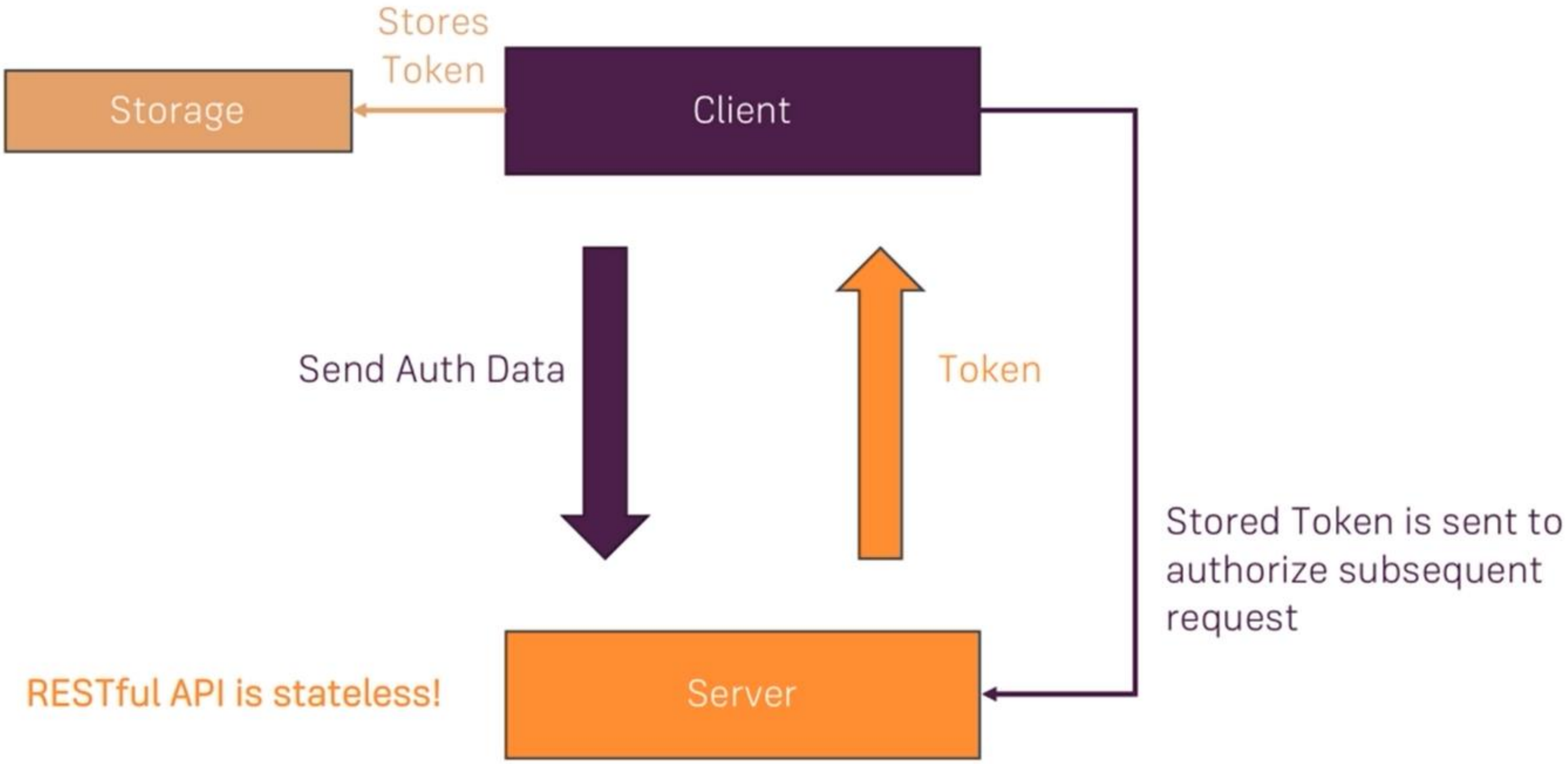
NODEJS

Express Framework

Authentication

HOW AUTHENTICATION WORKS



Stores Token

Storage

Client

