GG Stats — Stage 2: Capstone Project Requirements

---

1. Problem Statement

GG Stats aggregates professional Dota 2 match data from external sources (e.g., **OpenDota**), computes highlights and aggregations (e.g., hero item popularity by game phase, hero pair synergies), and exposes them via REST APIs consumed by a **React UI**. Users browse heroes, teams, and highlights; operators can trigger data refresh, monitor rate limits and circuit breakers, and validate configuration.

Business goals:

- Provide **timely, accurate, and actionable stats** for pro-scene watchers.
- Offer discoverable highlights (top heroes, duos, item builds) per patch/week.
- Maintain **reliability** despite upstream API rate limits or outages.

---

2. Functional Requirements (FRs)

2.1 Public API: Data Retrieval (User Facing)

ID	Description
FR-2.1	The system **shall** expose a REST API endpoint to **retrieve a list of all professional teams**, supporting **pagination** (limit and offset parameters).
FR-2.2	The system **shall** expose a REST API endpoint to **retrieve a list of all heroes**.
2.3	The system **shall** expose a REST API endpoint to **retrieve popular item builds** per game phase for a specified hero, allowing an **optional limit** on the number of item builds returned.
FR-2.4	The system **shall** expose a REST API endpoint to **retrieve aggregated highlights**, **bucketed by patch or week**, with configurable **limit and sorting** parameters.
FR-2.5	The system **shall** expose a REST API endpoint to **retrieve hero pair highlights** (e.g., synergy/counter metrics) for a specified **week offset**, with a configurable **limit**.

ID	Description
FR-2.6	The system **shall** expose a REST API endpoint to **retrieve new and popular hero picks**, based on current/recent professional matches.
FR-2.7	The system **shall** act as an **image proxy** by serving external images through a backend endpoint, accepting a validated external URL as a parameter.

2.2 Data Ingestion and Persistence (Internal)

ID	Description
FR-2.8	The system **shall** ingest match data, hero data, team data, player data, and rankings from the **OpenDota API**.
FR-2.9	The system **shall** persist all domain data (e.g., `Hero`, `Team`, `Player`, `ApiRateLimit`) using **PostgreSQL** and **JPA Entities**.
FR-2.10	The system **shall** compute and persist the **statistical aggregations** (e.g., item popularity, hero pair synergies) required by the public API highlight endpoints.

2.3 Operational and Administration (Operator Facing)

ID	Description
FR-2.11	The system **shall** expose an administrative endpoint to **manually trigger a refresh** of all computed aggregations (heroes, matches, rankings, etc.).
FR-2.12	The system **shall** expose an administrative endpoint to **display the current rate limit status** (e.g., remaining calls, reset time) for all external API clients.
FR-2.13	The system **shall** expose an administrative endpoint to **display circuit breaker statuses** and allow manual **open, close, or reset** actions per configured service.
FR-2.14	The system **shall** expose an administrative endpoint to display a **summary of the running system configuration** for diagnostics.
FR-2.15	The system **shall** expose standard **health and metrics endpoints** for automated monitoring.

---

3. Frontend Requirements (FRs)

ID	Description
FR-3.1	The frontend **shall** provide a user interface to **browse the Heroes list** and view the **Hero item popularity page**.
FR-3.2	The frontend **shall** provide a user interface to view all **Highlights** (single and pair views).
FR-3.3	The frontend **shall** provide a user interface to **browse professional Teams** with support for **pagination**.
FR-3.4	The frontend **shall** call the backend REST APIs and **render all results responsively** across standard desktop and mobile screen sizes.

---

4. Non-Functional Requirements (NFRs)

4.1 Performance and Scalability

ID	Description	Measurable Criteria
NFR-4.1	**Response Time (Public Read)**: Public read endpoints **shall** respond within **500ms** at the **p95** percentile for *cached* or *hot* data paths.	Threshold: 500ms (p95)
NFR-4.2	**Response Time (Cold Path)**: Public read endpoints **shall** respond within **1.5s** at the **p95** percentile for *cold* data paths (requiring internal recalculation/full DB lookup).	Threshold: 1.5s (p95)
NFR-4.3	**Throughput**: The system **shall** sustain a concurrent load of at least **100 Requests Per Second (RPS)** across all public read endpoints on modest hardware.	Threshold: 100 RPS
NFR-4.4	**Batch Tunability**: The batch ingestion process **shall** allow horizontal tuning (e.g., chunk size, concurrency limits) via **external configuration** (not code changes).	Achieved via: External configuration.

4.2 Reliability and Availability

ID	Description	Implementation Strategy
NFR-4.5	**Failure Isolation**: All external API calls **shall** be protected by **circuit breakers** and configured **timeouts** to prevent failure propagation.	Circuit Breaker pattern.

ID	Description	Implementation Strategy
NFR-4.6	**Rate Limit Adherence**: The system **shall** actively track and respect external API rate limits to **prevent upstream bans** and throttle requests to avoid "thundering herd" issues.	Rate limit tracking/throttling.
NFR-4.7	**Self-Recovery**: The system **shall** be capable of **automatically recovering** from transient upstream network or service failures without requiring operator intervention.	Retry mechanisms, circuit breakers.

4.3 Observability

ID	Description	Interface/Tooling
NFR-4.8	**Metrics and Health**: The system **shall** expose health checks and operational metrics via an industry-standard interface (e.g., **Spring Boot Actuator**).	Actuator/Prometheus compatible.
NFR-4.9	**Structured Logging**: All application logs **shall** use a **structured JSON format**.	JSON.
NFR-4.10	**Upstream API Metrics**: The system **shall** record and expose application metrics specifically tracking calls to upstream APIs (e.g., count, latency, success/failure rate of `opendota.api.call`).	Tracked: Upstream API call stats.

