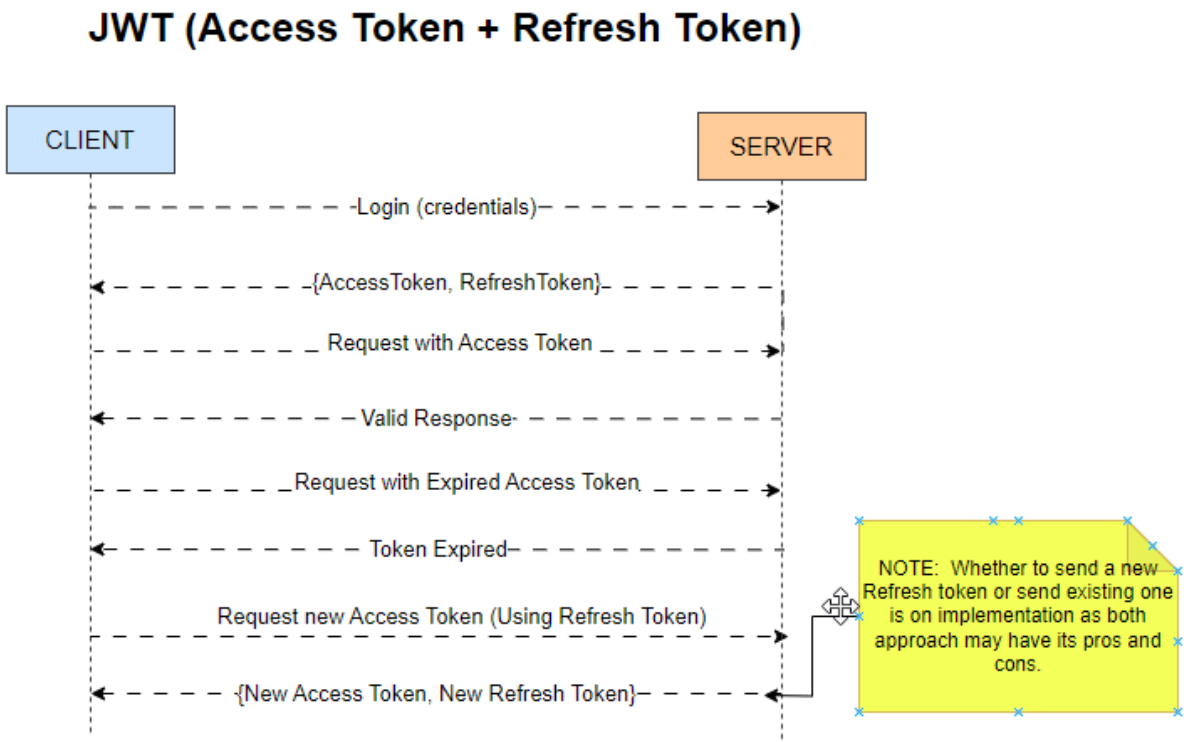
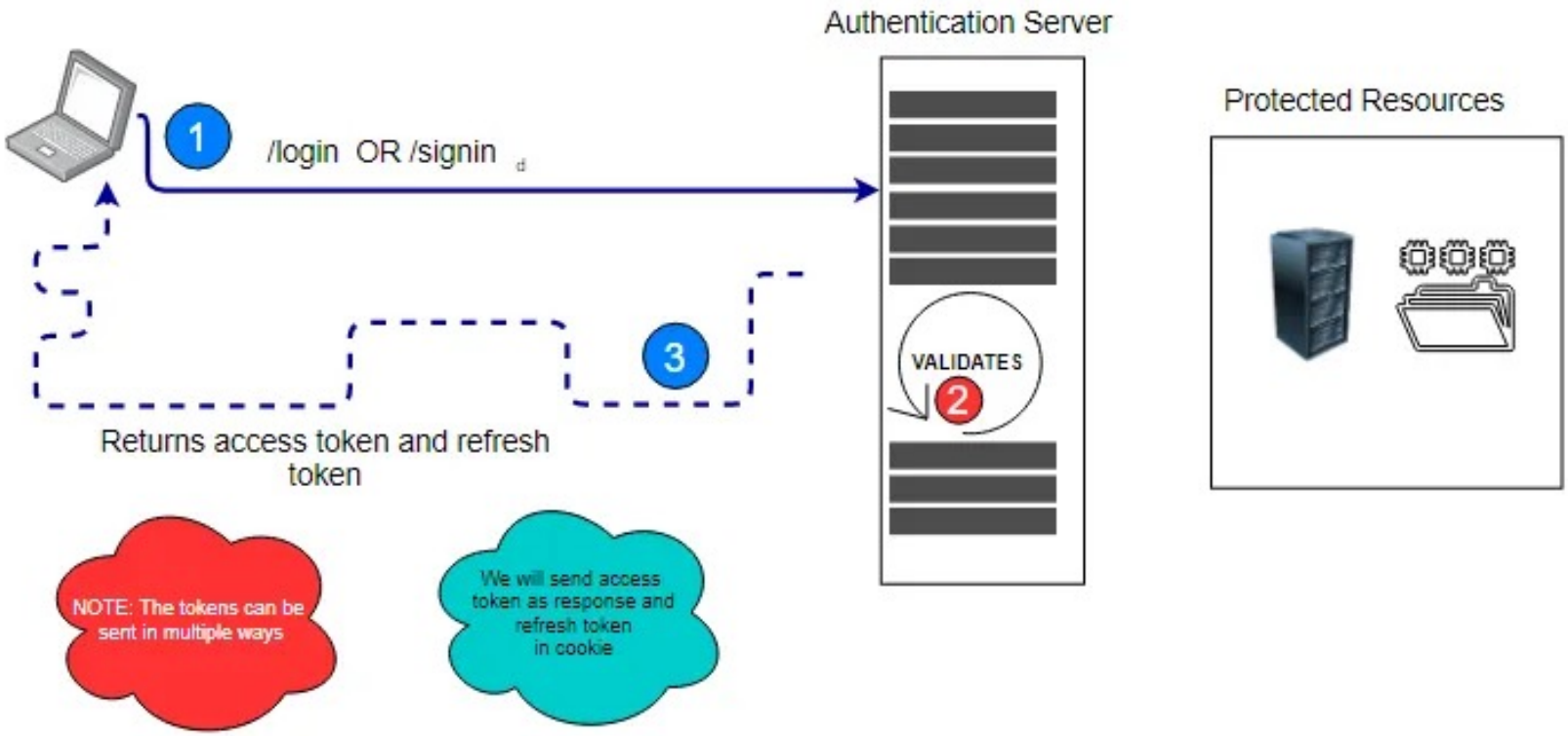
JWT & cookies

