NEXT JS 13.4.19

Next -> 13.0

DATE: PAGE: 1

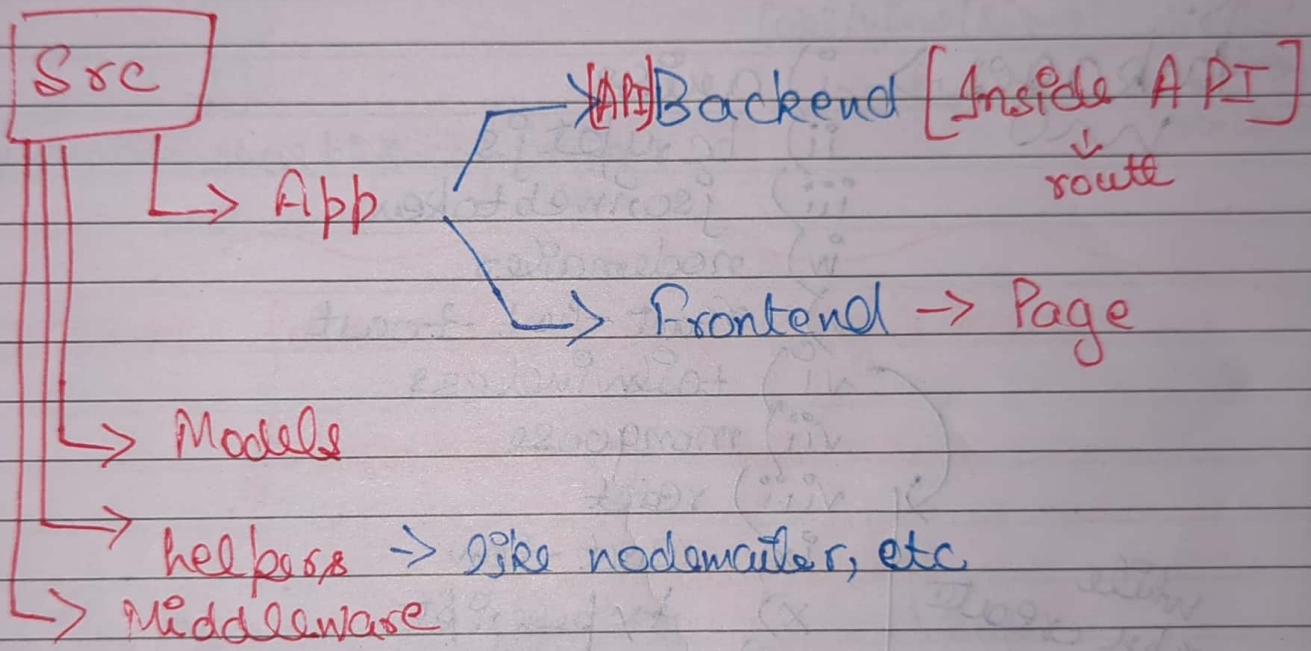
5/9/23

Next JS file structure (Hitesh Chaudhary)

↳ Edge Run framework

Tech Stack-> nextjs
mongoDB
nodemailer
moongose

Typescript javascript
ts -> js
[ts -> jsx (components)]



Command to start Project

↳ npx create-next-app@latest .

Making Authentication & profile page

Good Write

8/9/23

App -> layout.tsx -> Layout that display their children

page.tsx -> Home

To Run -> npm run dev -> painfully slow on first build

Packages ->
i) axios
ii) bcrypt.js
iii) jsonwebtoken
iv) nodemailer
v) react-hot-toast
vi) tailwindcss
vii) mongoose
viii) react
ix) react-dom
x) typescript
xi) eslint

while npx create next@latest

Creating Directories ->

-> src
-> App
helpers
models
dbConfig
new

Good Write

8/9/23

DATE: PAGE: 5

Frontend App — Backend

(Folder) login

(Page.tsx)

Rule: If this is a frontend file you have to call it page.tsx

Path -> localhost:3000/login

api (routes)

users (api/users)

Signup (api/users/signup)

route.ts

home (api/home)

login (api/users/login)

route.ts

☆ -> create folder for every single Route

folder Name = Route name

↳ Inside it page.tsx

-> File Naming is very very important

Edge Run framework -> Do not get always connected DB

Next ↔ (Send some files) request → DB -> every single time

response (receive some files)

6/9/23

DATE: PAGE: 4

src
↳ dbConfig
↳ dbConfig.ts
↳ Code to connect to DB.

