A blue and white logo

AI-generated content may be incorrect.

National College of Ireland

Project Proposal

Momentum

24/9/25

Computing

Software Development

2025/2026

Eoin Andrew Fitzsimons

X23151374

X23151374@student.ncirl.ie

Contents

[1.0 Objectives 2](#_Toc209646186)

[2.0 Background 2](#_Toc209646187)

[3.0 State of the Art 3](#_Toc209646188)

[4.0 Technical Approach 4](#_Toc209646189)

[4.1. Simulation Engine 4](#_Toc209646190)

[4.2. Procedural Content Generation 4](#_Toc209646191)

[4.3. AI Decision-Making 4](#_Toc209646192)

[4.4. Unity Front-End Interface 4](#_Toc209646193)

[4.5. Iteration, Feedback, and Scalability 5](#_Toc209646194)

[5.0 Technical Details 6](#_Toc209646195)

[5.1. Race Outcome Simulation 6](#_Toc209646196)

[5.2. Driver Progression 6](#_Toc209646197)

[5.3. AI Decision-Making 6](#_Toc209646198)

[5.4. Procedural Content Generation 6](#_Toc209646199)

[5.5. Performance Optimisation & Integration 6](#_Toc209646200)

[6.0 Special Resources Required 8](#_Toc209646201)

[6.1. Hardware Requirements 8](#_Toc209646202)

[6.2. Software and Libraries 8](#_Toc209646203)

[6.3. Optional Resources 8](#_Toc209646204)

[6.4. Summary 8](#_Toc209646205)

[7.0 Project Plan 9](#_Toc209646206)

[7.1. Project Overview 9](#_Toc209646207)

[7.2. Requirements and Design 9](#_Toc209646208)

[7.3. Core Simulation Engine Development 9](#_Toc209646209)

[7.4. Procedural Content Generation 9](#_Toc209646210)

[7.5. AI Systems Development 9](#_Toc209646211)

[7.6. Front-End Integration 10](#_Toc209646212)

[7.7. Iterative Testing and Refinement 10](#_Toc209646213)

[7.8. Final Evaluation and Documentation 10](#_Toc209646214)

[8.0 Testing 11](#_Toc209646215)

[8.1. Testing Strategy 11](#_Toc209646216)

[8.2. Unit Testing 11](#_Toc209646217)

[8.3. Integration Testing 11](#_Toc209646218)

[8.4. System Testing 11](#_Toc209646219)

[8.5. User Evaluation 11](#_Toc209646220)

# Objectives

Momentum aims to design and implement a prototype motorsport management simulator that captures the scale, depth, and unpredictability of real-world racing ecosystems. Unlike existing titles, which currently focus on a limited range of disciplines (e.g., Formula 1 and limited feeder systems), it will model a wide variety of racing disciplines, each with its own unique challenges and culture. For example, rally racing will feature dynamic terrain and weather, requiring teams to adapt strategies for gravel, tarmac, or snow. Endurance racing will simulate human factors such as driver fatigue, teamwork, and mechanical reliability over long events. Street racing will emphasise customisation, driver personality, and the underground culture that shapes rivalries and alliances. By capturing these distinctions, the simulator will aim to represent a more varied and context-sensitive motorsport environment, where each series presents different management challenges.

The objective is to simulate a persistent game world where events unfold across all leagues, regardless of whether the player is directly involved. By doing so, the system mirrors the emergent qualities of Football Manager, supporting diverse career trajectories that may differ across playthroughs.

A further objective is to offer a high degree of customisation. Players will be able to modify or generate new teams, drivers, and competitions. Procedural generation will be a cornerstone, enabling large-scale fictional ecosystems that feel authentic while avoiding licensing restrictions.

The project will demonstrate implementation across key areas including simulation design and AI-driven decision-making. The intended prototype will model core systems such as driver progression, financial management, league standings, and race outcomes, with a focus on achieving computational efficiency.

The project’s objective is to produce a management simulator that can serve as a foundation for future extension. It seeks to address an identified gap in current games by emphasising simulation depth and customisation, rather than prioritising graphical fidelity.

# Background

My passion for motorsport spans a wide range of disciplines, from open-wheel racing to rally, endurance, and street racing. While some recent games, such as The Crew Motorfest, successfully capture the unique feel and culture of different racing series, they focus on driving experiences rather than management. Existing management games, on the other hand, tend to concentrate on a single discipline or lack the depth and variety needed to authentically represent the full spectrum of motorsport. No current title combines the breadth of racing types, the distinctiveness of each discipline, and the emergent, persistent world of a management simulation. This project seeks to address that gap by integrating the unique characteristics of multiple racing series into a single, dynamic management experience.

