Enhancing Sports Fandom Through IoT: Backend Dashboard, Monitoring, and Embedded Client Development

Project Requirements and Specifications

Sports IOT LLC

Sports IOT

Enhancing the fan experience through internet-connected sports memorabilia

By: John Swensen and James Swensen

30-FA25-SP26-SIL-GEN

Gaynay Doo

Modeste Mahouna Houenou

September 25, 2025

TABLE OF CONTENTS

I. Introduction 3

II. System Requirements Specification 3

II.1. Use Cases 3

Use Case 1: Register Device 4

Use Case 2: Receive Live Score Updates 4

Use Case 3: Monitor Dashboard 5

Use Case 4: Push OTA Firmware Update 5

Use Case 5: Scrape and Store Sports Data 5

II.2. Functional Requirements 6

II.2.1 Device Registration and (Initial) Connectivity 6

II.2.2 Data Storage 7

II.2.3 API Layer 7

II.2.4 Dashboard / Admin UI 7

II.1.5 Device Connectivity and Behavior 8

II.3. Non-Functional Requirements 8

III. System Evolution 9

IV. Glossary 10

V. References 11

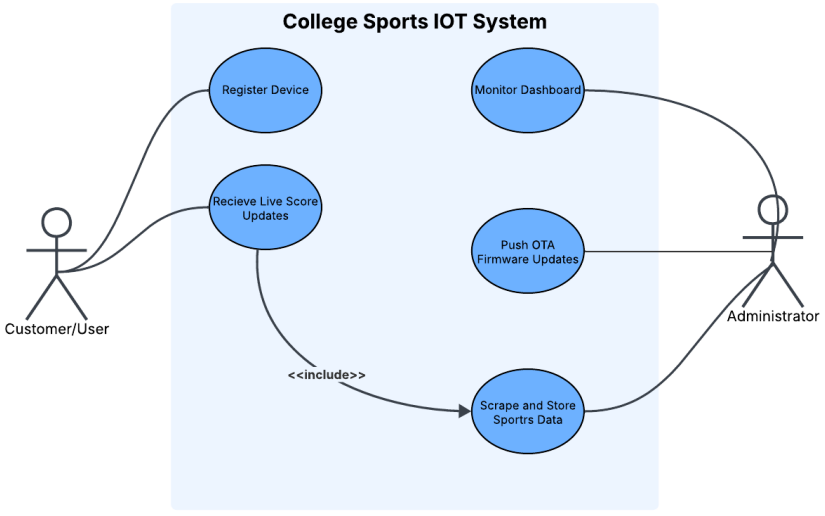
I. Introduction

This project is an Internet of Things (IoT) platform designed to enhance college sports fandom by connecting small desk-sized display devices to a centralized backend and web dashboard. Each device can be registered to a school or team and automatically shows real-time scores, team colors, and event information during games. The system integrates embedded clients, a backend API and database, a live-score scraper, and a browser-based dashboard to provide seamless device onboarding, timely updates, remote monitoring, and over-the-air firmware updates, ensuring fans and administrators receive accurate and engaging information with minimal effort.

II. System Requirements Specification

This section outlines the key functional and non-functional requirements, user stores and acceptance scenarios of the Sports IOT system. The project delivers a desk-sized, logo-branded display that lights up to celebrate a user's chosen college sports team, supported by a modular backend for real-time score updates. The functional requirements describe how the system will scrape live NCAA and other sports data, parse and persist scores to a Postgres database, notify an API layer, and push updates to devices via WebSockets or polling. Non-functional requirements define performance, scalability, and security constraints, such as update latency within ten seconds, support for hundreds of devices, and secure authentication of HTTP and WebSocket endpoints. Complementary user stories capture the perspectives of customers, administrators, and operators, thus covering device registration, live score updates, dashboard monitoring, OTA firmware updates, and scraper operations with positive and negative acceptance scenarios for each. Together, these requirements provide a comprehensive view of the system's goals, expected behaviors, and interactions between users, devices, and the back-end services.