Frontend → signup (page.ts)

-> I have grab Data & send it to backend
Axios Request

Latest Next -> Everything is server component
& server component doesnot have
access to frontend.

Now, usually outside the API folder stuff
is client side.

How to make a component/folder client side?

"use client" -> just this in tsx
or jsx file
↳ yes, just do this.

(New) * -> useNavigator -> In React
-> use Router -> next/navigation
*
Prev -> use Router -> next/Router;

Good Write

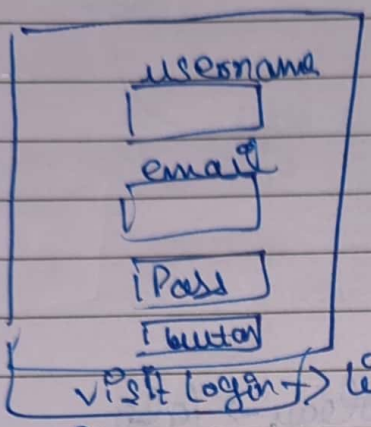
8/9/23

DATE: 5

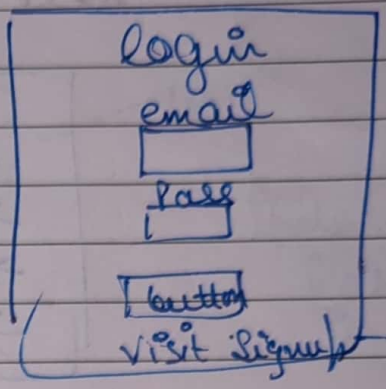
Link -> next/Link

<Link href="/"> Name </Login>

↳ Link



Signup



Login

↳ app

↳ profile

page.tsx (:3000/profile)

Profile

(:3000/profile/abc)

To grab this

In React -> use params

folder -> [id name]

page.tsx

export fun (params)

{ return (

 {params.id name}

); }

abc

But in Next, there is a rule to grab this

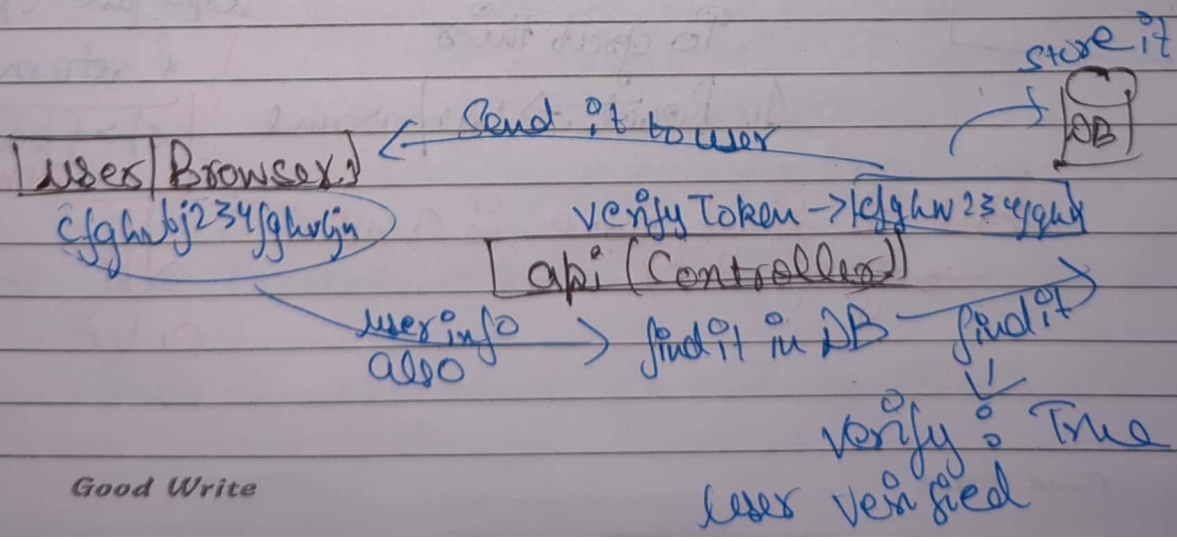
10/9/23 Mongodb, signup & login

DATE: / PAGE: 6

→ Src
  ↳ models
    ↳ userModel.js

- username
- email
- password
- isVerified → false
- isAdmin → Boolean → false
- forgotPasswordToken : String,
- forgotPasswordTokenExpiry : Date,
- verifyToken : String,
- verifyTokenExpiry : Date,

