EECS 581 – Project 3 (Fanalytics)

Team 27 – Sprint 2 Artifact Document

Team Members: Asa Maker | Brandon Dodge | Zach Sevart | Josh Dwoskin | Ebraheem AlAamer
Sprint Number: 2
Date: November 2 2025

Sprint 2 Objectives

Sprint 2 extended the system’s functionality by implementing live updates, advanced user interactions, and data automation features. The focus shifted from foundational setup to **real-time reliability, intelligent data retrieval, and user-defined engagement features**.

Primary goals:

1. Expand live API integration and normalize responses across sports.
2. Enable real-time client communication through WebSocket channels.
3. Implement search, filtering, and alert mechanisms for enhanced usability.
4. Design background job framework for automated data refresh.
5. Integrate and store injury reports with team and player records.

Requirement Stacks

Sprint 2

ID	Requirement Description	Story Points	Priority	Sprint Number
R6	Integrate live data feeds from APIs for football, basketball, baseball, and UFC. (Begin connecting selected API endpoints and normalizing response data.)	13	1	2
R7	Build WebSocket interface for real-time game and stat updates to connected clients. (Start	8	1	2

	implementing live update mechanism.)			
R8	Implement search and filtering by sport, team, or player across datasets. (Develop frontend components once live data is integrated.)	3	1	2
R9	Implement user-defined alerts for news mentions and injury updates of subscribed teams. (Set groundwork for alert logic.)	8	2	2
R10	Implement scheduled background jobs (APScheduler) to poll for new data, injuries, and odds updates. (Plan job scheduling structure.)	5	2	2
R11	Store and display recent injury reports per team/player. (Initial database schema design.)	3	2	2

Requirement Descriptions

R6 – Expanded API Integration (13 SP)
Objective: Strengthen connections to multiple sports data sources and standardize incoming JSON responses for uniformity.
Implementation Details:

- Added more API endpoints to pull in live score and injury information.

- Developed middleware to clean and standardize key identifiers such as player IDs and team codes.

- Introduced detailed logging to track API response times, monitor rate limits, and capture error events.
 Deliverable: A dependable, cross-sport data ingestion framework with verified response mappings.
 Rationale: Establishes a stable backend foundation supporting consistent front-end updates and analytics.

R7 – WebSocket System Upgrade (8 SP)
Objective: Provide continuous real-time updates through persistent WebSocket connections.
Implementation Details:

- Created a modular WebSocket manager to oversee connection handling, broadcasting, and client subscriptions.

- Leveraged Redis Pub/Sub to deliver instant updates for scores and player injuries.
- Implemented client-side subscription logic for following specific teams or events.
  Deliverable: A reliable WebSocket layer capable of transmitting live sports data to concurrent users.
  Rationale: Improves responsiveness and eliminates the need for users to manually refresh the interface.

R8 – Enhanced Search and Filtering (3 SP)
Objective: Introduce more refined search and filtering tools following the completion of live data integration.
Implementation Details:

- Improved query endpoints to support pagination and case-insensitive lookups.
- Added front-end filters for sport type, player statistics, and performance metrics.
- Incorporated a debounce mechanism to limit unnecessary backend requests during active typing.
  Deliverable: An optimized search interface seamlessly connected to FastAPI endpoints.
  Rationale: Enhances usability by enabling fast, precise data exploration across large datasets.

R9 – Customizable Alerts (8 SP)
Objective: Build an alert feature that informs users about team injuries or relevant news.
Implementation Details:

- Defined a new alert schema capturing user, team, and event metadata.
- Integrated third-party news and injury data sources to initiate automated alert triggers.
- Designed a prototype user interface for managing alert settings and delivery preferences.
  Deliverable: A working proof-of-concept for the alert engine with database logging and event hooks.
  Rationale: Adds personalization and helps users stay informed about the teams and players they follow.

R10 – Automated Job Scheduling (5 SP)
Objective: Implement background automation for fetching and refreshing sports data.
Implementation Details:

- Configured APScheduler to execute periodic data collection tasks.
- Added tracking for job runs, including performance metrics and error reporting.
- Developed administrative controls to pause or resume scheduled jobs as needed.
  Deliverable: A fully functional scheduler maintaining continuous data synchronization.
  Rationale: Guarantees up-to-date system information without requiring manual processes.

R11 – Injury Data Management (3 SP)
Objective: Capture and display current injury details for players and teams within the database and interface.
Implementation Details:

- Designed a dedicated injury_reports table linked to the players and teams entities.
- Built a UI component that shows injury summaries and estimated recovery timelines.
- Connected scheduled tasks to automatically refresh stored injury data.
  Deliverable: A functional module for tracking and presenting injury information within the app's dashboard.
  Rationale: Provides analytical depth and context for users monitoring player health and team readiness.