Send Auth Data

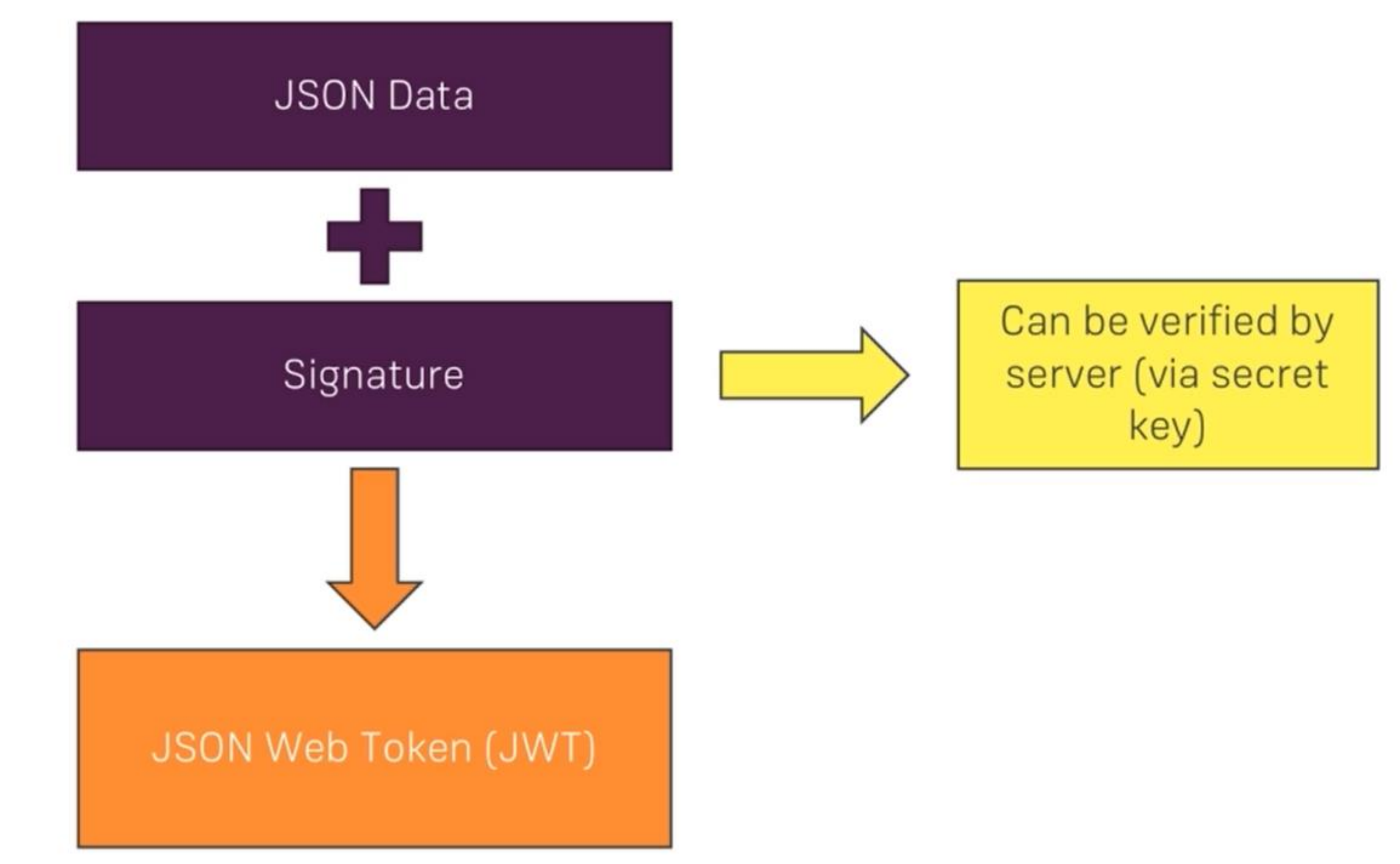
Token

Stored Token is sent to authorize subsequent request

RESTful API is stateless!

