Project :

1. First we apply npm init.
2. We create git keep and git ignore for push onto github.
3. Now we create .evn file which store into git ignore so it is not push into git .we just create evn.sample for our reference.
4. Now we create new folder src and we create our required file.
5. Then we apply commond npm I -D nodemon onto main

Terminal.

1. After apply commond we change script of json package

“dev”: “nodemon src/index.js.

1. Now we create Folders into src.

Video 7 :

1. Copy data base link.
2. Put copied link into .evn and .evn.sample files.
3. Go to constant file and export data base name.
4. Install mongoose dotenv express
5. Go to index and write the code upto line 1 to 34.
6. Go to .env file change script.
7. Go to terminal and run file using npm run dev.

Video 8 :

1. First import express into app.js
2. Apply then catch on index.js
3. Install cookie-parser and cors.

Cors: a mechanism by which a front-end client can make requests for resources to an external back-end server.

cookie-parser:  Extracts the cookie data from the HTTP request and converts it into a usable format that can be accessed by the server-side code.

1. Import cookie-parser and cors.
2. Use function is use for middle waves.
3. When user enter data into from then express.json convert into usable data.
4. To encoder url we use express.unlencoder().
5. To wrapper connectioMongoo function we create asynhandler into util fiile . write the code.
6. Now to represent error we create another file into

Util folder.

1. Write the code into apierror and apiresponse.

Video 9 :

1. Make user and video schema.
2. Install mongoose aggerate.
3. Create pluging method.

Pluging: Plugins are a tool for reusing logic in multiple schemas.

1. To secure password we use bcrypt and jsonwebtoken .

Jsonwebtoken: It is code of block which contain header,information,

Some.which use in when user enter any information

That information convert into block of code and go to

Mongoodb.so install bcrypt and jwt.

1. Import them.
2. Now we want to encrypt the password. So better understanding go to mongoose middle ways

And check all function. Here we use pre function

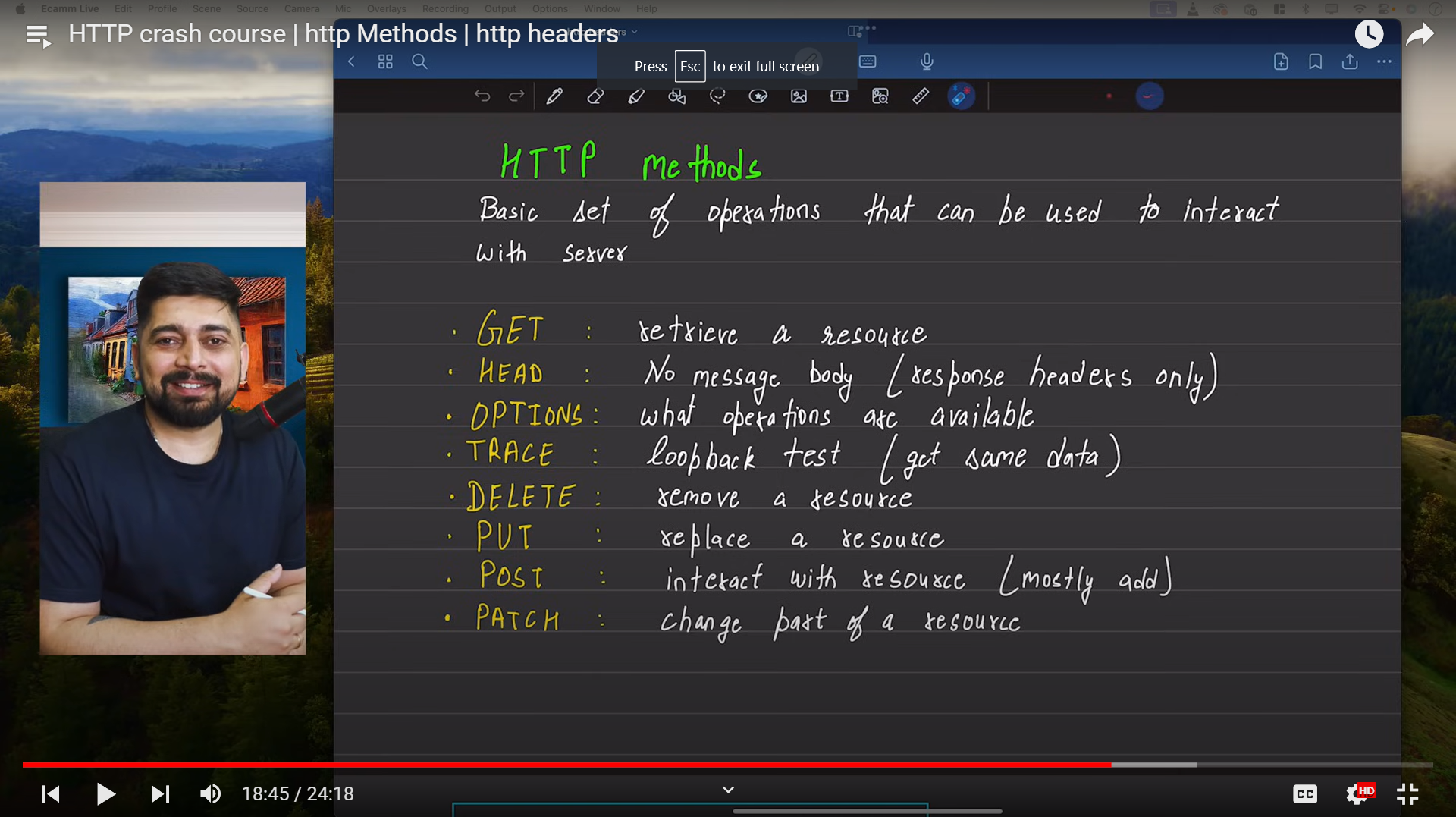
Because when user enter any password when that password go to database just before we encrypt that password.

