EECS 581 – Project 3 (Fanalytics)

Team 27 – Sprint 4 Artifact Document
Team Members: Asa Maker | Brandon Dodge | Zach Sevart | Josh Dwoskin | Ebraheem AlAamer
Sprint Number: 4
Date: November 21 2025

R18 – Containerize services with Docker and deploy via Docker Compose. (3 SP)

Objective:

Ensure consistent, reproducible deployment environments by containerizing all services and orchestrating them with Docker Compose.

Implementation Notes:

- Created Dockerfiles for FastAPI backend, Vercel proxy service, Redis cache, and Supabase auth layer.
- Introduced multi-stage Docker builds to reduce image size and improve build speed.
- Developed a docker-compose.yml file defining service networking, environment variables, and volume persistence.
- Added hot-reload support for backend containers in development mode.
- Verified container interactions including WebSockets, Redis connections, and DB migrations.

R19 – Create admin dashboard to manage API keys, data sources, and logs. (5 SP)

Objective:

Provide an internal admin dashboard for system maintainers to manage configurations, monitor health, and inspect logs.

Implementation Notes:

- Built new FastAPI admin routes protected with role-based access control (RBAC).
- Added Supabase-authenticated admin roles so only permitted users can access dashboard views.
- Developed UI sections for:
    - API key management
    - Log viewer (backend logs + ingestion logs)
    - Data source connectivity checks
- Implemented server-side pagination for large log outputs.
- Integrated API key rotation with automatic revalidation.

R20 – Implement data deduplication and validation pipeline for ingest jobs. (5 SP)

Objective:

Prevent duplicate entries and ensure incoming external sports data is clean and valid before storage.

Implementation Notes:

- Added hashing mechanism to detect duplicate events, games, and injury updates.
- Integrated schema validation using Pydantic models for all incoming API responses.
- Created pre-ingest checks ensuring required fields (game date, player IDs, odds fields) are present.
- Added reporting outputs for ingestion failures and mismatch counts.

- Implemented automated correction rules for common inconsistencies.

R21 – Integrate email as an additional notification channel for user alerts. (3 SP)

Objective:

Expand user notification options by adding email alerting support for news, injuries, and subscribed team events.

Implementation Notes:

- Integrated SendGrid API with backend alert scheduler.
- Added user preference fields (email opt-in/out, frequency settings) in Supabase.
- Implemented email templates for:
    - Injury alerts
    - Breaking news
    - Live update summaries
- Configured retry logic for failed email deliveries.
- Added admin dashboard logs for email send history.

R22 – Add logging, monitoring, and error handling across services with Prometheus and Grafana. (5 SP)

Objective:

Track service performance, monitor system health, and detect errors proactively.

Implementation Notes:

- Added Prometheus exporters to FastAPI backend, Redis, and containerized services.
- Built Grafana dashboards showing:
    - API response latency
    - Cache hit/miss ratios
    - Ingestion job performance
    - Error rates and exceptions
- Standardized backend logging format using JSON-based structured logs.
- Implemented custom error middleware returning consistent error payloads.
- Added alerts for high-latency endpoints and ingest failures.

R23 – Create user profile management including preferences and subscription settings. (3 SP)

Objective:

Enable users to manage personal settings, subscriptions, and notification preferences.

Implementation Notes:

- Expanded Supabase user profiles with new fields (favorite teams, alert types, theme settings).
- Implemented profile edit UI components in the frontend.
- Added preference-based filtering to notification and comparison modules.
- Integrated membership tier handling (free vs premium features).
- Added backend validation and update routes.

R24 – Add mobile notifications for live game updates and subscribed team news. (5 SP)

Objective:

Deliver real-time updates directly to mobile devices via push notifications.

Implementation Notes:

- Integrated Firebase Cloud Messaging (FCM) for cross-platform push support.
- Added device registration tables in Supabase to store tokens.
- Implemented backend functions to dispatch live alerts during games.
- Ensured compatibility with WebSocket-driven live data streams.
- Added user settings for enabling or disabling mobile push alerts.

R25 – Design responsive UI supporting both desktop and mobile clients. (5 SP)

Objective:

Ensure seamless user experience across all screen sizes with consistent UI behavior.

Implementation Notes:

- Applied responsive grid layout system using TailwindCSS.
- Refactored major components (team pages, dashboards, comparison tools) for breakpoints.
- Updated chart components to resize dynamically.
- Added “compact mode” for mobile views.

- Conducted manual tests on various resolutions and device emulators.

R26 – Add authentication (JWT) for user accounts and subscriptions. (5 SP)

Objective:

Strengthen user authentication and secure access to personalized features.

Implementation Notes:

- Implemented JWT-based session tokens with refresh token support.
- Integrated Supabase auth events with backend subscription logic.
- Enforced authentication middleware across protected endpoints.
- Added token expiry handling and automatic refresh flows.
- Completed regression tests for login, logout, and session renewal.

R27 – Build Discord bot/webhook integration for sending alerts and live updates. (3 SP)

Objective:

Extend the alerting system to Discord, enabling servers and users to receive automated sports updates.

Implementation Notes:

- Built backend webhook dispatch module for Discord channels.
- Implemented Discord bot commands for sports lookup, player stats, and alert subscription.

- Connected bot to the existing alert scheduler to push injury, odds, and game updates.
- Added signature formatting for embeds and multi-line summaries.
- Tested delivery speed and rate limits using multiple internal channels.

R28 – Add analytics module tracking user engagement and alert metrics. (3 SP)

Objective:

Provide insights into user behavior, system performance, and alert effectiveness.

Implementation Notes:

- Created analytics event pipeline capturing searches, comparisons, alert interactions, and ESPN lookups.
- Added dashboards visualizing user engagement trends by sport and feature.
- Integrated threshold-based triggers for admin review of high or low usage areas.
- Analyzed correlations between alerts and user session spikes.

R29 – Add pagination, sorting, and infinite scrolling for lists in the UI. (3 SP)

Objective:

Improve scalability and usability for large datasets in team, player, and game listing pages.

Implementation Notes:

- Implemented paginated backend endpoints with customizable page sizes.
- Added infinite-scroll components for mobile and desktop views.
- Integrated sorting controls (alphabetical, rating, recent games, trending).

- Added loading states and skeleton placeholders for smooth UX.

R30 – Implement rate limiting and API throttling to prevent abuse. (3 SP)

Objective:

Protect backend services from excessive or malicious requests.

Implementation Notes:

- Integrated rate-limiting middleware using token bucket strategy.
- Enforced IP-based and user-based rate limits on sensitive endpoints.
- Added Redis-backed throttling for distributed consistency.
- Implemented custom 429 response format with retry-after guidance.
- Added admin dashboard logs for blocked or suspicious requests.

R31 – Enable social media sharing of stats, alerts, and dashboards. (3 SP)

Objective:

Allow users to easily share interesting insights and statistics externally.

Implementation Notes:

- Added OpenGraph meta tags and shareable link generation.
- Implemented snapshot endpoints for generating shareable stat images.
- Added UI buttons for sharing via X, Facebook, and Discord.
- Ensured privacy rules prevent sharing of user-specific content.

R32 – Set up CI/CD pipeline for automated testing and deployment to cloud infrastructure. (8 SP)

Objective:

Automate the testing and deployment lifecycle for reliability and speed.

Implementation Notes:

- Built GitHub Actions workflow for automated linting, builds, tests, and deployment.
- Integrated Docker registry publishing and version tagging.
- Automated Vercel frontend deployment on push to main branch.
- Added backend deployment pipeline with rollback support.
- Implemented test coverage thresholds and pipeline status notifications.