NAMASTE NODE.JS SEASON 2

Episode - 9
Hand Written Notes

-By Shanmuga Priya

www.linkedin.com/in/shanmuga-priya-e-tech2

Episode - 9

Encrypting Password

1) What are all the steps has to be done before sending data to DB when Signing up ?

There are 2 steps involved before sending the signup data to DB

*) Validating the input Data

*) Encrypting the Password.

2) How to Encrypt the Password?

-> we can encrypt the password using "bcrypt" library.

-> using bcrypt we can hash the password and also can validate the password.

"npm i bcrypt"

-> we use "hash" function to hash the password. This has fn returns a promise.

syntax / eg:

```
bcrypt.hash( password, salt )
```

-> Salt is the random piece of data that gets added to the password to make it unbreakable and unique.

-> Higher the No. of. Salt longer the time it takes, if it is less it is easy to break. So, the standard salt value is 10.

-> Once the Password is encrypted we cannot decrypt it.

-> we can store the hashed password in the DB.

eg:

```
const bcrypt = require ("bcrypt")
app.post ("/signup", async(req, res) => {
```

```
try{  //validating data
      validateSignup(req)

      // Encrypting password

      const {firstName, lastName, email, Password} = req.body

      const PasswordHash = await bcrypt.hash(password, 10)

     //storing the hashed password to DB.
      const   user = new User({
                 firstName, lastName, email,
                 Password: PasswordHash
            })

      await User.save()
      res.send("user added successfully")
   }catch(err){
      res.status(400).send("Error" + err.message)
   }
})
```

3) <u>How to compare the hashed password with the incoming password while logging in?</u>

→ there is a fn called "compare" which is used to compare the incoming password with the hashed password. It returns a promise.

<u>syntax:</u>

```
bcrypt.compare(password, hashedPassword)
```

<u>eg:</u>

```
app.post("/login", async(req, res) => {
 try{ //validate incoming data
        validateLogin(req)

     // get the document based on ~~pass~~ email Id from DB for password
        comparison
const { email, password } = req.body

const user = await User.findOne ({ email : email })
  if (!user) {
        throw new Error ("Invalid credentials")
    }
//comparing the password

  const isPasswordValid = await bcrypt.compare( password, user.password)
  if ( isPasswordValid) {
          res. send ("user logged in successfully")
  } else {
        throw new Error ("Password not correct")
  }
} catch(err) {
      res. status (400) send ( "Error" + err. message)
  }
})
```

———————— × ————————