Note on Web Back-End Development

Mahir Shadid

MERN STACK – BATCH 1

1. **Rest API** Best Practices:

I. Use JSON as the Format for Sending and Receiving Data.

II. Use Nouns Instead of Verbs in Endpoints:

An endpoint should not look like this:

`https://mysite.com/getPosts` or `https://mysite.com/createPost`

Instead, it should be something like this:

`https://mysite.com/posts`

III. Name Collections with Plural Nouns:

If we have an endpoint like https`://mysite.com/post/123`, it might be okay for deleting a post with a DELETE request or updating a post with PUT or PATCH request, but it doesn't tell the user that there could be some other posts in the collection. This is why our collections should use plural nouns.
So, instead of https://mysite.com/post/123, it should be `https://mysite.com/posts/123`.

IV. Use Status Codes in Error Handling:

We should always use regular HTTP status codes in responses to requests made to our API. This will help our users to know what is going on – whether the request is successful, or if it fails, or something else.

V. Use Nesting on Endpoints to Show Relationships:

In the case of a multi-user blogging platform, different posts could be written by different authors, so an endpoint such

as `https://mysite.com/posts/author` would make a valid nesting in this case.
In the same vein, the posts might have their individual comments, so to retrieve the comments, an endpoint like `https://mysite.com/posts/postId/`comments would make sense.

VI. Use SSL for Security:

SSL stands for secure socket layer. It is crucial for security in REST API design. This will secure our API and make it less vulnerable to malicious attacks.

VII. Clear with Versioning:

REST APIs should have different versions, so you don't force clients (users) to migrate to new versions. This might even break the application if you're not careful.

VIII. Well Compiled Documentation:

Documentation is one of the important but highly ignored aspects of a REST API structure. The documentation is the first point in the hands of customers to understand the product and critical deciding factor whether to use it or not. One good documentation is neatly presented in a proper flow to make an API development process quicker.

IX. Return Error Details In The Response Body:

It is convenient for the API endpoint to return error details in the JSON or response body to help a user with debugging.

2. **Http methods** Best Practices:

- Main **5** methods are:

Method 1: Post

POST is the only RESTful API HTTP method that primarily operates on resource collections. When creating a subordinate resource in a collection, applying POST to the parent resource prompts it to create a new resource, associate it with the proper hierarchy and return a dedicated URL for later reference. However, keep in mind that POST is not idempotent; you can't use this method more than once and expect a consistent outcome or result.

A significant benefit of POST is that it enables developers to explicitly define resources. This feature helps prevent teams from accidentally creating subordinate resources that pollute code, muddy references and cause applications to experience problems.

Method 2: Put

The single-resource equivalent of POST is PUT, which updates a resource by replacing its content entirely. As a RESTful API HTTP method, PUT is the most common way to update resource information.

It's possible to create a resource with a PUT method, but this approach carries the risk of creating resources by accident, as noted above. If PUT is applied to a collection of resources, the entire resource collection gets replaced, which usually isn't the intention.

Method 3: Patch

PATCH is another HTTP method used to update resources. As opposed to replacing resources, like the PUT method does, PATCH only modifies resource contents. As a general rule, these modifications should be expressed in a standard format like JSON or XML.

Much like in PUT, it's poor practice to specifically apply PATCH methods to a whole resource collection -- that is, unless you truly intend to update every resource it contains.

Method 4: Get

The most common HTTP method is GET, which returns a representational view of a resource's contents and data. GET should be used in read-only mode, which keeps the data safe and the resource idempotent. You should get the same results no matter how many times you use this method, unless it is modified by another client in the interim.

The GET method is sometimes used to change the contents of a resource, but this is a precarious use of the method. It's common to compromise a client's ability to PATCH a resource if the resource detects a change since the PATCH client last conducted a GET.

Method 5: Delete

The last HTTP method to examine is DELETE. When a DELETE method targets a single resource, that resource is removed entirely.

