3. JSON best practices

\*JSON: Java script object notation. It is a lightweight data interchanging format for any language.

\*json media type: application/json.

##Uses of JSON

1. Java script-based application বানানোর জন্য অনেক গুরুত্বপূর্ণ ।
2. Json format is used for serializing and transmitting data over network.
3. It is used to transmit data between client and server.
4. Web services and api s use json format to provide public data.

## Characteristics of JSON

1. Easy to read and write.
2. Lightweight text-based interchange format.
3. It should be language independent.

## JSON data types: Number, String, Boolean, Array, Object, null, whitespace.

## JSON object সবসময় {} এর মধ্যে থাকবে । property: value থাকবে।

## json এর মধ্যে কোন এরর থাকলে json-lint বা অন্য কোন tools ব্যবহার করতে হবে।

## JSON লিখার নিয়ম

1. Always enclose the key: value pair with double cotation.
2. Never use hyphen in key fields.
3. Use camelCase when writing json data.
4. Always create root elements.

4.Request response best practices

\*\*\* HTTP request response communication model

1. One software module sends a request to a second software module and waits for a response.
2. The First software module performs the role of the client.
3. The second, the role of the server.

\*\*\*HTTP client

1.Browser level client: Browser is the primary http client responsible for load the web platform. It takes only get () request.

2.Application level client: HTTP Client libraries varies from platform to platform. HTTP Client is an application library used in client-side application to generate request and receive response.

\*\*\* Postman Http client: Postman is an HTTP Client application, used to test request-response communication. Postman is widely used for API testing and generating documentation.

\*\*\*HTTP request: HTTP Request is the first step to initiate web request/response communication. Every request is a combination of request header, body and request URL.

1. REST API best practice

\*\*\* API naming

1. Forward slashes for hierarchy: Forward slashes are conventionally used to show the hierarchy between individual resources and collections.
2. Punctuation for lists: punctuation marks such as the semicolon, or, more frequently, the comma should be used.
3. Query parameters where necessary.
4. Leave file extensions (such as .xml, .html etc.) out of your URIs.
5. keeping URIs clean, do not add a trailing forward slash.

\*\*\*API response header

1. Provide http response status code.
2. Provide proper content type, file type if any.
3. Provide cache status if any.
4. Only string data is allowed for response header.
5. Provide content length if any.
6. Provide response date and time.

\*\*\*Response body

1. Avoid providing response status, code, message via response body.

2. For single result, can use String, Boolean directly.

\*\*\*Response cookie

1. Avoid using response cookies as it is violate stateless principle.
2. If required use cookie encryption, decryption and other policies.

\*\*\*When use get()

1. GET is used to request something from server with less amount of data to pass.
2. When nothing should change on the server because of your action.
3. Get method only carries request url & header not request body.

\*\*\* When use post()

1. POST should be used when the server state changes due to that action.
2. When request needs its body, to pass large amount of data.

\*\*\*APIControllerBestPractices

1. Consider API versioning.
2. Use async/await if at all possible.
3. Mention which method is responsible for GET() and which for POST().

2.HTTP methods

GET (): The method requests a representation of the specified resource. Requests using GET should only retrieve data.

HEAD (): The HEAD method asks for a response identical to a GET request, but without the response body.

POST (): The POST method submits an entity to the specified resource, often causing a change in state or side effects on the server.

PUT (): The PUT method replaces all current representations of the target resource with the request payload.

DELETE (): The DELETE method deletes the specified resource.

5. REST API security best practice

\*\*\*Output validation

1. Output header
2. Provide proper http response status code.
3. Provide proper content type, file type if any.
4. Provide cache status if any.
5. Provide content length if any.
6. Provide response date and time.
7. Authentication token should provide via response header.

B. Output body

1. Avoid providing response status, code, message via response body.

2. For single result, can use String, Boolean directly.

3. Provide proper JSON encode-decode before writing JSON Body.

\*\*\*CSRF/XSRFProtection**:** Cross-site request forgery attacks (CSRF or XSRF for short) are used to send malicious requests from an authenticated user to a web application.

\*\*\*User Agent Protection: User agent is a request header property, describe client identity like operating system, browser details, device details etc. More over every web crawler like Google crawler, Facebook crawler has specific user-agent name.

\*\*\* API key

1. This is the most straightforward method and the easiest way for auth
2. With this method, the sender places a username: password/ ID / Keysinto the request header.
3. The credentials are encoded and decode to ensure safe transmission.

\*\*\* **JWT (JSON web token):** Information can be verified and trusted because it is digitally signed.

**Uses**

1. **Authorization:** Allowing the user to access routes, services, and resources.
2. **Information Exchange**: Way of securely transmitting information between parties.

**6.** **50 Interview Question on Web Back-End Development**

1. **What REST stands for?**

## 2.What are NoSQL databases?

## 3.What are the different types of NoSQL databases?

## 4. What do you understand by NoSQL databases?

## 5. What is SQL injection?

## 6. What is meant by Continuous Integration?

## 7. How to mitigate the SQL Injection risks?

## 8. Name some performance testing steps

## 9. Name the difference between Acceptance Test and Functional Test

## 10. What are the advantages of Web Services?

## 11. What does Containerization mean?

## 12. Name some Performance Testing best practices

**13. What is your favorite programming language? And why?**

**14.What do you understand by NoSQL databases? Explain.**

**15.What is the difference between software architecture and software design?**

**16.What is JavaScript, and why is it used?**

**17.What is your approach to debugging?**

**18.How would you find the most expensive queries in an application?**

**19.What is the difference between an acceptance test and a functional test?**

**20.Which sorting algorithm should you use and when?**

**21.What is the meaning of “high cohesion” and “loose coupling”?**

**22.Why should you use microservices architecture?**

**23.Explain the difference between cohesion and coupling?**

**24.When is refactoring useful?**

**25.** **What is your experience with object-oriented programming (OOP)?**

**26.Tell us about a time when you received criticism about your work and how you handled it.**

**27.If you have to deliver negative feedback to members of your development team, how would you do it?**

**28.Tell me about your favorite development project to date and what it was like from start to finish.**

**29.What is your experience with GoTo, and do you prefer structured programming?**

**30.Tell me about the largest web application you have ever worked on? What coding were you responsible for?**

**31.What's your preferred type of development environment?**

**32.Which programming languages do you want to work with and why?**

**33. What are NoSQL databases? Mention different types of NoSQL databases.**

**34.** **Mention the benefits of using Node.js.**

**35.** **Explain global installation of dependencies.**

**36.** **Explain how blocking is prevented in Node.js.**

**37.** **How does Node.js handle concurrency if it is single-threaded?**

**38. Name some important applications in IT where Node.js can be used.**

**39.** **What causes server latency and prevents scalability?**

**40.** **Define Control function in Node.js.**

**41. What is the Modularize option and when do you use it in Node.js?**

**42. Are there any disadvantages of Node.js?**

**43. How can you build scalability into a software program?**

**44. Explain SQL injection.**

**45. Differentiate between acceptance and functional tests.**

**46. Mention some performance testing steps.**

**47. Mention some advantages of Web Services.**

**48. Differentiate between Clustered and Non-clustered indexes?**

#### **49. Describe your experience with object-oriented programming.**

#### **50. Explain callback in Node.js.**