Project Assignment 1:

Defining Functional Requirements

and

Writing User Stories with Acceptance Scenarios

Team Name: 26-FA25-SP26-SIL-GEN

Team Members**:** Gaynay Doo, Modeste Mahouna Houenou

Course Code: CPT_S-421-PULLM-1-LEC

Date: September 19, 2025

1. Functional Requirements (FRs)

(List at least 8–10. Use FR-01, FR-02 ... numbering)

- **FR-01:** The system shall periodically scrape live NCAA (or other) stats and persist parsed data to the database.
- **FR-02:** The system shall support multiple sports/schools through pluggable parsers that can be added or replaced without changing core logic.
- **FR-03:** The scraper shall record raw HTML snapshots alongside parsed results for debugging purposes.
- **FR-04:** The database shall store game scores, events, device registrations, and parsed errors.
- **FR-05:** The database shall publish changes to the API layer via event triggers (e.g. Postgres NOTIFY/LISTEN).
- **FR-06:** The FastAPI layer shall act as the single hub for communication between devices, dashboards, and browsers, exposing HTTP and WebSocket endpoints.
- **FR-07:** The dashboard shall show live game status, connected devices, parsing errors, and system health. It shall allow administrators to add/remove sports or schools.
- **FR-08:** 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.
- **FR-09:** Devices shall change LED colors based on received updates:
    - White = idle/normal
    - Blue = win condition
    - Yellow = debug/error
- **FR-10:** Devices shall support initial setup via temporary access point + mDNS and receive firmware updates over-the-air (OTA) when available.

2. Non-Functional Requirements (NFRs)

(List at least 5–7. Use NFR-01, NFR-02 ... numbering)

- **NFR-01:** 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:** The system shall deliver score updates from DB -> API -> device within 10 seconds.
- **NFR-03:** The system shall support hundreds of devices and multiple schools without major re-architecture.
- **NFR-04:** All device registrations, HTTP, and WebSocket endpoints shall require secure authentication tokens and use HTTPS/WSS.
- **NFR-05:** The system shall use modular, Dockerized services and support pluggable parsers for new sports and minimal code changes.
- **NFR-06:** The dashboard shall be simple enough for non-technical staff to monitor devices and system health.
- **NFR-07:** Event-driven modules (scraper, DB, API) shall remain loosely coupled to enable independent scaling and debugging.

3. User Stories with Acceptance Scenarios

*(Write as many as reasonably possible at this stage. Each story must be peer-reviewed in the team. Add **at least two scenarios per story** — one positive, one negative/edge case.)*

US-01: Device Registration

As a customer, I want to register my school's device through the API so that it displays the correct colors for our games.

Acceptance Scenarios (Gherkin):

Scenario 1: Positive Flow

Given a new device connects to the API

When the user registers it with a school and sport

Then the device is saved in the DB and updates reflect correctly

Scenario 2: Negative/Edge Case

Given a device attempts registration with invalid credentials

When the API validates

Then registration is rejected, and an error is returned.

US-02: Live Score Updates

As a customer, I want to receive score updates instantly so that my LEDs reflect the current game status.

Acceptance Scenarios (Gherkin):

Scenario 1: Positive Flow

Given a registered device connected via WebSocket

When the DB registers a score change

Then the device updates its LEDs within 10 seconds

Scenario 2: Negative/Edge Case

Given a registered device loses WebSocket connectivity

When the connection fails

Then the device falls back to polling every 5 minutes and eventually shows the updated score

---

US-03: Dashboard Monitoring

As a dashboard operator, I want to see which devices are online and any parsing errors so that I can troubleshoot quickly.

Acceptance Scenarios (Gherkin):

Scenario 1: Positive Flow

Given the system is running normally

When the operator is logged in through the browser to view the dashboard

Then the dashboard shows connected devices, parsing status (latest scraped data is visible), and system health

Scenario 2: Negative/Edge Case

Given the scraper fails or DB stops sending notifications

When the operator views the dashboard

Then error alerts are visible for investigation

US-04: OTA Firmware Updates

As an administrator, I want to push firmware updates to devices so that they stay current without manual intervention.

Acceptance Scenarios (Gherkin):

Scenario 1: Positive Flow

Given a device is online

When an OTA is triggered

Then the device donwloads and installs the new firmware successfully

Scenario 2: Negative/Edge Case

Given a device has poor connectivity

When an OTA update is pushed

Then the update fails gracefully and retries later

US-05: Browser Access

As a customer, I want to log in through the browser and set my sport/event preferences so that I only receive updates relevant to me.

Acceptance Scenarios (Gherkin):

Scenario 1: Positive Flow

Given a valid/registered user is logged in through the browser

When they select their preferred sports/teams/events and save changes

Then the system stores the preferences in the database

And notifications are filtered according to the saved preferences

Scenario 2: Negative/Edge Case

Given a registered user is logged in through the browser

When they attempt to save new preferences

Then the system fails to connect to the database

And an error message is shown

And the user’s previous preferences remain unchanged

US-06: Scraper Operations

As a system operator, I want the scraper to periodically pull live sport stats and store them in the database so that devices and dashboard always have the latest data.

