

The image features a glowing green padlock centered on a dark background with a complex, glowing blue circuit board pattern. The padlock has a textured, almost crystalline appearance. The word "Authentication" is written in a clean, white, sans-serif font, positioned horizontally across the middle of the padlock.

Authentication

Topics

- User REST API
- Password & Salt
- JWT Token
- Authentication
- Cookie
- Role Authorization & Route Protection

Let's start!

1. `> npm i jsonwebtoken bcryptjs`
2. Create a file: `models/User.js`
3. Create a file: `routes/auth.js`
4. Create a file: `controllers/auth.js`

```
const mongoose=require('mongoose');

const UserSchema=new mongoose.Schema({
  name:{
    type:String,
    required:[true,'Please add a name']
  },
  email:{
    type: String,
    required:[true,'Please add an email'],
    unique: true,
    match: [
      /^(([<>()[]\.\,\;\:\s@""]+(\.["<>()[]\.\,\;\:\s@""]+)*)|("\.+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.([0-9]{1,3})|((([a-zA-Z\-[0-9]+\.))+[a-zA-Z]{2,}))$)/,
      'Please add a valid email'
    ]
  },
  role: {
    type:String,
    enum: ['user','admin'],
    default: 'user'
  },
  password: {
    type:String,
    required:[true,'Please add a password'],
    minlength: 6,
    select: false
  },
  resetPasswordToken: String,
  resetPasswordExpire: Date,
  createdAt:{
    type: Date,
    default:Date.now
  }
});

module.exports = mongoose.model('User',UserSchema);
```


server.js

...

//Route files

```
const hospitals = require('./routes/hospitals')
```

```
const auth = require('./routes/auth');
```

```
app.use('/api/v1/hospitals',hospitals);
```

```
app.use('/api/v1/auth',auth);
```

...

routes/auth.js

```
// routes/auth.js

const express= require('express');
const
{register}=require('../controllers/auth');

const router=express.Router();

router.post('/register',register);

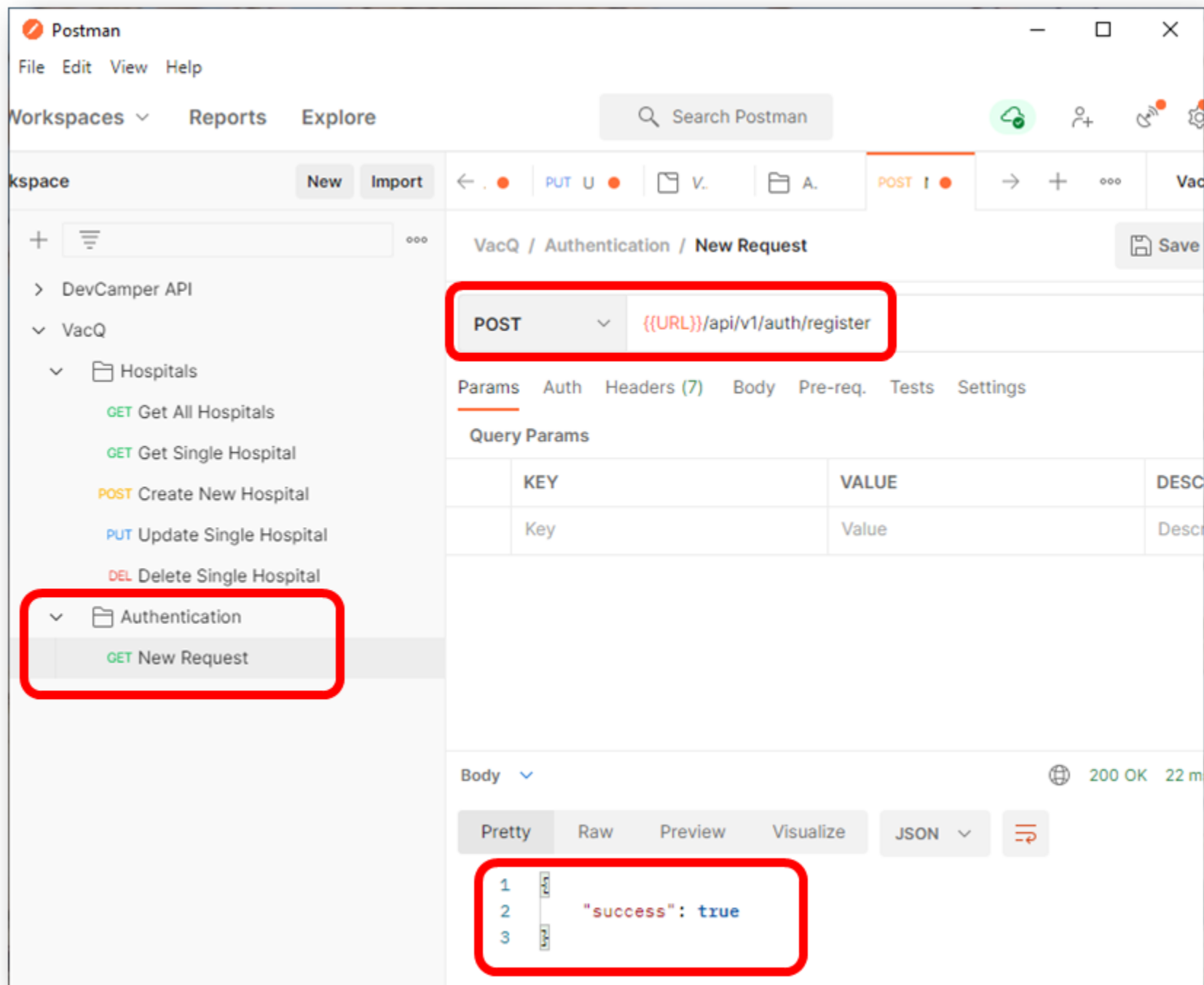
module.exports=router;
```