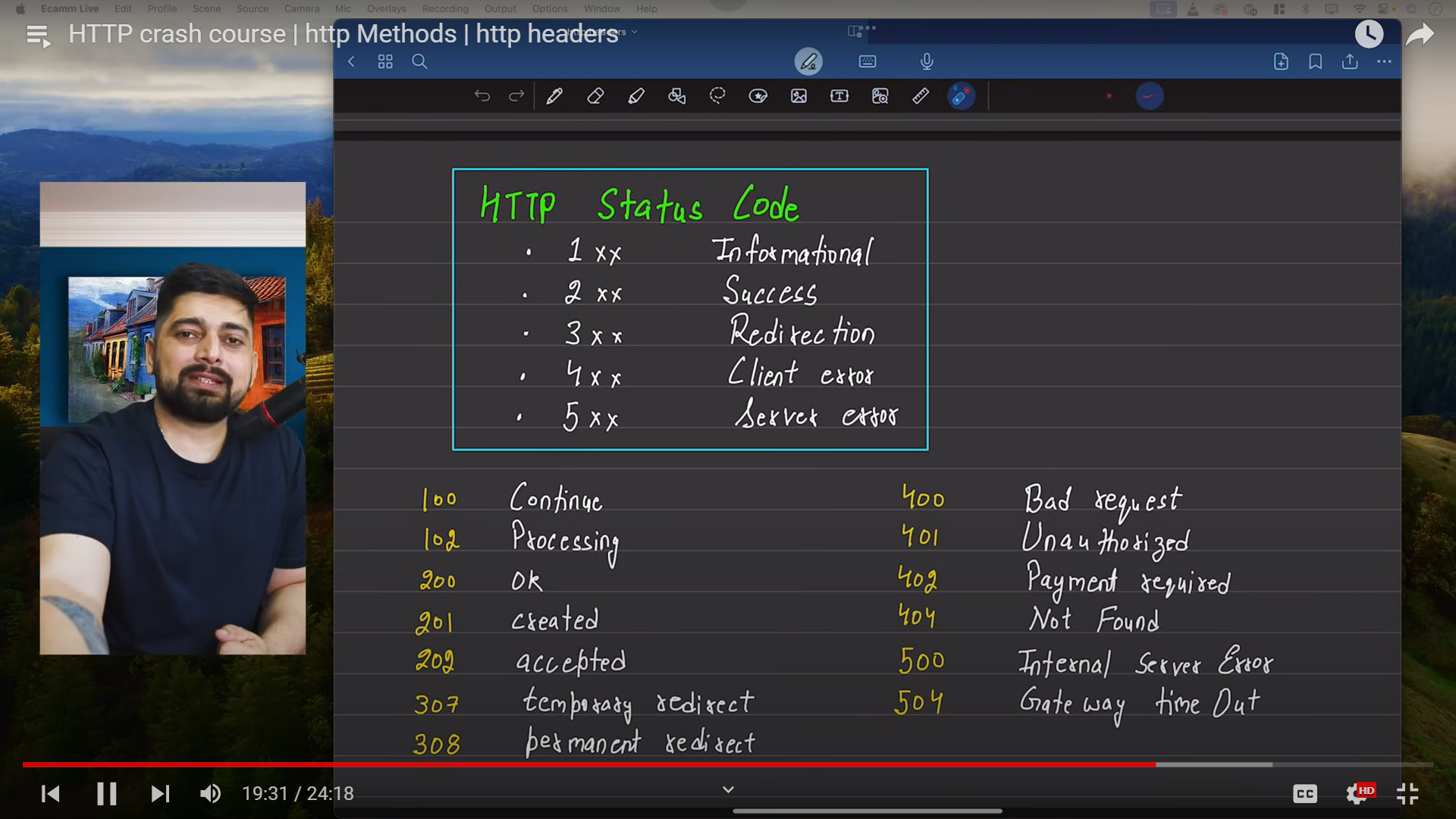
1. After we check password is password is correct or not.
2. Then we write the code of token in env and user.model.js file.
3. When a user logs in my app, a token is generated and saved in a mongoDB collection. This token is supposed to be valid for 100 hours and then expire, so that the user has to generate a new one. This is my schema code :

