Case Overview:

Create a simple yet scalable leaderboard system for a game application. This system will allow players to submit scores, view their ranking, and retrieve the top players. A straightforward authentication mechanism (such as basic token-based authentication) is sufficient, as the focus is on implementing the leaderboard functionality. You may either deploy the services or set up a local development environment using Docker.

Requirements:

Define and implement the endpoints to fulfill each business requirement below. If working locally, please prepare a `Dockerfile` and a bash script to run the application, including clear setup instructions.

Business Requirements

1. Score Submission

- Players must be able to submit their scores for a specific game.
- Only the highest score for each player should be recorded; new submissions should update the score only if it is higher than the current one.
- Update the player's rank on the leaderboard accordingly.

2. Leaderboard Retrieval

- Players should be able to view a list of top-ranked players.
- This list should include each player’s rank, username, and score.
- Include pagination to browse the leaderboard.

3. Individual Ranking

- Players should be able to query their individual ranking within the leaderboard.
- If the player is not in the top ranks, the response should still include their position and score.

Optional:

- Simple token-based authentication to validate players submitting scores.
- Optional deployment to a cloud provider

Technical Instructions

Stack Recommendations:

- **Backend**: Node.js (with Express or NestJS)
- **Database**: Redis (for leaderboard storage), MongoDB (for user/game data if needed)

Note: The recommended technologies provided are just suggestions—you're welcome to use alternatives (e.g., AWS DynamoDB instead of MongoDB).

Implementation Steps

1. **Example API Endpoint Requirements**
   For each endpoint, include example requests and responses.
   - **Score Submission**

Request:

```
POST /leaderboard/submit-score

{
  "userId": "user123",
  "score": 250,
  "gameId": "game456"
}
```

Response:

```
{
  "userId": "user123",
  "gameId": "game456",
  "score": 250,
  "rank": 10
}
```

- **Leaderboard Retrieval**

Request:

```
GET /leaderboard/top?limit=10&page=1
```

Response:

```
[
  {
    "rank": 1,
    "username": "player1",
    "score": 300
  },
  {
    "rank": 2,
    "username": "player2",
    "score": 290
  }
]
```

- **Individual Ranking**

Request:

```
GET /leaderboard/rank?userId=user123
```

Response:

```
{
  "userId": "user123",
  "rank": 10,
  "score": 250
}
```

Submission

1. Include a README.md file with setup instructions and dependencies.
2. Share any sample API requests and responses as JSON files.
3. If deploying, provide the endpoint URL and any required access details.