GJG Backend Coding Challenge

Design a REST API endpoint that manages a game that uses a leaderboard with players submitting new scores from around the world. The service will only be responsible for submitting player scores and returning leaderboard data either globally or country-specific.
Players gain points by submitting scores and they are placed on the leaderboard by their scores. The player with the highest score will be at the top.

Requirements

- The endpoint is exposed at **/leaderboard**
- By default it will return global leaderboard (no specific country)
- For country specific queries, it is exposed at **/leaderboard/{country_iso_code}**
- Score submission endpoint is exposed at **/score/submit** (POST)
- User profile endpoint is exposed at **/user/profile**
- Each score submission can **affect the ranking** of any player below them.
- All endpoints will accept **JSON** and/or return **JSON** responses.

Rules

- You can use the language/technology you are comfortable with. We don't care if you use something that we don't use. The goal is not to validate your knowledge of a certain technology but to understand your problem solving abilities.
- The end result should be deployed on public (AWS, Heroku, GCP etc. they all have free tiers)
- The source code must be either available on a platform like Github, or sent to us by email.

Advice

- Try to design and implement your solution as you would do for real production code. Show us how you create clean, maintainable code that does awesome stuff. Build something that we'd be happy to contribute to. This is not a programming contest where dirty hacks win the game.
- Documentation and maintainability are a plus, and don't you forget those unit tests.
- We don't want to know if you can do exactly as asked (or everybody would have the same result). We want to know what you bring to the table when working on a project, what is your secret sauce. More features? Best solution? Thinking outside the box?

Evaluation

- We will first test the **endpoints deployed on cloud** and check for expected results.
- **Duration** of responses will be very important. Please try to keep your responses below 1 second. Scaling is very important. We will insert hundreds of thousands of data to your backend and see how it changes response times.
- We will lastly look at the project and see what **technologies/solutions** you came up with.

Behaviours

- Imagine that I'm rank #1500000 and there are a total of 10 million players. When I submit a score (**/score/submit**) with a score worth of 325 points (**score_worth**), it will grant me 300,000 in rankings. I will be at rank #1200000 and the rankings of people in between my old and new rank will change.

Final Notes

- Please think about scaling. The leaderboard can have millions of players and there can be millions of score submissions.
- Ignore the consequences of security related things in the endpoints. You do not need to do any authorization as this is not the point of our challenge. For example, score submission POST body has user_id for simplification purposes as that would normally be derived from JWT or other authorization mechanisms.

Sample Responses

These responses are meant to provide guidance. These properties will at least need to be present.

GET /leaderboard


```
[
  {
    "rank": 1,
    "points": 5011,
    "display_name": "gjg",
    "country": "tr"
  },
  {
    "rank": 2,
    "points": 4992,
    "display_name": "gjg_2",
    "country": "us"
  }
  ...
]
```


GET /leaderboard/{country_iso_code}


```
[
  {
    "rank": 498,
    "points": 3534,
    "display_name": "gjg_5",
    "country": "tr"
  },
  {
    "rank": 9848,
    "points": 2309,
    "display_name": "gjg_7",
    "country": "tr"
  }
  ...
]
```


POST /score/submit


```
{
    "score_worth": 125.6,
    "user_id": "d926636e-4cf1-4e19-93a0-3689d22c70a9",
    "timestamp": 1573040751
}
```

GET /user/profile