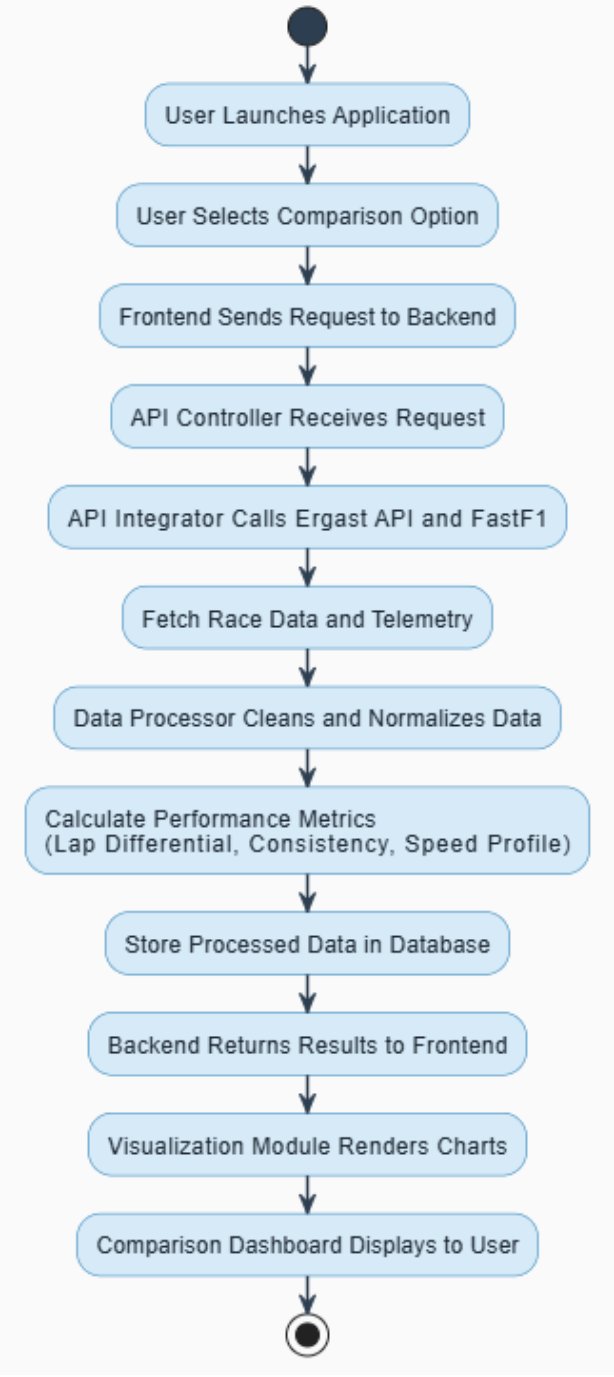

Fig. 2. Operational Workflow: Activity Diagram of the F1 Analytics System

with missing telemetry or flagged as inlaps or outlaps; `calculateMetrics()`, which applies Equations (1), (2), and (3); and `prepareAnalytics()`, which structures the output as comparison-ready JSON objects.

3) Storage Layer: A MySQL database stores normalized tables for Driver, Team, Race, Lap, LapPerformance, and Telemetry entities as modelled in the ER diagram. Foreign key constraints enforce referential integrity between LapPerformance records and their parent Driver and Race entries. Telemetry sample documents are stored in MongoDB with fields for TelemetryID, LapPerfID, SampleTime, Speed, and Gear. This hybrid storage design balances relational integrity for structured race records with document flexibility for variable-length telemetry sequences.

4) Presentation Layer: The ReactJS frontend maintains component-level state for session selection and driver pair configuration. On user interaction, it dispatches asynchronous fetch calls to Flask API endpoints. Returned JSON arrays are parsed and bound to chart data structures rendered by interactive visualization libraries. Chart types include line charts for lap-time trends and speed traces, bar charts for sector comparisons, and overlay charts for throttle and brake traces. All charts support hover tooltips that display precise numerical values at any data point.

IX. Results and Discussion

The implemented platform was evaluated using historical F1 sessions accessed through FastF1 and Ergast. Multiple race rounds from recent seasons were loaded and tested across the full data pipeline to verify correctness and usability.

Lap-time differential charts rendered accurately for all tested driver pairs, with ΔT_n values aligning with published race timing records. The consistency metric σ produced meaningful rankings, with drivers who maintained stable pace across stints showing significantly lower standard deviation values compared to those involved in incidents or tyre degradation phases. These numerical outputs are rendered directly on the dashboard alongside the corresponding lap-time charts, providing users with both a visual and quantitative view of driver consistency.