Implementations of DELETE are typically somewhat inconsistent: The URL for the resource may remain available even if the actual resource is absent. In this type of scenario, it's possible the server or resource implementation will still change the state of the vanished resource using the URL, and likely react differently to subsequent DELETE executions.

While it's certainly possible, you should generally avoid using the DELETE method in a resource collection since it will delete all the contents within. Remember, the method isn't idempotent, and shouldn't be treated as such.

3. **JSON** Best Practices:

I. **Enclose within DOUBLE Quotes:**

```
{'id': '1','name':File} is not right ✗
```
`{"id": 1,"name":"File"} is okay ✓`
`{"id": "1","name":"File"} is the best ✓`

II. No Hyphens please:

Never use Hyphens in your Key fields. It breaks python, scala parser and developers have to escape it to use those fields. Instead of Hyphens use underscores (_). But using all lower case or camel case is the best. See samples below.

```
{"first-name":"Rachel","last-name":"Green"}  is not
right. ✗
{"first_name":"Rachel","last_name":"Green"} is okay ✓
{"firstname":"Rachel","lastname":"Green"} is okay ✓
{"firstName":"Rachel","lastName":"Green"} is the best. ✓
```

III. Always create a Root element.

Creation of Root element is optional, but it helps when you are generating complicated JSON.

JSON with root element
```
{
"menu": [
    {
        "id": "1",
        "name":"File",
        "value": "F",
        "popup": {
            "menuitem": [
                {"name":"New", "value": "1N", "onclick":
"newDoc()"},
                {"name":"Open", "value": "1O", "onclick":
"openDoc()"},
                {"name":"Close", "value": "1C",
"onclick": "closeDoc()"}
```

```
                ]
            }
    },
    {
        "id": "2",
        "name":"Edit",
        "value": "E",
        "popup": {
            "menuitem": [
                {"name":"Undo", "value": "2U", "onclick":
"undo()"},
                {"name":"Copy", "value": "2C", "onclick":
"copy()"},
                {"name":"Cut", "value": "2T", "onclick":
"cut()"}
                ]
            }
    }
    ]
}
```

JSON without root element

```
[
    {
        "id": "1",
        "name":"File",
        "value": "F",
        "popup": {
            "menuitem": [
                {"name":"New", "value": "1N", "onclick":
"newDoc()"},
                {"name":"Open", "value": "1O", "onclick":
"openDoc()"},
                {"name":"Close", "value": "1C",
"onclick": "closeDoc()"}
                ]
            }
    },
    {
        "id": "2",
        "name":"Edit",
        "value": "E",
        "popup": {
            "menuitem": [
                {"name":"Undo", "value": "2U", "onclick":
"undo()"},
                {"name":"Copy", "value": "2C", "onclick":
"copy()"},
                {"name":"Cut", "value": "2T", "onclick":
"cut()"}
                ]
            }
    }
]
```

IV. **Provide META sample:**

The idea of JSON is flexibility so you don’t have to restrict your data feed within few columns. But at the same time, if you are providing large data set with nested levels, the consumer will go crazy. Provide them with the meta / sample, so it helps them to understand what data to look for and what to skip.

V. **Validating JSON output:**

Using command line tools like ajv-cli / jsonlint (can be installed via any package manager) will eliminate trouble for consumers.

4. **Request - Response** Best Practices:

1. Describe resource functionality with HTTP methods:

All resources have a set of methods that can be operated against them to work with the data being exposed by the API. REStful APIs comprise majorly of HTTP methods which have well defined and unique actions against any resource. Here’s a list of commonly used HTTP methods that define the CRUD operations for any resource or collection in a RESTful API.

Method	**Description**
GET	Used to retrieve a representation of a resource.
POST	Used to create new new resources and sub-resources
PUT	Used to update existing resources
PATCH	Used to update existing resources
DELETE	Used to delete existing resources

2. Responses should give feedback to help developers succeed

   Providing good feedback to developers on how well they are using your product goes a long way in improving adoption and retention.

