

Final Year Project Thesis – B.Sc. in Software Development

Development of a Sports Analytics Platform for Rugby

Authors:

Steven Nolan

April 30th, 2025

Supervisors:

Brendan Watson

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Brendan Watson, for their invaluable guidance, patience, and insightful feedback throughout this research process. Their expertise and encouragement were instrumental in bringing this project to completion.

In addition, my heartfelt thanks go to my parents, and my friends for their unwavering support, understanding, and encouragement during my time at university, especially during the demanding phases of this research. Your belief in me has been a constant source of strength.

Ethical Declaration

I declare that this project and document is wholly my own work except where I have made explicit reference to the work of others. I have read the Department of Information Technology Final Year Project guidelines and relevant institutional regulations, and hereby declare that this document is in line with these requirements.

I have discussed, agreed, and complied with whatever confidentiality or anonymity terms of reference were deemed appropriate by those participating in the research and dealt appropriately with any other ethical matters arising, in line with the LIT Research Ethics Guidelines for Undergraduate and Taught Postgraduate Programmes policy document.

[Steven Nolan]                     [30/04/2025]

Abstract

This project presents the development of a sports analytics platform focused on rugby, aiming to enhance data accessibility and visualization for players, teams, and enthusiasts. As the demand for real-time performance insights grows, especially in professional sports, the integration of data collection, analysis, and visualization becomes critical for informed decision-making.

The platform utilizes web scraping techniques—leveraging Python libraries such as BeautifulSoup and Selenium—to gather live and historical data on matches, player performance, and league standings. Additionally, data visualization tools including Django, Plotly, and Chart.js are employed to create an interactive dashboard, enabling users to explore match trends, player metrics, and predictive analytics. Ethical considerations in web scraping were central to this project, adhering to robots.txt directives and privacy standards.

Key features of the platform include dynamic match statistics, head-to-head team comparisons, player performance tracking, and visually engaging representations such as heat maps, radar charts, and score progression graphs. Through these tools, users gain deeper insights into team strategies and individual contributions, enriching their understanding of the game.

This project lays the foundation for future enhancements, including machine learning models for predicting match outcomes and player form, ultimately contributing to the evolving landscape of sports analytics in rugby.

Table of Contents

[Acknowledgements 2](#_Toc196891181)

[Ethical Declaration 3](#_Toc196891182)

[Abstract 4](#_Toc196891183)

[Table of Contents 5](#_Toc196891184)

[Table of Figure 7](#_Toc196891185)

[Chapter 1 Introduction 8](#_Toc196891186)

[1.1 Objectives 9](#_Toc196891187)

[1.1.1 The project objectives 9](#_Toc196891188)

[1.1.2 Academic objectives 9](#_Toc196891189)

[1.1.3 Problem Domain 10](#_Toc196891190)

[1.2 Product title: 10](#_Toc196891191)

[1.3 The Scope of the solution 10](#_Toc196891192)

[1.4 Report Structure 10](#_Toc196891193)

[Chapter 2 Literature Review 12](#_Toc196891194)

[2.1 Introduction 12](#_Toc196891195)

[2.2 Existing Applications and Competitive Analysis 12](#_Toc196891196)

[2.3 Data Collection Techniques 13](#_Toc196891197)

[2.4 Data Visualization 13](#_Toc196891198)

[2.5 Ethical and Security Considerations in Web Scraping 14](#_Toc196891199)

[2.6 Machine Learning in Sports Analytics 15](#_Toc196891200)

[2.7 Conclusion 15](#_Toc196891201)

[Chapter 3 Analysis and Design 16](#_Toc196891202)

[3.1 Introduction and focus 16](#_Toc196891203)

[3.2 Weekly Meetings and Agile Approach 16](#_Toc196891204)

[3.3 Project Management 16](#_Toc196891205)

[3.3.1 Source code management (SCM) 16](#_Toc196891206)

[3.3.2 Collaboration Tools 16](#_Toc196891207)

[3.3.3 Code Style Guide 17](#_Toc196891208)

[3.4 Application Overview 17](#_Toc196891209)

[3.5 Potential Blockers and Considerations 17](#_Toc196891210)

[3.6 Scope of the Project 17](#_Toc196891211)

[3.7 Prerequisites and Background Knowledge 18](#_Toc196891212)

[3.8 Tools and Frameworks Considered 18](#_Toc196891213)

[3.9 Dataset Considerations 18](#_Toc196891214)

[3.10 System Architecture 20](#_Toc196891215)

[3.11 Data Visualization and Exploration 21](#_Toc196891216)

[3.12 Data Visualization and Exploration 22](#_Toc196891217)

[3.13 Use Cases 23](#_Toc196891218)

[3.14 Conclusion 24](#_Toc196891219)

[Chapter 4 Implementation 25](#_Toc196891220)

[4.1 Development Environment 25](#_Toc196891221)

[4.2 Tools Used 25](#_Toc196891222)

[4.3 Source Control and Project Management 26](#_Toc196891223)

[4.4 Data Integration 26](#_Toc196891224)

[4.5 Visualization and Frontend Features 28](#_Toc196891225)

[4.6 Challenges and Solutions 29](#_Toc196891226)

[4.7 Ethical Considerations in Implementation 30](#_Toc196891227)

[4.8 Conclusion 32](#_Toc196891228)

[Chapter 5 Rough Testing 33](#_Toc196891229)

[5.1 Academic Aims 33](#_Toc196891230)

[5.2 Testing Methodologies 34](#_Toc196891231)

[5.3 Functional Requirements 35](#_Toc196891232)

[5.4 Non-Functional Requirements 35](#_Toc196891233)

[5.5 Data Integrity and Validation 35](#_Toc196891234)

[5.6 Challenges in Testing 36](#_Toc196891235)

[5.7 Conclusion 36](#_Toc196891236)

[Chapter 6 Results 38](#_Toc196891237)

