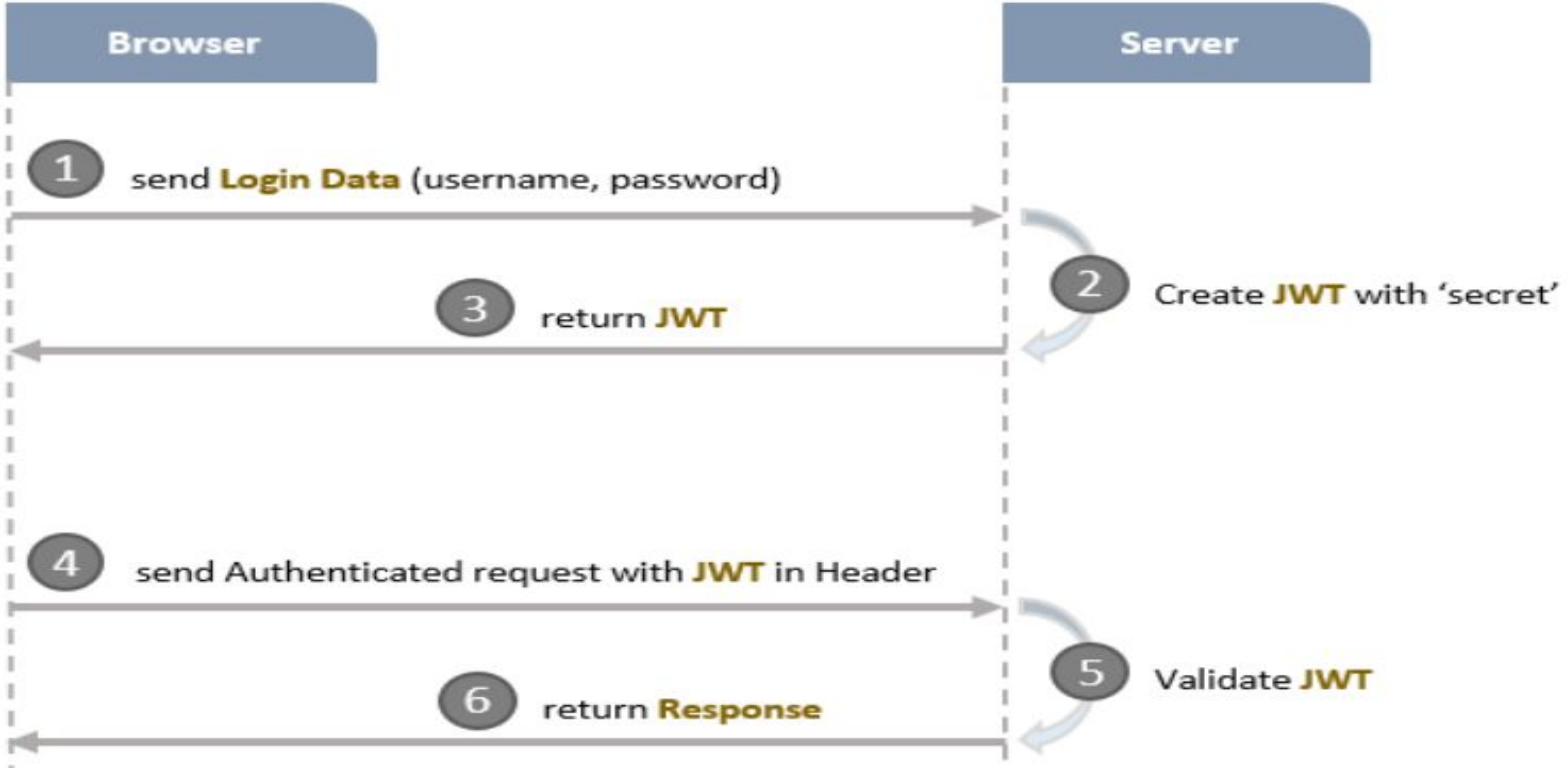
Back-end

Created by :
Riyamol Sara Jayan
Minnu Augustine

Token Based Authentication



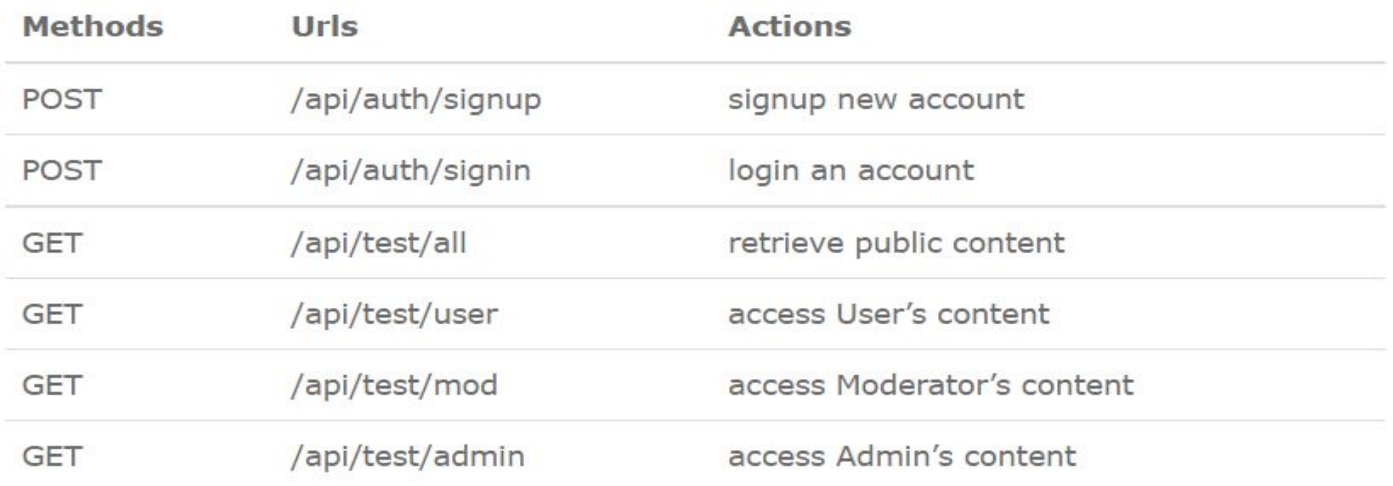
Browser
Server
1
send Login Data (username, password)
2
Create JWT with 'secret'
3
return JWT
4
send Authenticated request with JWT in Header
5
