Sprint 1 Artifact Document

Project: EECS 581 – Project 3 (Fanalytics)
Team 27: Asa Maker | Brandon Dodge | Zach Sevart | Josh Dwoskin | Ebraheem AlAamer
Sprint Number: 1
Date: October 23 2025

Sprint 1 Objectives

Sprint 1 established the project's technical foundation by focusing on backend infrastructure, real-time data ingestion, and client-side presentation pipelines.

The primary goals were:

1. Build a RESTful FastAPI backend integrated with PostgreSQL and Redis.
2. Connect live sports data APIs (football, basketball, baseball, UFC) and normalize their structures.
3. Enable real-time updates through WebSocket channels.
4. Develop an initial front-end interface capable of displaying live statistics.
5. Provide search and filtering across all datasets.

Requirement Stacks

ID	Requirement Description	Story Points	Priority	Sprint Number
R1	Locate RESTful FastAPI backend to serve sports data (teams, players, stats, odds) using PostgreSQL and Redis caching.	13	1	1
R2	Integrate live data feeds from APIs for football, basketball, baseball, and	13	1	1

	UFC. Normalize and store in unified schema.			
R3	Build WebSocket interface for real-time game and stat updates to connected clients.	8	1	1
R4	Create front-end interface (React Native or PWA) to display teams, players, and stats.	8	1	1
R5	Implement search and filtering by sport, team, or player across datasets.	3	1	1

Requirement Descriptions

R1 – RESTful FastAPI Backend Setup (13 SP)

Objective: Establish a modular backend service that exposes endpoints for teams, players, stats, and odds.

Implementation Notes:

- Created FastAPI application with route groups (/teams, /players, /games).
- Integrated PostgreSQL (via SQLAlchemy) for structured storage and Redis for caching heavy reads.
- Configured CORS, request validation, and Pydantic models for consistent schemas.

Deliverable: Running FastAPI instance connected to PostgreSQL and Redis on local Docker compose stack.
Rationale: Provides the data backbone required for all subsequent sprint goals.

R2 – Live Data Feed Integration (13 SP)

Objective: Ingest and normalize multi-sport data from external APIs.

Implementation Notes:

- Connected to public/free sports APIs (e.g., SportsRadar or API-Sports) for football, basketball, baseball, UFC.
- Implemented ETL pipeline to standardize incoming JSON into a canonical schema (Player, Team, Game, Metric).
- Built scheduler (using Celery or FastAPI background tasks) to refresh data periodically.

Deliverable: Unified PostgreSQL tables (teams, players, games) with up-to-date entries.
Rationale: Ensures real-time reliability and supports analytics engine accuracy.

R3 – WebSocket Interface for Real-Time Updates (8 SP)

Objective: Enable clients to receive live game and stat changes instantly.

Implementation Notes:

- Added WebSocket endpoint (/ws/updates) to FastAPI.
- Subscribed Redis Pub/Sub channels to broadcast stat changes to connected clients.
- Created test client scripts to simulate live score updates.

Deliverable: Real-time dashboard view where scores auto-refresh without page reload.
Rationale: Provides live experience for sports fans and analysts alike.

R4 – Front-End Interface (8 SP)

Objective: Develop cross-platform UI for visualizing teams, players, and stats.

Implementation Notes:

- Designed React Native (PWA variant for desktop) frontend.
- Implemented screens: Home, Teams, Players, Game Stats.
- Connected to FastAPI backend via REST and WebSocket.
- Used Recharts and Tailwind for data visualization and responsive layout.

Deliverable: Functional prototype showing data pulled from API with live updates.
Rationale: Provides user interaction and validates backend data flow end-to-end.

R5 – Search and Filtering (3 SP)

Objective: Allow users to find specific teams or players quickly across sports.
Implementation Notes:

- Added query parameters (/players?name=, /teams?sport=).
- Created frontend search bar and dropdown filters for sport selection.
- Implemented server-side indexing and Redis cache for fast responses.

Deliverable: Responsive UI filter capability.
Rationale: Improves usability and data navigation across large datasets.

Sprint 1 Artifacts

Artifact	Description / Deliverable
System Architecture Diagram	Visualized relationships between FastAPI, PostgreSQL, Redis, and React UI components using the defined data flow.
API Specification (OpenAPI)	Auto-generated FastAPI docs (/docs) showing endpoints for teams, players, games.
Database ERD	PostgreSQL ER diagram detailing primary/foreign keys for Player, Team, Game, Metric.
WebSocket Demo Script	Python client showing live score push events over WebSocket.
Frontend Prototype	React Native/PWA demo screen showing team list and stat dashboard.
Redis Cache Metrics	Performance benchmarks before/after caching with latency reductions logged.