Having spent extensive hours in Football Manager, I have experienced the appeal of large, complex simulations that generate emergent stories. Despite having a stronger interest in Formula 1 and motorsport generally than in football, I find myself drawn more to Football Manager than to existing motorsport management titles. This reveals an important insight: the attraction lies not in the sport itself, but in the richness of the simulated world.

By comparison, current motorsport titles such as F1 Manager and Motorsport Manager are limited. Their focus is on presentation and visuals rather than world depth. The lack of customisation and the narrow scope of simulated leagues lead to repetitive experiences over time.

This project seeks to address this gap by exploring how the thematic appeal of motorsport can be combined with the systemic depth characteristic of Football Manager. The objectives outlined in Section 1.0 will be met by designing algorithms that support persistent multi-league simulation, driver development models, and financial/strategic systems. Procedural content generation will be used to overcome licensing limitations and create a fictional but believable motorsport ecosystem.

# State of the Art

Current motorsport management titles include Motorsport Manager (2016) and F1 Manager (2022–2024). These games present detailed races and authentic visuals, but their simulation depth is fundamentally limited. Driver development is linear and predetermined, making long-term play predictable. Once a player has learned the optimal strategies, the outcomes become repetitive. In contrast, Football Manager maintains replayability by simulating an immense global ecosystem of leagues, players, and staff, where no two careers are the same. This difference highlights the gap between motorsport management games and the broader standard of emergent sports simulations.

Another limitation of existing motorsport titles is their lack of customisation. Players cannot easily expand leagues, create fictional scenarios, or reshape the competitive environment. The games are tightly constrained by licensing agreements and static structures, which prevents the freedom that makes Football Manager engaging. Furthermore, the game worlds are small in scale, restricted to a handful of categories such as Formula 1 or Formula 2, with little simulation occurring outside the top tiers. By comparison, Football Manager tracks hundreds of leagues and thousands of clubs across multiple nations, with events unfolding regardless of direct player involvement.

This project aims to distinguish itself by adopting that philosophy. The simulation will extend beyond open-wheel racing, with multiple tiers within each discipline. Every league will progress whether the player manages there or not, creating a sense of a living world. Because official licensing is not available, all teams, drivers, and championships will be procedurally created, which also allows experimentation with unique structures, rivalries, and narratives. By prioritising breadth, depth, customisation, and emergent systems, this project will investigate how principles seen in Football Manager’s large-scale simulation might be adapted to a motorsport context.

It is worth noting that some recent non-management racing games, such as The Crew Motorfest, have made significant strides in capturing the unique feel, culture, and challenges of different racing disciplines. These titles demonstrate that it is possible to authentically represent the diversity of motorsport, from rally to endurance to street racing. However, they focus on the driving experience and do not offer the persistent, emergent management simulation that is central to this project. No current management title integrates the breadth of racing types, the distinctiveness of each discipline, and a persistent simulation world. This project aims to explore how these elements can be combined within a prototype, developing a management simulator that reflects both the variety and complexity of motorsport ecosystems.

# Technical Approach

Momentum is developed using a modular, iterative methodology that combines agile software principles with a technically rigorous framework for multi-discipline motorsport simulation. The system architecture is divided into four interdependent subsystems: **Simulation Engine**, **Procedural Content Generation**, **AI Decision-Making**, and **Unity Front-End Interface**. Each module is engineered to operate independently while remaining fully interoperable, allowing isolated development, testing, and optimization without disrupting overall system functionality. Data exchange between Unity and the Python back-end occurs via structured JSON messages, and persistent data is maintained using SQLite, providing a durable foundation for multi-season and multi-league simulations.

## Simulation Engine

The simulation engine forms the core computational framework. Requirements were identified by analysing existing motorsport management games, reviewing multi-agent systems research, and evaluating emergent gameplay mechanics. Key functions include driver skill progression, team financial management, dynamic race outcome modelling, and concurrent league support. Race outcomes are computed using Monte Carlo simulations, integrating driver attributes, car performance metrics, track-specific parameters, weather conditions, and probabilistic random events. Financial models employ stochastic methods to simulate operational budgets, sponsorship volatility, and unexpected expenditures, producing realistic emergent league economics. For example, an underfunded team may be forced to adopt aggressive pit strategies, affecting competitor behaviour and league standings.

