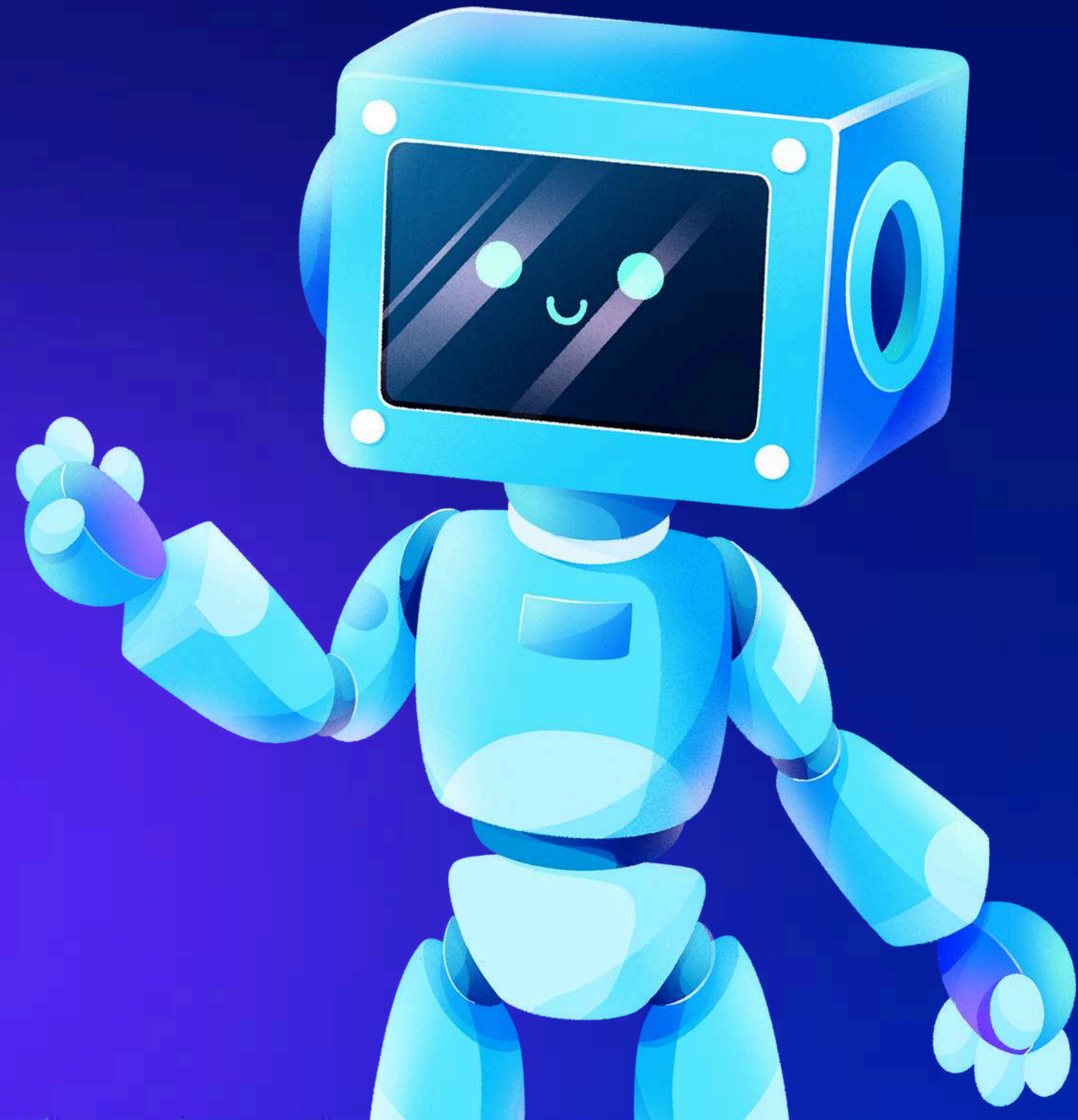


pemrograman berbasis platform

AUTENTIKASI JSON WEB TOKEN (JWT)

