⇒ bcrypt:-
- It is a package which allows us to encrypt our password with hashes.

⇒ JWT (JSON web tokens):
- It ~~allows us to~~ based on cryptography.
- It is used for send token to clients.
→ There is three parts when encoded with secret keys.

① Header ② payload & data ③ secret key
encoded data based on it.

⇒ Middleware hook: it is the some kind of method or hook which used to perform on specific requirments.

→ ex: Pre, Post, etc (mongoose Docs)

~~→ use of pre & bcrypt in User Schema~~

User Schema.pre("save", async function(next){
this.password

Page No.
Date

=> Use of pre & bcrypt in schema.

=> We use pre for we want to perfor encrypt the password before they save in DB. So we do it as...

~~User.Sc~~ Here, we don't use arrow func due to the

UserSchema.pre("save", asyncFunction(next) {

~~if(!this.password~~

if(!this.isModified("password")) ~~return~~ next();

this.password = await bcrypt.hash(this.password, 10)

next();

});

isModified property

=> For Validation checking we use the password is correct or not we use it as below.

=> We make isPasswordCorrect name function in the UserSchema.method as like below.

Here, we don't use arrow fun because it not provide context of this keyword

userSchema.methods.isPasswordCorrect = async function(password) {

return await bcrypt.compare(password, this.password)

}

→ userSchema.methods in we add our methods as above we write isPassword...

⇒ JWT :-

- JWT is bearer token.

- means who have this token it would be right. (જેની પાસે ટોકન તે બોરો)

→ .env we define secret key as like below.

ACCESS_TOKEN_SECRET = 0T4XPWZ.....UXPNX

ACCESS_TOKEN_EXPIRY = 1d

REFRESH_TOKEN_SECRET = CJK1OU.....PTXNY

REFRESH_TOKEN_EXPIRY = 10d

→ So, as above we write method fun in method for bcrypt same we writ for our jwt token.

Page No. ______

Date ______

userSchema.methods.generateAccessToken = function () {

return jwt.sign (

{

id : this._id,

email : this.email,

userName : this.userName

fullName : this.fullname

},

process.env.ACCESS_TOKEN_SECRET,

{

expiresIn : process.env.ACCESS_TOKEN_EXPIRY

}

)

}

{ as we want we can write prope

=> AS, same way we write for REFRESH TOKEN...

userSchema.methods.generateRefreshToken = function () {

} There is only we pass _id

}

Page No.
Date

9] Q Have to upload file in backend (multer)

⇒ Cloudinary : It's provide storage for our video & files assets..
- We most companies use this third party for its assets and also we used it for our Proj.

⇒ 1] create account on it.

⇒ 2] we use also middware for file uploading is multar.

→ express-fileupload is earlier used but today most popular is multar.

⇒ lets start working.

⇒ npm i cloudinary multer.

⇒ Strategy for file uploading

→ we take file from user and temporary keep in our server with using of multer.

Page No.
Date

→ After, with using cloudinary we take file from our local server and upload in cloudinary.

⇒ why this 2 steps instead of direct uploading in cloudinary ?

→ because, we give first it's production grade and recommended way in industry for attempting the reuploading if any problem occurrence.

⇒ In, [utils] we create utility of cloudinary as name cloudinary.js.

→ In this file we give the path of ~~our~~ local ~~local~~ ~~server~~ ~~file~~ files which are in uploaded the local server.

→ After upload we ~~xx~~ unlink (remove) from our local server for cleaning

⇒ fs module in nodejs:
- it's pre defined module in node js for file operation.

→ unlink :

- when we want do delete file so we unlink it. (It's terminologies for delete file in OS)

→ So after uploading file we unlink from our server. using this fs mod module method.

→ In, file. code.....

```
Import fs2 as cloudinary from 'cloudinary'
Import fs from 'fs'.

cloudinary.config({
  cloud_name: P.e.CLOUDINARY_CLOUD_NAME,
  api_key: P.e.CLOUDNARY_API_KEY,
  api_secret: P.e.CLOUDINARY_API_SECRET.
})
```

⇒ Now is uploading is too easy in cloudinary just use .V.2.uploader.upload method but,

→ we make it some additional to more efficent and for unlink the file from local.

→ So, for that we make func. as...