## Procedural Content Generation

Procedural generation algorithmically produces teams, drivers, tracks, and championships using constraint-based and probabilistic rules. Driver traits, including skill, temperament, and fatigue progression, are assigned according to league difficulty and balance requirements. Tracks are generated with variable weather, geographic diversity, and surface types, influencing race strategy and outcome. Team budgets and resources are procedurally allocated, generating dynamic competitive hierarchies that evolve over seasons. Metrics such as driver attribute diversity and competitive balance are monitored, and automated adjustments ensure ecosystem stability, enabling a replayable and unique motorsport environment for each simulation.

## AI Decision-Making

AI agents integrate deterministic rule sets, probabilistic heuristics, and reinforcement learning to manage pit stops, driver rotations, race tactics, and financial decisions. Emergent behaviours arise when AI teams respond dynamically to competitor actions. For instance, a fatigued AI driver may miscalculate braking in wet conditions, causing chain reactions that affect pit timing and rival team strategies. Reinforcement learning allows agents to optimise long-term league performance, producing unpredictable yet balanced outcomes. Real-time monitoring ensures AI decisions remain consistent with procedural constraints and league regulations.

## Unity Front-End Interface

Unity renders race telemetry, standings, driver statistics, financial dashboards, and optional 3D track previews. Iterative integration ensures simulation outputs, AI decisions, and procedural updates are reflected accurately. User interactions, including strategy adjustments and managerial decisions, are transmitted to the Python back-end, which updates persistent league data. Visualization metrics provide immediate feedback on emergent behaviour, AI impact, and procedural ecosystem balance.

## Iteration, Feedback, and Scalability

The development plan divides tasks into measurable phases with continuous feedback loops. Milestones track driver progression, AI adaptability, financial consistency, and procedural diversity. Modular isolation enables future expansion, including additional motorsport disciplines, extended driver attributes, or enhanced AI complexity without redesigning the core engine. Persistent storage and optimised computational methods sustain emergent, multi-league behaviour across multiple seasons, ensuring Momentum remains a robust, scalable, and highly replayable motorsport management simulator.

# Technical Details

Momentum is implemented using a **Unity 2023 front-end** for real-time visualization and dashboard interfaces, paired with a **Python 3.11 back-end** for simulation logic, AI decision-making, procedural content generation, and persistent data management. Python leverages **NumPy** for high-performance numerical computations, **Pandas** for structured data manipulation, and **SciPy** for advanced statistical operations. AI modules additionally employ **PyTorch** for reinforcement learning and neural network modelling, while procedural generation routines rely on constraint-solving libraries to maintain balanced ecosystems. **SQLite** provides a lightweight, cross-platform database for persistent storage of drivers, teams, tracks, championship states, and multi-season histories.

## Race Outcome Simulation

Race results are computed using **Monte Carlo simulations**, integrating multiple weighted factors including driver skill, vehicle performance parameters, track-specific attributes, dynamic weather, and stochastic random events. Each simulated race consists of thousands of iterations to generate probabilistic distributions of finishing positions. Edge cases, such as mechanical failure or accidents, are represented as low-probability events, introducing realistic uncertainty while maintaining competitive balance. Cached results for commonly accessed league states reduce redundant computation and ensure real-time responsiveness when updating leaderboards and race summaries.

## Driver Progression

Driver attributes—speed, consistency, aggression, adaptability, and fatigue resilience—are represented as **multi-dimensional vectors** with probabilistic evolution based on historical performance. Past performance is stored in SQLite and influences future development trajectories using weighted stochastic growth functions. Fatigue and injury are modelled using decay algorithms combined with recovery cycles, producing emergent career progression patterns that differ for each driver. This ensures that each season presents novel strategic challenges and realistic driver dynamics.

## AI Decision-Making

AI agents use a **hybrid approach**, combining deterministic rule-based behaviour with probabilistic decision heuristics and reinforcement learning. Rule sets govern fundamental decisions, such as pit stop eligibility and safety car reactions, while probabilistic heuristics allow for variable strategies under uncertainty. Reinforcement learning optimises long-term league objectives, such as championship points accumulation, resource allocation, and rivalry management. AI outputs are transmitted in real time to Unity via JSON, enabling coherent front-end visualization of emergent behaviours and adaptive strategy execution.

## Procedural Content Generation

Procedural algorithms generate teams, drivers, tracks, and championships based on constraint satisfaction models. Attribute ranges are normalised to maintain competitive balance, and geographic or thematic variations are algorithmically assigned to tracks and championships. Procedural generation pipelines iterate over candidate entities, validating diversity and ecosystem stability before committing data to SQLite. This ensures that each simulated league is unique, replayable, and emergent.