controllers/auth.js

```
// controllers/auth.js

const User = require('../models/User');

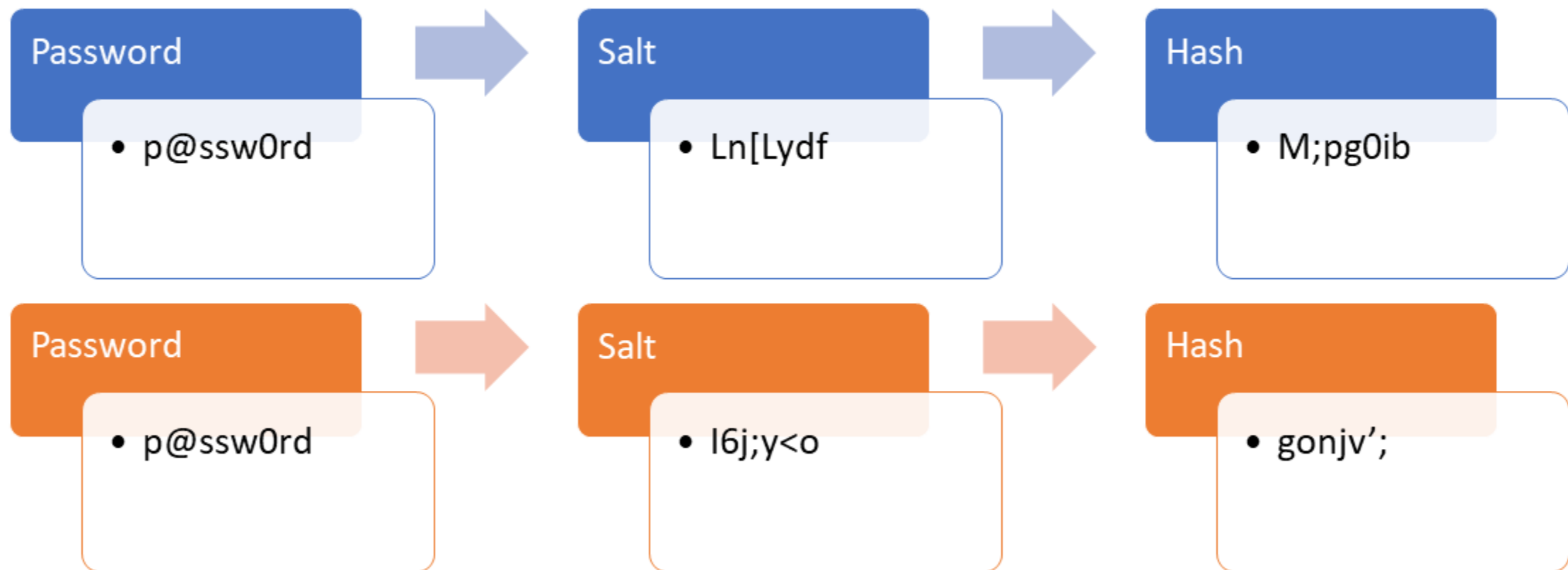
//@desc    Register user
//@route    POST /api/v1/auth/register
//@access   Public
exports.register=(req,res,next)=>{
    res.status(200).json({success:true});
}
```



User Register & Encrypting Passwords: models/User.js

```
const mongoose=require('mongoose');  
const bcrypt=require('bcryptjs');  
  
const UserSchema=new mongoose.Schema({  
  ...  
});  
  
//Encrypt password using bcrypt  
UserSchema.pre('save',async function(next){  
  const salt=await bcrypt.genSalt(10);  
  this.password=await bcrypt.hash(this.password,salt);  
});  
  
module.exports = mongoose.model('User',UserSchema);
```


Password + Salt



Ref: <https://www.npmjs.com/package/bcrypt>

controllers/auth.js

```
...
...
exports.register=async (req,res,next)=>{
  try{
    const {name, email, password, role}=req.body;

    //Create user
    const user=await User.create({
      name,
      email,
      password,
      role
    });
    res.status(200).json({success:true});
  } catch(err){
    res.status(400).json({success:false});
    console.log(err.stack);
  }
};
```

POST {{URL}}/api/v1/auth/register Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies </>

	KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Content-Type	application/json	JSON Type			

POST {{URL}}/api/v1/auth/register

Params Authorization Headers (10) Body Pre-request

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw

```
1 {
2   "name": "John Doe",
3   "email": "john@gmail.com",
4   "password": "123456",
5   "role": "publisher"
6 }
```