II.1. Use Cases



Use Case 1: Register Device

Actors	Customer (or Administrator)
Pre-Conditions	Device is powered on and connected to Wi-Fi; user has valid credentials.
Post-Conditions	Device is registered to the correct school and sport; registration stored in database.
Basic Path	• User powers on new device. • Device creates a temporary access point and is discovered via mDNS. • User enters credentials and selects school and sport through API/browser. • Device registers with API. • API saves registration details in the database and confirms success.
Alternative Path	Registration fails due to invalid credentials → API returns an error; device stays in idle (white LED) mode.
Related Requirements	FR-08, FR-10; NFR-04

Use Case 2: Receive Live Score Updates

Actors	Device
Pre-Conditions	Device is registered and connected via WebSocket.
Post-Conditions	LED colors reflect the current game status within 10 seconds.
Basic Path	• Scraper pulls live stats from NCAA site and stores parsed data in the database. • Database triggers notification to API layer. • API pushes update over WebSocket to device. • Device changes LED color (white → blue on win).
Alternative Path	Device loses WebSocket connectivity → falls back to polling every 5 minutes.
Related Requirements	FR-01, FR-05, FR-06, FR-09; NFR-01, NFR-02

Use Case 3: Monitor Dashboard

Actors	Dashboard Operator / Admin
Pre-Conditions	Operator is logged into the dashboard through a browser.
Post-Conditions	Operator sees current device status, parsing errors, and system health.
Basic Path	• Operator navigates to dashboard page. • Dashboard shows live game status, connected devices, and parsing status. • Operator can add or remove sports/schools.
Alternative Path	Scraper or DB stops sending notifications → dashboard displays error alerts for investigation.
Related Requirements	FR-07; NFR-06

Use Case 4: Push OTA Firmware Update

Actors	Administrator
Pre-Conditions	Device is online and authenticated.
Post-Conditions	Device downloads and installs new firmware.
Basic Path	• Admin selects firmware version in dashboard/API. • OTA update is triggered. • Device downloads firmware and installs it. • Device restarts with updated firmware.
Alternative Path	Device has poor connectivity → update fails gracefully and retries later.
Related Requirements	FR-10; NFR-01

Use Case 5: Scrape and Store Sports Data

Actors	Scraper Service
Pre-Conditions	Scraper service is running normally; source site is reachable.
Post-Conditions	Parsed data and raw HTML snapshots stored in the database.
Basic Path	• Scheduled pull is triggered. • Scraper downloads live stats from NCAA site. • Data is parsed and stored in the database. • Raw HTML snapshot saved for debugging. • Database sends notifications to API layer
Alternative Path	Source site unavailable → system logs the error, does not save invalid data, shows alert on dashboard.
Related Requirements	FR-01, FR-03, FR-05; NFR-07

II.2. Functional Requirements

The functional requirements define the core capabilities and behaviors of the Sports IoT system, focusing on how it interacts with users, devices, and external data sources. These requirements specify essential operations such as device registration, real-time score updates, data storage, dashboard monitoring, and firmware management. Most of the requirements are derived from the needs of our client and system use cases, ensuring the solution is both practical and scalable. The following subsections detail the specific functionalities that the system must support to deliver a seamless and engaging sports fandom experience.

II.2.1 Device Registration and (Initial) Connectivity

ID	Requirement	Source	Priority
FR-02	**[Pluggable Parsing]** The system shall support multiple sports/schools through pluggable parsers that can be added or replaced without changing core logic.	Client Requirement for flexibility	Level 1 (Desirable)

II.2.2 Data Storage

ID	Requirement	Source	Priority
FR-01	**[Scraper]** The system shall periodically scrape live NCAA (or other) stats and persist parsed data to the database.	Client requirement for real-time data	Level 0 (Essential)
FR-03	**[Snapshot Storage]** The scraper shall record raw HTML snapshots alongside parsed results for debugging purposes.	Admin need for troubleshooting	Level 0 (Essential)
FR-04	**[Database Storage]** The database shall store game scores, events, device registrations, and parsed errors.	Business requirement	Level 0 (Essential)
FR-05	**[Database Notification]** The database shall publish changes to the API layer via event triggers (e.g. Postgres NOTIFY/LISTEN).	Design requirement for real-time updates	Level 0 (Essential)