## Performance Optimisation & Integration

Python back-end modules leverage **vectorised computations and in-memory caching** for computational efficiency, while Unity employs asynchronous data polling to avoid frame drops. Multi-league persistence is maintained by serializing league states at season checkpoints. JSON-based communication guarantees synchronised updates between front-end visualization, AI agents, and procedural generators. The architecture allows future expansion of AI sophistication, additional motorsport disciplines, and enhanced procedural attributes without disrupting core functionality.

# Special Resources Required

## Hardware Requirements

Development and execution of Momentum require a high-performance workstation with a minimum of a 6-core CPU (Intel i7-12700 or AMD Ryzen 7 5800X) to handle concurrent back-end simulations and AI computations, 16 GB of RAM for caching simulation states, and a dedicated GPU (NVIDIA RTX 3060 or equivalent) to render real-time Unity dashboards and optional 3D track previews. For testing multi-league persistence and large procedural ecosystems, additional cores or cloud-based virtual machines (VMs) are recommended to parallelize race simulations across multiple seasons without performance degradation. SSD storage is necessary for fast SQLite read/write operations, particularly when managing hundreds of driver and team records across concurrent leagues.

## Software and Libraries

The front-end is built in Unity 2023, providing 2D/3D rendering, UI management, and telemetry visualization. The back end is implemented in Python 3.11, utilizing NumPy and Pandas for numerical and data processing, SciPy for statistical modelling, and PyTorch for AI reinforcement learning. Persistent storage is managed via SQLite, offering a lightweight, cross-platform relational database sufficient for multi-season and multi-league state persistence without requiring enterprise-level database infrastructure. SQLite supports ACID compliance, enabling safe rollback during iterative simulation testing, though for extremely large-scale simulations, optional migration to PostgreSQL or MySQL can be considered.

## Optional Resources

Cloud-based VMs (AWS, Azure, or GCP) allow large-scale stress testing of AI agents, procedural content generation, and multi-league simulations, enabling automated benchmarking for performance optimization. Git version control with branching ensures collaborative development and rollback safety. Continuous integration tools (GitHub Actions or GitLab CI) automate unit and integration testing pipelines. Procedural generation eliminates the need for licensed assets, while optional telemetry visualization plugins enhance dashboard clarity for advanced performance analysis.

## Summary

These resources collectively provide sufficient computational capacity, software infrastructure, and development support to enable persistent, emergent, and scalable motorsport management simulation without reliance on high-cost enterprise solutions.

# Project Plan

## Project Overview

The development of Momentum will follow a phased, iterative methodology, combining agile principles with the requirements of a multi-discipline motorsport management simulation. The project is divided into seven primary phases, each building on the previous phase and enabling measurable milestones. Dependencies between phases ensure that core systems are functional before advanced AI and procedural content systems are integrated. Feedback loops are implemented at each stage to refine algorithms, validate simulation balance, and confirm emergent behaviours.

## Requirements and Design

The initial phase focuses on defining the full scope of the project, architecture of the system, and algorithmic approaches for AI and procedural generation. Functional requirements include realistic driver progression, dynamic race outcomes, persistent multi-league simulation, and procedurally generated teams, drivers, tracks, and championships. Non-functional requirements focus on computational efficiency, scalability, and maintainability. Design documentation, UML diagrams, and flowcharts will be produced to ensure all team members have a collective understanding of the system architecture. This phase establishes clear performance targets and constraints, including database schema designs for SQLite and communication protocols between Unity and Python back-end modules.

## Core Simulation Engine Development

The next phase implements the simulation engine, which forms the backbone of the project. League structures, driver progression mechanics, financial systems, and race resolution algorithms are developed and unit tested. Driver attributes such as skill, fatigue, consistency, and personality traits are modelled using probabilistic weighted systems. Financial operations for teams, including budgets, sponsorships, and operational costs, are represented using stochastic models to reflect real-world variability. Race outcomes are calculated using Monte Carlo simulations, factoring in driver skill, car performance, weather conditions, and track characteristics. Data structures and in-memory caching strategies are optimized to maintain high performance during multi-league, multi-season simulations.

## Procedural Content Generation

Once the simulation engine is operational, procedural content generation routines are integrated. Teams, drivers, tracks, and championship schedules are algorithmically generated based on constraint satisfaction models to ensure diversity and balance. Geographic, thematic, and competitive variations are embedded to create a believable motorsport ecosystem. Generated entities are validated for ecosystem stability before being committed to the SQLite database. This phase also implements pipelines to allow batch generation and automated verification of content, supporting replayability and emergent behaviours across multiple simulated seasons.

