1. Difference between Authorization and Authentication

Feature	Authentication	Authorization
Definition	Verifying identity	Determining access permissions
Purpose	Confirm user identity	Control access to resources
Process	Verifying credentials	Checking permissions based on identity
Sequence	First step in security process	Second step, follows authentication
Scope	Identity verification	Permissions and access control
Data Used	User credentials (e.g., passwords, biometrics)	Permissions, roles, policies
Security Risks	Impersonation if compromised	Unauthorized access if misconfigured
Examples	Logging in with a username and password	Accessing admin features based on user role
Tools/Protocols	OAuth, OpenID Connect, SAML	OAuth, RBAC, ABAC

2. Differences Between JSON and XML

Feature	**JSON**	**XML**

Definition	JavaScript Object Notation, a lightweight data interchange format.	eXtensible Markup Language, a markup language for encoding documents in a format that is both human-readable and machine-readable.
Syntax	Uses key/value pairs and arrays.	Uses a tree structure with nested elements and attributes.
Readability	More concise and easier to read, especially for those familiar with JavaScript.	More verbose and can be harder to read due to extensive use of tags.
Data Types	Supports strings, numbers, objects, arrays, booleans, and null.	All data is treated as text; must explicitly convert types.
Schema and Validation	Uses JSON Schema for defining structure and validation.	Uses DTD (Document Type Definition) and XSD (XML Schema Definition) for structure and validation.
Attributes	Does not support attributes. All metadata must be included in the object structure.	Supports attributes, which can provide additional information about elements.
Namespaces	Does not have built-in support for namespaces.	Supports namespaces, allowing differentiation of elements and attributes with the same name in different contexts.
Usage and Interoperability	Widely used in web applications and APIs due to its lightweight nature.	Commonly used in configuration files, document storage, and enterprise applications.

Transformation and Querying	JavaScript functions (`JSON.parse`, `JSON.stringify`), libraries (e.g., lodash), JSONPath for querying.	Robust tools like XSLT for transformations and XPath for querying XML documents.
Comments	Does not support comments.	Supports comments using `<!-- comment -->`.
Serialization	Easier and faster to serialize and deserialize due to simpler structure.	More complex and slower to serialize and deserialize.
Size	Generally smaller in size due to less verbose syntax.	Larger in size due to verbose tagging.
Error Handling	Less tolerant to errors; small errors can prevent parsing.	More tolerant to errors; can often recover from certain errors.
Parsing	Typically faster parsing due to simpler and less verbose structure.	Slower parsing due to more complex and verbose structure.
Tooling Support	Excellent support in modern web technologies, native support in JavaScript.	Wide support across many platforms, strong in enterprise environments.
Human Readability	Easier for humans to read and write, especially for simple data structures.	Can be harder to read due to verbosity but is still human-readable.
Extensibility	Limited extensibility; designed for data interchange.	Highly extensible; designed for document formatting and data interchange.
Compatibility	Best suited for web applications and lightweight data exchange.	Suitable for complex document handling and data interchange across diverse systems.