3. Give examples for all your GET responses:

   A well- designed API also has an example of the kind of response one can expect on successful call against a URL. This example response should be simple, plain, and quick to comprehend

4. Requests handle complexity elegantly

   The data you're trying to expose can be characterized by a lot of properties which could be beneficial for the end consumer working with your API. These properties describe the base resource and isolate specific assets of information that can be manipulated with the appropriate method.

5. **Web Security** Practices:

1. Provide Everyone with Application Security Training

The first and foremost step to guarantee web application security is to offer software development security training in every level. The training shouldn't be restricted to web application developers, however, include all related personnel involved in the procedure, like QA Specialist, Operational Staff, and Project Management. This sort of training to all personnel linked with the development lifecycle helps in building a culture of security within the company.

2. It Is Better to Back-up All our Web App Information

When any sort of security breach or malware infection occurs, and one requires restoring the web app after that, it would be terrible if anyone won't store the recent or updated version of the site. When the time arrives to live the site again, you will be pleased you postponed it. Thus return your information as much as feasible. It is worth noted that majority of the web hosting service suppliers will give backups from their servers if this mishappening occurs.

3. Regularly Scan our Web App for Threats and Vulnerabilities

Security checks and scans should be made frequently to stay on top of the safety and protection of your web app. It would be a sensible choice to do security scans on your sites at least one time each week. Besides, you must scan after every alteration that you have completed in your web app. It is essential that several security scanners, even experienced folks, will not be able to discover all the security complexities. Scanners are either heuristic or pattern-based, and malware is all the time engineered to be undetectable from the scanners. A lot of security scanners locate malware better than others, and conversely, some struggle with false positives. The majority of the security scanners don't work at all. Furthermore, you should still learn about several security glitches and flaws.

4. Utilize a Web Application Firewall

The WAF (web application firewall) is principally a filter for HTTP traffic amid a client and a server. It does not allow any malicious requests experience and infiltrate your databases. Firewall is the most significant way for safeguarding software at the entrance points to your network, as they scrutinize all incoming traffic and stop all doubtful activity. WAFs do not necessitate developers to transform anything in the source code, which also makes them suitable to make use of.

But, traditional firewalls have their flaws: they cannot determine some sorts of attacks. To ensure higher security, make use of advanced WAFs that can guard your app from cross-site scripting and SQL injection attacks.

5. Prioritize our Web Apps Is the Logical Next Step

After ending the inventory of your existing web apps, sorting them according to priority is the sound step. You may disbelief it now, but your list is expected to be extended. Without prioritizing which apps to concentrate on initially, you will struggle to make any significant advancement.

Sorting the apps into three categories is significant:

1. Normal

2. Critical

3. Serious

6. The Use of Cookies Securely

Another most significant area that many companies do not think about while addressing web app security best practices is the employ of cookies. Cookies are extremely convenient for users and businesses similarly. They let users to be remembered by websites that they browse so that future visits are rapid and, in several cases, extreme personalized. But, cookies can also be utilized by hackers to gain access to secured areas.

7. Introduction of a Bounty Program

One of the better ways to get feedback from the community concerning potential web app security glitches is to maintain a bounty program. Even if you handle a company with devoted security staffs employed, they often might not be able to recognize all potential security threats. Hence, to help support the community to find security threats and report them, offer a "bounty" of financial worth.

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

1. What is object-oriented programming?

Object-oriented programming (OOP) is a programming language structure where the data and its associated processing, also known as methods, are defined as self-contained entities. These entities are called "objects." OOP involves modeling a system as a collection of various objects, with each object representing some specific aspect of the system.

2. What is a constructor?

A constructor in Java is a special method used to initialize objects. It is called when an object of a class is created. A constructor has the same name as its class. Also, it is syntactically similar to a method. However, constructors don't have any explicit return type.