II.2.3 API Layer

ID	Requirement	Source	Priority
FR-06	**[API Hub]** The FastAPI layer shall act as the single hub for communication between devices, dashboards, and browsers, exposing HTTP and WebSocket endpoints.	Client Requirement	Level 0 (Essential)

II.2.4 Dashboard / Admin UI

ID	Requirement	Source	Priority
FR-07	**[Dashboard Functions]**	Client requirement	Level 0 (Essential)

	The dashboard shall show live game status, connected devices, parsing errors, and system health. It shall allow administrators to add/remove sports or schools.		

II.1.5 Device Connectivity and Behavior

ID	Requirement	Source	Priority
FR-08	**[Device Connectivity]** Devices shall connect to the API via Wi-Fi and register with a school and sport. Devices shall primarily receive updates through WebSockets, with a fallback to polling every 5 minutes if the connection drops.	Client Requirement for reliability	Level 0 (Essential)
FR-09	**[LED State Management]** Devices shall change LED colors based on received updates: White = idle/normal; Blue = win condition; Yellow = debug/error.	Client requirement for clear visual cues	Level 0 (Essential)
FR-10	**[Device Setup & Updates]** Devices shall support initial setup via temporary access point + mDNS and receive firmware updates over-the-air (OTA) when available.	Business requirement for easy setup/maintenance	Level 0 (Essential)

II.3. Non-Functional Requirements

These requirements describe *how well* the system should do every single thing it's supposed to do. They cover qualities like performance, reliability, scalability, and security, all of which are the aspects that make our system run smoothly, stay secure, and keep up as it grows. By setting these expectations, we make sure the Sports IoT system isn't just capable, but also fast, dependable, and easy to maintain over time.

Non-Functional Requirement	Description
[NFR-01] Reliability / Fault Tolerance	The system shall handle scraper or API outages gracefully; devices shall revert to idle mode or polling fallback when no updates are received.
[NFR-02] Performance	The system shall deliver score updates from DB → API → device within 10 seconds under normal operating conditions.
[NFR-03] Scalability	The system shall support hundreds of devices and multiple schools without requiring major re-architecture.
[NFR-04] Security	All device registrations, HTTP, and WebSocket endpoints shall require secure authentication tokens and use HTTPS/WSS.
[NFR-05] Modularity	The system shall use modular, Dockerized services and support pluggable parsers for new sports with minimal code changes.
[NFR-06] Usability	The dashboard shall be simple enough for non-technical staff to monitor devices and system health effectively.
[NFR-07] Maintainability / Loose Coupling	Event-driven modules (scraper, DB, API) shall remain loosely coupled to enable independent scaling, maintenance, and debugging.

III. System Evolution

The Sports IoT system is designed with flexibility and adaptability in mind to accommodate anticipated changes in hardware, software, and user needs. Several fundamental assumptions underlie the current design, and potential evolution points have been identified to mitigate risks and support future growth.

Assumptions:

- Devices will have reliable Wi-Fi connectivity for real-time updates, with fallback to periodic polling.
- The system will initially support a limited number of sports and schools, using pluggable parsers for each sport.
- Firmware updates will be delivered over-the-air (OTA) to ensure devices remain current without manual intervention.
- Postgres will serve as the database, supporting event-driven notifications via NOTIFY/LISTEN.
- Users will primarily access the system via a web dashboard, with basic administrative and monitoring functionality.

Anticipated Evolution:

1. **Hardware Evolution:**

    a. New microcontroller models or LED hardware may require updated drivers or firmware adjustments.
    b. Devices may need additional connectivity options, such as cellular or Bluetooth, if Wi-Fi becomes unavailable or unreliable.
    c. Future devices may include more sensors or indicators, necessitating firmware and API expansion.
2. **Software Evolution:**
    a. The scraping system may need updates if the NCAA or other data sources change their HTML structure or API endpoints.
    b. Additional sports, schools, or event types may be added, requiring new parsers or modifications to the API and database schema.
    c. Dashboard features may evolve based on administrator feedback, including mobile-friendly access or enhanced reporting.
3. **User and Client Evolution:**
    a. Customers may request more customization for LED colors, notification preferences, or device grouping.
    b. Increased adoption may lead to scaling requirements for the API, database, and WebSocket infrastructure.
    c. Security requirements may evolve, necessitating stronger authentication, encryption, or audit logging.