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "success": true
3 }
```

+ Create Database

VacQ

hospitals

users

devcamper

firstAPIproj

VacQ.users

COLLECTION SIZE: 186B TOTAL DOCUMENTS: 1 INDEXES TOTAL SIZE: 40KB

Find Indexes Schema Anti-Patterns 0 Aggregation Search

FILTER { field: 'value' }

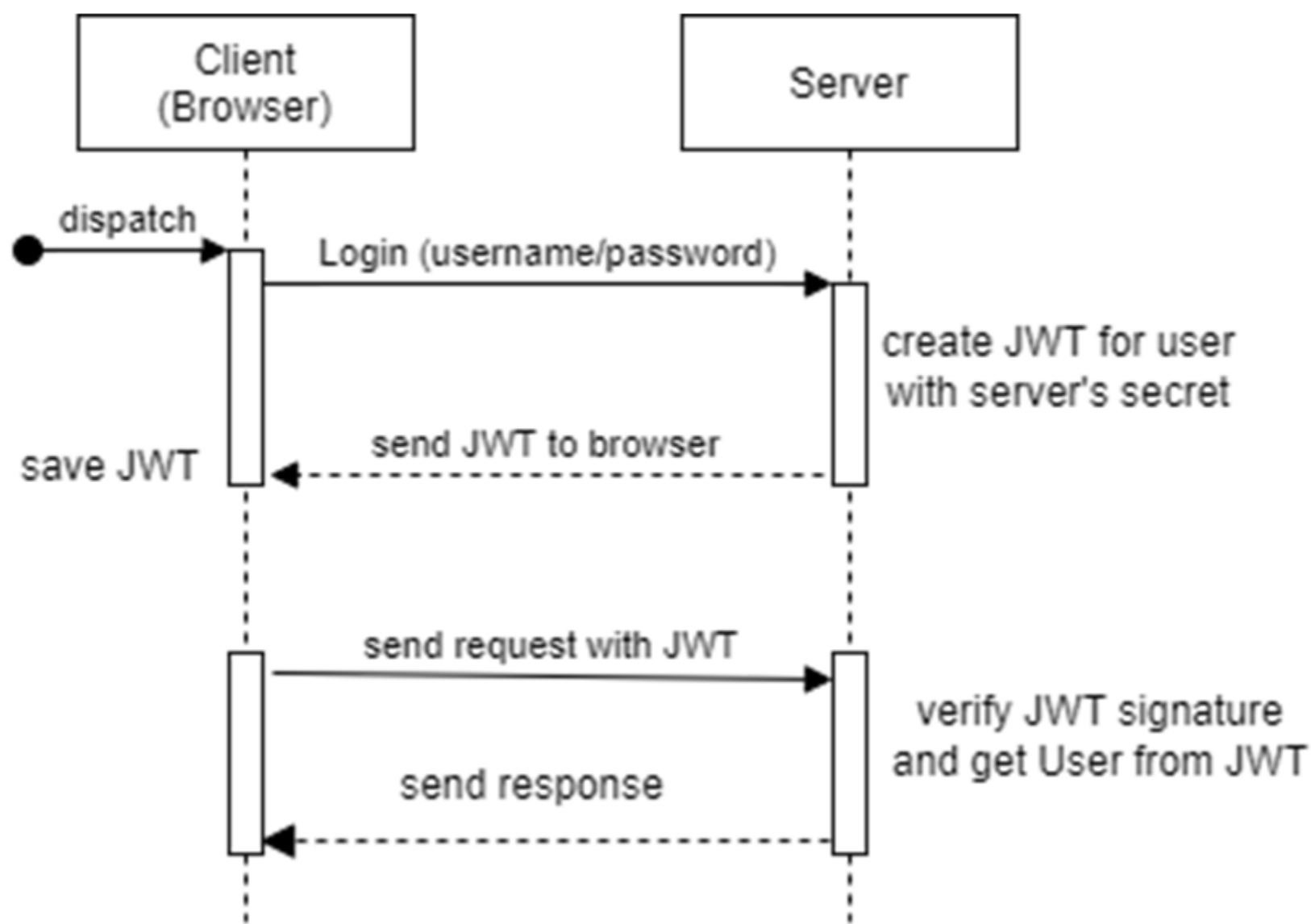
QUERY RESULTS 1-1 OF 1