Server

WHAT'S THAT TOKEN?



OBJECTIVES

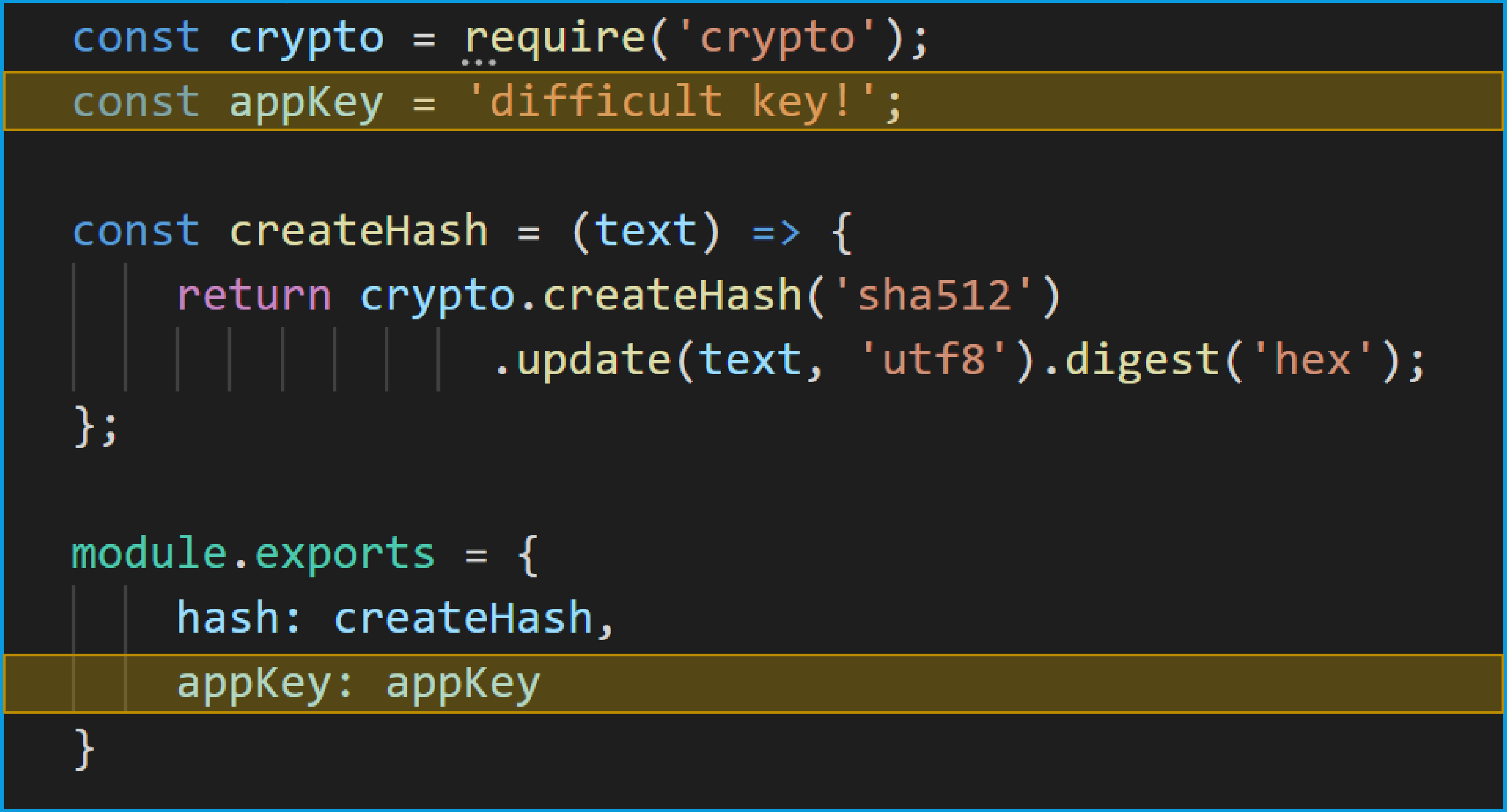
1. Installing package
2. Adding application key
3. Creating token in login api
4. Protecting apis

1. INSTALLING PACKAGES

```
npm install jsonwebtoken
```

2. ADDING APPLICATION KEY

Add appKey constant in “helper” module.

