Week 9 exercises

Authentication

Next project is to create an application that has a functional authentication using javascript web tokens (jwt). This week we will focus only on the server-side, and in the next week we will create the front end for this authentication.

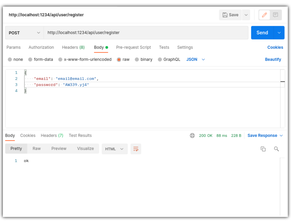
Requirements and Scoring

1. Registration

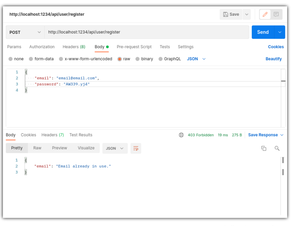
Create an web application using express generator, and connect it to MongoDB. Remember that CodeGrade has the mongoDB running in **mongodb://localhost:27017/testdb**. You may use view engine if you want to. Then create an POST route '/api/user/register' in which the user can register. Create a mongoose model for "Users" schema. This schema should have at least email and password attributes. The password should be crypted to bcrypt-format. Recommended library for doing this is bcrypt.js

Also, it should not be possible to create two users with same email. If user tries to make an account for an email that is already used, it should return status 403.

example requests for registration done with postman.



Duplicate user registration:

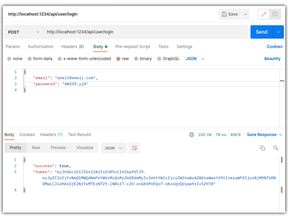


Example user in the database. Remember that CodeGrade has MongoDB running in **mongodb://localhost:27017/testdb** Otherwise it will not find the documents and send error message "*timed out for finding documents in collection: users*"

Note: Do not use escape() or any other sanitizer with the password!

2. Login

Implement Login using JWT token. Create a POST route '/api/user/login', then, with a correct login information, it should return JWT token as a JSON object. Be careful with the sanitizers! If you use some sanitizers, such as escape(), remember that it has to be used both in login and register, otherwise they won't match. Here is an example login:



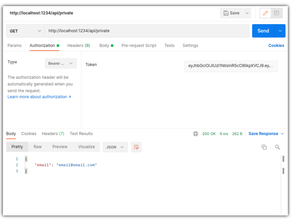
The payload of the token should include email of the user. You can use jsonwebtoken to create the token.

Note: Let requires a secret or private key. Set it as an environment variable named SECRET. you can then use it in your application calling **process.env.SECRET**.

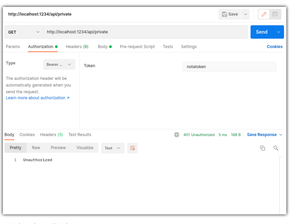
3. Route that requires authentication

Now that we have a token that we can use for authentication, let's create a route that can be used only if the user has logged in. Create a GET route '/api/private'. This route can only be accessed if the token is valid. If the authentication is valid, the server should send the email of the user as json. If the authentication failed, it should response with status 401. For this authentication, passport.js and more specifically passport-jwt is recommended.

Example request:



Example of unauthorized access:



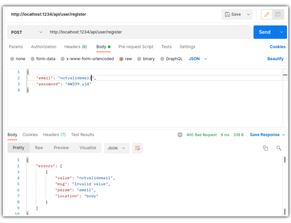
4. Registration validation

Now that the login and registration are implemented, now they have to be validated. Modify the registration so that actually only accepts valid email formats, and the password should be strong. Strong password should fulfill the following requirements.

- minimum length 8
- at least one lowercase letter
- at least one uppercase letter
- at least one number
- at least one symbol +`!@#$%^&*()-_=~{}[]:";'<>,.?/

You may use 3rd party validator package(s), such as express-validator, to achieve these requirements. If the registration is not valid, it should send an error status 400.

Example of failing registration. Note that the body of the response can be any text or json you like, but it has to have the error status 400.

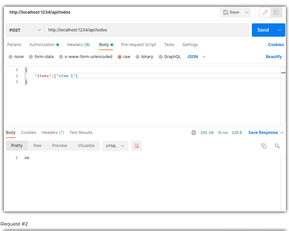


5. Todos

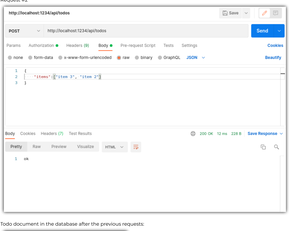
Authenticated user should be able to create a todo list, and add items to existing list. Make route POST '/api/todos' and Model Todo. Todo -model should have fields "user" and "items". Items is a array of strings. Authenticated user should be able to send a list of todos in JSON to '/api/todos'. the format of the body is: {items: ["]}. If there is already a array of items for this user, it should then instead append these items to the existing array.

Two request made by the same user:

Request #1



Request #2



Todo document in the database after the previous requests:

Useful documents and links

- bcrypt
- express-validator

Last modified: Monday, 9 January 2023, 10:19 AM