SportNinja Mini Stats API Assignment

Build a Laravel project with a simple HTTP API to persist and retrieve individual player stats in a leaderboard format. Keep in mind that performance of stat persistence and retrieval are key aspects for this project. The Mini Stats assignment should implement the following two API endpoints:

- **Add Player Stats**: This POST or PUT API call accepts an array of name/value pairs of 'stat_name' => value. Values will always be numeric. These stats will be persisted in a SQL database and can support positive and negative numbers. The API call needs to accept the API input and r**espond as fast as possible** since this will be called millions of times a day and the client requires super low latency. When implementing this, the developer should pay attention to the persistence method since a player could have millions of individual stat values (but for a limited set of stat types (goals, shots, etc)). Concepts like stat aggregation within SQL should be considered here if to store stats in individual rows or utilize SQL aggregation techniques.

Example input data:

```
{
      "player_id": 2345,
      "stats": [
          {
              "name": "goals",
              "value": 4
          }, {
              "name": "shots",
              "value": 45
          }, {
              "name": "penalties",
              "value": 8
          }
      ]
  }
```

- **Retrieve Player Standings**: This GET API call will add up individual stats and group them by the player ID. This is basically a player leaderboard based on the defined available stats (goals, shots, etc). By passing in a stat_name into the request querystring, the API response should be sorted based on the **aggregate stat value** (goals, for example), and **ordered by player_id from most -> least**. Let's assume that there could be millions of player stat values. Possible

utilization of Redis to cache API results could be used here. When implementing this, be cognizant of invalidating cached results when new stat values are introduced to the system.

Example output data:

```
{
 "standings": [
   {
     "player_id": 2345,
     "stats": [
       {
         "name": "goals",
         "value": 460
       },
       {
         "name": "shots",
         "value": 7857
       }
     ]
   },
   {
     "player_id": 2356,
     "stats": [
       {
         "name": "goals",
         "value": 320
       },
       {
         "name": "shots",
         "value": 557
       }
     ]
   }
 ]
}
```

The below list of suggested tech for this assignment is only a suggestion. If you know of better tools or techniques for performant stat persistence and retrieval…go for it!

Suggestions for tech utilization:

- Utilizes PHP (7.x/8.x)

- Utilizes Laravel (8.x/9.x)
    - Models, Controllers, Eloquent or DB, Requests, Responses, Queue
- Utilizes SQL for stat persistence (We use MySQL, Postgres is fine)
    - Tables, Schemas, Indexes, Aggregates
    - Utilizes Redis for caching
    - Caching, Queue