```
{
  _id: ObjectId("61388457f87bdf7aa3a682ee")
  name: "John Doe"
  email: "john@gmail.com"
  role: "admin"
  password: "$2a$10$IflGVtOKQsA34Dph.sYGb.oss02int5tCAMXZaglPYlMhIQ2MP216"
  createdAt: 2021-09-08T09:37:27.179+00:00
  __v: 0
}
```



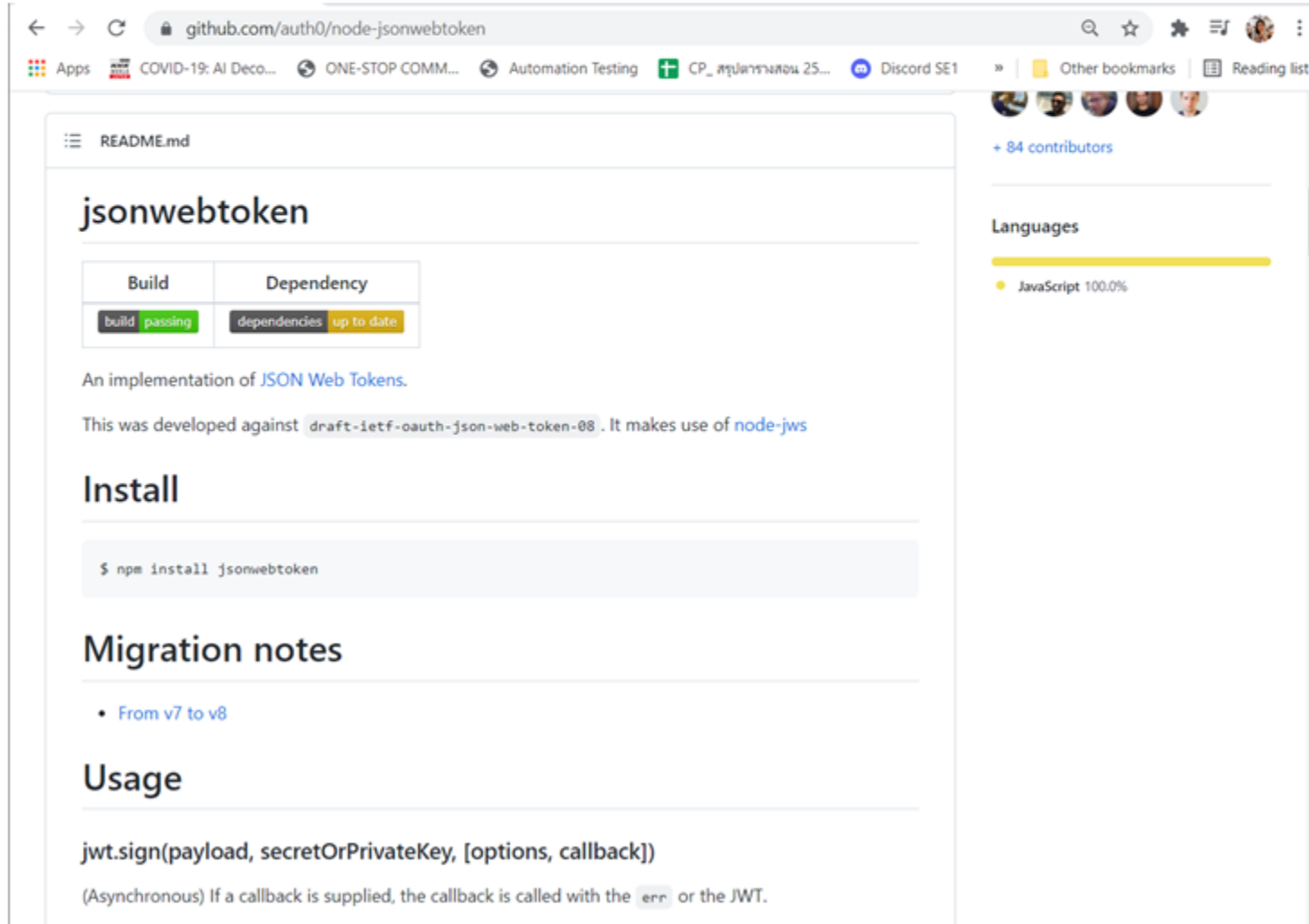
JWT: JSON Web Token

JWT – JSON Web Token



JSON Web Token Reference

- <https://github.com/auth0/node-jsonwebtoken>



The screenshot shows the GitHub repository page for `auth0/node-jsonwebtoken`. The browser's address bar displays the URL `github.com/auth0/node-jsonwebtoken`. The repository's README is visible, featuring the title `jsonwebtoken` and a table with build and dependency status. The build status is 'passing' (green) and the dependencies are 'up to date' (yellow). Below the table, the README describes the package as an implementation of JSON Web Tokens, developed against `draft-ietf-oauth-json-web-token-08` and using `node-jws`. The 'Install' section provides the command `$ npm install jsonwebtoken`. The 'Migration notes' section lists a link for 'From v7 to v8'. The 'Usage' section shows the `jwt.sign` function signature: `jwt.sign(payload, secretOrPrivateKey, [options, callback])`, followed by a note that it is asynchronous and calls the callback with an error or the JWT.

github.com/auth0/node-jsonwebtoken

Apps COVID-19: AI Deco... ONE-STOP COMM... Automation Testing CP_ สรุปการสอบ 25... Discord SE1 Other bookmarks Reading list

README.md

+ 84 contributors

Languages

- JavaScript 100.0%

Build	Dependency
build passing	dependencies up to date

An implementation of [JSON Web Tokens](#).

This was developed against `draft-ietf-oauth-json-web-token-08`. It makes use of [node-jws](#).

Install

```
$ npm install jsonwebtoken
```

Migration notes

- [From v7 to v8](#)

Usage

`jwt.sign(payload, secretOrPrivateKey, [options, callback])`

(Asynchronous) If a callback is supplied, the callback is called with the `err` or the JWT.

config/config.env

```
NODE_ENV=development
```

```
PORT=5000
```

```
MONGO_URI=mongodb+srv://ohm123:ohm123@traversymedia.o0hi2.mongodb.net/devcamper?retryWrites=true&w=majority
```

```
JWT_SECRET=asdfjkl;;lkjfdsa
```

```
JWT_EXPIRE=30d
```

models/User.js

```
const mongoose=require('mongoose');
const bcrypt=require('bcryptjs');
const jwt=require('jsonwebtoken');

const UserSchema=new mongoose.Schema({
  ...
});