[6.1 Project Plan: Priorities and Milestones 38](#_Toc196891238)

[6.1.0 The Data Structure 38](#_Toc196891239)

[6.1.1 Populating the System with Data 38](#_Toc196891240)

[6.1.2 Machine Learning 38](#_Toc196891241)

[6.1.3 Testing 38](#_Toc196891242)

[6.1.4 Paths to completion 39](#_Toc196891243)

[6.2 Data Structures 39](#_Toc196891244)

[6.3 System Architecture 39](#_Toc196891245)

[6.3.1 Object Identification 40](#_Toc196891246)

[6.4 Conclusion 40](#_Toc196891247)

[Chapter 7 Conclusion and Recommendations 41](#_Toc196891248)

[7.1 Conclusion 41](#_Toc196891249)

[7.2 Limitations 41](#_Toc196891250)

[7.3 Recommendations 41](#_Toc196891251)

[7.4 Final Reflection 43](#_Toc196891252)

[Chapter 8 References 44](#_Toc196891253)

Table of Figure

[Figure 1 ERD Entity-Relationship Diagram 19](#_Toc196891257)

[Figure 2 System Architecture Diagram 20](#_Toc196891258)

[Figure 3 Sequence Diagram 21](#_Toc196891259)

[Figure 4 Data Flow Diagram 22](#_Toc196891260)

[Figure 5 Use Case Diagram 24](#_Toc196891261)

# Introduction

In recent years, sports analytics has transformed how athletes, teams, and fans engage with competitive sports. Among the various disciplines, rugby analytics remains an evolving field, with untapped potential for data-driven insights. Traditional rugby statistics often provide surface-level data, lacking the depth and interactivity required for comprehensive performance analysis. This project addresses this gap by developing a rugby-specific sports analytics platform that integrates real-time data collection, visualization, and predictive analytics.

Literature search and review will be undertaken in the areas of sports analytics frameworks, data visualization techniques, machine learning applications in sports performance, and domain-specific rugby metrics. Emphasis will also be placed on existing tools used in other sports (e.g., soccer and American football) to identify transferable approaches, best practices for real-time data handling, and user experience design for analytics dashboards. This review will inform the system architecture, ensure the relevance of selected metrics, and support the development of accurate and actionable insights for coaches, analysts, and players.

The core aim of this platform is to empower users—coaches, players, analysts, and enthusiasts—with intuitive visualizations and detailed performance metrics, facilitating a deeper understanding of individual and team dynamics. The platform utilizes web scraping and API integration to gather rugby match data and player statistics. Data is processed and visualized using Python’s Django framework, along with libraries such as Plotly, Matplotlib, and Chart.js. These tools allow users to explore trends, compare players, and analyse match outcomes interactively.

A key challenge in this project involves the ethical and legal considerations of web scraping. Web scraping is a technique used to automatically extract data from websites, often by simulating how a user would browse and interact with web pages. In the context of sports analytics, it allows for the collection of valuable information such as match statistics, player data, and live scores from public sources. However, not all websites permit automated data extraction, and scraping without permission can raise ethical and legal concerns. This issue is addressed in the literature search, which explores best practices, case studies, and legal frameworks surrounding data collection. To ensure compliance, the project adheres to responsible data scraping methods, including respecting robots.txt directives, using rate limiting to avoid server overload, and ensuring no personal or sensitive user data is collected. Furthermore, the platform is built using Django for the backend and React for the frontend, offering a robust and scalable architecture capable of real-time data processing and interactive visualizations. This technical stack not only supports performance and flexibility but also ensures the system can evolve to meet future data compliance standards and user demands.

The scope of this thesis focuses on match insights, team form comparison, player performance, and data visualization. Looking ahead, machine learning models may be integrated to predict match outcomes and player performance.

This report is structured as follows: Chapter 1 introduces the project; Chapter 2 reviews related literature; Chapter 3 details design considerations; Chapter 4 explains the implementation process; Chapter 5 covers testing; Chapter 6 presents results; and Chapter 7 concludes with findings and recommendations.

## Objectives

### The project objectives

Develop an interactive rugby analytics web application capable of collecting, processing, and visualizing player and team performance data.

Implement web scraping and API integration for real-time and historical data acquisition from rugby sources.

Design and deploy data visualizations (e.g., line graphs, radar charts, heat maps) for intuitive performance insights.

Explore the integration of machine learning models for predictive analytics, such as match outcome prediction and player performance trends.

Ensure ethical data collection practices, respecting privacy, robots.txt directives, and copyright laws.

### Academic objectives

Gain hands-on experience in data engineering, including data scraping, cleaning, and preprocessing for large datasets.

Conduct a comprehensive literature review to explore existing research in sports analytics, focusing on rugby, and examine relevant studies in data visualization, machine learning, and ethical data scraping practices.

Apply software development methodologies (Agile) and tools (GitHub, Django, React) in a structured project setting.

Demonstrate the ability to present data through interactive visual interfaces using Python libraries and JavaScript frameworks.

Critically evaluate the ethical implications of web scraping and data privacy in real-world applications.

Contribute to academic discourse on sports analytics by proposing an innovative, user-centered platform focused on rugby.

### Problem Domain

Despite the growth of sports analytics platforms, rugby-specific solutions remain limited, especially those that offer real-time, interactive data tailored to team strategies and individual player insights. Existing tools often prioritize basic statistics, overlooking visual analytics and predictive models. Moreover, access to data is fragmented, with limited API support and varying website formats, making data collection inconsistent and ethically complex. This project aims to address these challenges by developing a comprehensive, ethical, and scalable analytics tool for rugby.

## Product title:

Development of a Sports Analytics Platform for Rugby

## The Scope of the solution

Data collection via web scraping/APIs for rugby match data, player stats, and league standings.

Visual analytics with dynamic charts, tables, and dashboards.