Video 10:

1. Go to cloudinary website and install extension.

Video 11:





Video 12:

1. Create user.controller. import required thing.
2. Write the code upto 11 line.
3. Create user.routes. import required thing.
4. Write the code.
5. Go to main app.js and

import router from "./routes/user.routes.js";

app.use("/api/v1/users",router)

1. Run the dev.

Video 13:

1. Go to usercontroller and ask request form body and print them.
2. We want to upload file in middle so we import middle

Wares in user.routes file.

1. This is middle wares so we add before resiteruser function.
2. Full feed register from:
3. Checking validation:
4. if([fullname,email,username,password].some((field)=>
5. field?.trim() === "")){
6. throw new Apierror(400,"All Field are required")
7. }

1. Check user already exist or not:
2. const userexisted = User.findOne({
3. $or: [{ username }, { email }]
4. })
5. if(userexisted) {throw new Apierror(409,"User Already existed")}

3. get local path of avtar and cover image:

const avtarloaclpath=req.files?.avtar[0]?.path;

   console.log(avtarloaclpath);

   const coverImageloaclpath=req.files?.coverimage[0]?.path;

   console.log(coverImageloaclpath);

   if (!avtarloaclpath) {

    throw new Apierror(400,"Avtar is required")

   }

1. Upload on couldnary :

  const avtar = await uploadfileoncloud(avtarloaclpath);

  const coverimage = await uploadfileoncloud(coverImageloaclpath);

  if (!avtar) {

    throw new Apierror(400,"Avtar is required")

   }

1. Create data base object:
2. Const user = await User.create({
3. fullname,
4. email,
5. password,
6. avtar:avtar.url,
7. coverimage:cover?.url || "",
8. username : username.toLowerCase()
9. })
10. Hide the password :
11. const createuser = await User.findById(User.\_id).select(
12. "-password -refrenceToken"
13. )
14. Check object is create or not .