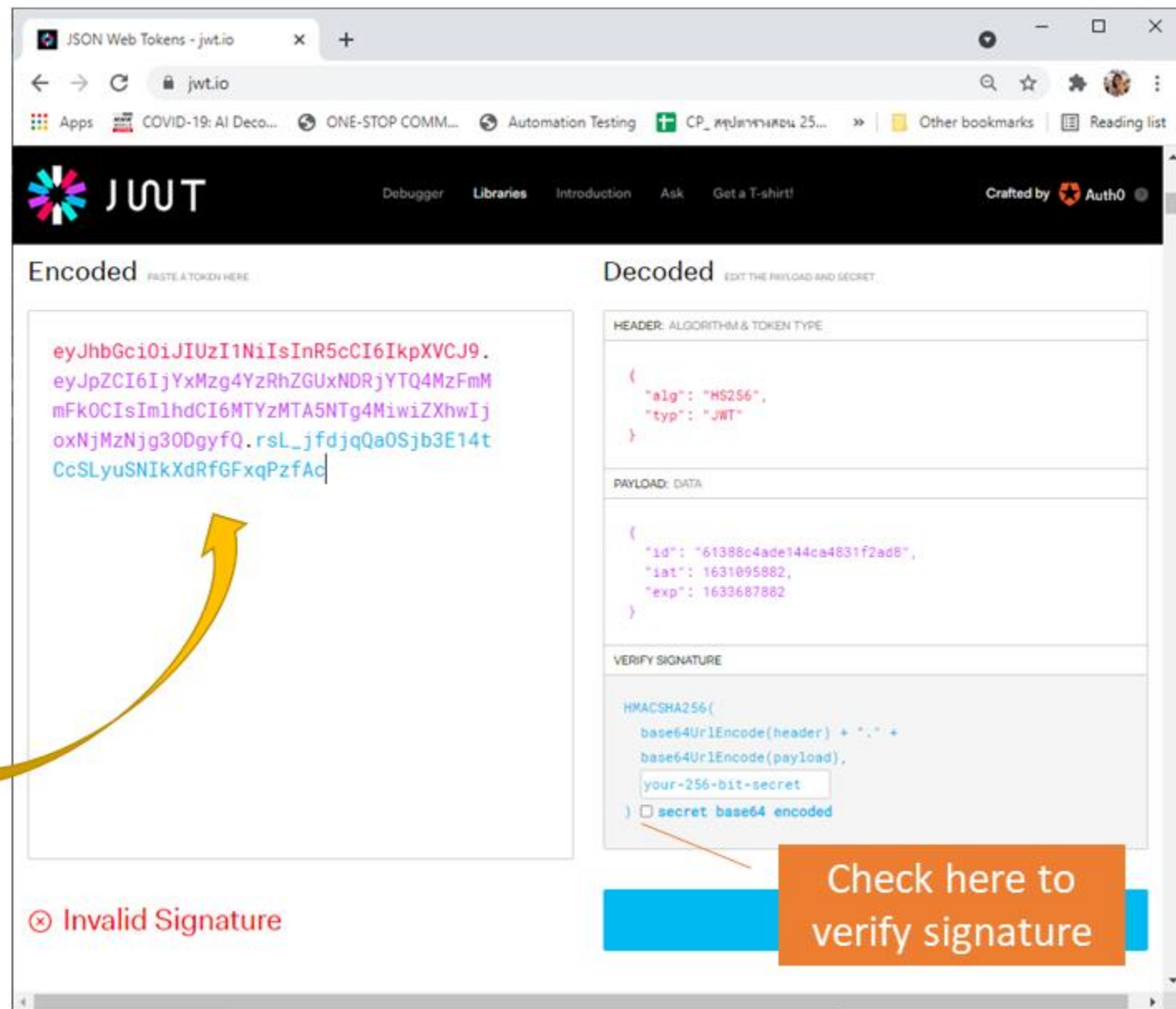
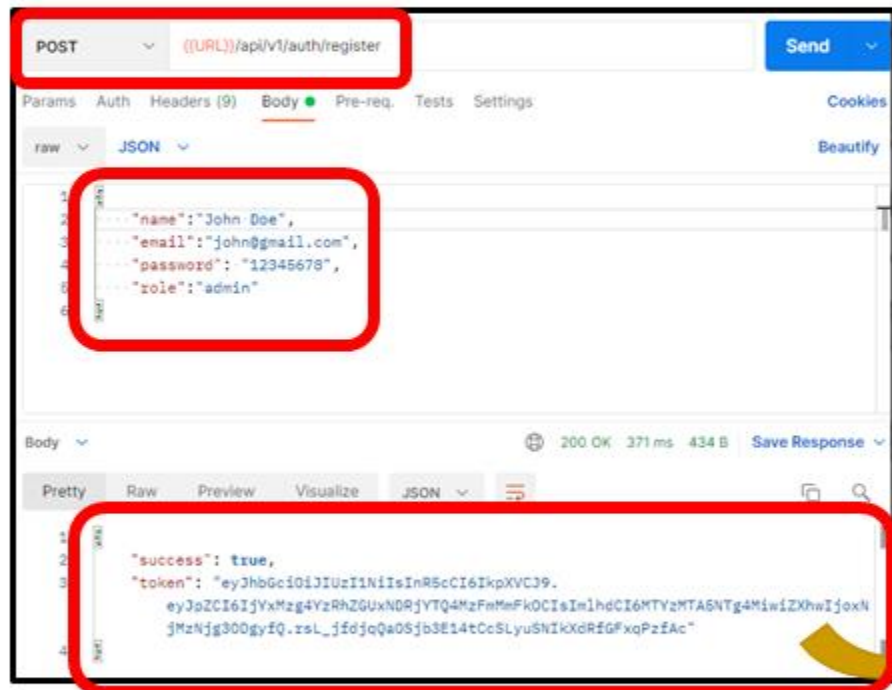
//Encrypt password using bcrypt
UserSchema.pre('save',async
  ...
);

//Sign JWT and return
UserSchema.methods.getSignedJwtToken=function() {
  return jwt.sign({id:this._id},process.env.JWT_SECRET,{
    expiresIn: process.env.JWT_EXPIRE
  });
}

module.exports = mongoose.model('User',UserSchema);
```

controllers/auth.js

```
...  
  