3. What are the different types of constructors?

There are three main types of constructors: No Argument Constructor, Default Constructor, and Parameterized Constructor.

4. Why is multiple inheritance not possible in Java?

In Java, multiple inheritance isn't supported due to the ambiguity problem. Multiple inheritance is also not supported in Java to simplify the language and reduce the complexity. When one class, such as class B, extends more than other classes, such as class A and class C, this is called multiple inheritance. One example of a problem that occurs in multiple inheritance is the diamond problem.

5. What is the difference between Wrapper Classes and Primitive Data Types?

Wrapper classes offer a simple way to use various primitive data types, such as int and boolean, as objects. In other words, the wrapper class converts a primitive type into an object. On the other hand, a primitive type is a predefined data type that the Java programming language provides.

6. What is serialization?

Serialization is a method or mechanism of converting the state of an object into a byte stream. Deserialization does the opposite. Serializing an object ensures that the byte stream can be easily reverted into a copy of the object.

Serialization in Java is often used in JMS, Hibernate, EJB, and JPA and helps transport the code from one JVM to another.

7. What is the difference between finally, final, and finalize?

Finally is a block or code block. We use it with a try-catch block to handle the exception. In contrast, final is a keyword. We can use this keyword to mark a variable "unchangeable." Final is also used to apply restrictions on method, class, and parameters.

Finalize is a deprecated method of the Object class. Finalize was used for performing clean-up processing before the object is garbage-collected. In recent Java versions (9+) it is recommended to inherit the Closable interface and use its method close() for resource management.

8. What is the difference between LinkedList and ArrayList?

The key difference between LinkedList and ArrayList is that LinkedList exercises LinkedList Data Structure within the class with multiple variations in every element. In contrast, ArrayList falls under the category of collection framework of dynamic arrays that are distinct from standard arrays.

9. How does a HashMap work?

In Java, HashMap works on various hashing principles. It uses its static inner class to store the entries in the map. HashMap uses several buckets, with each bucket pointing to a Singly Linked List. In Java 8, however, HashMap replaces Linked List with a binary tree when the amount of bucket elements reaches a certain threshold. HashMap allows multiple null values and at most one null key.

10. What is a thread?

Thread as an independent path of execution through program code. Threads are used to make Java applications faster and more responsive by doing multiple things simultaneously.

11. What are the different types of threads?

There are two kinds of threads: daemon threads and user threads. A daemon thread is a low-priority thread whose only role is to offer services to various user threads. In contrast, a user thread is a high-priority thread.

12. What is the difference between a process and a thread?

A process can be defined as the execution of a program, allowing users to perform the appropriate actions that are specified in a program. In other words, it means a program is in execution.

In contrast, a thread is an execution unit. It is part of a process and can be managed independently by a scheduler. A process may have multiple threads, which all execute at the same time.

13. What is exception handling?

Exception handling is the process of responding to the occurrence of exceptions like ClassNotFoundException, SQLException, IOException, and RemoteException.

14. What is a design pattern?

A design pattern is a general repeatable or reusable solution to common problems occurring in software design. They are similar to blueprints, and the pattern often shows interactions and relationships between objects or classes.

15. What are the types of Creational design patterns?

You can implement five well-known Creational design patterns in the scope of programming languages. These are as follows:

- Builder pattern
- Abstract factory pattern

- Factory method pattern
- Singleton pattern
- Prototype pattern

16. What is a lambda expression?

Lambda expressions are a feature in Java and its first step into the world of functional programming. It is a short block of code that takes in various parameters and then returns a value. A lambda expression comprises a comma-separated list of multiple formal parameters that are enclosed in parentheses.

17. What are NoSQL databases? What are the different types of NoSQL databases?

A NoSQL database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases (like SQL, Oracle, etc.).