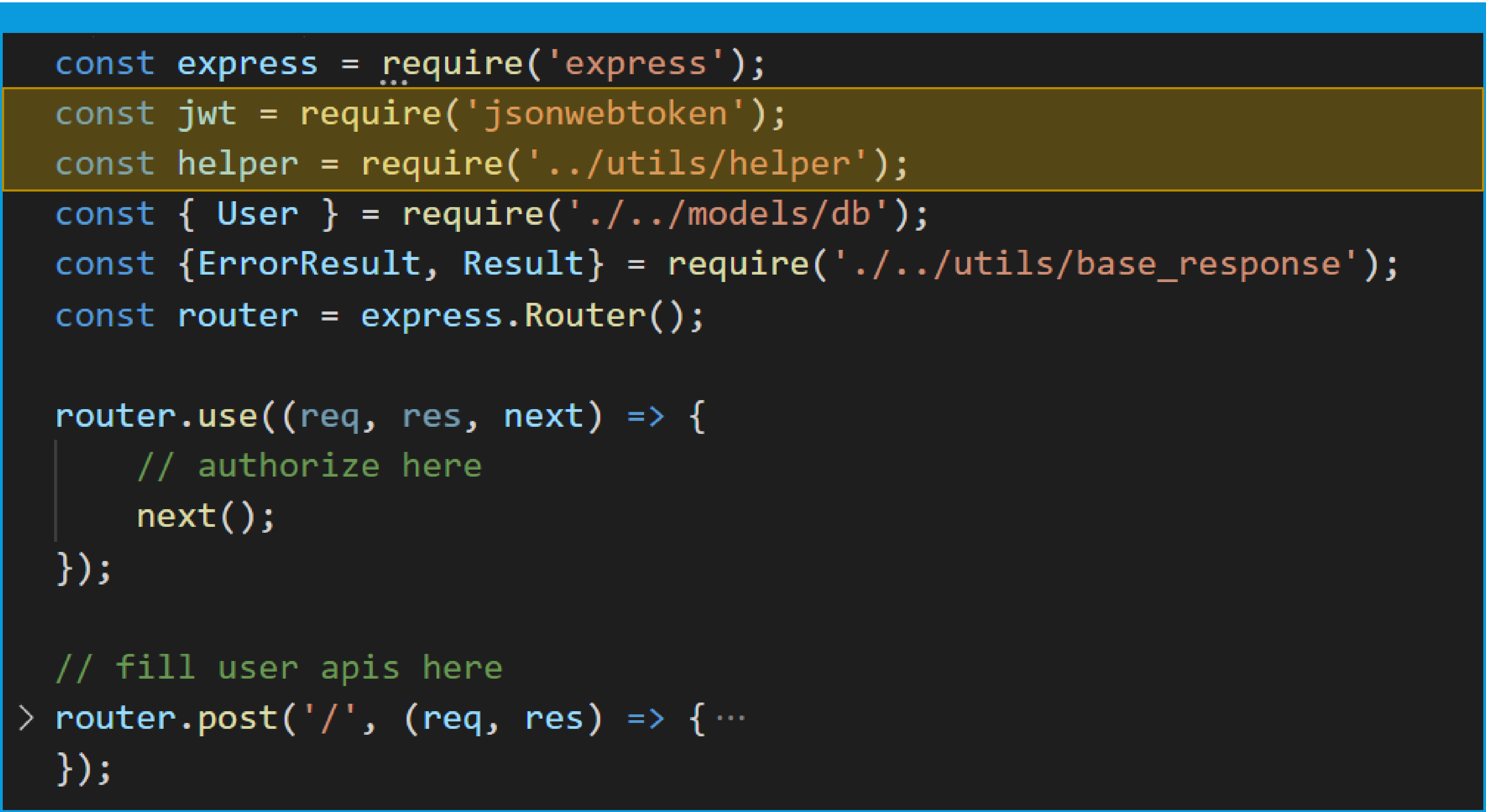


```
const crypto = require('crypto');
const appKey = 'difficult key!';

const createHash = (text) => {
    return crypto.createHash('sha512')
                .update(text, 'utf8').digest('hex');
};

module.exports = {
    hash: createHash,
    appKey: appKey
}
```