User features like match comparisons, player profiles, and league overviews.

Lays groundwork for future machine learning integration for predictive analysis.

## Report Structure

Chapter 1 introduces the project scope, objectives, and problem domain.

Chapter 2 provides a literature review covering data collection, visualization, and ethics in sports analytics.

Chapter 3 explores the design and analysis of system architecture and tools.

Chapter 4 details implementation, including technical tools and challenges.

Chapter 5 discusses testing methodologies and results.

Chapter 6 presents outcomes and visualizations.

Chapter 7 concludes the report with recommendations and future work.

The Chapters following styles

Paragraphs are 12pt Arial Justified with 1.5-line spaces and 6pt before with 3 pt after.

# Literature Review

## Introduction

Sports analytics is a rapidly evolving field, providing athletes, teams, and fans with increasingly deep insights into performance and strategy. However, within rugby, dedicated analytics platforms often lag behind those developed for sports like soccer. This chapter presents a comprehensive review of the existing literature, methodologies, and technologies shaping sports analytics platforms, with a specific focus on the rugby landscape. It explores current approaches to data collection, analysis (including machine learning), and visualization, examining existing tools and frameworks. By evaluating these elements, alongside the potential integration of techniques like web scraping and APIs, this review aims to identify key areas for innovation while addressing crucial ethical and security considerations inherent in data handling.

## Existing Applications and Competitive Analysis

Several sports analytics platforms serve as benchmarks for this project, notably FotMob, Ultimate Rugby, RugbyPass, and Flashscore.

FotMob, a soccer application, inspired the core user interface design and feature set. It aggregates player statistics, match data, and league tables, enabling users to view upcoming fixtures, lineups, and live stats at a glance.

Ultimate Rugby provides detailed match analytics, including possession, territory, attack, and defence metrics. This structured presentation is crucial for effective rugby data visualization.

RugbyPass stands out for its graphical insights, including points flow, ruck speed, set play stats, and head-to-head comparisons, informing the development of similar visual elements in this project.

Flashscore contributes historical data displays and player nationality insights, enhancing user engagement with global data perspectives.

This analysis identified a need for the current project: a dedicated rugby analytics platform emphasizing robust data visualization, in-depth analysis, and ethical data handling.

## Data Collection Techniques

Robust data collection is the foundation of any effective analytics platform. This project primarily utilizes web scraping to gather player and match statistics from relevant rugby websites. Key Python libraries employed include Beautiful Soup, chosen for its simplicity in parsing static HTML content, alongside Selenium and Playwright, which are necessary for navigating JavaScript-heavy websites and retrieving dynamic data. These tools collectively enable the extraction of both structured and unstructured content (Mitchell, 2015). Where available, public APIs are preferentially used to enhance data consistency, ensure efficient retrieval, and minimize the need for subsequent data cleaning. While this project relies on publicly accessible data, it's noted that professional rugby increasingly leverages wearable technology (e.g., GPS trackers, heart rate monitors) and smart ball technology (capturing metrics like speed, hang time, and trajectory). Although these advanced data streams are inaccessible for this project, they inform the aspirational goals for potentially integrating such telemetry data in future versions.

## Data Visualization

Effective visual representation is crucial for transforming raw data into actionable insights within rugby analytics. This platform employs specific tools for this purpose: Django is utilized for backend data handling, complemented by the Pandas library for data manipulation. On the frontend, a suite of libraries including Chart.js, Plotly, Matplotlib, and Seaborn are used to generate a range of interactive visualizations. Key visualization types implemented include:

Line and Area Graphs: To track score progression and potentially illustrate match momentum over time.

Bar and Pie/Stacked Bar Charts: For comparing team statistics, individual player metrics (like tackles or tries), and representing proportions such as win/loss ratios.

Radar Charts: Offering multi-metric views for comprehensive analysis of individual player performance.

Heat Maps and Bubble Charts: To enable spatial analysis, visualizing elements like possession zones or key action areas on the pitch.

Timeline and Stacked Area Charts: Visualizing the flow of the match, sequences of events, and shifts in control or momentum.

Collectively, these visualization tools enable users to perform real-time updates, conduct historical analysis, and interactively explore team performance, player statistics, and match trends, providing insights specifically tailored and essential for rugby analytics.

## Ethical and Security Considerations in Web Scraping

Web scraping, while powerful, inherently raises important ethical, legal, privacy, and security concerns that must be addressed responsibly. This project is committed to ethical data collection and adheres to established best practices. A primary principle is the use of public APIs whenever available, as this method is preferred for structured data retrieval, enhances consistency, and significantly reduces server load on the target websites (Densmore, 2017).

When scraping is employed, several key guidelines are followed:

Respecting Site Owners: robots.txt directives are strictly adhered to, ensuring that only publicly allowable data is accessed and restricted areas are avoided (Zachary Gold & Mark Latonero, 2018). Clear User Agent strings are used to identify the scraper.

Preventing Disruption: Requests are carefully rate-limited to prevent server overload and mitigate Denial-of-Service (DoS) risks (Khder, 2021).

Upholding Privacy: Collection of Personally Identifiable Information (PII) is strictly avoided, and compliance with regulations like GDPR is maintained, often by focusing on aggregated rather than individual data (Ali Hussain Ahmad & Gafar Zen Alabdeen Salh Hassan, 2024).

Maintaining Integrity & Avoiding Circumvention: Data accuracy is validated where possible, and circumvention tactics like aggressive IP rotation, which can breach ethical norms and Terms of Service, are avoided.

Furthermore, ethical scraping involves pursuing constructive relationships with site administrators (Densmore, 2017)and aiming for value creation, such as properly crediting data sources and avoiding the wholesale replication of proprietary content. By integrating these technical and ethical safeguards, this project ensures a transparent, responsible data collection approach that aligns with academic values and legal standards.