//Create user  
  
const user=await User.create({  
  name,  
  email,  
  password,  
  role  
});  
  
//Create token  
  
const token=user.getSignedJwtToken();  
  
res.status(200).json({success:true,token});  
  
});
```





User Login

User Login: models/User.js

...

```
//Sign JWT and return
```

```
UserSchema.methods.getSignedJwtToken=function(){...
```

```
}
```

```
//Match user entered password to hashed password in database
```

```
UserSchema.methods.matchPassword=async function(enteredPassword){
```

```
  return await bcrypt.compare(enteredPassword, this.password);
```

```
}
```

```
module.exports = mongoose.model('User',UserSchema);
```

User Login: controllers/auth.js

ref: https://mongoosejs.com/docs/api.html#query_Query-select

```
...
exports.register=async (req,res,next)=>{
...
};

//@desc    Login user
//@route    POST /api/v1/auth/login
//@access   Public
exports.login=async (req,res,next)=>{
  const {email, password}=req.body;

  //Validate email & password
  if(!email || !password){
    return res.status(400).json({success:false,
msg:'Please provide an email and password'});
  }

  //Check for user
  const user = await
User.findOne({email}).select('+password');

  if(!user){
    return res.status(400).json({success:false,
msg:'Invalid credentials'});
  }

  //Check if password matches
  const isMatch = await user.matchPassword(password);

  if(!isMatch){
    return res.status(401).json({success:false,
msg:'Invalid credentials'});
  }

  //Create token
  const token=user.getSignedJwtToken();

  res.status(200).json({success:true,token});
};
```

routes/auth.js

```
const express= require('express');  
const {register, login}=require('../controllers/auth');  
  
const router=express.Router();  
  
router.post('/register',register);  
router.post('/login',login);  
  
module.exports=router;
```

Postman

File Edit View Help

Home Workspaces Reports Explore

Search Postman

My Workspace

New Import

VacQ / Authentication / Login

POST `{{URL}}/api/v1/auth/login` Send

Params Auth Headers (9) Body Pre-req. Tests Settings Cookies Beautify

raw JSON

```
1  ... "email": "john@gmail.com",
2  ... "password": "12345678"
3
4
```

Body 200 OK 423 ms 434 B Save Response

Pretty Raw Preview Visualize JSON

```
1
2  "success": true,
3  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYxMzg4YzRhZGUxNDRjYTQ4MzFmMmFkOCIsIm1hdCI6MTYzMTEwNzA4NiwiZXhwIjoxNjMzNjk5MDg2fQ.X0VpwZLhalpMPHskAZhcXlXKiyepWDXgkYU-Cw1qQz0"
4
```




Sending JWT in a Cookie

server.js

> npm i cookie-parser

```
const express = require('express');
const dotenv = require('dotenv');
const cookieParser = require('cookie-parser');
const connectDB = require('./config/db');

//Load env vars
dotenv.config({path: './config/config.env'});

...

//Body parser
app.use(express.json());

//Cookie parser
app.use(cookieParser());

...
```


config/config.env

```
NODE_ENV=development
```

```
PORT=5000
```

```
MONGO_URI=mongodb+srv://ohm123:ohm123@trav  
ersymedia.o0hi2.mongodb.net/devcamper?retr  
yWrites=true&w=majority
```

```
JWT_SECRET=asdfjkl;;lkjfdsa
```

```
JWT_EXPIRE=30d
```

```
JWT_COOKIE_EXPIRE=30
```

controllers/auth.js (1 of 2)

```
exports.register=async (req,res,next)=>{  
  ...  
  //const token=user.getSignedJwtToken();  
  //res.status(200).json({success:true});  
  sendTokenResponse(user,200,res);  
  
  ...  
};  
  
exports.login=async (req,res,next)=>{  
  ...  
  //const token=user.getSignedJwtToken();  
  //res.status(200).json({success:true});  
  sendTokenResponse(user,200,res);  
  
  ...  
};
```

controllers/auth.js (2 of 2)

```
//Get token from model, create cookie and send response
const sendTokenResponse=(user, statusCode, res)=>{
  //Create token
  const token=user.getSignedJwtToken();

  const options = {
    expires:new Date(Date.now()+process.env.JWT_COOKIE_EXPIRE*24*60*60*1000),
    httpOnly: true
  };

  if(process.env.NODE_ENV==='production'){
    options.secure=true;
  }

  res.status(statusCode).cookie('token',token,options).json({
    success: true,
    token
  })
}
```

Postman

File Edit View Help

Home Workspaces Reports Explore

Search Postman

Invite Upgrade

My Workspace New Import

VacQ / Authentication / Login

POST {{URL}}/api/v1/auth/login

Send

Params Auth Headers (10) Body Pre-req. Tests Settings

raw JSON

Cookies (1)

Name	Value	Domain	Path	Expires	HttpOnly	Secure
token	eyJhbGciOiJI...	localhost	/	Fri, 08 Oct 20...	true	false

Body Cookies (1) Headers (8) Test Results

200 OK 426 ms 682 B Save Response

Find and Replace Console

Bootcamp Runner Trash



Auth-Protect Middleware

middleware/auth.js