## AI Systems Development

AI modules are implemented concurrently with procedural generation. Teams and drivers utilize hybrid decision-making models combining rule-based heuristics, probabilistic strategies, and reinforcement learning. AI agents determine pit stop timing, race tactics, driver rotations, and long-term strategic decisions to maximize championship points while responding dynamically to rival behaviour. Reinforcement learning models are trained on historical simulation data to optimize performance over multiple seasons. AI outputs are communicated to Unity in real time via JSON, ensuring coherent visualization of strategies, standings, and emergent rivalries.

## Front-End Integration

The Unity front-end is connected to the Python back-end to visualize race telemetry, leaderboards, team dashboards, and driver statistics. Asynchronous data polling is implemented to prevent frame drops and maintain smooth user interactions. User interface components are tested for usability, clarity, and responsiveness. Integration ensures that procedural generation and AI decisions are reflected accurately in the visualized simulation state.

## Iterative Testing and Refinement

Unit, integration, and system testing occur in iterative cycles, with milestones for driver progression accuracy, financial consistency, AI diversity, race outcome realism, and persistent multi-league simulation. Iterative feedback informs adjustments to algorithms and simulation parameters. Emergent behaviours are monitored to ensure realistic rivalries, variable race outcomes, and balanced progression across multiple leagues and seasons. Performance profiling and optimization are performed to maintain efficiency under high computational loads.

## Final Evaluation and Documentation

The final phase evaluates the emergent behaviour, user experience, system stability, and overall performance. All code is documented, and comprehensive reports on simulation accuracy, AI performance, procedural generation stability, and database integrity are produced. This phase confirms that Momentum operates as a coherent, persistent, and scalable motorsport management simulator, meeting all functional and non-functional requirements.

The structured project plan ensures clear progression from design through deployment, with measurable milestones and dependencies at each stage. Iterative feedback and testing cycles guarantee that Momentum achieves realistic, emergent, and persistent gameplay while remaining maintainable and extensible for future development.

# Testing

## Testing Strategy

Testing for Momentum is structured across multiple levels to ensure system stability, performance, and fidelity of simulation outcomes. Unit, integration, system, and user testing are all incorporated to validate both functional and non-functional requirements. Each level of testing is designed to produce measurable results and inform iterative refinement cycles throughout development. Ethical considerations, including data privacy, informed consent, and participant anonymity, are integrated into all user evaluation procedures.

## Unit Testing

Unit testing validates individual modules of the simulation. Driver progression, financial systems, race outcome calculations, and AI decision-making routines are tested against expected behaviours and edge cases. Probabilistic models for driver skill development, fatigue, and personality traits are verified for correct statistical distribution and consistency. Financial operations are checked for accuracy, ensuring budgets, sponsorships, and costs are correctly computed under stochastic conditions. Race outcome algorithms, including Monte Carlo simulations, are validated to maintain realistic variability while ensuring balance between drivers and teams.

## Integration Testing

Integration testing ensures proper communication between the Python back-end, Unity front-end, AI modules, and procedural content generation systems. JSON-based data exchanges are verified for accuracy and latency, and SQLite storage operations are tested for integrity across multi-season and multi-league scenarios. Metrics include data consistency, transaction success rates, and persistence reliability. Integration tests also verify that AI decisions, procedural generation outputs, and race telemetry are correctly reflected in the user interface, maintaining coherent emergent behaviours across the simulation.

## System Testing

System-level testing evaluates Momentum under full operational conditions. Multi-league, multi-season simulations are run to assess emergent behaviour, computational performance, and scalability. Metrics include league variability, AI decision diversity, race outcome distribution, system load, and memory usage. Stress testing is performed to evaluate performance under high concurrency and extended simulation periods. Any bottlenecks in AI computation, procedural content generation, or database operations are identified and addressed to maintain real-time responsiveness and persistent state accuracy.

## User Evaluation

User testing measures interface usability, engagement, and comprehension of the management systems. Participants interact with the full Unity front-end, reviewing dashboards, race summaries, and team management tools. Quantitative metrics include task completion time, accuracy of actions, and interaction frequency. Qualitative metrics are gathered via structured surveys, focusing on perceived realism, clarity, and enjoyment. Ethical safeguards are enforced, including informed consent, voluntary participation, and secure handling of collected data. Feedback informs both interface improvements and algorithmic refinements to maintain realistic, emergent gameplay.

Overall, the testing framework ensures that Momentum achieves persistent, scalable, and believable motorsport simulation. Iterative results from unit, integration, system, and user testing directly guide algorithm adjustments, AI behaviour tuning, procedural content balancing, and front-end enhancements, guaranteeing a coherent, high-quality, and engaging management experience.