Backend 4.2_CW Exercises

understanding

Up until here, we have completed authentication concepts. This is it. You have a secret between client and server. If both know the secret, they can talk. Else, no talka talkie.

But do you see the glaring problem with this? If one client loses the secret, everyone's security is compromised. This is where cryptography comes to rescue. It can generate different token for different people and then validate the token too. We will see how that works soon, but first let's look at how to implement login functionality on backend.

ex01: login route

challenge

- For now, use a simple array of username/password.
- On your `/login` route, take username and password from body and check if the credentials are correct using `array.find`.
    - If yes, then send them a token - `{ token: "abcdefghi"}` & message that the user is found.
    - If not, then send them a 401 response
- Test this from the frontend by making a Postman call.

solution

https://replit.com/@tanaypratap/BE42CW-ex01

```
app.use(express.json())

const users = [
  { username: 'user1', password: 'bhokaal' },
  { username: 'user2', password: 'eehuinabaat' },
]

app.post('/login', (req, res) => {
  const { username, password } = req.body

  const user = users.find(
    (user) => user.username === username && user.password === password,
  )
```

```
  if (user) {
    const token = 'abcdefghi'
    res.json({ message: 'User Found', token })
  } else {
    res.status(401).json({ message: 'Authentication failed' })
  }
})
```

COPY

understanding

- This is what you'll do for each user. Just that instead of looking up an array you would do it from the database.
- Pay attention that we use login, and username-password pair only once. Later, we use only the token to authenticate the user.
    - This way you don't send credentials in every request.
    - Also, it saves time for each request as you don't need to do an extra query to DB per request.
- Another thing to notice is that there's no logout. Since there's no state saved, no token saved on server, server can't log you out. Expiry of your token is logout.
    - Therefore, JWT works best on HTTPS. If you're not using HTTPs on your site in 2021 then you have bigger issues.

ex02: sign up route

Challenge:

- Create a sign up route `/signup` that allows users to register with a username and password.
- Store the registered users in an array. Ensure that usernames are unique.
- Respond with a success message if registration is successful with a token or an error message if the username is already taken.

solution

https://replit.com/@tanaypratap/BE42CW-ex02

```
const users = []

app.post('/signup', (req, res) => {
  const { username, password } = req.body

  // Check if the username is already taken
  const userExists = users.some((user) => user.username === username)

  if (userExists) {
    res.status(400).json({ message: 'Username already taken' })
```

```
  } else {
    // Add user to user array
    const token = 'abcdefghi'
    res.status(201).json({ message: 'Registration successful', token })
  }
})
```

COPY

ex02.1: sign up route with bcrypted password

challenge

- If the username is not taken, hash the provided password securely using bcrypt with a salt.
  - Import the `bcrypt` library at the beginning of your file.
  - Generate a salt using `bcrypt.genSalt()` with a cost factor of 10.
  - Hash the user's password using `bcrypt.hash()` with the generated salt.
  - Store the hashed password in the `users` array along with the username.

understanding

Bcrypting is required for password security. It hashes passwords with unique salts, making them resistant to attacks, such as brute force and rainbow tables. The cost factor adds an extra layer of security, and bcrypt can adapt to evolving hardware for long-term protection.

1. Generate a Salt: Use `bcrypt.genSalt(10)` to create a random "salt" with a cost factor of 10. The salt enhances security by making each password hash unique.
2. Hash the Password: Apply `bcrypt.hash(password, salt)` to securely hash the user's password with the generated salt. Store this hashed password in your database.

solution

https://replit.com/@tanaypratap/BE42CW-ex021

```
const bcrypt = require('bcrypt')

app.post('/signup', async (req, res) => {
  const { username, password } = req.body

  // Check if the username is already taken
  const userExists = users.some((user) => user.username === username)

  if (userExists) {
    res.status(400).json({ message: 'Username already taken' })
  } else {
    const salt = await bcrypt.genSalt(10)
```

```
    const hashedPassword = await bcrypt.hash(password, salt)
    users.push({ username, password: hashedPassword })
    console.log(users) // to test
    const token = 'abcdefghi'
    res.status(201).json({ message: 'Registration successful', token })
  }
})
```

COPY

ex02.2: using bcrypt in login route

challenge

Enhance the login route by using `bcrypt` to securely compare passwords.

understanding

During login, the provided plain text password is compared with the stored hashed password using `bcrypt.compare()`.

solution

https://replit.com/@tanaypratap/BE42CW-ex022

```
app.post('/login', async (req, res) => {
  const { username, password } = req.body

  const user = users.find((user) => user.username === username)

  if (!user) {
    return res.status(401).json({ message: 'Authentication failed 1' })
  }

  try {
    const passwordMatch = await bcrypt.compare(password, user.password)

    if (passwordMatch) {
      const token = 'abcdefghi'
      res.json({ message: 'User Found', token })
    } else {
      res.status(401).json({ message: 'Authentication failed 2' })
    }
  } catch (error) {
    console.error(error)
    res.status(500).json({ message: 'Authentication failed 3' })
  }
})
```

COPY

ex03: fetch all orders

challenge

- Create a route `/orders` that returns a list of all orders.
- Make sure to update the users array with orders.
- Ensure that only authenticated users can access this route.
    - Check for `abcdefghi` in the `authorization` header. `const authHeader = req.headers.authorization;`
- Respond with a success message & all the users or an error message saying authentication failed.

solution

https://replit.com/@tanaypratap/BE42CW-ex03

```
const authToken = 'abcdefghi'

const users = [
  {
    username: 'user1',
    password: 'password1',
    orders: [
      { id: 1, product: 'Product A', quantity: 2 },
      { id: 2, product: 'Product B', quantity: 1 },
    ],
  },
  {
    username: 'admin1',
    password: 'adminpassword1',
    orders: [{ id: 3, product: 'Product C', quantity: 3 }],
  },
]

app.get('/:userId/orders', authVerify, (req, res) => {
  if (req.user.userId === req.params.userId) res.json(orders)
})
```