```
const jwt = require('jsonwebtoken');
const User = require('../models/User');

//Protect routes
exports.protect=async (req,res,next)=>{
  let token;

  if(req.headers.authorization && req.headers.authorization.startsWith('Bearer')){
    token = req.headers.authorization.split(' ')[1];
  }

  //Make sure token exists
  if(!token) {
    return res.status(401).json({success:false, message:'Not authorize to access this route'});
  }

  try{
    //Verify token
    const decoded = jwt.verify(token,process.env.JWT_SECRET);

    console.log(decoded);

    req.user=await User.findById(decoded.id);

    next();
  }catch(err){
    console.log(err.stack);
    return res.status(401).json({success:false, message:'Not authorize to access this route'});
  }
};
```


routes/hospitals.js

...

```
const router = express.Router();
```

```
const {protect} = require(' ../middleware/auth');
```

...

```
router.route('/').get(getHospitals).post(protect, createHospital);
```

```
router.route('/:id').get(getHospital).put(protect, updateHospital).delete(protect,  
deleteHospital);
```

```
module.exports=router;
```

controllers/auth.js

```
//At the end of file
//@desc      Get current Logged in user
//@route      POST /api/v1/auth/me
//@access     Private
exports.getMe=async(req,res,next)=>{
    const user=await User.findById(req.user.id);
    res.status(200).json({
        success:true,
        data:user
    });
};
```

routes/auth.js

```
const express= require('express');  
const {register, login,getMe}=require('../controllers/auth');  
  
const router=express.Router();  
  
const {protect}= require('../middleware/auth');  
  
router.post('/register',register);  
router.post('/login',login);  
router.get('/me',protect,getMe);  
  
module.exports=router;
```

Postman

File Edit View Help

Home Workspaces Reports Explore

Search Postman

Invite Upgrade

My Workspace

New Import

POST Login GET Get Logged In Us... GET Register user

VacQ

Collections

- DevCamper API
- VacQ
 - Hospitals
 - GET Get All Hospitals
 - GET Get Single Hospital
 - POST Create New Hospital
 - PUT Update Single Hospital
 - DEL Delete Single Hospital
 - Authentication
 - GET Register user
 - POST Login
 - GET Get Logged In User with Token

VacQ / Authentication / Get Logged In User with Token

Save

GET `{{URL}}/api/v1/auth/me` Send

Params Authorization Headers (11) Body Pre-request Script Tests Settings Cookies

Headers 9 hidden

	KEY	VALUE	DESCRIPTION		Bulk Edit	Presets
<input checked="" type="checkbox"/>	Content-Type	application/json	JSON Type			
<input checked="" type="checkbox"/>	Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVC...				

Body Cookies (1) Headers (7) Test Results

200 OK 3.24 s 399 B Save Response

Pretty Raw Preview Visualize JSON

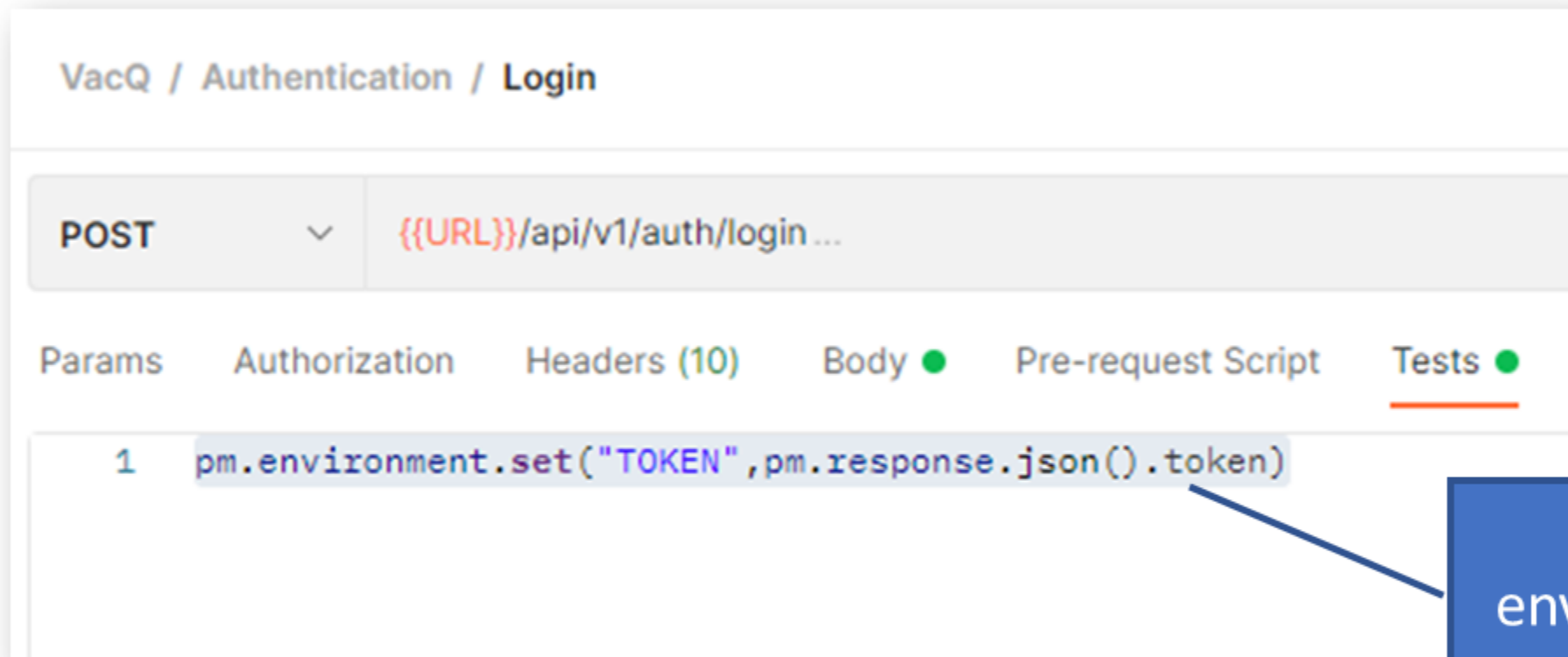
```
1  {
2    "success": true,
3    "data": {
4      "_id": "61388c4ade144ca4831f2ad8",
5      "name": "John Doe",
6      "email": "john@gmail.com",
7      "role": "admin",
8      "createdAt": "2021-09-08T10:11:22.223Z",
9      "__v": 0
10   }
11 }
```

Copy Token from Login Response (without "quote")



Storing Token in Postman

Storing The Token In Postman



Postman

File Edit View Help

Home Workspaces Reports Explore

Search Postman

Upgrade

My Workspace

New Import

VacQ / Authentication / Login

POST Login

VacQ

Edit

VARIABLE	INITIAL VALUE	CURRENT VALUE
URL	http://localhost:5000	http://localhost:5000
TOKEN		eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYxMzg4YzRhZGUxNDRjYTQ4MzFmMmFkOCIsImhhdCI6MTYzMTYwODU3MSwiZXhwIjojM0MjAwNTcxZQJJCQ03LbsO...

1 pm.environment.set(

Body Cookies (1) Headers (8)

Pretty Raw Preview

