**Backend Part:**

In the backend part, I learned how to create a API server and some important npm package like express, mongoose, dotenv, bcrypt, cors, multer, Image-downloader, cookie-perser, jwt(JSON web token)………..

Before install npm package :

1st, in the terminal type npm init -y to create a new file ‘package.json’ in the project directory that contains all the basic information about the project like name, version, dependencies and scripts

Explanation of the important npm package:

Mongoose:

npm I mongoose

It is a npm package to install mongoose for connecting mongoDB database with express server. By using this npm packge we can connect express to mongoDB database send, get , update, delete data to mongoDB database .

Dotenv:

npm I dotenv

it is a env file that contains all the basic information like port, db\_url, secret\_key etc.

Bcrypt:

npm i bcrypt

it is a npm package to encrypt the password and other thing in a secure way and that is very useful to encrypt password

let express = require("express")

let app = express()

Require the bcrypt and saltRounds and bcryptSecrect in the env file

let bcrypt = require("bcrypt")

const saltRounds = 10;

const bcyprtSecret = process.env.bcrypt\_secret;

app.post("/register", async (req, res) => {

    try {

        let user = await usermodel.findOne({ email: req.body.email })

        if (user) {

            res.send("User already exists")

        }

        else {

            bcrypt.hash(req.body.password, saltRounds, async function (err, hash) {

                // Store hash in your password DB.

                let newUser = usermodel({

Here bcrypt.hash is use for encrypting the password

                    email: req.body.email,

                    password: hash

                })

                await newUser.save()

                res.status(202).json(newUser)

            });

        }

    } catch (error) {

        res.status(404).json(error)

    }

})

app.post("/login", async (req, res) => {

    let user = await usermodel.findOne({ email: req.body.email })

    if (user) {

        const result=  bcrypt.compareSync(req.body.password, user.password)

            // result == true

            if (result) {

Here bcrypt.compareSync() is used to compare the given from password with the database password. If password matched then it return true otherwise it return false

                jwt.sign({

                    name:user.name,

                    email: user.email,

                    id: user.\_id

                }, process.env.jwt\_secret,{expiresIn:"2d"},(err,token)=>{

                    if(err) throw err;

                    res.cookie("token", token,{secure:true,sameSite:"none", }).json(user)

                } )

            }

            else {

                res.send("password is not ok")

            }

    }

    else {

        res.send("not found")

    }

})

<https://www.npmjs.com/package/bcrypt>

go to the link to learn more

cors:

cors(Cross Origin Resource Sharing) is a security mechanism that is implemented in web browser. Suppose we want to use API server for a specific client side then we use cors. And in the cors, we define the origin of the client side and when the specific client give a request to the api then the api only response to that specific client otherwise it block the client request

we use cors in the server side and define the origin of the specific client side and If we do not define the specific origin then any client side can give request to the api.

let express = require("express")

let app = express()

let cors = require("cors")

app.use(cors({

    origin: "https://airbnb-hotel-management-app.netlify.app",

    // origin: "http://localhost:3000",

    credentials:true

}))

Here we define a specific origin of the client side that only make request to the api. Otherwise it will block the client request.

Before uploading image

Make sure the folder where the images are downloaded is declared static of express in the app.use ()

app.use("/upload", express.static(\_\_dirname + "/upload/"))

Multer:

npm i multer

We use multer to upload the image/file in the local folder from our pc or it helps us to upload file in the Node.js application.

multer.diskStorage is used to set up a disk storage engine for file uploads in the node.js application specifically where the uploaded files should be saved and how their filenames shouldbe generated

How we can use multer:

let express = require("express")

let app = express()

Require the multer

let multer = require("multer")

// storage file locally in the uploads folder

const storage = multer.diskStorage({

    destination: function (req, photo, cb) {

        cb(null, 'upload/')

    },

    filename: function (req, photo, cb) {

        const normalizedFileName = Date.now() + '-' + photo.originalname

        cb(null, normalizedFileName);

    }

});

// Initialize multer middleware with the storage engine

Upload single file or photo in the destination folder

const upload = multer({ storage: storage });

app.post("/upload", upload.single("photo"), (req, res) => {

    let name = req.file.filename

    if(name)

    {

        res.json(name)

    }

    else{

        res.json(null)

    }

})

Learn more to go to the link

<https://www.npmjs.com/package/multer?activeTab=readme>

Image-downloader:

npm i image-downloader

we use this npm package to download image in the specific folder from a given link.

How we can use this package:

let express = require("express")

let app = express()

Require the image-downloader package

let imageDownloader = require("image-downloader")

// download the image using the link

When we get a link from the give form then it download the image from the link and stored it to upload folder

app.post("/uploadByLink", async (req, res) => {

    let { link } = req.body

    let newName = "photo" + Date.now() + ".jpg"

    await imageDownloader.image({

        url:link ,

        dest: \_\_dirname + "/upload/" + newName

    })

    res.json(newName)

})

Learn more to go to the link

<https://www.npmjs.com/package/image-downloader>

cookie-parser

we use cookie-parser to set the cookie in the browser and also get the cookie from the browser.

How to use this package:

let express = require("express")

let app = express()

let cookieParser = require("cookie-parser")

// this is how we can set the cookie

  res.cookie("keyName", value).json(user)

// how to get cookie

let value  = req.cookies.keyname

If we want to set cookie in the server side then we must follow the following steps

1st, in the client side in the app.js make sure axios.defaults.withCredentials=true ,

axios.defaults.baseURL = "https://airbnbhotels.up.railway.app"

axios.defaults.withCredentials=true

2nd , in the server side make sure the cors origin is the specific client url and

credentials:true

app.use(cors({

    origin: "https://airbnb-hotel-management-app.netlify.app",

    // origin: "http://localhost:3000",

    credentials:true

}))

3rd , if we want to store cookie in the client side from the server then we cannot make it happen and it will be very risky and security issue. So we store token in the server side so that no one can access the token easily and it is very much safety

How we can make it happen:

  res.cookie("token", token,{secure:true,sameSite:"none", }).json(user)

here to store the cookie in the server side we must declare the cookie to be

secure: true

samesite:none

if we can do that then we can store the cookie in the server side otherwise it will give error

Jwt(Json web Token):

Jwt is used to create a information as token and again we can get the information from the cookie

To create the token we use jwt.sign(payload, jwt\_secrect,{other info},(err,token)=>{

That’s how token is generated

}  
)

How we can use it:

let express = require("express")

let app = express()

let jwt = require("jsonwebtoken")

//creating jwt or token

jwt.sign({ foo: 'bar' }, privateKey, { algorithm: 'RS256' },

function(err, token) {

console.log(token);

});

How to get the token

**jwt.verify(token, secretOrPublicKey, [options, callback])**

jwt.verify(token, 'shhhhh', function(err, decoded) {

console.log(decoded.foo) // bar

});

For more information go to the link,

<https://www.npmjs.com/package/jsonwebtoken>