## Machine Learning in Sports Analytics

Machine learning presents considerable potential in rugby analytics, for tasks ranging from performance prediction to injury forecasting, though it remains under exploration for this specific project. Potential future applications being considered include: using classification models like Random Forests for player role identification/categorization; applying regression models to predict player performance metrics; and leveraging time-series models such as LSTMs for match outcome predictions. Further research, dataset expansion, and detailed discussion on the integration of these ML techniques will be addressed in the implementation and future work chapters.

## Conclusion

This chapter reviewed the landscape of sports analytics, highlighting the specific need and opportunity for a dedicated rugby platform. Analysis of benchmark applications informed the project's scope, emphasizing data visualization and ethical practices. The core technical approach relies on carefully managed web scraping and APIs for data collection, alongside a suite of visualization tools (e.g., Chart.js, Plotly) to generate actionable insights. Adherence to ethical scraping standards (Densmore, 2017; Zachary Gold & Mark Latonero, 2018; Khder, 2021; Ali Hussain Ahmad & Gafar Zen Alabdeen Salh Hassan, 2024) is foundational. While machine learning integration is a future goal, this review confirms the project's basis in current best practices and its potential to contribute valuable analytical capabilities to the rugby domain, providing a solid foundation for the work described in subsequent chapters.

# Analysis and Design

## Introduction and focus

This chapter explores the design philosophy, technical architecture, and planning strategies for the development of a rugby analytics platform. It outlines key challenges, chosen solutions, and the rationale behind the frameworks, tools, and methodologies utilized. The goal is to build a scalable, user-centric, and ethically responsible web application that transforms rugby data into actionable insights.

## Weekly Meetings and Agile Approach

The development process followed an Agile methodology, with weekly meetings and iterative sprints to ensure continuous progress and adaptability. These sessions involved:

Progress tracking via GitHub.

Task assignment and sprint planning.

Feedback from the project supervisor to guide development.

This iterative process allowed the project to respond flexibly to challenges, incorporating ongoing testing and user feedback for refined functionality.

## Project Management

### Source code management (SCM)

All code was version-controlled using Git and hosted on GitHub. This enabled structured collaboration, backup, and progress tracking.

### Collaboration Tools

#### GitHub

GitHub for submissions and code review.

#### Microsoft Office

This was used for creating all documentation related to this project

### Code Style Guide

To ensure consistency and readability, a coding style guide was followed:

Meaningful variable naming and method declarations.

Avoidance of magic constants—using named constants instead.

Proper documentation, comments, and indentation for maintainability.

## Application Overview

The platform aims to analyse rugby match and player data, offering features such as:

Real-time and historical match statistics.

Player performance tracking.

Predictive analytics (future integration).

Interactive data visualizations.

By combining web scraping, API integration, and data visualization, the platform empowers users to derive strategic insights.

## Potential Blockers and Considerations

Data Availability: Reliable, consistent data is scarce, and scraping quality varies across sources.

Dynamic Content: JavaScript-rendered websites require Selenium or Playwright for effective scraping.

Computational Complexity: Large datasets and machine learning models may require substantial processing power.

Ethical Scraping: Scraping must respect robots.txt, rate limits, and data ownership.

## Scope of the Project

The platform will focus on:

Data visualization of match and player statistics.

Head-to-head comparisons, historical analysis, and performance dashboards.

Future machine learning integration for performance predictions.

## Prerequisites and Background Knowledge

Key skills and tools required:

Web Scraping: Familiarity with Python libraries (BeautifulSoup, Selenium).

Machine Learning: Understanding of models for classification, regression.

Data Visualization: Proficiency with Plotly, Chart.js, and Django templating.

## Tools and Frameworks Considered

Backend: Python with Django for API creation and data processing.

Frontend: React for a dynamic, interactive interface.

Database: Initially SQLite3, with potential to scale to PostgreSQL or MongoDB.

Visualization: Matplotlib, Plotly, Chart.js.

Machine Learning: scikit-learn for basic models; TensorFlow or PyTorch for future development.

## Dataset Considerations

Data collected via web scraping and public APIs.

Preprocessing includes formatting, cleaning, and anonymizing data.

Data stored in a relational database, optimised for queries and visualization.

To support the development of the rugby analytics platform, a structured relational database schema was designed. The following Entity-Relationship Diagram (ERD) illustrates the key entities involved in the system—Match, Player, and Statistics—and how they relate to each other. This model ensures efficient storage and retrieval of rugby performance data. Each player is associated with multiple matches, and each match contains various player statistics. The diagram also highlights primary keys (PK) and foreign keys (FK), which help maintain data integrity across the platform.

A diagram of a computer

AI-generated content may be incorrect.

Figure ERD Entity-Relationship Diagram

## System Architecture

This project adopts a modular architecture designed for flexibility, scalability, and ease of maintenance. The following diagram outlines the key components of the system and how they interact:

The frontend is built using React and communicates with a Django-based REST API backend.

Data is collected via a Web Scraping Module from external sources or public APIs and stored in a relational database.

The SQLite3 database is used as a lightweight solution for prototyping, with future migration to PostgreSQL planned for scalability.

A Machine Learning Module runs server-side to generate predictive insights based on historical data.

All communication between layers uses JSON for structured data exchange, supporting potential mobile app integration in later stages.

A diagram of a machine learning module

AI-generated content may be incorrect.

Figure System Architecture Diagram

As shown in Figure 2, the system promotes separation of concerns across the UI, data handling, and analytics layers. This architecture not only enables independent module development but also supports future scaling through API-based integration and potential cloud deployment.

## Data Visualization and Exploration

To enhance user engagement and decision-making, the platform includes interactive data visualizations that present rugby statistics in an accessible and intuitive format. These visual tools include score progression graphs, player performance metrics, heat maps, and team comparisons.

The sequence diagram below illustrates the data flow and interaction between the system's layers when a user requests match statistics. This includes how data is requested by the frontend, processed by the backend, and retrieved from the database before being visualized in the user interface.