Types of NoSQL databases:

- Document Oriented
- Key Value
- Graph
- Column Oriented

18. What is SQL injection?

Injection attacks stem from a lack of strict separation between program instructions (i.e., code) and user-provided (or external) input. This allows an attacker to inject malicious code into a data snippet.

SQL injection is one of the most common types of injection attack. To carry it out, an attacker provides malicious SQL statements through the application.

19. What is meant by *Continuous Integration*?

Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.

20. Name some performance testing steps

Some of the performance testing steps are:

- Identify the testing environment
- Identify performance metrics
- Plan and design performance tests
- Configure the test environment
- Implement your test design
- Execute tests
- Analyze, report, retest

21. What are the advantages of Web Services?

Some of the advantages of web services are:

- Interoperability: Web services are accessible over network and runs on HTTP/SOAP protocol and uses XML/JSON to transport data, hence it can be developed in any programming language. Web service can be written in java programming and client can be PHP and vice versa.
- Reusability: One web service can be used by many client applications at the same time.
- Loose Coupling: Web services client code is totally independent with server code, so we have achieved loose coupling in our application.
- Easy to deploy and integrate, just like web applications.
- Multiple service versions can be running at same time.

22. Name some Performance Testing best practices

- Test as early as possible in development.
- Conduct multiple performance tests to ensure consistent findings and determine metrics averages.
- Test the individual software units separately as well as together
- Baseline measurements provide a starting point for determining success or failure
- Performance tests are best conducted in test environments that are as close to the production systems as possible
- Isolate the performance test environment from the environment used for quality assurance testing
- Keep the test environment as consistent as possible
- Calculating averages will deliver actionable metrics. There is value in tracking outliers also. Those extreme measurements could reveal possible failures.

23. What Is ACID Property Of A System?

ACID is a acronym which is commonly used to define the properties of a relational database system, it stand for following terms

- Atomicity - This property guarantees that if one part of the transaction fails, the entire transaction will fail, and the database state will be left unchanged.
- Consistency - This property ensures that any transaction will bring the database from one valid state to another.
- Isolation - This property ensures that the concurrent execution of transactions results in a system state that would be obtained if transactions were executed serially.
- Durable - means that once a transaction has been committed, it will remain so, even in the event of power loss.

24. What are disadvantages of REST web services?

Some of the disadvantages of REST are:

- Since there is no contract defined between service and client, it has to be communicated through other means such as documentation or emails.
- Since it works on HTTP, there can’t be asynchronous calls.
- Sessions can’t be maintained.

25. What are the DRY and DIE principles?

In software engineering, Don't Repeat Yourself (DRY) or Duplication is Evil (DIE) is a principle of software development.

26. What are the difference between *Clustered* and a *Non-clustered* index?

- With a Clustered index the rows are stored physically on the disk in the same order as the index. Therefore, there can be only one clustered index. A clustered index means you are telling the database to store close values actually close to one another on the disk.
- With a Non Clustered index there is a second list that has pointers to the physical rows. You can have many non clustered indices, although each new index will increase the time it takes to write new records.
- It is generally *faster to read* from a clustered index if you want to get back all the columns. You do not have to go first to the index and then to the table.
- *Writing* to a table with a clustered index can be *slower*, if there is a need to rearrange the data.

27. What are the differences between *Continuous Integration*, *Continuous Delivery*, and *Continuous Deployment*?

- Developers practicing continuous integration merge their changes back to the main branch as often as possible. By doing so, you avoid the integration hell that usually happens when people wait for release day to merge their changes into the release branch.
- Continuous delivery is an extension of continuous integration to make sure that you can release new changes to your customers quickly in a sustainable way. This means that on top of having

automated your testing, you also have automated your release process and you can deploy your application at any point of time by clicking on a button.

- Continuous deployment goes one step further than continuous delivery. With this practice, every change that passes all stages of your production pipeline is released to your customers. There's no human intervention, and only a failed test will prevent a new change to be deployed to production.