How Token works →

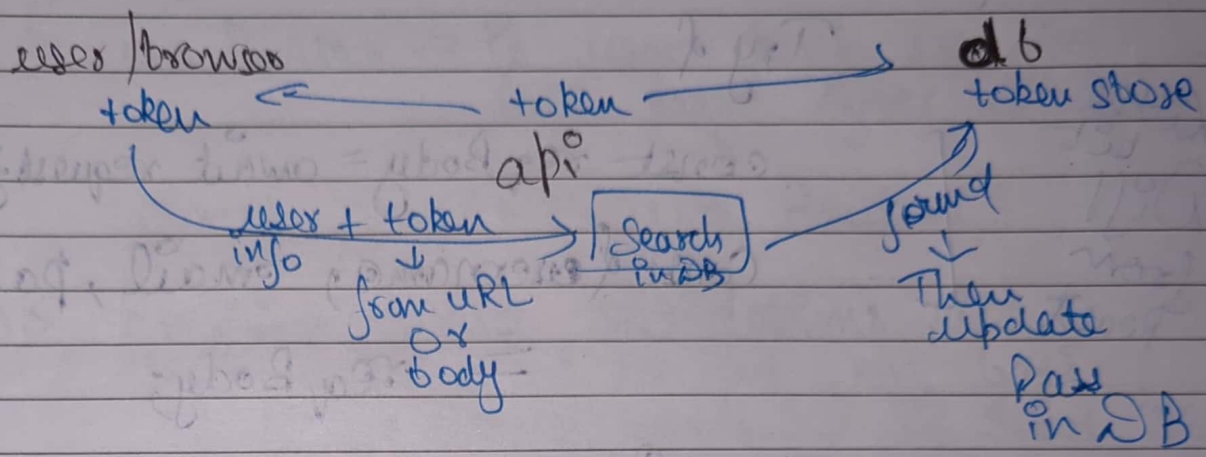


Good Write

10/9/23

DATE : PAGE : 17

forgot Pass Token



/api/user/signup get

post

/api/user/create user put

delete

Do not this because Not standarised approach

Src → app → api

↳ Signup → route.ts

imports
- Connect from → db
- User → @/models/userModel
- NextRequest, NextResponse → next/server
- Bycrypt

connect()

Good Write

19/9/23

for DB

Type of Request

DATE: PAGE: 8

```
export async function POST(request: NextRequest)
{
    Try {
        const reqBody = await request.json();
        Const {username, email, pass} = reqBody;
        // validate
            -> email exist or Not
        // -> hashed your pass
        // -> Save it DB
        return NextResponse.json({
            message: "User created",
            success: True
        }, {status: 201})
    } catch (error) {
        return NextResponse.json({
            error: True},
            {status: 500})
    }
```

Little bit Diff from React

19/9/

Ap

Good Write

19/9/23

DATE: PAGE: 9

App -> Signup -> page.tsx

-> use Router -> is used push user to login page after signin

Const router = use Router()

(1) -> creating some validations like button disabled or not on basis of if we entered anything in Input field

↳ use state & useEffect

(2) -> Also create Loading use state for api call

☆ -> finally -> It is added with try-catch
↳ it should run if our code Catch error or runs.

try {
await axios.post("/api/users/signup", user);

router.push("/login")

} catch (error) {

} finally {
setLoading(false);
}

Good Write

19/9/23

☆ -> always connect -> while writing handlers

src
↳ app
↳ api
↳ users
↳ login
↳ route.ts

export async function POST(request: NextRequest)
{
try {

(S-1) -> get Data from req body

(S-2) -> check if user email exist or Not -> Validation

(S-3) -> check pass is correct or Not using bcrypt compare

(S-4) -> create token data

(S-5) -> create token using jwt

Access Cookies ->

const response = NextResponse.json({
message: ->
success: true,
})

response.cookies.set("token", token, {
httpOnly: true,
})

}

Good Write

19/5/23

DATE: PAGE: 11

-> At last Return Response

Note -> Next JS -> 13.4.12 -> greater than this version doesnot support cookies.set