The design is influenced by leading sports applications such as FotMob and RugbyPass, with future plans for real-time updates and historical trends tracking.

A diagram of a process flow

AI-generated content may be incorrect.

Figure Sequence Diagram

As shown in Figure 3, the process begins when a user interacts with the frontend to request specific match data. This triggers an API call to the backend, which then queries the database for the required statistics. Once the data is retrieved and processed, it is returned to the frontend and presented visually to the user. This layered communication ensures data integrity while supporting real-time responsiveness. To improve print readability

## Data Visualization and Exploration

To support intelligent analysis and predictions, the platform integrates various machine learning models. Classification techniques such as Random Forest and Decision Trees are used for categorizing player roles and performance profiles, while Long Short-Term Memory (LSTM) models are implemented for time-series analysis, enabling match outcome predictions based on historical data.

The diagram below illustrates the system’s data flow, from the user interface down to data retrieval and processing layers. It highlights how user interactions trigger backend processes, web scraping operations, and database queries, all of which work together to deliver updated and relevant visual statistics.

A diagram of a software development

AI-generated content may be incorrect.

Figure Data Flow Diagram

As shown in Figure 4, when a user requests data, the frontend communicates with the Django backend through an API call. The backend queries the SQLite3 database for existing data or, when necessary, activates the web scraping module to fetch data from external sources. Once the required data is gathered and processed, it is returned to the frontend for visualization. This modular flow ensures responsiveness, data accuracy, and the ability to scale with additional data sources or predictive logic.

## Use Cases

This section outlines the core interactions between the system and its users, represented through use cases. The primary actor is the user, who engages with the platform to retrieve, analyse, and visualize rugby-related data for insights into player performance, team comparisons, match predictions, and more. The system also acts independently by performing background tasks such as automated data scraping.

The diagram below visually represents these key use cases, and the actors involved, illustrating how the system supports both user-driven and autonomous operations.

**Primary Actor: User**

View Match Statistics

Track Player Performance

Compare Teams

Predict Match Outcomes

Analyse Team Form

View Fixtures and Predictions

Visualize Set Plays

**Primary Actor: System**

Real-time Data Collection via Web Scraping

As shown in Figure 5, users can perform several actions such as viewing match statistics, tracking performance, and predicting outcomes, while the system autonomously handles real-time data collection. This structure helps ensure the platform meets the practical needs of analysts and coaches while maintaining continuous data flow through automated backend processesA diagram of a diagram

AI-generated content may be incorrect..

Figure Use Case Diagram

## Conclusion

This chapter outlined the design blueprint of the platform, detailing tools, frameworks, and the strategic decisions made. By leveraging Agile development, ethical scraping, and powerful visualization, the platform is positioned to offer a compelling, data-driven experience for rugby analysis.

# Implementation

This chapter documents the practical implementation of the rugby analytics platform, detailing the specific tools, frameworks, and methodologies employed, alongside the challenges encountered during the development process. The focus spans the critical areas of backend and frontend development, data scraping techniques and integration workflows, the creation of visual analytics, and the practical adherence to ethical considerations outlined previously. Project management practices guiding this phase are also noted. This implementation effort culminated in a fully functional prototype that demonstrates key platform features, such as match insights, team comparisons, and player profiles, while establishing a scalable technical foundation for future enhancements.

## Development Environment

The development environment setup was key to ensuring consistency, scalability, and efficiency across development phases. It was configured to support modular development and seamless integration between components. A core architectural decision was to decouple the backend and frontend, primarily to enhance scalability and allow for potential future extension into mobile platforms. Development activities were primarily conducted using VS Code as Integrated Development Environments (IDEs), with GitHub utilized for version control and workflow management. The specific technical environment comprised:

Operating Systems: Windows 11 and Ubuntu (via Windows Subsystem for Linux 2 - WSL2)

Python Version: 3.11.x

Node.js Version: 21.6.1

Browser & Testing Tools: Chrome DevTools (for debugging and inspection) and Selenium WebDriver (for testing dynamic content interaction).

## Tools Used

Backend: Python, Django REST Framework (API, routing, auth, data access).

Frontend: React (interactive/responsive UI).

Database: SQLite3 (PoC, Django compatible), designed for potential PostgreSQL migration.

Web Scraping: BeautifulSoup (static HTML), Selenium (dynamic JS content), Requests/Playwright (experimental).

Data Handling: Pandas, NumPy.

Visualization: Matplotlib, Plotly, Seaborn, Chart.js.

Dependencies: pip (Python/backend), npm (Node.js/frontend).

Others: Django ORM, CORS Headers, Axios (frontend HTTP), python-dotenv (config).

## Source Control and Project Management

The project adopted Agile methodologies to guide the development process and ensure iterative progress. Version control was managed using Git, with the repository hosted on GitHub. This platform served as the central hub for code management and tracking changes throughout development. The development cycle was organised into weekly sprints, each typically concluding with a review meeting or feedback session with the supervisor to ensure alignment and gather input. Task management and issue tracking were also handled within GitHub, utilizing its built-in features to monitor progress and manage development tasks. This combination facilitated iterative development, clear versioning, and steady progress.

Repository: [<https://github.com/realstevennolan/fyp-thesis.git>]

## Data Integration

The platform's functionality relies heavily on robust data scraping, integration, and processing workflows.

Data Scraping:

Match and player data were gathered from multiple public, rugby-specific websites. Python libraries, primarily BeautifulSoup (for static content) and Selenium (for dynamic, JavaScript-rendered content), were employed for this task. The targeted data included:

Match scores

Team names

Detailed player statistics (e.g., tackles, carries, lineouts won/lost)

League standings and fixture details

Ethical scraping practices were a core consideration throughout. This involved strict adherence to each site's robots.txt rules, implementation of rate-limiting delays to avoid overloading source servers, respecting data ownership rights, and maintaining logs for each scrape session to aid debugging and ensure compliance.