Design Considerations:
To accommodate these potential changes, the system has been designed with modularity and loose coupling:

- Pluggable parsers allow adding new sports without changing core scraper logic.
- Dockerized services facilitate deployment across different hardware and operating systems.
- Event-driven architecture enables independent scaling and debugging of scraper, database, and API modules.
- Firmware is designed for OTA updates, ensuring long-term device compatibility and feature enhancements.

By documenting these assumptions and potential evolution points, we, the current design team, and any future teams that take over can better anticipate risks, avoid design decisions that would constrain future growth, and ensure that the system remains adaptable to changing requirements and technologies.

IV. Glossary

WebSockets: Also known as the WebSocket API, it is a computer communications protocol that makes it possible to open a two-way (bidirectional) interactive

communication between the client (like the user's web browser) and a server over a long-lived connection.

mDNS: This stands for Multicast DNS, which is a protocol that helps devices on the same local network to resolve hostnames (like printer.local) to IP addresses, without needing a central DNS server.

API: An Application Programming Interface is a set of rules, protocols, and tools that allow different software programs or components to communicate and interact with each other. They define the way requests and responses are formatted and exchanged, making it possible for one program to ask another for specific data or actions.

SQL Database: A type of database that stores and organizes data in highly structured tables of rows and columns, similar to a spreadsheet, and lets you manage that data using the SQL (Structured Query Language).Scraper: A piece of software (sometimes called a "web scraper") that automatically collects data from websites or APIs.

Docker: This is an open-source platform that allows developers to package applications and all their dependencies into a standard unit called a "container". These containers can then run on any system that supports Docker, regardless of the underlying hardware or operating system, ensuring consistent behavior across different environments. Dockerized or containerized services refer to software applications that have been bundled into containers using Docker.

Wi-Fi: A wireless networking technology that uses radio waves to provide high-speed internet and network connections to devices like laptops, smartphones, tablets, and smart gadgets, without cables.

IoT: Refers to the network of physical objects that are embedded with sensors, software, and other technologies to connect and exchange information with other devices and systems over the Internet.

LED: It stands for Light-Emitting Diode and is a semiconductor device that emits light when an electrical current passes through it.

V. References

[1] J. Schneider and I. Smalley, "What is a microcontroller? | IBM," *www.ibm.com*, Jun. 04, 2024. https://www.ibm.com/think/topics/microcontroller

[2] Mozilla, “The WebSocket API (WebSockets),” *MDN Web Docs*, Nov. 28, 2019. https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API

[3] “Multicast DNS,” *Wikipedia*, Nov. 19, 2020. https://en.wikipedia.org/wiki/Multicast_DNS
[4] “Introduction to web APIs - Learn web development | MDN,” *MDN Web Docs*, Dec. 19, 2024. https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Client-side_APIs/Introduction

[5] Microsoft, “What is a SQL Database? | Microsoft Azure,” *azure.microsoft.com*, 2025. https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-sql-database

[6] Tafara Muwandi, “What Are Scrapers and Why Should You Care? | F5 Labs,” *F5 Labs*, Aug. 02, 2024. https://www.f5.com/labs/articles/threat-intelligence/what-are-scrapers-and-why-should-you-care

[7] GeeksforGeeks, “What is Docker?,” *GeeksforGeeks*, Jul. 24, 2020. https://www.geeksforgeeks.org/devops/introduction-to-docker/ (accessed Sep. 12, 2025).

[8] “The future of work is here with Wi-Fi 7.,” *Cisco*, Jul. 2025. https://www.cisco.com/site/us/en/learn/topics/networking/what-is-wi-fi.html (accessed Sep. 12, 2025).

[9] Wikipedia Contributors, “Light-emitting diode,” *Wikipedia*, May 27, 2019. https://en.wikipedia.org/wiki/Light-emitting_diode