Validate JWT
6
return Response

Node.js & MongoDB User Authentication

We will build a Node.js Express application in that:

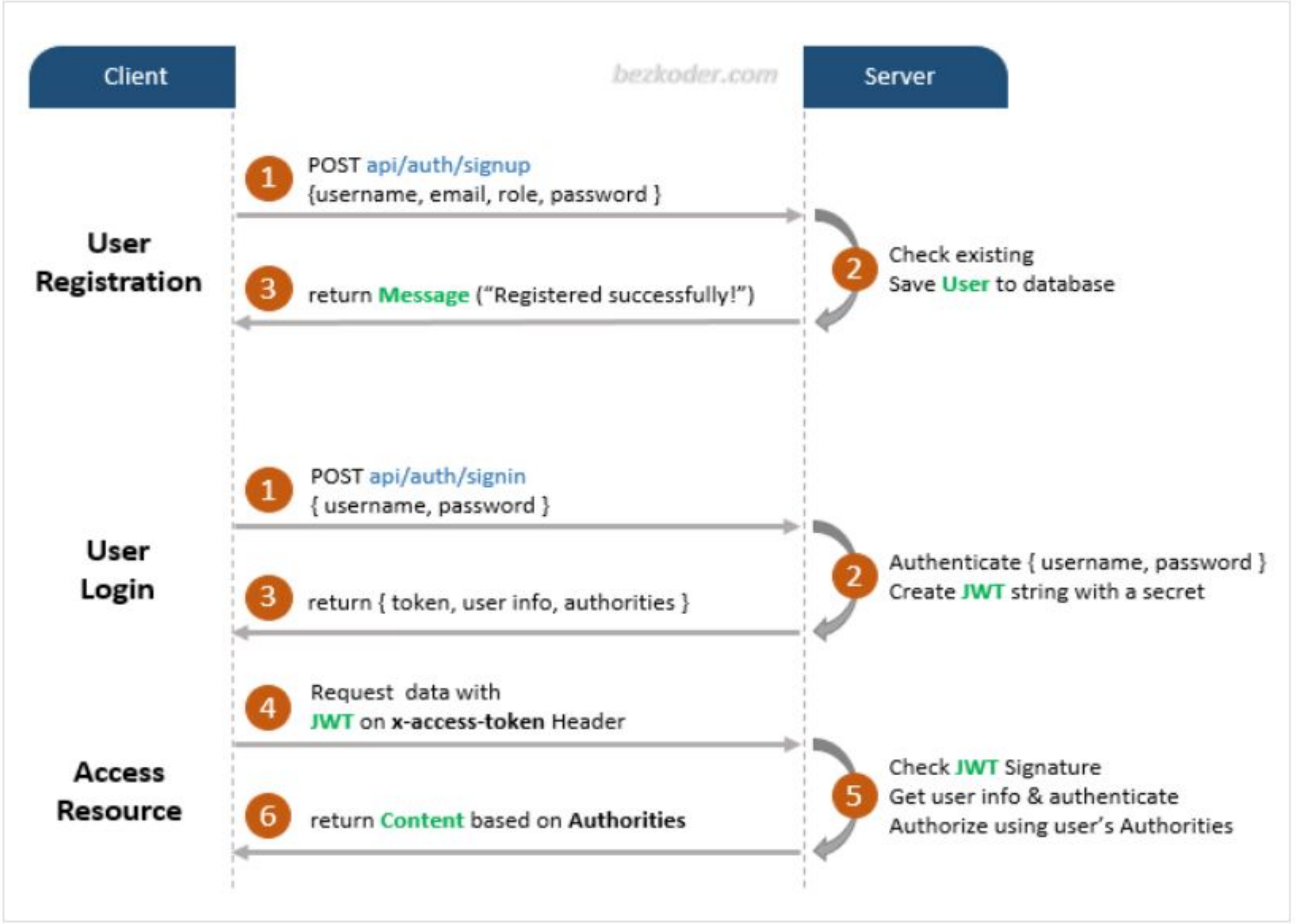
- User can signup new account, or login with username & password.
- By role (admin, moderator, user), the User has access to protected resources or not