Page No.
Date

```
const uploadOnCloudinary = async (localFile
Path) => {

try {
    if (!localFilePath) return null
    // File upload on the cloudinary.

    const response = await cloudinary.
    uploader.upload(localFilePath, {
        resource_type: "auto"
    })
    // we also need remove the file
    after uploading
    return response;
} catch (error) {
    fs.unlinkSync(localFilePath)
    return null;
}
}

export {uploadOnCloudinary}
```

→ For any format and JPG, Video etc

→ This is also same property.

→ Remove uploaded file if the upload is failed.

Page No.
Date

=> Multer middleware writing code:

=> when ever we perform file uploading we use multer middleware so we use one middleware for it.

→ In middleWare folder we make file multer.middleware.JS

→ There is three storage memoryStorage, diskStorage but we use diskStorage for prevent to load on memory.

→ code:

```
import multer from "multer"

const storage = multer.diskStorage({

    destination : function (req, file, cb) {
        cb(null, "./public/temp")
    },

    filename : function (req, file, cb) {
        cb(null, file.originalname)
    }
})

export const upload = multer({storage})
```

(learn more about)

10] HTTP crash Course

→ HTTP | HTTPS : Hyper text transfer protcol | Secured.

→ In, HTTP data transfer in plain text & in HTTPS it encrypted.

→ URL | URN | URI :- uniform source Locator | Name | Identifier.
⇒ It's just address that where we intersect.

⇒ Headers :-

→ It is metadata. which keep the data about data.
→ Headers are sent along with req & res.

→ Headers are used for many ways some of are belows.

Caching : recently network req can data in header used for any purpose.

→ authitication : ex this session cookies, bearer token, refresh token, any kind of auths.

→ manage State : for checking is it logged user, guest user, have they perform any action (add to cart, or etc)

Tips : ⇒ the caller X used as Prefix in the header name. X-name, etc.
- it deprecated

⇒ There are no any types of headers but we categorize as below.

→ Request Headers : It's come from the client browsers.

→ Respond Headers : It's come from the servers. there some specific rule of success data. (200 code) not found (404) etc. for standerdize.

→ Representational Headers : ~~encoding~~ / comm encoding / compression
→ it's define based on our used data. ex: Zerodha, razorpay, etc used graph chart, so there data is too much.

Page No.
Date

so, we compressed because there is some limit data for transfer otherwise large issue occurs.

→ Payload Headers : it's just fancy name of our passed data.
→ an example : we passed same of id, name, email,

⇒ There are many types of headers. Security Headers etc.....

⇒ most common Headers :

→ Accept : application/json.
Html/text.
- mostly now used api/json due to api's but we can accept html.

→ User-Agent : It's give data that where is come from: means from, mobile, ios, android, mac, postman... basis of it data we got some times suggestion of it app for downloads. etc.

→ Authorization : mostly used for front-end authentication.
eg: token : Bearer xyVPz.....

→ content-type : when type data are send pdf, img, text etc.

→ cookie :- is object which store data in key-value pairs.

→ cache-control : for controlling the network data.
(eg: in which time it caches this data)

⇒ CORS Headers :-
- Access-control-Allow-origin : from which origin to allow. like as operation.
- Access-control-Allow-credential,
- Access-control-Allow-Method, etc.

⇒ Security Headers:
- Cross-origin-Embedded-Policy
- cross-origin-opener-Policy
- content-security-policy
- X-xss-protection.

Page No.
Date

⇒ HTTP methods:

- Basic set of operations that can be used to interact with server.

- GET : for retrieve resource.
- HEAD : No message body response headers only.
- OPTIONS : what options are available
- TRACE : loopback test
  - It most used for debugging for finding which proxy to pass of connection
- DELETE : delete source
- PUT : replace whole resource.
- PATCH : 11 specific part of source
- POST : for new resource add.

⇒ HTTP Status code:

x-mean 2 digit

- 1xx - Intermational
- 2xx - Succes
- 3xx - Redirect
- 4xx - Client error
- 5xx - Server Error

102 - continue
102 - Processing
200 - OK
201 - created
202 - accepted
307 = temporary redirect
308 - permanent

400 - Bad req
401 - Unauthorized
404 - ~~Payment~~ Not Found ~~Invalid~~
500 - Internal ser. Err
504 - Gateway Time out
402 payment required