```
1 {
2   "success": true,
3   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjYxMzg4YzRhZGUxNDRjYTQ4MzFmMmFkOCIsImhhdCI6MTYzMTYwODU3MSwiZXhwIjojM0MjAwNTcxZQJJCQ03LbsOYVd"
4 }
```

Globals

Add

2. เมื่อ login แล้ว ค่า token ใน response ถูกบันทึกใน environment อัตโนมัติ

Use variables sensitive values with your team. Learn more about variable values

Find and Replace Console

Bootcamp Runner Trash

Storing The Token In Postman: Register Request

Params Authorization Headers (10) Body ● Pre-request Script Tests ●

```
1 pm.environment.set("TOKEN",pm.response.json().token)
```

1. Set postman environment ด้วยค่าที่ได้จาก token ใน response

3.บันทึก token ไว้ในทุก request
ที่ต้องให้ login ก่อนใช้งาน

2.Authorization

3. Bearer Token

4. ใส่ตัวแปร TOKEN

1. เลือก request

The screenshot shows the Postman interface with the following elements and annotations:

- Left Sidebar:** Contains a list of API collections. The 'Authentication' collection is expanded, showing endpoints like 'Register user', 'Login', and 'Get Logged In User with Token'. The 'Get Logged In User with Token' endpoint is highlighted with a red box and labeled '1. เลือก request'.
- Top Bar:** Includes a 'Save' button with a dropdown arrow, highlighted with a red box.
- Request Editor:** The URL is set to `{{URL}}/api/v1/auth/me`. The 'Params' tab is active, and the 'Authorization' dropdown is set to 'Bearer ...', both highlighted with red boxes and labeled '2.Authorization' and '3. Bearer Token' respectively.
- Token Field:** The 'Token' input field contains the placeholder `{{TOKEN}}`, highlighted with a red box and labeled '4. ใส่ตัวแปร TOKEN'.
- Bottom:** A cartoon astronaut icon is present with the text 'Click Send to get a response'.

Role Authorization



middleware/auth.js

```
..  
//at the end of file  
//Grant access to specific roles  
exports.authorize=(...roles)=>{  
  return (req,res,next)=>{  
    if(!roles.includes(req.user.role)){  
      return res.status(403).json({success:false,  
message:`User role ${req.user.role} is not authorized to access  
this route`});  
    }  
    next();  
  }  
}
```


routes/hospitals.js

```
...  
const {protect, authorize} = require('../middleware/auth');  
  
router.route('/').get(getHospitals).post(protect, authorize('admin'), createHospital);  
router.route('/:id').get(getHospital).put(protect, authorize('admin'), updateHospital).delete(protect, authorize('admin'), deleteHospital);  
  
module.exports=router;
```


VacQ / Hospitals / Create New Hospital

Save

POST

{{URL}}/api/v1/hospitals/

Send

Params

Authorization

Headers (11)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Beautify

```
1 {
2   "name": "World Medical",
3   "address": "Changwattana Pakkret",
4   "district": "Pakkret",
5   "province": "Nonthaburi",
6   "postalcode": "10110",
7   "tel": "02-8369999",
8   "region": "กรุงเทพมหานคร (Bangkok)"
9 }
```

Body

Cookies (1)

Headers (7)

Test Results

403 Forbidden

278 ms

325 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "success": false,
3   "message": "User role user is not authorized to access this route"
4 }
```

Conclusions

User REST API

Password & Salt

JWT Token

Authentication

Cookie

Role Authorization & Route Protection