-> Show error -> Cookies is Not Iterable

src
└ app :-> Login
└ page.tsx

(S-1) -> Made Loading & button disabled states like signup

(S-2) -> Route ->

```
try {
  await axios.post("/api/users/login", user);
  router.push("/profile");
} catch (—) {
} finally {
}
```

Good Write

21/9/23

JWT, cookies & protected routes, Middleware

app
↳ api
↳ users
↳ logout → route.js

```
import { NextResponse } from "next/server"

export async function GET() {
    try {
        const response = NextResponse.json({
            message: "Logout successful",
            success: true,
        })

        response.cookies.set("token", "", {
            httpOnly: true, expires: new Date(0)
        });

        return response;

    } catch (error: any) {
        return NextResponse.json({ error: error.message }, { status: 500 })
    }
}
```

21/9/23:

src
↳ app
↳ profile
↳ page.tsx → Make it client component

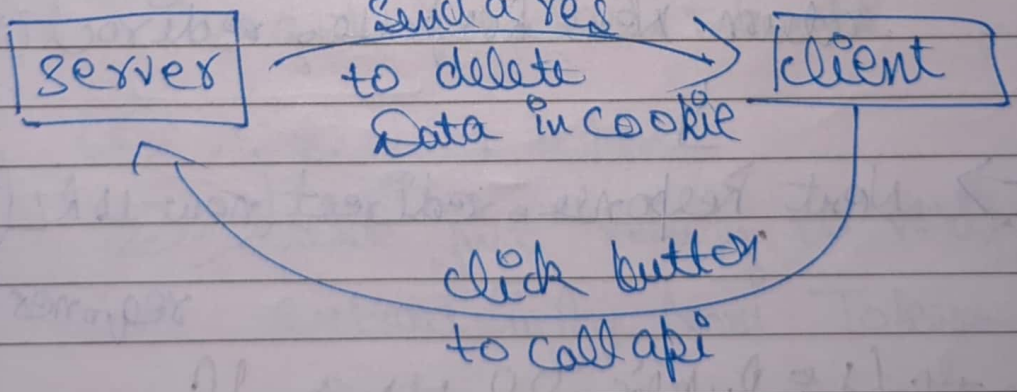
(1) → Adding a button logout

(2) > const logout = async () =>{
await axios ("api/users/logout")
↳ response

toast.success(—);

router.push ("/login")

↳ use Router()
↳ next/navigation



Good Write

2/9/23 # Middleware →

→ They run before cached content & routes are matched.

src
↳ middleware.ts (or js)

```
export function middleware (request : NextRequest) {
    Const path = request.nextUrl.pathname
                    ↓
                gives path (current path)

Const isPublicPath = path === '/login'
                  || path === '/signup'

(2) -> get token

If (isPublicPath && token) {
```

(M-1) →
```
    return NextResponse.redirect('/')
}
```

(M-2) →
```
Next Response.redirect (new URL ('/',
                        req.nextUrl)

If (!isPublic && !token) {
return NextResponse.redirect (new URL ('/', request.
                                    nextUrl)
}
```

Good Write

21/9/23

// see 'Matching Paths' below

export const config = {

matcher: '/about/:path*',

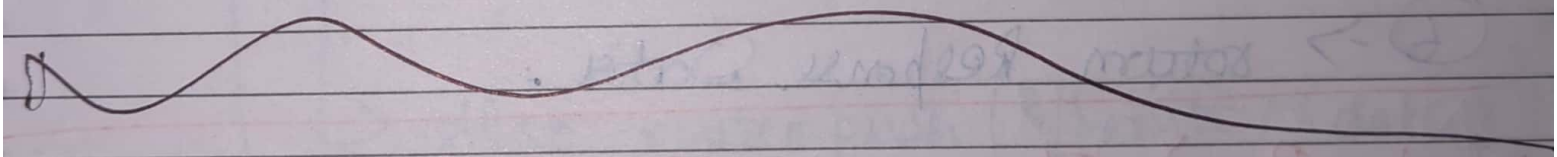
Can be only one ← matcher: ['/about/:path*', '/dashboard/:path*'],

Or

matches: ['/', '/login', '/profile/:path*', etc]