These are APIs that we need to provide:



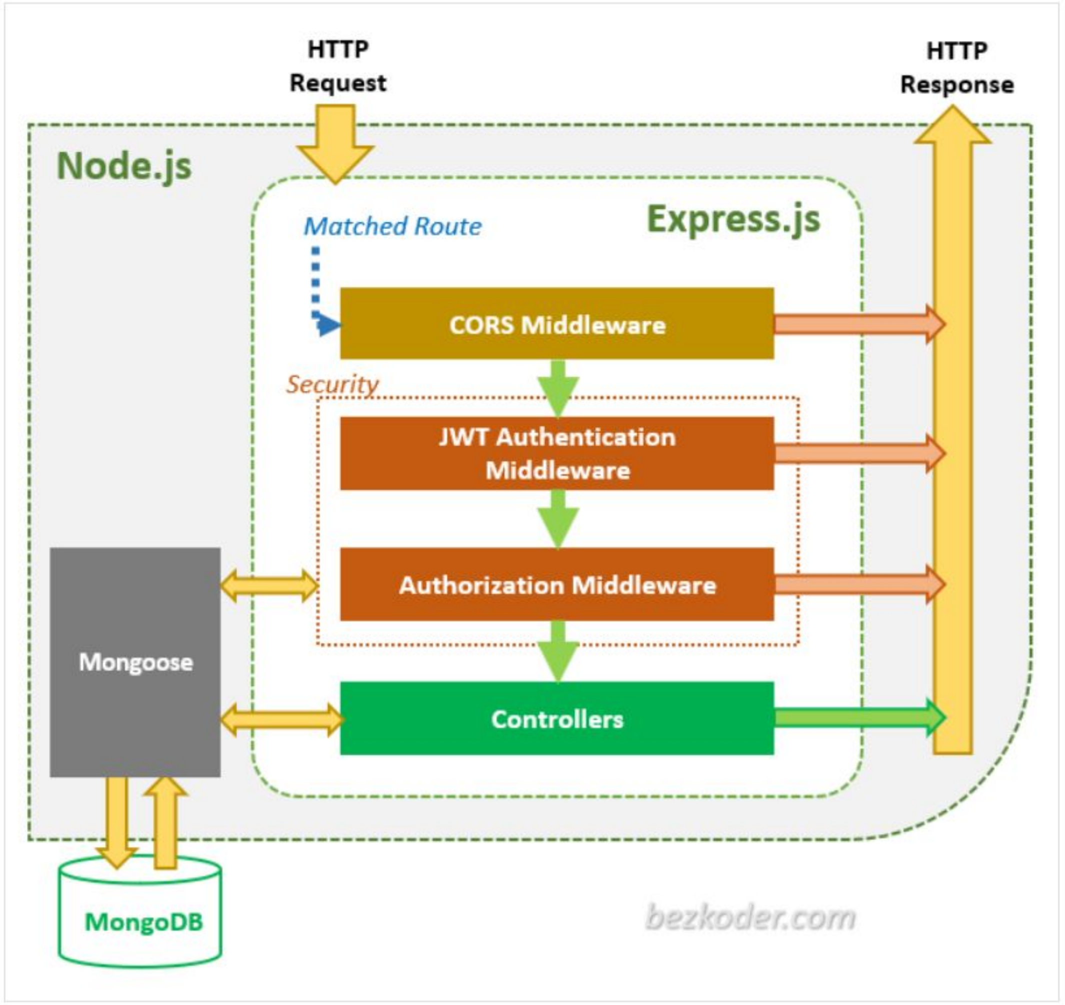
Methods	Urls	Actions
POST	/api/auth/signup	signup new account
POST	/api/auth/signin	login an account
GET	/api/test/all	retrieve public content
GET	/api/test/user	access User’s content
GET	/api/test/mod	access Moderator’s content
GET	/api/test/admin	access Admin’s content

Flow for Signup & Login with JWT Authentication


Client
bezkoder.com
Server
User Registration
1
POST api/auth/signup
{username, email, role, password }
2
Check existing
Save User to database
3
return Message ("Registered successfully!")
User Login
1
POST api/auth/signin
{ username, password }
2
Authenticate { username, password }
Create JWT string with a secret
3
return { token, user info, authorities }
Access Resource
4
Request data with
JWT on x-access-token Header
5
Check JWT Signature
Get user info & authenticate
Authorize using user's Authorities
6
return Content based on Authorities

Node.js Express Architecture with Authentication & Authorization

Here is an overview of our Node.js Express App:



An overview of Express app

Via *Express* routes, **HTTP request** that matches a route will be checked by **CORS Middleware** before coming to **Security** layer.