Weather Data Integration:

To add a unique analytical dimension, historical weather data corresponding to match dates and locations was integrated. This was achieved using an open weather API. The relevant weather fields (temperature, atmospheric pressure, wind speed, humidity) were retrieved and joined to the respective match data. This join process relied on careful normalization of datetime fields using Python to ensure accuracy. This integrated data allows for potential comparative analysis of match performance under different weather conditions.

Data Processing and Storage:

Once collected, both scraped rugby data and API-sourced weather data underwent a processing pipeline using the Pandas library. This involved:

Cleaning: Handling inconsistencies or errors in the raw data.

Normalization/Formatting: Standardizing formats, particularly crucial for datetime fields to ensure consistency for joining and analysis.

Validation: Checking for missing or clearly malformed entries.

The cleaned, validated, and integrated data was then stored in the SQLite3 database, utilizing Django's Object-Relational Mapper (ORM) models for structured storage and interaction. Finally, this processed match, player, and associated weather data were served to the frontend application through custom-built API endpoints created with the Django REST Framework.

## Visualization and Frontend Features

Visualization is critical to user engagement and generating actionable insights from the rugby data. The frontend was developed to provide interactive exploration capabilities.

Tools & Techniques:

Interactive Dashboards: Built using React as the primary frontend framework. Plotly.js and Chart.js components were embedded within React to render dynamic and interactive visualizations.

Data Fetching: Dashboards and components dynamically fetch processed data from the backend Django APIs via secure JSON communication.

Key Visualization Components:

Bar Charts: Used for comparing team performance metrics across different matches or seasons.

Line Graphs: Implemented to track player statistics over time, illustrating form and historical trends.

Radar Charts: Employed to create comprehensive all-round player ability profiles, visualizing multiple metrics simultaneously.

Scatter Plots: Used to explore potential correlations, such as team performance metrics versus weather conditions on match day.

Heat Maps: Explored experimentally for visualizing spatial data like field zones of action (primarily considered a future extension).

User-Facing Features:

Match Insights: Provides a visual summary of key match metrics, team performance indicators, relevant player statistics, and historical context.

Player Profiles: Offers detailed views of individual players, incorporating Radar charts for ability profiles and Line charts comparing form across current and potentially past seasons.

Team Comparisons: Enables side-by-side analysis of teams, including win ratios, league standings, and head-to-head performance graphs.

Weather Contextualization: Integrates weather data where available, potentially using tooltip overlays or dedicated sections to suggest how conditions might have impacted performance (noting this may be resource-intensive or remain experimental).

(Planned) Real-Time Updates: Future integration of live scraping or APIs is planned to enable near real-time updates during matches.

System Architecture Notes:

The frontend (React) and backend (Django) communicate exclusively via secure JSON APIs.

The modular design aims to support easier maintenance and potential future expansion into microservices or native mobile clients.

The current prototype is deployable on a local server environment. Docker containerization is planned for streamlined future deployment and scalability.

## Challenges and Solutions

Several challenges were encountered during the implementation phase, each requiring targeted research and technical solutions to maintain the platform’s performance, ethical integrity, and usability. Among these, handling dynamic web content proved to be the most technically demanding aspect of the project. Scraping data from JavaScript-heavy websites required specialized tools and a deeper understanding of browser automation, which introduced both performance and ethical considerations.

The table below summarises the key challenges, and the solutions implemented:

|  |  |
| --- | --- |
| **Challenges Encountered** | **Solutions Identified and implemented** |
| Inconsistent/Missing Data | Implemented robust data validation routines during processing. Handled missing entries using appropriate default values and applied interpolation techniques for numerical gaps where suitable. |
| Dynamic Web Content | Utilized Selenium to effectively interact with and scrape data from JavaScript-rendered website elements. Playwright was also explored as an alternative for potential performance improvements. |
| Backend Performance | Optimised Django ORM database queries to reduce execution time. Implemented caching strategies for frequently accessed API endpoints and minimized data payload sizes to improve response times and reduce latency. |
| Ethical Web Scraping | Strictly adhered to website robots.txt directives. Implemented appropriate rate-limiting delays between requests to avoid server overload and ensured transparency in scraping methods and user agent identification. |
| Visualization Clutter | Introduced pagination on tables and lists displaying large amounts of data. Implemented interactive filtering options on dashboards to allow users to refine displayed data, improving user experience (UX). |

## Ethical Considerations in Implementation

The ethical principles discussed in Chapter 2 were actively applied throughout the implementation phase to ensure responsible data handling:

Respecting Website Policies: All scraping activities strictly adhered to the guidelines outlined in website robots.txt files. Only publicly accessible data was targeted, and explicitly prohibited paths or sections were avoided.

Preventing Server Strain: Rate-limiting delays were consistently implemented between requests to prevent overloading source website servers and avoid causing disruptions. Clear User-Agent strings were used to identify the scraper's purpose.

Data Usage and Privacy: The collection of any Personally Identifiable Information (PII) was strictly avoided. The data gathered was used solely for the educational purposes of this project, focusing on value-added analysis rather than simple replication of proprietary content.

Attribution: Data sources were appropriately acknowledged and attributed where feasible within the platform or documentation, respecting the origin of the information.

Web Scrapers vs Web Developers A Broader Perspective

While web developers typically focus on creating websites and web applications that present content to users, web scrapers focus on retrieving and processing data from those websites, often without a formal API. The intent behind scraping can vary some use it for analysis, market research, academic research, or even to power competitive services.

The scale of web scraping on the internet is significant. Vast amounts of content are scraped daily product listings, reviews, social media posts, sports scores, and more. Some scrapers are sophisticated bots that operate at scale and frequency comparable to regular web users. This raises important ethical and legal concerns, especially when scraping is done without consent, at high volume, or to replicate someone else's content for commercial gain.

