Stateful applications and processes allow users to store, record, and return to already established information and processes over the internet. In stateful applications, the server keeps track of the state of each user session, and maintains information about the user's interactions and past requests.

A stateless process or application, however, does not retain information about the user's previous interactions.

**API** - API is an acronym that stands for “application programming interface,” and it allows apps to send information between each other.

**Rest API** – restful api – rest is an architectural constraints

* Client-server architecture with http requests
* [Stateless](https://www.redhat.com/en/topics/cloud-native-apps/stateful-vs-stateless) client-server communication, meaning no client information is stored between get requests and each request is separate and unconnected.

A **framework** is a set of pre-written code that provides a structure for developing software applications. A library, on the other hand, is a collection of pre-written code that can be used to perform specific tasks.

Express.js **is the most popular web framework for Node.js**.

Express JS uses **middlewares**.  
**Middleware** in Express. js is a function that is handles the request, response objects in a better way and has access to the next middleware function  
**Routing** – how an applications endpoints(URL’s) responds to client requests

app.use(express.static(“./public”)) 🡪 **this allows to serve static files within public folder without routes.**

**JSON Web Token 🡪** after comparing the passwords, if every thing is right, we use jwt.sign method that takes the first argument as object that takes what data is to be stored in the token, then the secret key. We can also set expiry time there. The secret key is saved on the server.

**the same secret key is used to both create and verify the signature**

**```**

**Q: How you provided confidential routes?**

Whenever a dedicated/ confidential route has to be provided, I used a middleware before it, to check if the token is valid, if valid the next() function was called so that other routes can be called else threw an error.

**Q: When to use next() and return next() ?**

 Use next() to pass control to the next middleware and continue executing the current function. Use return next() to ensure no further code in the current middleware function executes after passing control.

if you use return next(), it will jump out the callback immediately and the code below return next() in the callback will be unreachable.

**JWT detail verification process**

A JSON Web Token or JWT is made up of three parts:

1. The header: contains some metadata about the token itself.
2. The payload: contains the data that we want to encode into the token, so the more data we want to encode here the bigger is the JWT.
3. The signature.

These first two parts, the header and the payload, are just plain text that will get encoded, but not encrypted.

**So anyone will be able to decode them and read them**, we cannot store any sensitive data in here. But that's not a problem at all because in the third part, the signature, is where things really get interesting. The signature is created using the header, the payload, and the secret that is saved on the server.

[read from here](https://stackoverflow.com/questions/31309759/what-is-secret-key-for-jwt-based-authentication-and-how-to-generate-it)

```

**PUT / PATCH**

* PUT – all the data needs to be provided, if not then the others take default value or null
* PATCH – only the data provided is changed, others remains same.