Security layer includes:

- JWT Authentication Middleware: verify SignUp, verify token
- Authorization Middleware: check User's roles with record in database

An error message will be sent as HTTP response to Client when the middlewares throw any error, .

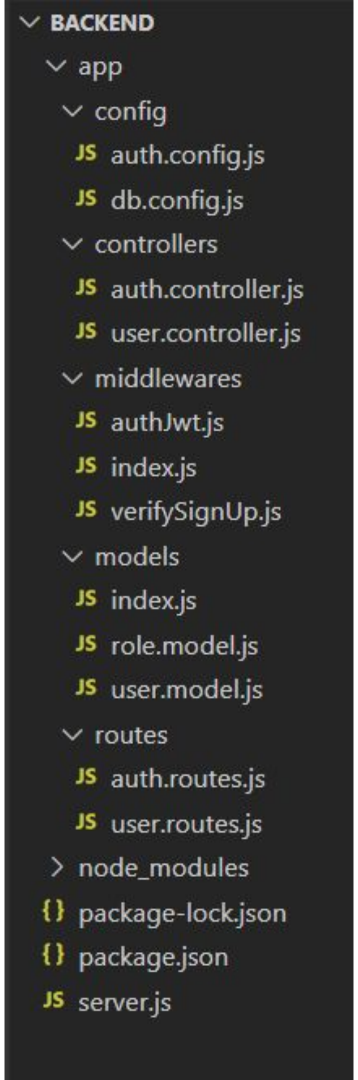
Controllers interact with MongoDB Database via *Mongoose* library and send **HTTP response** (token, user information, data based on roles…) to Client.

Technology

- Express 4.17.1
- bcryptjs 2.4.3
- jsonwebtoken 8.5.1
- mongoose 5.9.1
- MongoDB

Project Structure

This is directory structure for our Node.js Express & MongoDB application:



```
BACKEND
  app
    config
      auth.config.js
      db.config.js
    controllers
      auth.controller.js
      user.controller.js
    middlewares
      authJwt.js
      index.js
      verifySignUp.js
    models
      index.js
      role.model.js
      user.model.js
    routes
      auth.routes.js
      user.routes.js
  node_modules
  package-lock.json
  package.json
  server.js
```

Controller for Authentication

There are 2 main functions for Authentication:

- `signup`: create new User in database (role is **user** if not specifying role)

- `signin`:

- find `username` of the request in database, if it exists
- compare `password` with `password` in database using **bcrypt**, if it is correct
- generate a token using **jsonwebtoken**
- return user information & access Token

Defining Routes

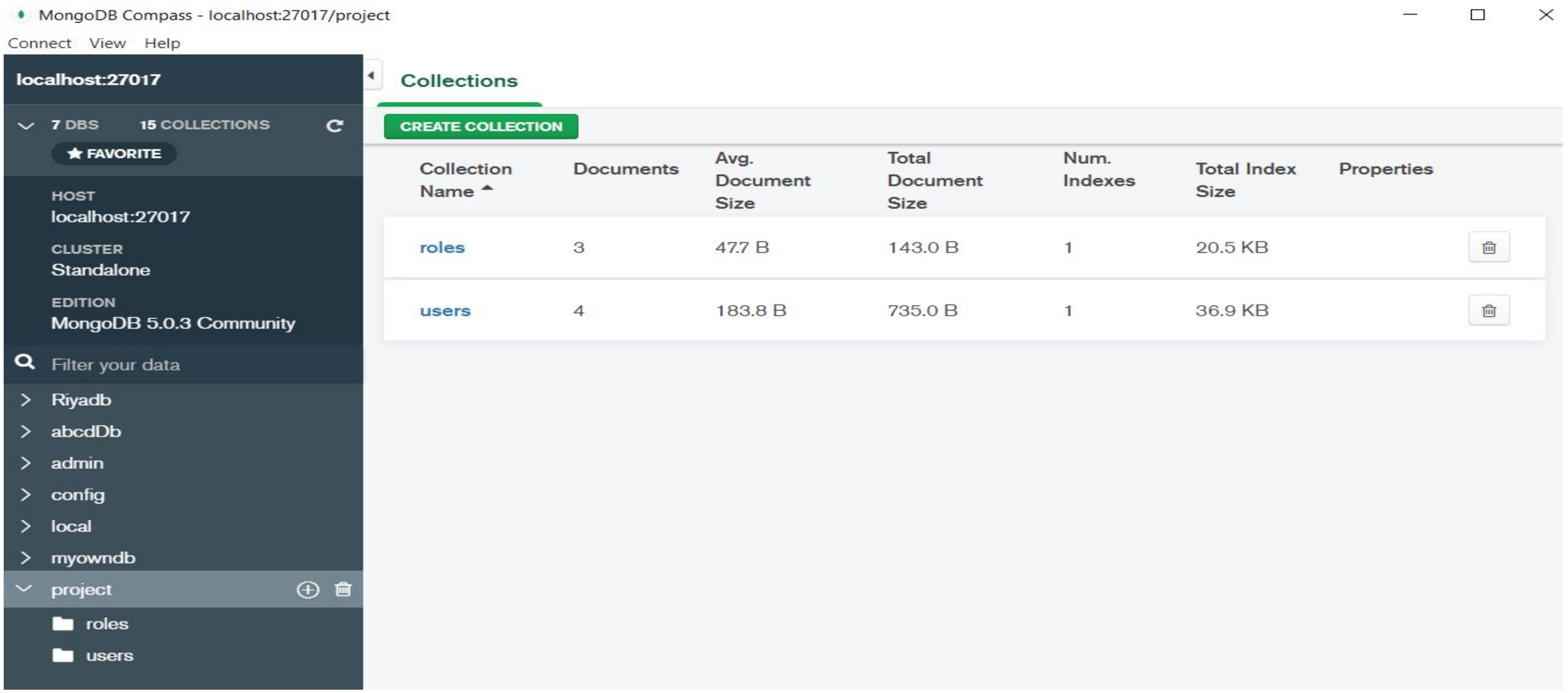
When a client sends request for an endpoint using HTTP request (GET, POST, PUT, DELETE), we need to determine how the server will response by setting up the routes.