4.4 Deployability

ID	Description	Standard/Tool
NFR-4.15	**Packaging**: The application **shall** be packaged as a runnable **OCI container image** (e.g., using Spring Boot's build-image functionality).	Image Format: OCI
NFR-4.16	**Local Environment**: The application **shall** be runnable locally via **Docker Compose** alongside its dependencies (PostgreSQL and the frontend).	Docker Compose
NFR-4.17	**Configuration**: All operational configuration **shall** be externalized and managed via standard **`application.properties` files or environment variables**.	Properties/Env Vars

5. Use Cases

UC-1 Browse Heroes

- **Actors**: Public Visitor
- **Preconditions**: Heroes are ingested and available.
- **Main Flow**:
  1. User opens Heroes page (frontend calls `GET /heroes`).
  2. System returns list of heroes (id, name, roles, images where applicable).
  3. UI renders heroes grid.
- **Alternatives**: If no heroes found, UI shows empty state.
- **Postconditions**: None.

UC-2 View Hero Popular Items

- **Actors**: Public Visitor, Power User
- **Preconditions**: Aggregations computed for hero items by phase.
- **Main Flow**:
  1. User opens a hero detail/items page (calls `GET /heroes/{heroId}/popular-items`).
  2. System returns top items per phase and top players for the hero.
  3. UI displays per-phase item chips/cards and top players list.
- **Alternatives**: User can adjust `limit` or `playersLimit` to refine data.
- **Errors**: Invalid `heroId` → 404/empty map; DB/connectivity issues → 5xx.

UC-3 Browse Highlights

- **Actors**: Public Visitor
- **Preconditions**: Highlights computed for the selected bucket (e.g., patch/week).
- **Main Flow**:
  1. User opens Highlights page (calls `GET /highlights?bucket&value&limit&sort&weekOffset`).
  2. System returns `HighlightsDto` with top entities and metrics.
  3. UI displays cards sorted by `lift` (default) or other metric.
- **Alternatives**: User chooses a different bucket or adjusts `limit` and `weekOffset`.
- **Errors**: If no highlights are found, API returns 400 with an `ErrorResponse`.

UC-4 View Pair Highlights (Synergy/Counter)

- **Actors**: Public Visitor
- **Preconditions**: Pair highlights computed for the requested view and week.

- **Main Flow**:
  1. User selects "Pairs" view (calls `GET /highlights/pairs?view&weekOffset&limit`).
  2. System returns `HighlightsDuoDto` with pair stats (e.g., synergy rank).
  3. UI renders pairs list with metrics.
- **Alternatives**: User switches between `synergy` and `counter` views.
- **Errors**: If unavailable, API returns 400 with an `ErrorResponse`.

UC-5 Browse Teams

- **Actors**: Public Visitor
- **Preconditions**: Teams are ingested and available.
- **Main Flow**:
  1. User opens Teams page (calls `GET /teams?page&size`).
  2. System returns paginated teams or the full list if no `page` is specified.
  3. UI renders team cards with logo and stats.
- **Alternatives**: Pagination size changed by user.

UC-6 Trigger Aggregations Refresh

- **Actors**: Operator/Admin, System Scheduler
- **Preconditions**: DB reachable; upstream limits respected.
- **Main Flow**:
  1. Actor calls `POST /api/aggregations/refresh`.
  2. Service schedules/executes batch jobs (heroes, teams, players, matches, rankings).
  3. Jobs read from OpenDota, process, and persist results.
  4. Aggregations/highlights are recomputed.
- **Alternatives**: Partial refresh via job parameters (future enhancement).
- **Errors**: Circuit open or rate limit exceeded → job is throttled or deferred.

UC-7 Monitor Rate Limits and Circuit Breakers

- **Actors**: Operator/Admin
- **Main Flow**:
  1. Call `GET /api/monitoring/rate-limits` to view current status.
  2. Call `GET /api/monitoring/circuit-breakers` to view statuses.
  3. Optionally call `POST /api/monitoring/circuit-breakers/{service}/open|close|reset`.
- **Errors**: None specified, since authorization is removed.

UC-8 View Configuration Summary

- **Actors**: Operator/Admin
- **Main Flow**: Call `GET /api/configuration` → config summary returned; used for troubleshooting.

---

6. Objects, Classes, and Relationships

6.1 Domain Entities (JPA)

- `Hero` (id, name, localizedName, primaryAttr, roles, images…)
- `Team` (id, name, tag, logoUrl, rating…)
- `Player` (id, name, teamId, rankTier, avatar…)
- `NotablePlayer` (id, playerId, heroId, note…)
- `HeroRanking` (id, heroId, rank, metric, patchWeek…)
- `ApiRateLimit` (id, serviceName, remaining, resetAt…)

6.2 Repositories/DAOs

- `HeroRepository` (JPA)
- `HeroItemsDao` (JDBC aggregation queries)
- `HeroTopPlayersDao` (JDBC)
- `TeamRepository` / `Teams DAO`

6.3 Services

- `AggregationService` (`refreshPatchesAndAggregations`)
- `HighlightsService` (`getHighlights`, `getPairHighlights`)

6.4 Controllers

- `HeroesController`
- `HighlightsController`
- `ProTeamsController` (Teams)
- `AggregationsController`
- `RateLimitController`
- `CircuitBreakerController`
- `ImageProxyController`
- `ConfigurationValidationController`

6.5 Relationships

- **Hero** 1..* — **HeroRanking** (per patch/week/metric)
- **Hero** 1..* — **NotablePlayer** (players known for a hero)
- **Team** 1..* — **Player**
- **Aggregations** derived from Matches → influence Highlights and Popular Items
- **ApiRateLimit** and **CircuitBreaker** relate to external clients (composition over services)