**SAAS**: Software as a service (SaaS) is a form of cloud computing in which the provider offers the use of application software to a client and manages all the physical and software resources used by the application.

# **Some Examples of SAAS in Nepal:**

Blanxer: <https://www.blanxer.com/>

Tigg: <https://www.tiggapp.com/>

**Multitenant Architecture:** In a multi-instance architecture, several companies will run their own separate instance of the application, with their own database. Each company will therefore have access to its data separately from another.

***Multi => Multiple***

***Tenant => Databases or Tables***

**Project: Forum Management System**

Start project (Setup): npm init -y

Install packages: npm install express mysql2 sequelize dotenv ejs nodemon

To start projet: npm start

For Google Login: <https://www.passportjs.org/packages/passport-google-oauth2/>

***authentication for Node.js***

Visit Google Developer Console: <https://console.cloud.google.com/>

Then, create project

Then, create credentials

Then, keep Client ID and Client Secret

Serialization and deserialization

**Serialization and Deserialization: A Comprehensive Guide**

**Serialization** and **deserialization** are essential processes in computer programming, especially when dealing with data storage and transmission. They involve converting data structures into a format that can be easily stored or transmitted, and then reconverting them back to their original form.

**What is Serialization?**

Serialization is the process of converting an object into a sequence of bytes that can be stored or transmitted. This sequence is typically called a **serialized representation**. Once serialized, the object can be stored on disk, sent over a network, or embedded within another data structure.

**Why is Serialization Used?**

* **Data Storage:** Serialization allows objects to be stored in databases, files, or other persistent storage mechanisms.
* **Data Transmission:** Serialized data can be transmitted over networks, such as the internet, to be processed by other systems.
* **Object Persistence:** It enables objects to be preserved across program executions, allowing for stateful applications.
* **Interoperability:** Serialized data can be shared between different programming languages and platforms.

**Common Serialization Formats**

* **JSON (JavaScript Object Notation):** A lightweight, human-readable format widely used for data exchange.
* **XML (Extensible Markup Language):** A more verbose format that provides a rich set of features for representing complex data structures.
* **Protobuf (Protocol Buffers):** A platform-neutral, extensible mechanism for serializing structured data.
* **Avro:** A data serialization system designed to be compact, efficient, and self-describing.
* **Thrift:** A scalable cross-language service development framework that includes a serialization protocol.

**Deserialization**

Deserialization is the reverse process of serialization. It involves taking a serialized representation and converting it back into its original object form. This allows the object to be used again within the program.

**Considerations for Serialization and Deserialization**

* **Performance:** The choice of serialization format can significantly impact performance, especially for large data sets.
* **Compatibility:** Ensure that the serialization format and deserialization process are compatible across different systems and programming languages.
* **Security:** Be mindful of security risks when serializing and deserializing data, especially when dealing with sensitive information.

Example:

Let user = {

name: “Bishal Rijal”,

Age: 23

}

**Serializing data**

JSON.stringify(user)

**Output:**

{

“name”: “Bishal Rijal”,

“age”: 21

}

**Deserializing data**

**JSON.parse({**

“name”: “Bishal Rijal”,

“age”: 21

**})**

**Output: {**

name: “Bishal Rijal”,

Age: 23

}