

Authentication and Authorization

Whats Auth

Authentication is the process of determining if the user is who he/she claims to be. It involves validating their email/password.

Authorization is the process of determining if the user has permission to perform a given operation.

Helping Code

<https://1drv.ms/f/s!AtGKdbMmNBGd0w6uAf8X7FHX584T>

This is a sample code with before and after.

Before section is a simple API built on top of mongoose, express without auth and in after a complete API with auth is implemented.

Users Model

Name

Email

Password: // must be hashed

isAdmin

Hashing Function

Npm | bcrypt

```
const bcrypt = require('bcrypt');  
const salt = await bcrypt.genSalt(10);  
const hashed = await bcrypt.hash('1234', salt);
```

```
//salt must be generated once in lifetime
```

Validating Password

```
const isValid = await bcrypt.compare('1234', hashed);
```

hashed is already stored password inside our db

JSON Web Token (JWT)

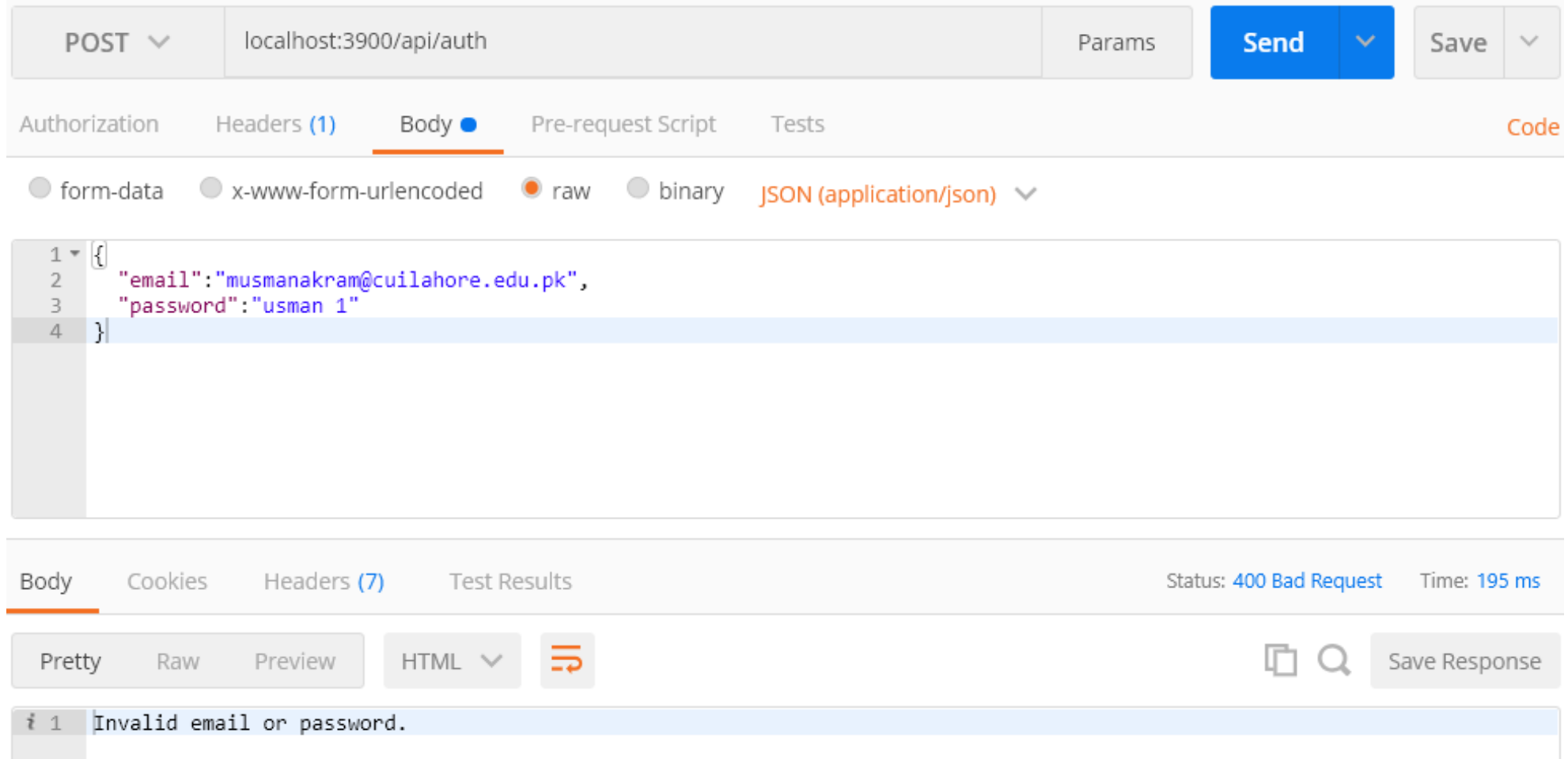
JSON object encoded as a long string

Similar to a passport or driver's license

includes a few public properties about a user

These properties cannot be tampered because doing so requires re-generating the digital signature.

Logging In Fail



POST localhost:3900/api/auth Params Send Save

Authorization Headers (1) **Body** Pre-request Script Tests Code

form-data x-www-form-urlencoded raw binary JSON (application/json)

```
1 {  
2   "email": "musmanakram@cuilahore.edu.pk",  
3   "password": "usman 1"  
4 }
```

Body Cookies Headers (7) Test Results Status: 400 Bad Request Time: 195 ms

Pretty Raw Preview HTML Save Response

```
1 Invalid email or password.
```


Logging In Success

The screenshot displays a REST client interface for a POST request to `localhost:3900/api/auth`. The request body is a JSON object with the following content:

```
1 {  
2   "email": "musmanakram@cui lahore.edu.pk",  
3   "password": "usman"  
4 }
```

The response status is `200 OK` with a response time of `196 ms`. The response body is a long alphanumeric string:

```
1 eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9  
i .eyJfaWQiOiI1YzZ0ZWYxZTQ2NzZ0MDI4NDh1MmI4OTYiLCJyZWY1IjoidXNtYW4iLCJlbWVpbCI6Im11c21hbmFrcmFtQGN1aWxhaG9yZS51ZHUucGsiLCJpYXQiOiJlNTExNjg5NzB9Lm5SWDFEArP5AbkeZ1qyokeJ65s0ytM6fyyv3qo28krM8
```