Backend – frontend interactions

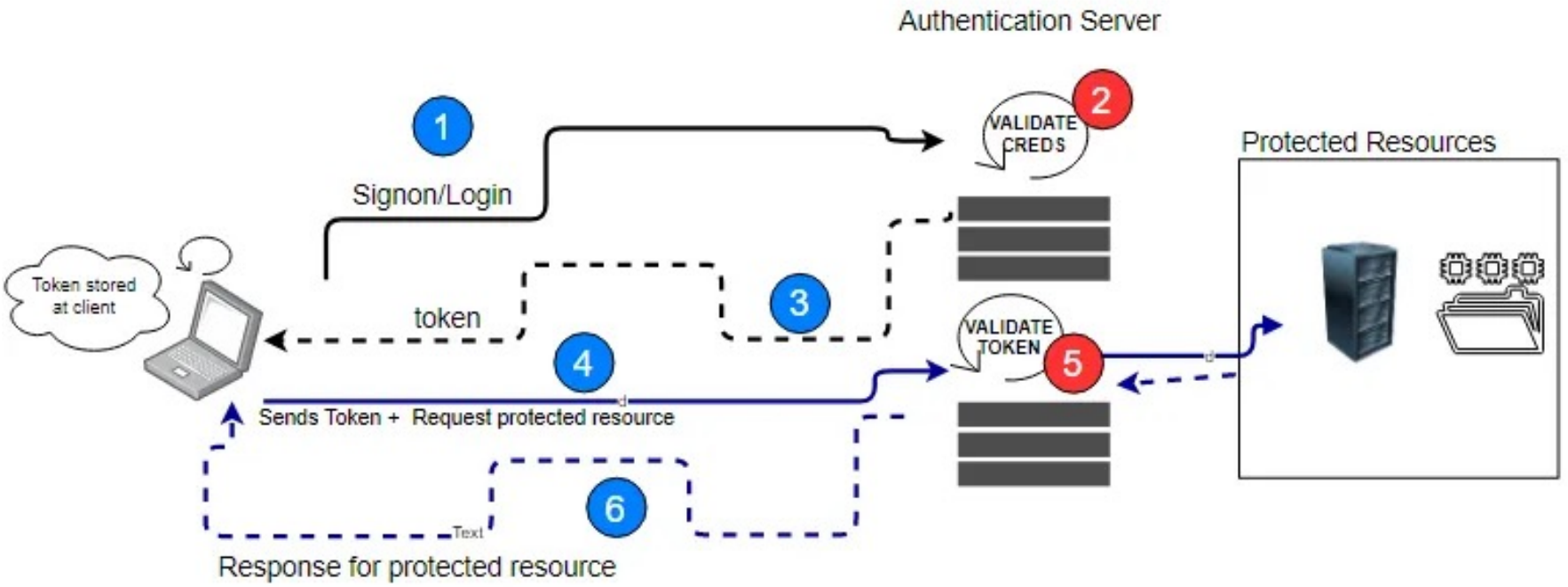
Overall Diagram



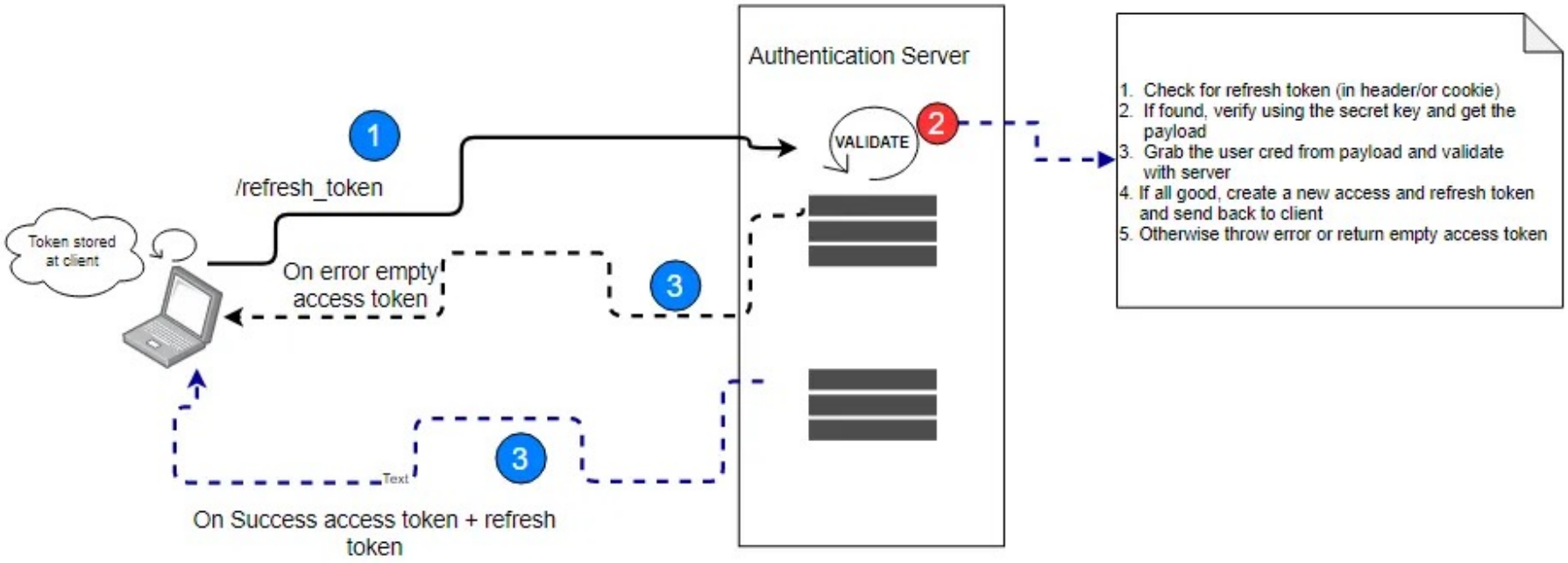
Login/sign-in flow



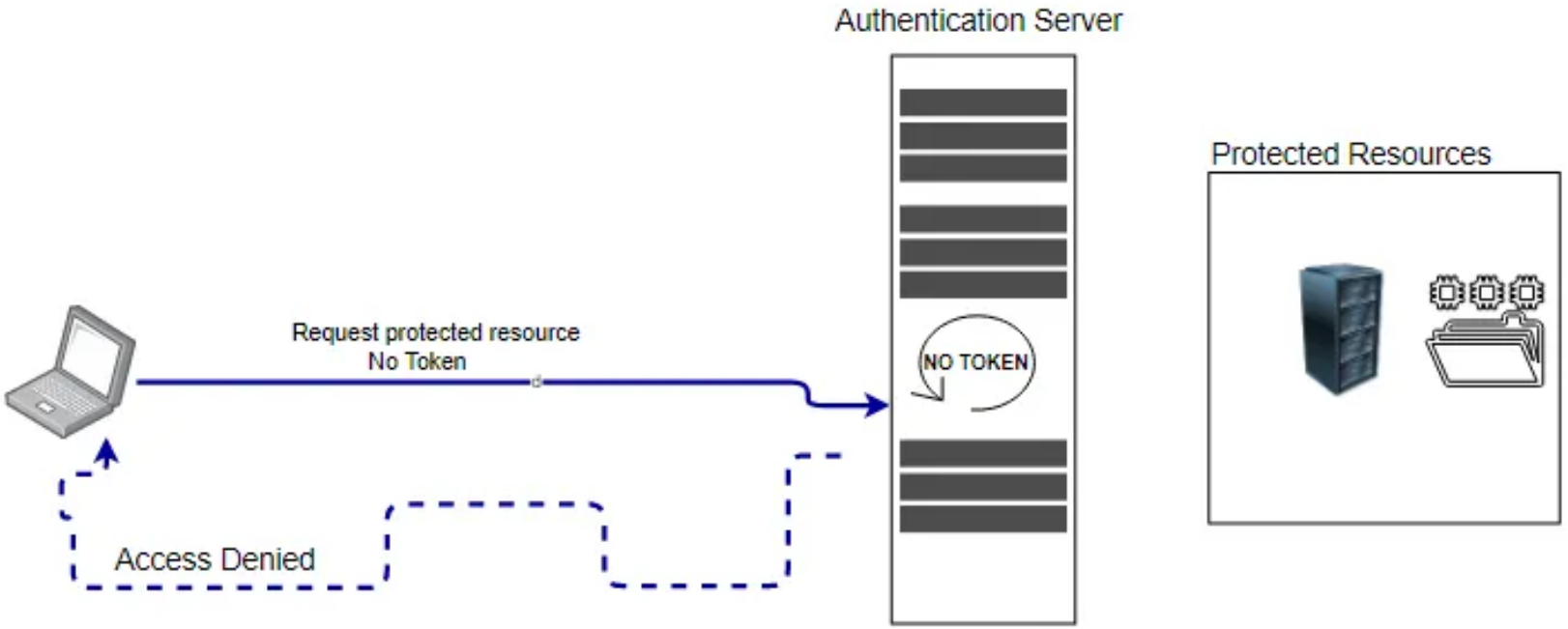
SUCCESSFUL ACCESS FLOW



REFRESH TOKEN FLOW



FAILED ACCESS FLOW



Set up basic **server.js file**

```
require('dotenv').config()
const express= require('express')
const app= express();

const jwt = require('jsonwebtoken')
let refreshTokens = [] //save to database

app.use(express.json())

app.get('/',authenticateToken,(req,res)=>{
       res.json({message:"welcome"})
})

app.listen(5400)
```

Setup token secrets in **.env**

```
ACCESS_TOKEN=
29af4b8139ae8539657f662268893bacee82c16b48dc93f370f83c495cc0c61dd5ac86fe4
8e57aa5e5ec7a35a9a1cd72318d3d2d5b1e96899948a31d80ae1ff1
REFRESH_TOKEN=7ac9fb52e8caf0dffe6aa24737ee75b175c967d36436f51ca5d18d7b477
88ff9aae6b864ff9a1d2eee5922aca53f7659fd28cb0e1230f233a266011fe87c7aa3