We can separate our routes into 2 part: for Authentication and for Authorization (accessing protected resources).

Authentication:

- POST `/api/auth/signup`
- POST `/api/auth/signin`

Mongo Db



MongoDB Compass - localhost:27017/project

Connect View Help

localhost:27017

7 DBS 15 COLLECTIONS

FAVORITE

HOST
localhost:27017

CLUSTER
Standalone

EDITION
MongoDB 5.0.3 Community

Filter your data

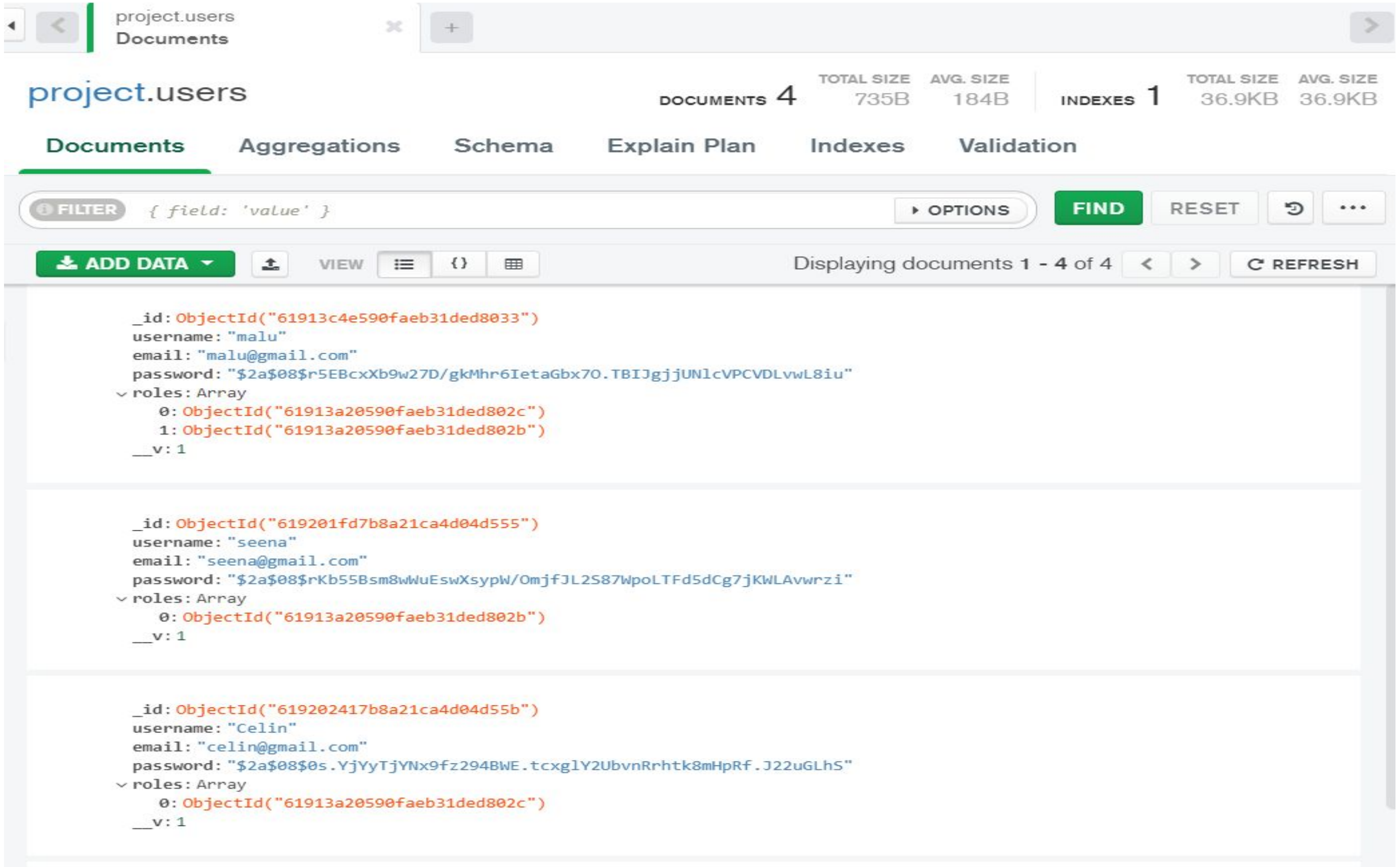
- Riyadb
- abcdDb
- admin
- config
- local
- myowndb
- project
  - roles
  - users