Is scraping responsible for viral content?

Web scraping isn’t directly responsible for making things “go viral,” but it can accelerate the spread of content. Developers or marketers may use scraped data like trending hashtags, popular posts, or breaking news to inform what they publish. This can lead to faster content reproduction, aggregation, or even manipulation, which in turn draws more attention and traffic. While this helps in keeping content discoverable and responsive to trends, it also blurs the line between content curation and content duplication, raising questions about originality, fairness, and ethical intent.

## Conclusion

This chapter detailed the successful implementation of the rugby analytics platform, transforming the design concepts into a functional prototype. Through the coordinated use of Django (backend), React (frontend), and various data handling libraries (Pandas, BeautifulSoup, Selenium), a system capable of ethically scraping, processing, integrating weather data, storing, and visualizing rugby statistics was constructed. Key features, including interactive match insights, player profiles, and team comparisons, were realised using tools like Plotly and Chart.js. Challenges related to data consistency, dynamic web content, performance, and visualization clarity were systematically addressed. Crucially, the implementation adhered strictly to the ethical guidelines established earlier. This process has resulted in a tangible, locally deployable prototype built on a scalable, decoupled architecture, providing a solid foundation for the evaluation phase and future enhancements such as machine learning integration and broader deployment strategies.

# Testing

Testing is a critical phase in software development, ensuring that the Rugby Analytics platform operates accurately, efficiently, reliably, and ethically, ultimately delivering valuable insights to its users. This chapter details the comprehensive testing strategy employed throughout the project lifecycle. It outlines the methodologies used—ranging from automated unit tests to manual user acceptance testing—the tools leveraged, the processes for validating data integrity, and the verification of both functional and non-functional requirements. The goal was to rigorously assess the system's correctness and usability while identifying areas for improvement.

## Academic Aims

From an academic perspective, testing validates not only the technical correctness but also the academic objectives of the project, including:

Validate Ethical Practices: Confirm adherence to ethical data collection standards, particularly concerning web scraping (robots.txt compliance, rate limiting).

Ensure Data Accuracy and Reliability: Verify that the data extracted, processed, stored, and presented is accurate and consistent across the platform.

Evaluate User Interaction and Clarity: Assess the effectiveness of the user interface, the clarity of data visualizations, and the overall user experience in conveying analytical insights.

Demonstrate Software Engineering Principles: Showcase the practical application of structured testing methodologies (unit, integration, UAT) as part of a robust software engineering process.

## Testing Methodologies

1. Unit Testing

Unit tests focused on individual components, such as:

Web scraping scripts: verifying accurate data extraction.

Data processing functions: ensuring correct data formatting and handling.

API endpoints: validating data retrieval and response formats.

Tools Used: Django’s built-in test framework, pytest.

1. Integration Testing

Tested the interaction between:

Backend and frontend components.

Data flow from scraping → processing → visualization.

Database interactions and data consistency.

1. User Testing

Informal user testing was conducted with peers and supervisors. Feedback was gathered on:

Usability of dashboards.

Clarity of visualizations.

Responsiveness and load times.

User feedback guided interface tweaks, improved navigation, and influenced chart selection.

## Functional Requirements

|  |  |  |
| --- | --- | --- |
| Requirement | Test Description | Status |
| Display real-time match stats | Scrape data during a live match and update dashboard dynamically. | Not Met |
| Visualize player performance (season) | Radar and line charts update as player data changes. | PartiallyMet |
| Load historical league tables | Retrieve and display past league standings correctly from DB. | Met |
| Adhere to ethical scraping practices | Verify robots.txt checks and rate-limiting delays in scrapers. | Met |

## Non-Functional Requirements

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Test Description** | **Status** |
| Performance | Backend response times < 1 second for standard queries. | Met |
| Accessibility | Dashboard usable across devices and browsers. | Met |
| Documentation | Maintain code comments/docstrings and provide README guidance. | Met |

## Data Integrity and Validation

Maintaining data quality was paramount. Validation steps included:

Cross-Validation: Manually comparing key scraped data points (e.g., scores, standings) against official URC sources during development cycles.

Database Constraints: Utilizing Django model field types, unique=True, and foreign key relationships to enforce basic data structure and relational integrity at the database level.

Serializer Validation: Implementing validation rules within Django REST Framework serializers to check data types, formats, and presence of required fields before saving to the database.

API Response Checks: Both automated tests (APIClient) and manual inspection (e.g., via browser dev tools or Postman) verified that API endpoints returned expected data structures and accurate values.

Consistency Audits: Periodically checking for consistency in naming conventions and references (e.g., ensuring player names matched across different data tables).

## Challenges in Testing

The testing process faced several practical challenges:

Live Data Dependency: Thoroughly testing real-time features was hampered by the dependency on live URC match schedules, making automated, repeatable tests difficult.

Data Source Variability: Occasional inconsistencies, missing data, or structural changes on the source websites required adaptive adjustments to scraping and processing logic.

Frontend Test Automation Scope: Due to project time constraints, comprehensive automated frontend testing (e.g., using Jest/React Testing Library or Cypress for component and end-to-end tests) was not fully implemented, necessitating greater reliance on manual UAT.

## Conclusion

The structured testing approach detailed in this chapter successfully validated the core functionalities of the Rugby Analytics platform. Unit and integration testing confirmed the robustness of backend data handling from scraping to API delivery, while User Acceptance Testing guided essential refinements to the frontend's usability.

Testing confirmed the achievement of primary objectives, demonstrating reliable loading, processing, and display of key rugby data and visualizations. However, it also highlighted challenges, notably with dynamic web scraping (necessitating the adoption of Selenium/Playwright) and backend performance (addressed through query optimization and caching), confirming the effectiveness of the solutions implemented.

