**Authentication**

**Authentication –** Verifying identity (error: 401 unauthorized)

**Authorization –** Verifying permissions (error: 403 forbidden)

Authentication can be:

1. Stateful (i.e. sessions using a cookie)
2. Stateless (i.e. JWT/oauth)

Stateful flow

1. User submits login credentials
2. Credentials are verified by database
3. Server creates temporary user session
4. Server issues cookie with a session ID
5. User sends cookie with each request
6. Server validates it against the session store and grants access
7. When user logs out, the server destroys the session and clears the cookie

Stateless flow

1. User submits login credentials
2. Credentials are verified by database
3. Server generates a temporary token and can embed user data in it
4. Server responds back with the token in the body header
5. User stores the token in client storage
6. User sends token along with each request
7. Server verifies the token and grants access
8. When the user logs out, the token is cleared from client storage

Token features

* Tokens are not stored server-side, only on client (stateless)
* Signed with a secret which can be verified and trusted by the server
* Tokens can be self-contained which means it can carry all the required user data with it in the body/payload
* Reduces database look ups and therefore increases performance, however can increase XSS attacks
* Typically sent in *authorization* header

JWT (JSON Web Tokens)

JWT is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object.

JWT consists of 3 parts:

1. Header – meta information about the token
2. Payload – data
3. Signature – guarantees token was not tampered with

JWT can be stored in client storage (local or session). Local storage is the more popular option.

JWT express example

*Npm install jsonwebtoken*

***App.js***

Const jwt = require(‘jsonwebtoken’)

App.post(‘api/login’, (req, res) => {

// mock user – would usually get from DB

Const user = {

Id: 1,

Username: ‘Rezaa91’

}

Jwt.sign({user}, ‘secretKey’, (err, token) => {

Res.json({token}) // view token in postman

})

})

// middleware

Function verifyToken(req, res, next) {

// get auth value

Const bearerHeader = req.headers[‘authorization’];

// check if defined

If (typeof bearerHeader !== ‘undefined’) {

Const token = bearerHeader.split(‘ ‘)[1];

Req.token = token; // set token

} else {

// forbidden

Res.sendStatus(403);

}

}

// protected route

App.post(‘api/post’, verifyToken, (req, res) => {

Jwt.verify(req.token, ‘secretKey’, (err, authData) => {

If (err) {

Res.sendStatus(403)’

} else {

Res.json({message: ‘post created’, authData});

}

})

})