Collections

CREATE COLLECTION

Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
roles	3	47.7 B	143.0 B	1	20.5 KB	
users	4	183.8 B	735.0 B	1	36.9 KB	

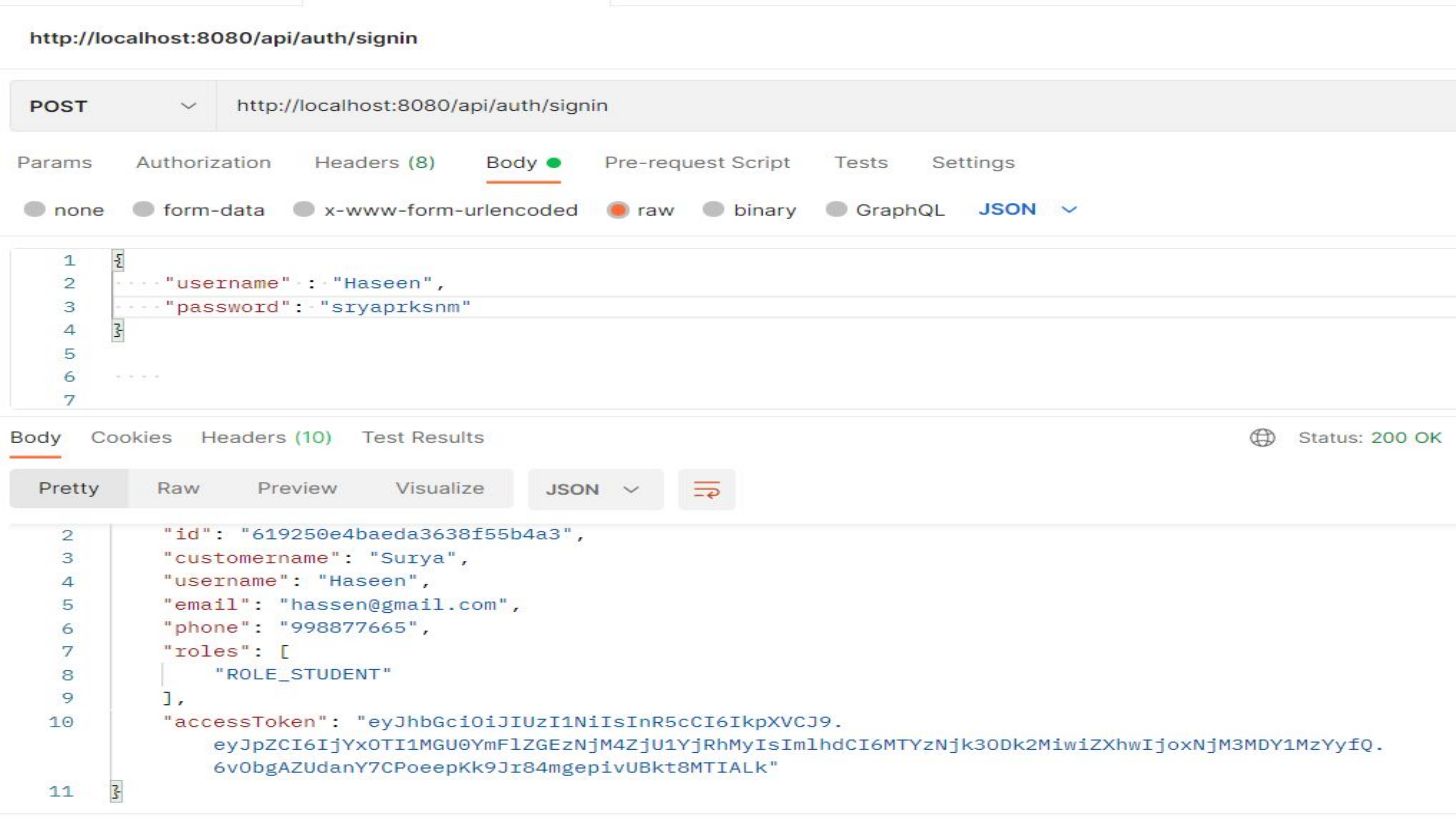
Users collection



```
  _id: ObjectId("61913c4e590faeb31ded8033")
  username: "malu"
  email: "malu@gmail.com"
  password: "$2a$08$r5EBcxXb9w27D/gkMhr6IetaGbx7O.TBIJgjjUNlcVPCVDLvwL8iu"
⌄ roles: Array
    0: ObjectId("61913a20590faeb31ded802c")
    1: ObjectId("61913a20590faeb31ded802b")
  __v: 1
```

```
  _id: ObjectId("619201fd7b8a21ca4d04d555")
  username: "seena"
  email: "seena@gmail.com"
  password: "$2a$08$rKb55Bsm8wWuEswXsypW/OmjfJL2S87WpoLTFd5dCg7jKWLAvwrzi"
⌄ roles: Array
    0: ObjectId("61913a20590faeb31ded802b")
  __v: 1
```

```
  _id: ObjectId("619202417b8a21ca4d04d55b")
  username: "Celin"
  email: "celin@gmail.com"
  password: "$2a$08$0s.YjYyTjYNx9fz294BWE.tcxglY2UbvnRrhtk8mHpRf.J22uGLhS"
⌄ roles: Array
    0: ObjectId("61913a20590faeb31ded802c")
  __v: 1
```



