**1. Please explain your choice of technologies.**

**Ans:**

I’ve used **PHP7** and **MySQL** 5.0.12 to develop this REST API without any framework dependency. For the response medium of this API I’ve used **JSON** data format.

**PHP** and **MySQL** are the most popular open source server side scripting and database in the current trend.

**The Major Benefits of using PHP and MySQL in Web Development**

* Open Source, Easy and fast maintenance.
* Superior performance, greater scalability, reliability.
* Compatible with operating system like IIS, Apache etc.
* Platforms independent and runs on Linux, Windows, or Unix.
* User-Friendly and programming offers multilingual support.
* Wider support for other popular databases like Informix, Oracle, Sybase, etc.

**JSON** stands for JavaScript Object Notation, it is a lightweight data-interchange format, "self-describing" and easy to understand and language independent.

**2. What is the difference between PUT and POST methods?**

**Ans:**

|  |  |
| --- | --- |
| **PUT** | **POST** |
| PUT puts a file or resource at a specific URI, and exactly at that URI. If there's already a file or resource at that URI, PUT replaces that file or resource. If there is no file or resource there, PUT creates one. | POST sends data to a specific URI and expects the resource at that URI to handle the request. The web server at this point can determine what to do with the data in the context of the specified resource. |
| PUT is idempotent, but paradoxically PUT responses are not cacheable. | The POST method is not idempotent, however POST responses are cacheable so long as the server sets the appropriate Cache-Control and Expires headers. |
| PUT is meant as a method for "uploading" stuff to a particular URI, or overwriting what is already in that URI. | POST, on the other hand, is a way of submitting data RELATED to a given URI. |
| In RESTful webservices, generally we use PUT to update an endpoint. | In RESTful webservices, generally we use POST to create an endpoint. |

**3. What approaches would you apply to make your API responding fast?**

**Ans:**

I would like to apply the following steps:

* Compress or remove unnecessary data.
* Simple and small request body or payload.
* Response body in a lightweight data-interchange format like JSON instead of XML.
* Cache data for fast retrieval.
  + Web Server Caching.
  + Client-Side Caching.

**4. How would you monitor your API?**

**Ans:**

I’ve used Google Analytics at my past employer organization. But there are many open source and paid analytics tools too. We can integrate any one of the analytics tools according to our employer/organization choices and available options. Some references given below:

1. <https://opensource.com/article/18/1/top-4-open-source-analytics-tools>
2. <https://nordicapis.com/monitor-the-status-of-apis-with-these-4-tools/>
3. <https://nordicapis.com/10-api-monitoring-tools/>

**5. Which endpoints from the task could by publically cached?**

**Ans:**

In this task, I’ve worked on users table. Indeed there is no real scenario of publicly cache at users table. However, I’ve used publicly cache for 10minutes/600seconds in the **Read or List all users** and **Read a specific user detail** to answer this question.