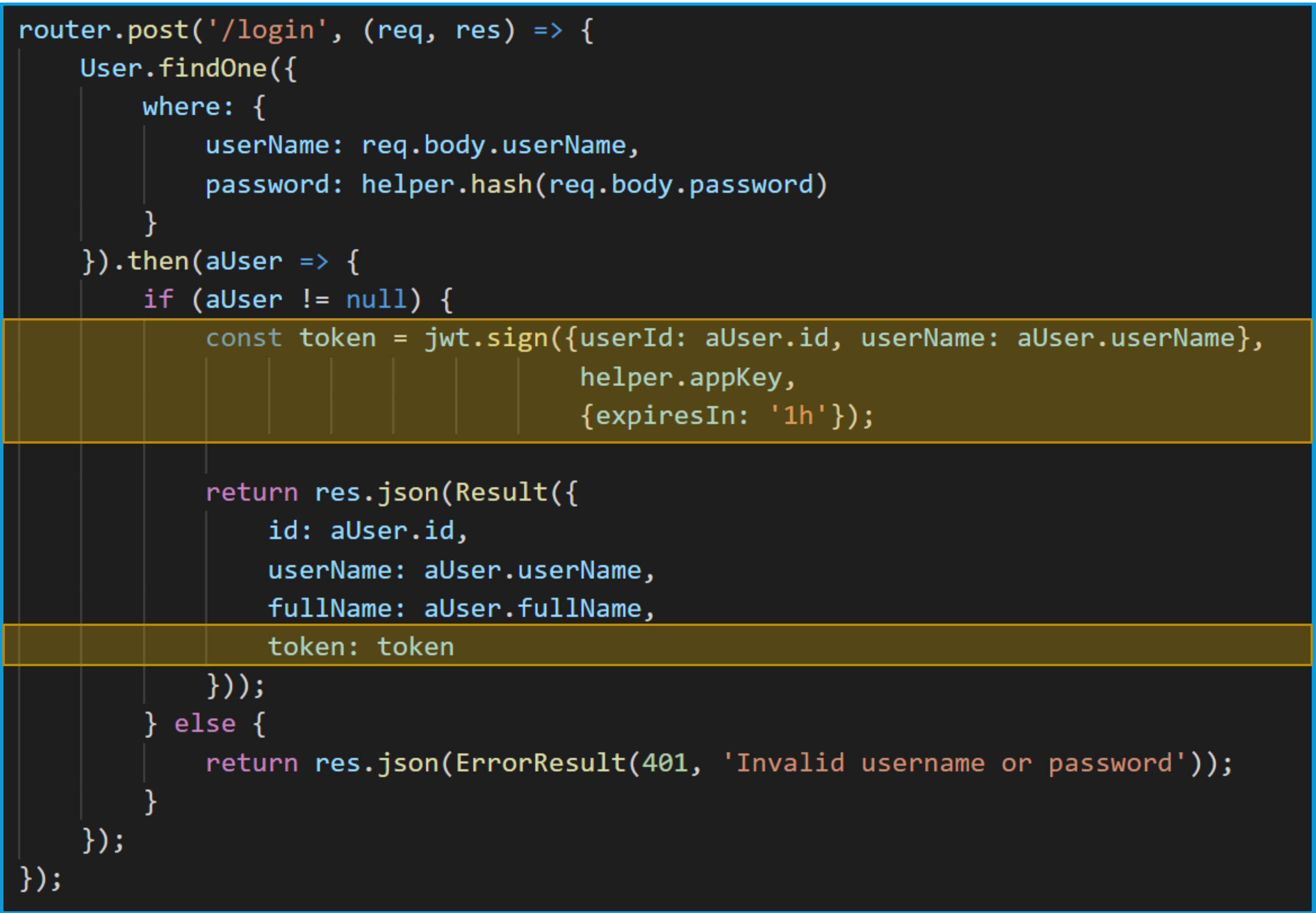
3. CREATING TOKEN IN LOGIN API



```
const express = require('express');
const jwt = require('jsonwebtoken');
const helper = require('../utils/helper');
const { User } = require('./../models/db');
const {ErrorResult, Result} = require('./../utils/base_response');
const router = express.Router();

router.use((req, res, next) => {
    // authorize here
    next();
});

// fill user apis here
router.post('/', (req, res) => {⋯
});
```



```
router.post('/login', (req, res) => {
    User.findOne({
        where: {
            userName: req.body.userName,
            password: helper.hash(req.body.password)
        }
    }).then(aUser => {
        if (aUser != null) {
            const token = jwt.sign({userId: aUser.id, userName: aUser.userName},
                                    helper.appKey,
                                    {expiresIn: '1h'});

            return res.json(Result({
                id: aUser.id,
                userName: aUser.userName,
                fullName: aUser.fullName,
                token: token
            }));
        } else {
            return res.json(ErrorResult(401, 'Invalid username or password'));
        }
    });
});
```