While advanced goals like predictive analytics and full accessibility were only partially realised due to complexity and time limits, testing affirms that the project met most core requirements. This iterative cycle of development and validation was instrumental in overcoming obstacles and drove significant practical learning in Django, React, data engineering, and API integration.

# Results

This chapter presents the outcomes of the project, structured around the objectives outlined in Chapter 1.1.

It evaluates the system’s development against original plans, highlights achievements, identifies challenges, and assesses future development opportunities.

## Project Plan: Priorities and Milestones

### The Data Structure

The foundation of the system was the data architecture.

Django models were created for Teams, Players, Matches, PlayerMatchStats, and TeamStandings.

Relationships between models ensured data consistency and efficient queries.

### Populating the System with Data

Data ingestion pipelines were developed using:

Web scraping techniques respecting ethical standards.

Manual data entry for teams and players where necessary due to limited public APIs.

### Machine Learning

Initial plans to integrate machine learning models for match outcome prediction and starting player prediction were deferred.

This was due to the limited amount of season data available and time constraints.

### Testing

Testing was performed systematically:

Unit tests verified API functionality, database integrity, and backend logic.

Integration tests ensured frontend and backend communication was reliable.

User feedback informed improvements in UI/UX design.

### Paths to completion

By adhering to Agile methodology, the project progressed iteratively:

Core backend functionality and database design were prioritized first.

API endpoints and React frontend development followed.

Testing and UI refinement continued through successive iterations.

## Data Structures

The data structures proved effective in supporting both backend operations and frontend displays:

The Player model captured comprehensive biographical and statistical data.

The Team model facilitated relationships with matches, standings, and stats.

Match records included detailed fixture information and scoring data.

TeamMatchStat entries aggregated key performance metrics like tackles, turnovers, and scrums.

Overall, the Django ORM (Object Relational Mapper) provided a flexible and scalable foundation for data operations.

## System Architecture

The final system architecture consisted of:

Backend: Django REST Framework provided APIs for players, teams, matches, and statistics.

Frontend: A React-based Single Page Application (SPA) rendered dashboards, visualizations, and detailed views.

Database: SQLite was used for rapid development, with future migration to PostgreSQL considered for scalability.

### Object Identification

The following key objects and their relationships were identified:

Team: Related to matches (home and away), standings, and team match statistics.

Player: Linked to PlayerMatchStat and their associated team (club).

Match: Connected to teams and included real-time scores and metadata.

Each object was carefully normalized to avoid redundancy and support efficient queries.

## Conclusion

This chapter outlined the project’s execution across data design, system architecture, and feature development.

Despite limitations in live data availability and time for machine learning experimentation, the project successfully achieved:

A functioning, interactive rugby analytics platform.

A scalable backend and frontend architecture.

A foundation for future real-time updates and predictive analytics.

# Conclusion and Recommendations

## Conclusion

The aim of this project was to develop a web-based analytics platform dedicated to rugby, capable of collecting, processing, and visualizing player and team performance data in an accessible and interactive manner.

The project achieved several key milestones:

A fully functioning backend powered by Django and RESTful APIs.

A React-based frontend that dynamically displays players, team stats, match results, and performance visualizations.

Custom visualizations, including doughnut charts and stat tables, tailored to rugby performance indicators.

Ethical considerations were respected during web scraping by adhering to guidelines set by data providers.

## Limitations

While the platform demonstrates strong core functionality, some limitations were encountered:

Lack of consistent, structured public APIs for rugby meant much of the data had to be manually input or scraped, reducing scalability.

Real-time updates were not fully implemented due to time constraints and ethical limits on live scraping.

Machine learning features were not deployed, although data pipelines for future use were established.

Additionally, testing revealed areas where data coverage or frontend responsiveness could be improved although I added cashing the initial load could be improved.

## Recommendations

* Machine Learning Integration

Now that clean, structured data exists, integrating models for:

* + Match outcome prediction,
  + Player form analysis, and
  + Injury risk assessment would add significant value.

Live Data API Partnerships

* + To ensure scalability and legality, partnerships with official data providers (e.g. Opta or URC) would provide structured, real-time match data, enabling advanced features like live dashboards.

Expand Visualizations

Add:

* + Heat maps for player positioning,
  + Time-series graphs for team form,
  + Head-to-head comparison charts between teams.

These additions would enhance interactivity and insight.

Deploy Publicly

Deploy the platform on a cloud environment (e.g. Heroku, Render, or AWS) and containerize using Docker for portability.

Improve Mobile Responsiveness

Although desktop responsiveness was tested, further refinement for mobile and tablet displays would ensure a broader reach.

## Final Reflection

This project provided valuable hands-on experience in full-stack web development, data engineering, and ethical data collection. From an academic standpoint, it bridged the gap between theoretical concepts and practical implementation. The knowledge and skills acquired throughout the development of this rugby analytics platform will serve as a strong foundation for future research, development, or professional work in data science and sports technology.

With a solid architecture and clear expansion roadmap, this platform stands as a promising step toward more advanced, ethical, and accessible analytics in the sport of rugby.

# References

Ali Hussain Ahmad & Gafar Zen Alabdeen Salh Hassan, 2024. Web Mining overview: Techniques, Tools, and Ethical Implications. *Journal of Computer Sciences,* pp. 86-94.

Densmore, J., 2017. *Ethics in Web Scraping.* [Online]   
Available at: https://towardsdatascience.com/ethics-in-web-scraping-b96b18136f01

Khder, M. A., 2021. Web scraping or web crawling: State of art, techniques, approaches, and application. *International Journal of Advances in Soft Computing & Its Applications,* Volume 13, p. 25.

Mitchell, R., 2015. *Web Scraping with Python: Collecting Data from the Modern Web.* Sebastopol, CA: O'Reilly Media.

Zachary Gold & Mark Latonero, 2018. Robots welcome: Ethical and legal considerations for web crawling and scraping. *Washington Journal of Law, Technology & Arts,* 13(3).