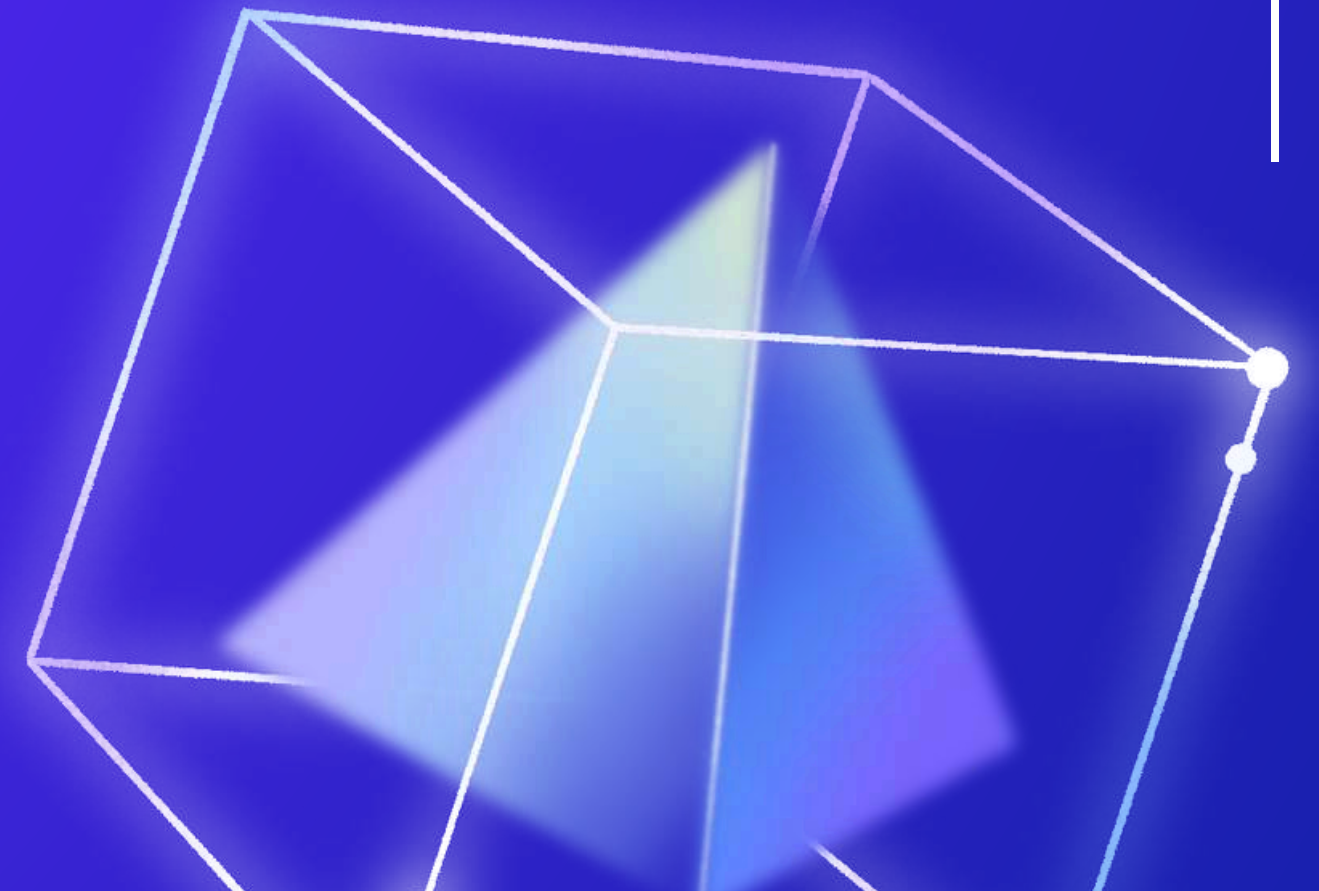
kelompok 3





ANGGOTA KELOMPOK

- Wiliana Irawan
- Khoirunnissa Salsa Julianti
- Syahril Gibran Wangsa Guna
- Rini Nurulsona
- M.Gilang Rija A
- Elia Jose Alvaro Rahayaan



PENDAHULUAN

Proses autentikasi adalah elemen penting dalam pembuatan aplikasi masa kini , khususnya ketika dihadapkan dengan permintaan dari pengguna atau perangkat klien. JSON Web Token (JWT) merupakan metode yang banyak dipakai untuk melindungi pertukaran data antara semua pihak yang berpartisipasi dalam sistem.

konsep JWT dan cara penerapannya dalam proses autentikasi di platform Node.js.

JSON Web Token (JWT) adalah sebuah standar terbuka yang menjelaskan metode aman untuk memindahkan informasi dalam format JSON di antara berbagai pihak dalam bentuk token .



TOKEN JWT TERDIRI DARI TIGA BAGIAN:



HEADER

Menyimpan metadata dan jenis Algoritma enkripsi yang digunakan.

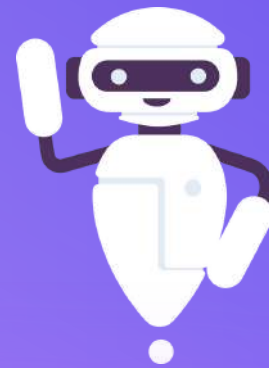
PAYLOAD

Berisi klaim (claim) yang dapat didefinisikan oleh pengguna, seperti informasi pengguna atau data tambahan.