Speed-versus-distance overlays revealed clear differences in driving style between driver pairs at specific track sections. Braking points, minimum corner speeds, and acceleration zones became visually identifiable from the plotted profiles, demonstrating the value of mapping telemetry to track distance rather than elapsed time. Throttle and brake trace comparisons further highlighted differences in driver input patterns through high-speed and technical sections.

Sector time comparisons exposed performance variations that lap-time totals alone do not reveal. In several tested sessions, drivers with comparable overall lap times exhibited opposing sector strengths, indicating different aerodynamic setups or driving approaches. The dashboard surfaces these differences through side-by-side sector bar charts.

Response performance was satisfactory for cached sessions, with visualization data loading within acceptable time thresh-

olds after the initial session fetch. Uncached first-load times were longer due to FastF1's session download requirement, which is addressed by pre-caching frequently accessed sessions in the database layer.

The platform successfully bridges the gap between raw public F1 datasets and accessible, interactive analytics, making detailed performance insights available without requiring users to write or understand the underlying Python data pipeline.

X. Conclusion and Future Work

This paper described the design, implementation, and evaluation of a web-based F1 race data visualization and performance analysis platform. The system integrates the FastF1 Python library and the Ergast Developer API into a unified backend pipeline, applies structured preprocessing and mathematical performance metric computation, and delivers results through an interactive ReactJS frontend. The platform enables lap-time differential analysis, driver consistency scoring, speed-versus-distance profile comparison, and sector time visualization for any supported historical F1 session.

The work demonstrates that combining open data sources with modular software design and interactive visualization can make professional-grade motorsport analytics accessible to students, independent researchers, and enthusiasts. The mathematical formulations for lap-time differential, consistency scoring, and distance-normalized speed profiling provide a reproducible analytical foundation that extends beyond visual display to support data-driven conclusions about driver performance.

Future enhancements to this platform include the integration of predictive models for pit stop strategy recommendation using reinforcement learning approaches as explored in related literature [3], extension to live session data when available through OpenF1 or similar real-time sources, addition of sector-wise telemetry breakdowns and tyre compound degradation curves, and support for circuit map overlays that annotate speed and braking data directly onto track geometry. Progressive web application support would also allow the platform to be accessed effectively on mobile devices, broadening its reach among the general F1 audience.

References

[1] Sky Sports F1, "Latest F1 news and live updates," *Sky Sports*, Nov. 8, 2025. [Online]. Available: https://www.skysports.com/f1

[2] T. Krause, "FastF1 - Python package for accessing and analysing Formula 1 timing data and telemetry," *PyPI*, Sep. 4, 2025. [Online]. Available: https://pypi.org/project/fastf1/

[3] International Journal for Research in Applied Science and Engineering Technology (IJRASET), "A F1 Strategy Predictor using Reinforcement Learning," Apr. 25, 2025. [Online]. Available: https://ijraset.com

[4] Tracing Insights, "F1 Racing Analytics: Lap Times, Telemetry, Timing Data," *TracingInsights.com*, Oct. 31, 2024. [Online]. Available: https://tracinginsights.com

[5] Catapult Sports, "How Data Analysis Transforms F1 Race Performance," *Catapult Sports Blog*, Feb. 13, 2024. [Online]. Available: https://www.catapult.com/blog/f1-data-analysis-transforming-performance

[6] T. Oehrly, "FastF1: A Python Package for Accessing and Analysing Formula 1 Data," GitHub Repository, 2024. [Online]. Available: https://github.com/theOehrly/Fast-F1

[7] M. Wright, "Formula 1 Technology and Data Analysis: Understanding Race Strategy and Performance," *Motorsport Engineering Journal*, vol. 6, no. 2, pp. 45–53, Jun. 2023. Available: https://www.motorsportengineer.net

[8] A. Singh and R. Patel, "Real-Time Telemetry Data Analysis for Formula 1 Racing," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 13, no. 9, pp. 112–118, Sep. 2022.

[9] A. Struthers, "Formula One Telemetry Analysis," in *SOURCE 2022: Scholar Works of University Research & Creative Expression*, Central Washington University, Ellensburg, WA, USA, May 2022. [Online]. Available: https://digitalcommons.cwu.edu/source/2022/COTS/99/

[10] R. Tavares *et al.*, "Real-Time Motorsport Analytics," *IEEE Xplore Digital Library*, 2021.