Sending Request with token

The screenshot shows a REST client interface for a POST request to `localhost:3900/api/movies`. The request is configured with the following headers:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> Content-Type	application/json	
<input checked="" type="checkbox"/> x-auth-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJfaWQiOiI1c74f8124677402848e1b1cc	
Key	Value	Description

The response is shown in the Body tab, with a status of `200 OK`, time of `23 ms`, and size of `407 B`. The response is displayed in JSON format:

```
1 {
2   "_id": "5c74f8124677402848e1b1cc",
3   "title": "Airplane WOW",
4   "genre": {
5     "_id": "5c749c241c412008b0aade22",
6     "name": "Comedy"
7   },
8   "numberInStock": 5,
9   "dailyRentalRate": 2,
10  "__v": 0
```

JSON Web Token (JWT) Process

If a user provide right credentials we generate a token and send back to client.

Client send that token back with each request

JSON Web Token (JWT) Process

```
const jwt = require('jsonwebtoken');  
const token = jwt.sign({ _id: user._id}, 'privateKey');
```

```
//install jsonwebtoken if not done yet
```

Private Keys

Never store private keys and other secrets in your codebase. Store them in environment variables.

Use the config package to read application settings stored in environment variables.

Fatty Models

```
// Adding a method to a Mongoose model
userSchema.methods.
generateAuthToken = function() {
}
const token = user.generateAuthToken();
```

Things to Do

Implement authorization using a middleware function.

Return a 401 error (unauthorized) if the client doesn't send a valid token.

Return 403 (forbidden) if

the user provided a valid token but is not allowed to perform the given operation.

You don't need to implement logging out on the server. Implement it on the client by simply removing the JWT from the client.

A Quick Knowledge Check

```
router.get('/me', auth, async (req, res) => {  
  const user = await  
User.findById(req.user._id).select('- password');  
  res.send(user);  
});  
//Remember this syntax What is it ?
```


Lets Comb The Code

<https://1drv.ms/f/s!AtGKdbMmNBGd0w6uAf8X7FHX584T>

This is a sample code with before and after.

Before section is a simple API built on top of mongoose, express without auth and in after a complete API with auth is implemented.