TEST LOGIN API

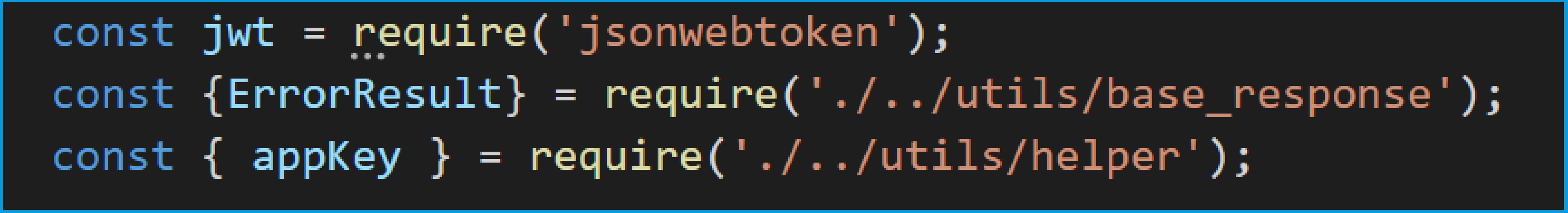
4. PROTECTING APIS

4.1. Creating Authentication module

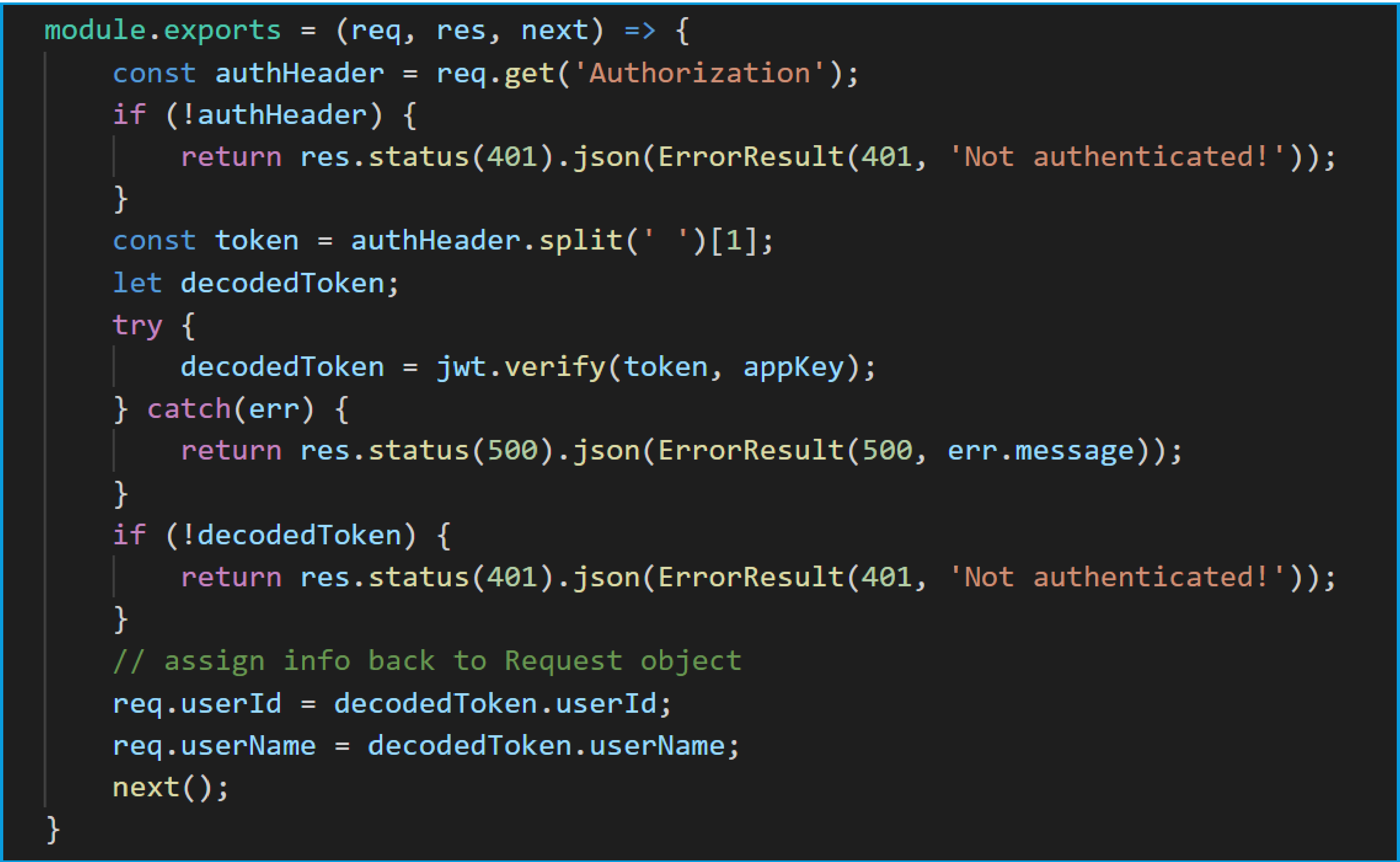
4.2. Modifying “index.js”

4.1. CREATING AUTHENTICATION MODULE

- Create a new folder named “middleware”
- Create a “auth.js” file in “middleware” folder
- Type code blocks below