http://localhost:8080/api/auth/signin

POST http://localhost:8080/api/auth/signin

Params | Authorization | Headers (8) | Body ● | Pre-request Script | Tests | Settings

none | form-data | x-www-form-urlencoded | ● raw | binary | GraphQL | JSON

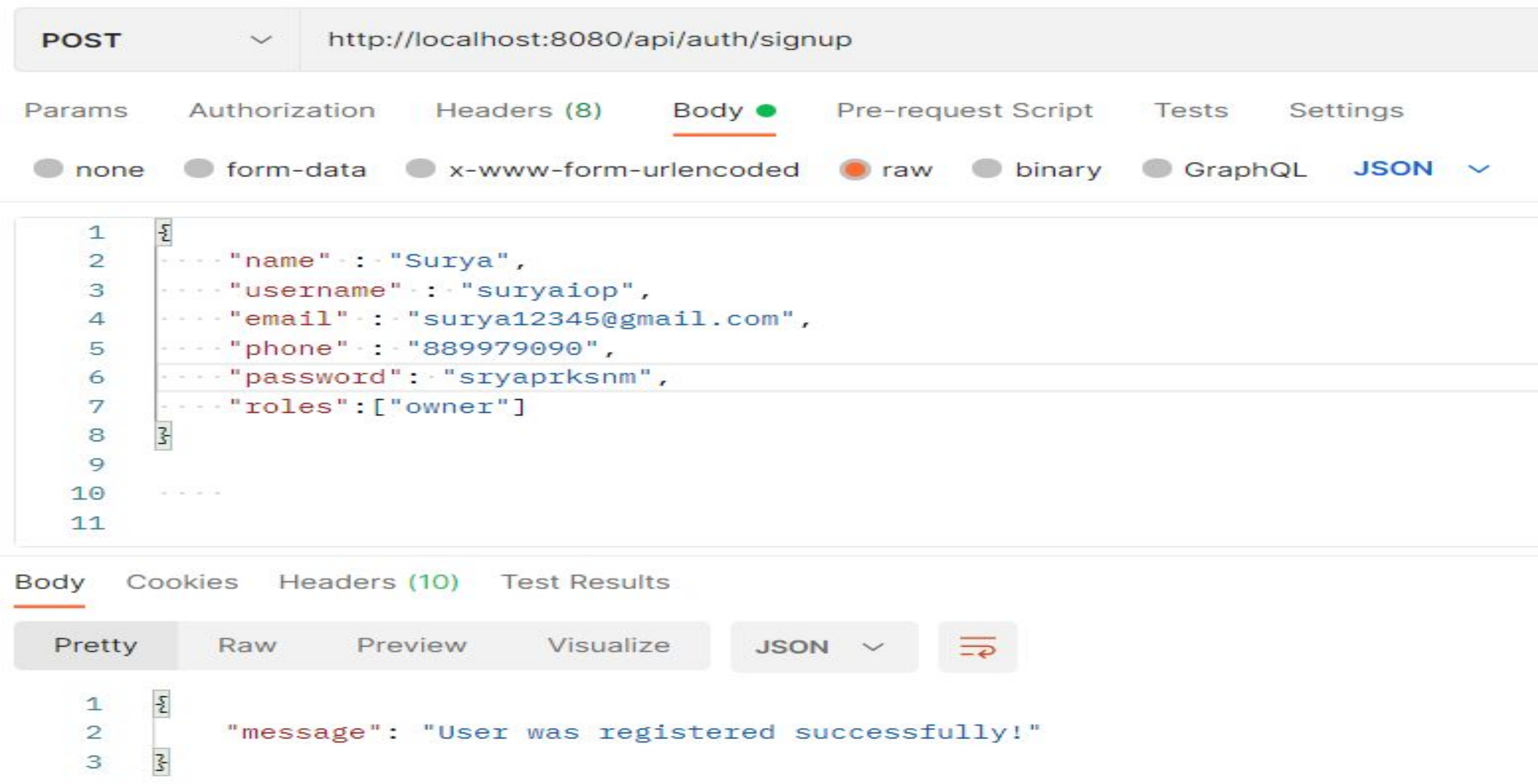
```
{
    "username" : "Haseen",
    "password": "sryaprksnm"
}


    
```