```

To generate secret tokens

const crypto = require('crypto’);
const token = crypto.randomBytes(64).toString('hex’);

Generate login secrets for each username in server.js

```
app.post('/login', (req, res) => {
// Authenticate User

const username = req.body.username
const user = { name: username }

const accessToken = generateAccessToken(user)
const refreshToken = jwt.sign(user, process.env.REFRESH_TOKEN)
refreshTokens.push(refreshToken)
res.json({ accessToken: accessToken, refreshToken: refreshToken })

})
```

Generator method in **server.js**

```
function generateAccessToken(user) {

return jwt.sign(user, process.env.ACCESS_TOKEN, { expiresIn: '15m' })

}
```

Return back to authentication – server.js

```
app.get('/',authenticateToken,(req,res)=>{
        res.json({message:"welcome"})
})

function authenticateToken(req,res,next){
    const authHeader=req.headers['authorization']
    const token = authHeader && authHeader.split(' ')[1]
    if(token ==null)
        return res.sendStatus(401)
    jwt.verify(token, process.env.ACCESS_TOKEN, (err,user)=>{
        if(err)
                return res.sendStatus(403)
        req.user = user
        next()
        })
}
```

Remove refresh token – server.js

```
app.delete('/logout', (req, res) => {

refreshTokens =
        refreshTokens.filter(token => token !== req.body.token)

res.sendStatus(204)

})
```

Front-end data handling

- Local storage Session Storage Cookies

```
// local storage
 localStorage.setItem('myData', JSON.stringify(data))

// cookie storage
res.cookie('authToken', token, { httpOnly: true, secure: true, sameSite: 'Strict' });

// session storage
sessionStorage.setItem('user', JSON.stringify(user));
```

Choosing the best option?

- **Security Considerations**
- **Data Size**