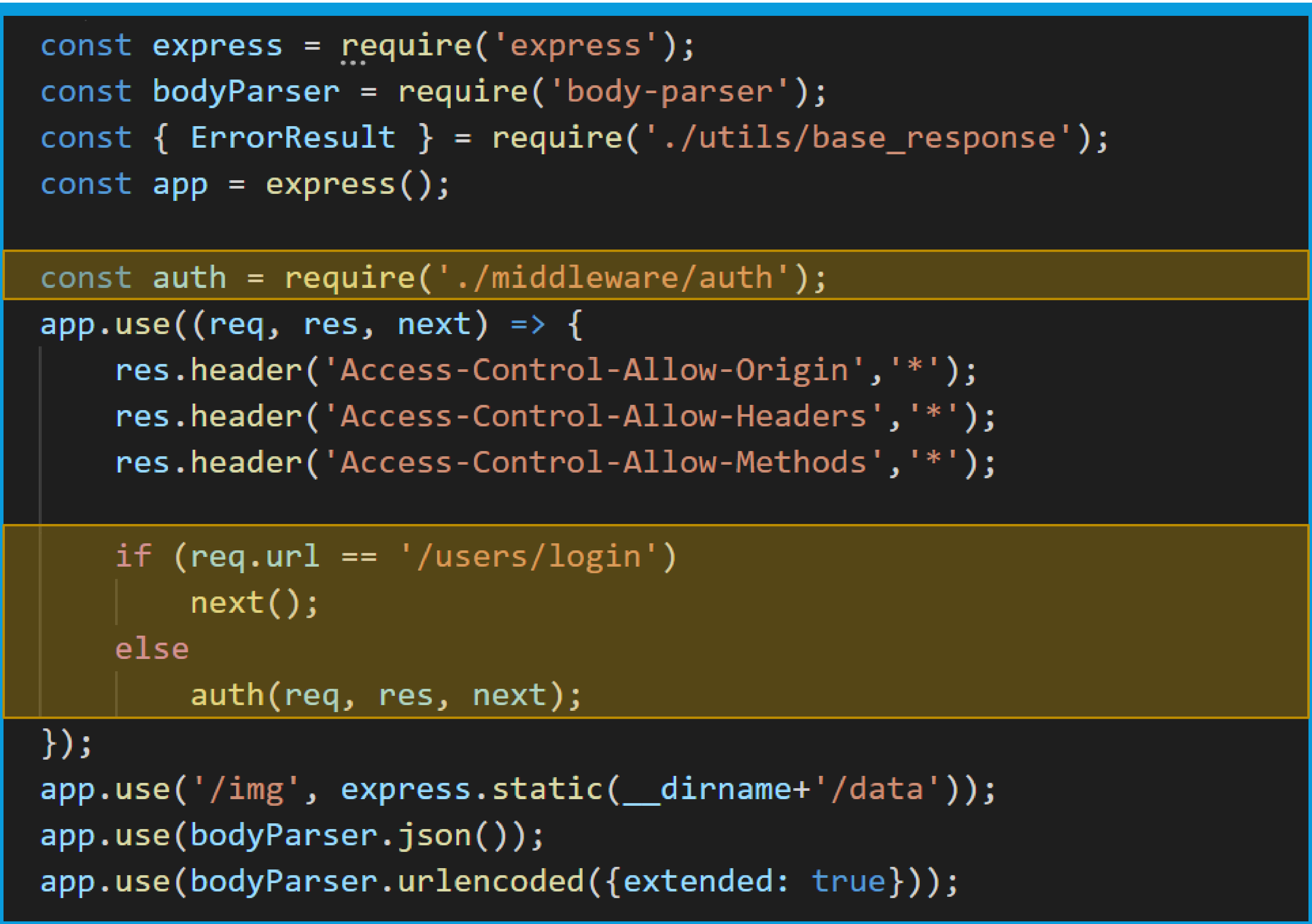


```
const jwt = require('jsonwebtoken');
const {ErrorResult} = require('./../utils/base_response');
const { appKey } = require('./../utils/helper');
```



```javascript
module.exports = (req, res, next) => {
    const authHeader = req.get('Authorization');
    if (!authHeader) {
        return res.status(401).json(ErrorResult(401, 'Not authenticated!'));
    }
    const token = authHeader.split(' ')[1];
    let decodedToken;
    try {
        decodedToken = jwt.verify(token, appKey);
    } catch(err) {
        return res.status(500).json(ErrorResult(500, err.message));
    }
    if (!decodedToken) {
        return res.status(401).json(ErrorResult(401, 'Not authenticated!'));
    }
    // assign info back to Request object
    req.userId = decodedToken.userId;
    req.userName = decodedToken.userName;
    next();
}
```

4.2. MODIFYING “INDEX.JS”



```
const express = require('express');
const bodyParser = require('body-parser');
const { ErrorResult } = require('./utils/base_response');
const app = express();

const auth = require('./middleware/auth');
app.use((req, res, next) => {
    res.header('Access-Control-Allow-Origin','*');
    res.header('Access-Control-Allow-Headers','*');
    res.header('Access-Control-Allow-Methods','*');

    if (req.url == '/users/login')
        next();
    else
        auth(req, res, next);
});
app.use('/img', express.static(__dirname+'/data'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended: true}));
```

TEST OTHER APIS