Body | Cookies | Headers (10) | Test Results — Status: 200 OK

Pretty | Raw | Preview | Visualize | JSON

```
    "id": "619250e4baeda3638f55b4a3",
    "customername": "Surya",
    "username": "Haseen",
    "email": "hassen@gmail.com",
    "phone": "998877665",
    "roles": [
        "ROLE_STUDENT"
    ],
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJpZCI6IjYxOTI1MGU0YmFlZGEzNjM4ZjU1YjRhMyIsImlhdCI6MTYzNjk3ODk2MiwiZXhwIjoxNjM3MDY1MzYyfQ.
        6vObgAZUdanY7CPoeepKk9Jr84mgepivUBkt8MTIALk"
}
```

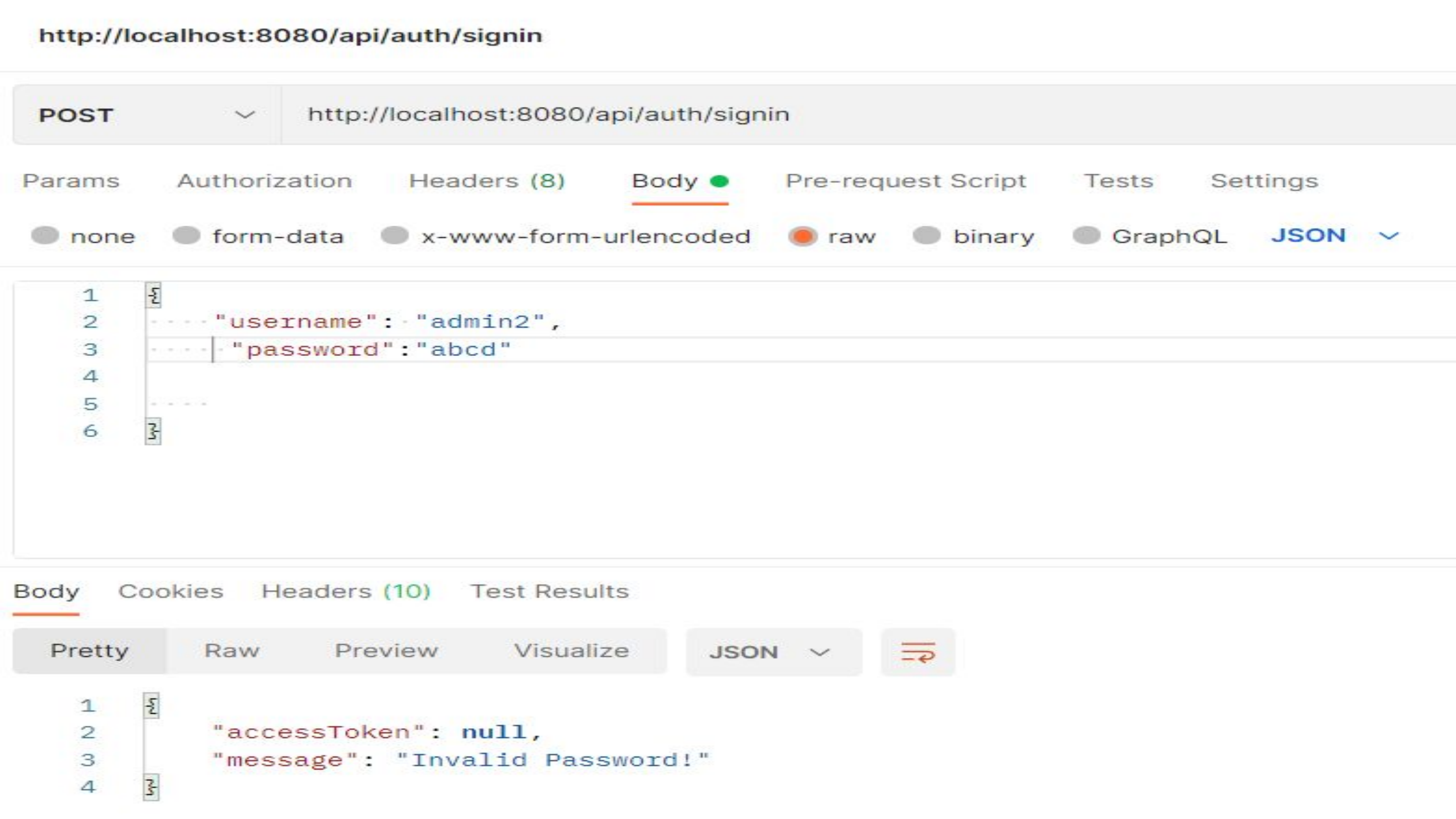


POST http://localhost:8080/api/auth/signup

```
{
    "name" : "Surya",
    "username" : "suryaiop",
    "email" : "surya12345@gmail.com",
    "phone" : "889979090",
    "password": "sryaprksnm",
    "roles":["owner"]
}


    
```

```
{
    "message": "User was registered successfully!"
}
```



http://localhost:8080/api/auth/signin

POST http://localhost:8080/api/auth/signin

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

```
{
    "username": "admin2",
    "password":"abcd"

    
}
```

Body Cookies Headers (10) Test Results

Pretty Raw Preview Visualize JSON

```
{
    "accessToken": null,
    "message": "Invalid Password!"
}
```