SIGNATURE

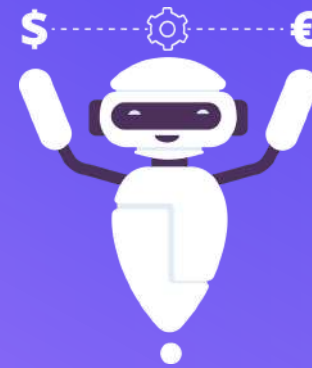
Digunakan untuk memverifikasi bahwa pesan tidak diubah selama transmisi.

KEUNTUNGAN JWT DALAM AUTENTIKASI



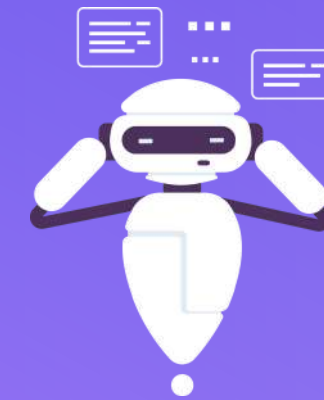
STATELESS DAN PORTABILITAS

JWT bersifat stateless, artinya server tidak perlu menyimpan sesi pengguna.



KEAMANAN

JWT dapat dilindungi dengan algoritma enkripsi yang kuat, sehingga data yang dikirimkan aman dari penyadapan.

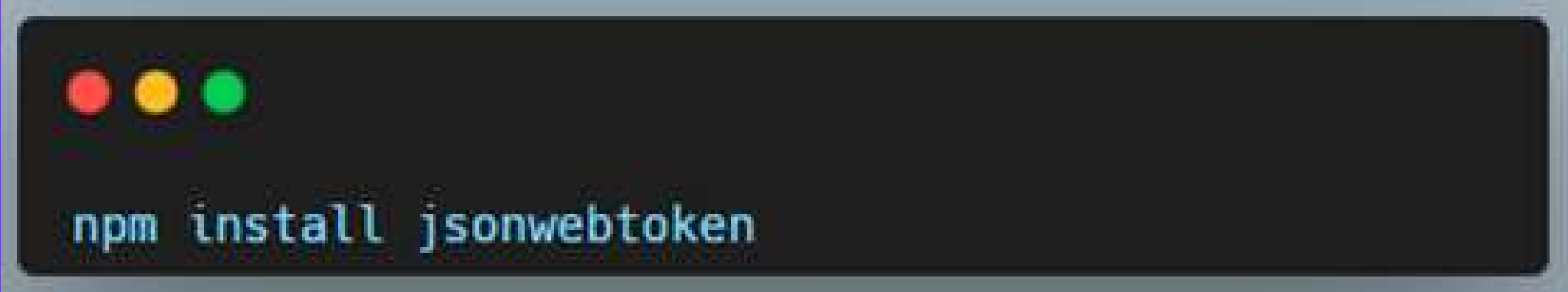


EFISIENSI

JWT mengurangi overhead jaringan karena tidak memerlukan penyimpanan sesi di server.

LANGKAH IMPLEMENTASI

(01) LAKUKAN INSTALASI PAKET JWT



```
npm install jsonwebtoken
```


LANGKAH IMPLEMENTASI

02

MEMBUAT DAN MEMVERIFIKASI TOKEN

```
const jwt = require('jsonwebtoken');

const secretKey = 'rahasiasecure'; // Isikan dengan kunci/password key

const user = {
  id: 1,
  username: 'Alun Sujjada'
};

const token = jwt.sign(user, secretKey, { expiresIn: '1h' });
console.log('Token:', token);
```

LANGKAH IMPLEMENTASI

(03) VERIFIKASI TOKEN

```
const decoded = jwt.verify(token, secretKey);  
console.log('Decoded:', decoded);
```


LANGKAH IMPLEMENTASI

(04)

INTEGRASI DENGAN EXPRESS.JS

```
const express = require('express');
const jwt = require('jsonwebtoken');

const app = express();
const secretKey = 'rahasiasecure'; // Gantilah dengan kunci rahasia yang kuat

// Middleware untuk verifikasi token
const verifyToken = (req, res, next) => {
  const token = req.header('Authorization');

  if (!token) {
    return res.status(401).json({ message: 'Akses ditolak, token tidak ada.' });
  }
}
```


LANGKAH IMPLEMENTASI

(04) INTEGRASI DENGAN EXPRESS.JS

```
try {  
  const decoded = jwt.verify(token, secretKey);  
  req.user = decoded;  
  next();  
} catch (error) {  
  return res.status(401).json({ message: 'Token tidak valid.' });  
}  
};  
  
// Contoh penggunaan middleware  
app.get('/secure-endpoint', verifyToken, (req, res) => {  
  res.json({ message: 'Akses diberikan.' });  
});  
  
app.listen(3000, () => {  
  console.log('Server berjalan di port 3000')  
});
```


PERTIMBANGAN KEAMANAN

01

SIMPAN SECRET KEY DENGAN AMAN:

Kunci rahasia (secret key) harus disimpan dengan aman dan tidak boleh disertakan dalam kode sumber publik.