Acceptance Scenarios (Gherkin):

Scenario 1: Positive Flow

Given the scraper service is running normally

When a scheduled pull is triggered

Then new stats are parsed and stored in the database

And notifications are sent to connected devices and dashboard

Scenario 2: Negative/Edge Case

Given the scraper service is running

When the external stats source is unavailable

Then the system logs the error

And no invalid/partial data is saved to the database

And an alert is shown on the operator dashboard

---

4. Brainstorming and GenAI Reflection

Step A: Team Brainstorming

(Document initial FRs, NFRs, stories, and scenarios created without AI. Record 3–4 key decision points from your discussion.)

Initial FRs (without AI):

- FR-01: Scraper periodically pulls NCAA stats and saves to DB.
- FR-02: Device registration tied to school and sports
- FR-03: The scraper shall record raw HTML snapshots alongside parsed results for debugging purposes.
- FR-04: Database stores scores, events, and device registrations.
- FR-05: The database shall publish changes to the API layer via event triggers (e.g. Postgres NOTIFY/LISTEN).
- FR-06: The FastAPI layer shall act as the single hub for communication between devices, dashboard, and browser, exposing HTTP and WebSocket endpoints.
- FR-07: Dashboard shows live game status, connected devices, and errors.
- FR-08: Devices support basic Wi-Fi setup with temporary AP/mDNS and devices fall back to polling every 5 minutes if connection fails.
- FR-09: LEDs colors would change appropriately based on received updates.

Initial NFRs (without AI):

- NFR-01: Updates must reach devices in <10 seconds.
- NFR-02: System handles scraper/API outages gracefully.
- NFR-03: Secure device registration and API endpoints (tokens, HTTPS).
- NFR-04: Secure authentication tokens are used during device registration.
- NFR-05: System scalable to hundreds of devices.
- NF5-06: The dashboard is simple to get accustomed to and to use by non-technical staff.
- NFR-07: System is easy to debug.

Initial User Stories (without AI):

- US-01: Device Registration
- US-02: Live Score Updates
- US-03: Dashboard Monitoring
- US-04: OTA Firmware Updates
- US-05: Browser Access

Acceptance Scenarios (without AI):

- Positive and negative flows for US-01 through US-05.

- Decision Point 1: We decided device registration must be linked to school and sport at setup time so that updates display the right team colors (avoiding mismatch issues).
- Decision Point 2: We decided the dashboard should also show parsing errors and overall system health metrics including whether all registered devices are getting updates as they should, not just scores. This would allow operators to troubleshoot quickly without having to constantly guess.
- Decision Point 3: We decided to require WebSockets for fast updates, but to also include make sure the dashboard fall back to polling every 5 minutes if WebSocket fails so that devices remain functional if connection drop.

Step B: GenAI-Assisted Brainstorming

(Summarize what GenAI suggested — new FRs, NFRs, user stories, or scenarios.)

- GenAI Suggestion 1: Add an explicit scraper user story (US-06), so that FR-10 (also suggested by GenAI) has coverage.
- GenAI Suggestion 2: Mention titles for each of the FRs and call out browser-based preference setting separately.
- GenAI Suggestion 3: Add OTA update requirement as its own story with postive/negative scenarios.

Step C: Refined Requirements & Stories

(Update Step A with insights from GenAI. Clearly mark new/modified items — e.g., FR-06 ***[Added after GenAI]****.)*

- **FR-06 [Part of it was reworded by GenAI]:** Provide a browser-based dashboard (different from the browser for the user to set preferences) for operators to monitor devices, scores, and errors.
- **FR-10 [Added after GenAI]:** Scraper must periodically pull live stats and push updates into DB.
- **US-05 [Part of it was reworded by GenAI]:** Browser Access for customer preferences.
- **US-06 [Added after GenAI]:** Scraper Operations story with error handling.

Step D: Reflection (200–300 words)

- How did you feel about using GenAI in this exercise (e.g., empowering, surprising, confusing, over-reliant)?
    - Honestly, using GenAI felt empowering because it helped us see gaps and possibilities that we hadn’t fully considered during our initial team brainstorming. It wasn’t confusing but rather acted like a smart assistant that could quickly organize our thoughts and suggest improvements, which helped a bit in reducing the risk of missing key requirements. It did also remind us not to be over-reliant on it though because at times it gave suggestions that weren’t necessarily appropriate or out of out of context.

- In what ways did GenAI change or improve your brainstorming compared to your team's initial work? (Consider clarity, creativity, and coverage.)
    - GenAI noticeably improved our brainstorming compared to our initial work. Ofr example it suggested adding explicit titles for each functional requirement and distinguishing between the operator dashboard and browser used by customer to set preferences. This clarified our documents and made the stories easier to read. It also encouraged us to think about cover as in ensuring that each requirement had a corresponding user story. This was something we partially overlooked in our first pass
- Did GenAI help you uncover new functional, non-functional, or business requirements that you had not considered before? Provide examples.
    - GenAI did help us uncover new or overlooked requirements. It proposed a dedicated scraper user story (US-06) to support a new functional requirement (FR-10) for periodic data pulls, and recommended creating an OTA update requirement as its own story with positive and negative scenarios. These additions expanded our scope beyond simple score updates and highting operational concerns such as error handling.