28. What is the difference between Monolithic, SOA and Microservices Architecture?

- Monolithic Architecture is similar to a big container wherein all the software components of an application are assembled together and tightly packaged.
- A Service-Oriented Architecture is a collection of services which communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity.
- Microservice Architecture is an architectural style that structures an application as a collection of small autonomous services, modeled around a business domain.

29. Explain the architectural style used to create the web API.

The architectural style for creating web API are:

- HTTP for client-server communication
- XML / JSON as formatting language
- Simple URI as the address for the services
- Stateless communication

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

Acceptance testing - this is a validation activity. Often developers are faced with questions, they built the right thing and or will it satisfy the needs of the client. With an acceptance test, you can determine if a product solves the problems for which it was designed. This can best be done by the user (customer), for instance, performing his / her tasks that the software assists with.

Functional testing - this is a verification activity. Developers also often have questions about whether they have built the product correctly and meet their business requirements. The functional test helps to answer these questions. It can be used to determine if the product works the way the developers think it does not.

31. What does REST stand for and what is a RESTful API?

REST states for REpresentational State Transfer and is a software architectural style that is meant to guide the design and development of web services.

A RESTful API is an architectural style for an application program interface that utilizes HTTP requests to access and process data. The API can be used for getting, putting, posting, or deleting various data types.

32. What is a reverse proxy?

A reverse proxy is a type of proxy that retrieves data from one or more servers on behalf of a client. This data is then returned to the client appearing as if it originated from the reverse proxy server itself. A reverse proxy is often used to balance the load.

33. What is the difference between the high-level comparison SQL and MongoDB databases?

SQL databases store all of their data as tables, columns, rows, and records. All this data is stored in one place, on a predefined data model, which is not flexible enough for modern real-world highly growing applications.

As for MongoDB databases, they are in many ways similar to SQL, because they also store all data in the form of tables, columns, rows, and records. The main difference is that MongoDB uses a flexible framework that can be easily extended and modified.

34. How do you understand the term Containerization?

Containerization is a type of virtualization strategy that was invented as an alternative method to traditional hypervisor-based virtualization.

During the containerization process, the operating system is used concurrently by different containers, so it does not need to be cloned for each virtual machine

35. If Node.js is single-threaded then how does it handle concurrency?

With Node, programmers have a single thread. Using it, code can be written very easily and without a bottleneck. Node also uses multiple POSIX threads, for various I / O operations such as File, DNS, Network calls, etc.

If Node gets an I / O request, it creates or uses a thread to perform that I / O operation. After performing this operation, it puts the result in the event queue. During each such event, an event loop starts and checks the queue. If the execution stack of Node is empty then it adds the queue result to the execution stack.

36. What's the difference between JOIN and UNION?

With the help of SQL JOIN, we can search for data in another table, based on the specified conditions between the tables.

UNION operation allows us to add 2 similar data sets to create the resulting data set that contains all the data from the source data sets. Union does not require any conditions for joining.

37. How can you swap the values of numeric variables without using other variables?

You can swap two values a and b without using any other variables as follows:

a = a + b;

b = a - b;

a = a - b;

38. Explain what the API Gateway pattern means?

An API Gateway is a server that is the only possible entry point to the system. It is similar to the Facade pattern from object-oriented design. API Gateway encapsulates the architecture of the entire system and provides an API that is already adapted for each client. This API can also have other functions such as authentication, caching, monitoring, and load balancing.

39. What are the benefits of using B-trees index?

Such indexes save a lot of time, because look-ups, deletions, and insertions can all be done in logarithmic time. Also, the data that is stored inside B-trees can be easily sorted.

40. What Is the BASE Property Of A System?

The acronym states for Basically Available, Soft state, Eventual consistency. BASE properties are general properties inherent in newly developed NoSQL systems. The BASE system does not guarantee consistency, but it is guaranteed to be available. Its soft structure suggests that the state of the system can change over time, even without data entry. This is due to the sequential model.

