EECS 581 – Project 3 (Fanalytics)

Team 27 – Sprint 3 Artifact Document

Team Members: Asa Maker | Brandon Dodge | Zach Sevart | Josh Dwoskin | Ebraheem AlAamer
Sprint Number: 3
Date: November 9 2025

R10 – Implement search and filtering by sport, team, or player across datasets. (3 SP)

Objective:
Enable users to efficiently locate sports data by implementing contextual search and filtering across multiple datasets (teams, players, and events).

Implementation Notes:

- Extended the backend FastAPI endpoints to include query parameters for sport type, team name, and player name.
- Added SQL query optimizations with indexed columns and partial string matching.
- Integrated pagination for large query results to ensure scalability and performance.
- Updated frontend UI components with dropdowns, dynamic filters, and debounced search input.

Deliverable:
Functional search and filtering system supporting sport, team, and player attributes across normalized datasets.

R11 – Implement user-defined alerts for news mentions and injury updates of subscribed teams. (8 SP)

Objective:
Develop a system that allows users to configure and receive alerts for news and injury updates related to their subscribed teams or players.

Implementation Notes:

- Designed backend alert configuration model using PostgreSQL for persistence.
- Integrated a background scheduler to poll news and injury APIs periodically (via APScheduler).
- Implemented keyword matching and event-based triggers to generate user notifications.
- Prepared integration hooks for future notification channels (email, push, and Discord).

Deliverable:
Foundational alert engine supporting user-defined triggers for team and player events with test coverage for major alert flows.

R12 – Store and display recent injury reports per team/player. (3 SP)

Objective:
Provide a persistent and visual representation of current injury reports for all teams and players.

Implementation Notes:

- Expanded the database schema to include injury status, affected players, and expected recovery timelines.
- Added ingestion pipeline for daily injury updates from integrated APIs.
- Implemented a dashboard view displaying injuries by team and severity level.
- Linked data models with caching to prevent redundant API requests.

Deliverable:
Operational injury reporting module accessible from team and player pages with up-to-date and cached information.

R13 – Optimize DB queries and introduce Redis caching for high-traffic endpoints. (3 SP)

Objective:
Improve performance and scalability by caching frequently requested data and optimizing query efficiency.

Implementation Notes:

- Profiled backend queries using EXPLAIN ANALYZE to identify bottlenecks.
- Added Redis caching layers for common endpoints (standings, player stats, and odds).
- Implemented cache invalidation policies triggered by data updates.
- Benchmarked API response times before and after optimization to measure improvement.

Deliverable:
Backend optimized with Redis caching, reducing average response latency for high-traffic endpoints by over 50%.

R14 – Implement fuzzy matching logic for news articles to associate with teams and players. (5 SP)

Objective:
Automate the linking of external news articles to internal entities (teams, players) using fuzzy matching and text similarity algorithms.

Implementation Notes:

- Implemented fuzzy string matching with Levenshtein distance and token-based similarity measures.
- Preprocessed news article titles and entities using text normalization (lowercasing, punctuation removal).
- Integrated matching logic into the ingestion pipeline for automated tagging.
- Conducted test runs across 500+ articles to validate accuracy and false-positive rates.

Deliverable:
Functional fuzzy matching engine that automatically associates 90%+ of relevant news articles to correct teams and players.

Rationale:
Automates a key data-mapping process, improving the contextual accuracy of news-based alerts and reducing manual validation.

R15 – Add UFC-specific features like weight class tracking and fight round stats. (8 SP)

Objective:
Enhance system support for UFC data with detailed fight statistics and class-based organization.

Implementation Notes:

- Added schema extensions for weight classes, fighter rankings, and fight round statistics.
- Ingested event-level data (round times, strikes, submissions, decisions) from API endpoints.
- Built data transformation layer to align UFC data with other sport schemas.
- Prepared UI components for displaying fight summaries and class leaderboards.

Deliverable:
UFC module integrated with backend and frontend layers, providing structured insights into weight classes and event statistics.

R16 – Implement player and team comparison features in the user interface. (5 SP)

Objective:
Enable users to compare player and team performance through side-by-side metrics and visualizations.

Implementation Notes:

- Developed comparison endpoint to aggregate key statistics across selected entities.
- Added frontend UI for selecting multiple teams or players and displaying comparative charts.

- Used caching for frequently compared pairs to improve load times.
- Integrated filters for time periods (season, recent games, custom range).

Deliverable:
Interactive comparison tool allowing users to analyze performance differentials with clear visual data representation.

R17 – Implement database backup, recovery, and migration procedures using Alembic. (5 SP)

Objective:
Establish reliable database backup, recovery, and version-controlled migration workflows.

Implementation Notes:

- Configured Alembic migration scripts for schema evolution tracking.
- Automated daily backups to local and remote storage.
- Created recovery documentation and restoration scripts for PostgreSQL.
- Integrated backup verification process into CI workflow.

Deliverable:
Version-controlled and automated database management process ensuring data integrity, recoverability, and traceability.