Encrypting Passwords.

write now we are storing our password in plain text - (we can read it) but our password should not be stored in plain text in our database.

There are lot of security issue if we store password like this - because this password is readable

☆ Password should be stored in hash format.
(encrypted) and nobody should be see it!

The first thing that should be happen when some one post the data is validation (throw error.)
once data is validated, then encrypt the password the store the user into database.

① validate
↓
② encrypt the password.
↓
② Storing the data into database.

① we can validate the data in 2 api also but it's not a good practic, we will create a helper function for this

Standard good practice =
create helper function.

(utils / helpers) ← folder.
↳ and create file validation.js
in this file we will do all type of validation

validation.js

```
const ValateSignup data = (req) => {
distructuring -> const {firstName, lastName, Email, password} = req.body

if(!firstName || !lastName){
  throw new error ("Enter Name");
  }
else if (firstName.length<4 || firstName.length>50){
  throw new error ("name Should be 4 to 50
                    character")
}
```

this all things we have done at schema level

```
else if (!validator.isEmail(emailId) {
  throw new Error ("Email is not valid")
  };

else if (!validator.isStrongPassword(password) {
  throw new error ("please enter Strong password");
  }

};
module.exports = {validateSignupdata}
```

and we will use this

```
// Validation the data
   ValidateSignupdata(req);
```

always do this in try catch...

No we will encrypt the password.
for this we will use ~~bycrypt~~ bcrypt.

NPM i bcrypt ⇐ to decrypt the password.

import. and we will create hash. function
```
const {password} = req.body;
const passwordHash = await bcrypt.hash(password,
                                       10);
```

The more the encryption level will be the more tough it will be (Salt round)
good number or basic number is 10

How the password get encrypted.

I Pravin@123, (we need a Salt) it could be
random string = 1AB2234@# @2SA
(it will generate the password)

now ~~son~~

now we will save the password.

till now we are doing

const user = ~~new~~ new user ({req.body})
↑
this is not a good pratic.

we have to do it like this.

```
const user = new user({
   firstName, LastName, emailId, Password : Password}
                                             hash
```
↑
only this field ~~will be~~ should be allowed.

whole code

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

      validateSignup(req);

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

const Hashpassword = await bcrypt.hash
                         (password, 10);

const user = new user ({ firstName, lastName,
           emailId, Password: Hashpassword})
```

Let's create login API
it ~~with~~ will validate.

How we will check if the email Id and password is correct or not

first we will check if ~~it~~ email id is present in DB or not
if present then we will find the data by email and we will use a function of bcrypt. compare (Password, usrpassword)
and then we will check if the password is same or not. If same then we will send login success message.

Process.

Database level validation
↓
API level validation and sanitization.
↓
Then storing to Database.