41. How can you explain Distributed Transactions?

Distributed Transaction is a situation where one event changes two or three separate data sources that cannot be captured. In the world of microservices, it becomes even more complex as each service is a unit of work, and most of the time multiple services have to work together to make a business successful.

42. Why should you always avoid GOD class?

The most efficient way to hack a program is to create a GOD class. These are classes that keep track of huge amounts of information and have multiple responsibilities. One code change will affect other parts of the class. This leads to chaos in the service because no one dares to make a quality change.

43. What's the difference between faking, mocking, and stubbing?

Fake objects have working implementations but will require a reduction, which makes them unusable for production.

Stubs are standard responses to calls made during a test. They can also record call information, such as an email gateway stub that remembers the messages it 'sent ''.

Mocks are objects with preprogrammed expectations that shape the specifics of the call.

44. When to use Redis and when to use MongoDB?

MongoDB and Redis are both NoSQL databases, however, they were built for different purposes.

MongoDB is a document-oriented, disk-based database that should be used for ensuring operational simplicity, creating a schema-free design, and processing very large data volumes.

Redis is an in-memory, persistent data structure store that should be used to enable the performance of common operations with minimal complexity and maximum performance.

45. Give examples of the mitigation tactics you'd use for various types of API attacks.

- Injection: I'd validate and sanitize all data in API requests as well as limit response data to prevent the unintentional leakage of sensitive data.
- Cross-Site Scripting (XSS): I'd validate input as well as use character escaping and filtering.

- Distributed Denial-of-Service (DDoS): I’d limit the number of requests and payload size.

Man-in-the-Middle (MitM): I’d encrypt traffic in transit.

46. How would you manage Web Services API versioning?

Versioning is a critical part of API design, as it gives developers the ability to improve their API without stopping the client’s applications whenever new updates are rolled out. The three types of API versioning are:

- URL Versioning or Route Versioning: This solution uses URI routing to point to a specific version of the API.
- Versioning using a custom header: REST APIs are versioned by providing custom headers with the version number included as an attribute.
- Query String Parameter: Considered to be the worst method, the version number is included as a query parameter.

47. What is CAP Theorem?

CAP stands for the attributes of Consistency, Availability, and Partition Tolerance. The CAP Theorem is a concept according to which, a distributed database system can have only two of the three mentioned attributes. It is useful to determine the choice of data manipulation tools used in Big Data, based on each unique use-case.
CAP Theorem is like the old joke about software projects: you can have it on TIME, in BUDGET, or CORRECT. Pick any two.

48. In what situation would you choose RDBMS and where would you choose NoSQL?

An RDBMS is used when you need ACID (Atomicity, Consistency, Isolation, Durability) compliance to reduce anomalies and protect data integrity and when the data is structured and unchanging.
On the other hand, NoSQL is recommended for high volume environments, cloud computing & storage, and when using unstructured

data. Examples of NoSQL databases are MongoDB, Cassandra, HBase, and CouchDB.

49. What is an MVC framework?

A Model-View-Controller is a software design pattern that separates an application into three logical interconnected components – the model, the view, and the controller. It is used to organize the code into simple organized components. MVC helps in letting your code interact with another developer's code, based on their functions.

50. What are the qualities any good backend developer must possess?

This is a great question with which to impress the interviewer, as you can tell them about your competencies and understanding of the position. However, many candidates make the mistake of only talking about a couple of their strengths. The ideal answer should include at least some of the following points:

- In-depth knowledge of server programming languages like Python, Ruby, Java, Perl
- Great acquaintance with NoSQL and RDBMS
- Good understanding of front-end technologies (easy to work with frontend developers)
- Basic understanding of cloud deployments
- Ability to develop business logic within any app
- Ability to easily create functional APIs
- Design of service architecture
- Ability to optimize web applications