02

WAKTU KEDALUWARSA (EXPIRATION)

Atur waktu kedaluwarsa token sesuai dengan kebutuhan keamanan aplikasi.

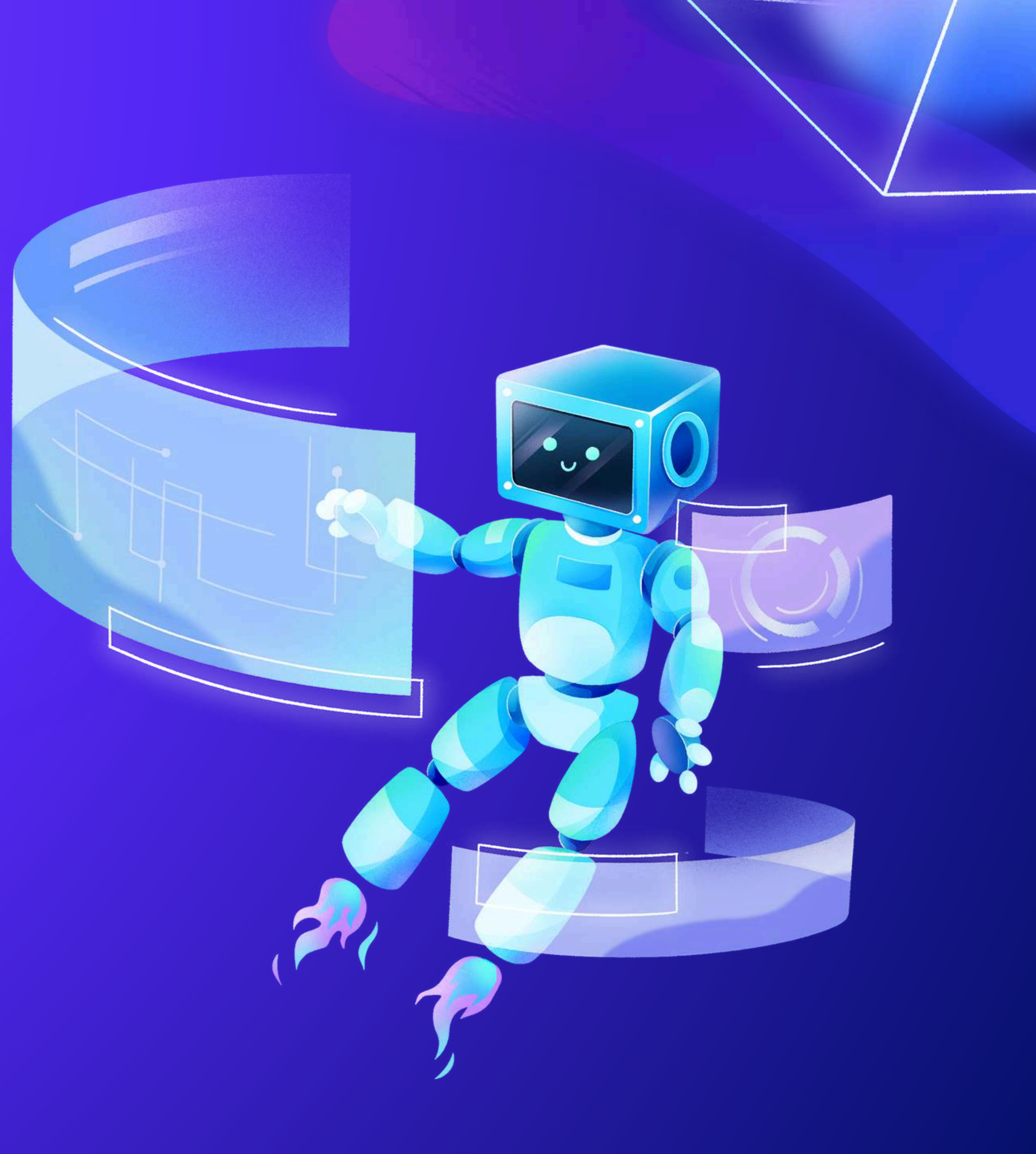
Token yang telah kedaluwarsa harus ditolak.

03

HTTPS UNTUK KEAMANAN KOMUNIKASI

Gunakan protokol HTTPS untuk mengamankan transmisi token antara klien dan server.

TANYA JAWAB



TERIMA

KASIH!