↑ Array

These are the Routes where middleware Runs ←



src

↳ helper -> extracting Data from Token

↳ getDataFromToken.ts

(1) -> extract token

(2) -> use jwt verify to verify and extract info from Token

↳ It uses token secret

(3) -> Return id from Data

21/9/23

DATE: PAGE: 16

src
↳ app → api
↳ users
↳ me/profile
↳ route.ts

(1) => Import
↳ getDataFromToken
→ NextRequest, NextResponse
→ User from model

(2) => connect() => db => also import it

(3) => await getDataFromToken(request);

(4) => Find user using id

(5) => Check user exist or Data

(6) => return Response Data:

Note:- => If you don't want some data from user

user.select("-password -email");

- => minus means remove them

Good Write

21/9/25

src
↳ app
↳ Profile
↳ page.tsx

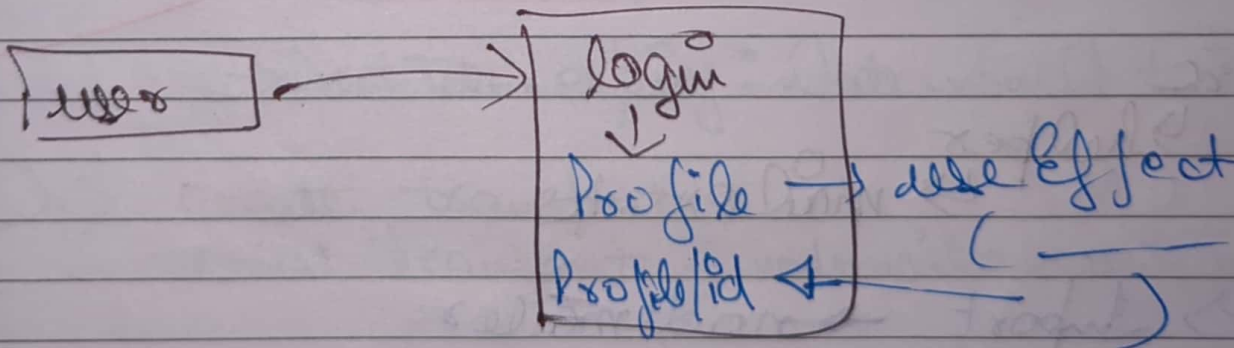
(1) -> Add a button -> get user Data

(2) -> Data -> use State

```
const getUserData = async () => {
  try {
    const res = axios.get("api/users/me")
    setData(res.data.data._id);
  } catch (error) {
    console.log(error);
  }
}
```

↳ also router.push(`/profile/${data}`)

(3) -> Do this in useEffect



Good Write

23/9/25 (4) -> User verification & emails

DATE: PAGE: 18

Nodemailer -> using for emails

-> helper -> only send mails but we are also generating token also

Mail-trap -> For email Testing (can take finbox of mails)

Approaches ->

(1) -> domain.com/verify token/asasadfdg -> Backend Approach

(2) -> domain.com/verify token? token = addfdf -> client Approach

Note :-> Both approaches can be used in both fontend & Backend but preferred wit

src
↳ helper
↳ mail.ex.ts

(1) -> Import -> nodemailer
↳ user
↳ Bcrypt

Good Write

23/9/23

DATE: PAGE: 19

(2) -> Take email, emailTypa, user Id
verify email → Reset Pass → hash it & use as Token

(3) -> bcrypt.hash(userId, 10)
Some time user Id can be in Bson form so -> use .toString()

(4) -> If emailTypa === "verify"
-> find user using userId & update
verifyToken:
verifyTokenExpiry: Date.now() + 3600

(5) -> If emailTypa === "Reset"
-> ResetToken:
->ResetTokenExpiry: Date.now() + 3600;

(6) -> create transporter
const transport = nodemailer.createTransport({
host: sandbox,
port: ,
auth: { user: pass: } -> from, to, Mailtrap

Good Write

23/9/23

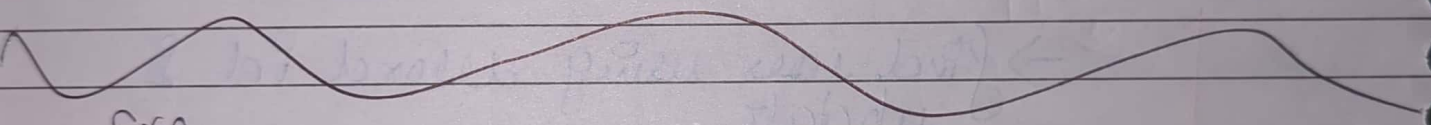
(7) -> Mail options

```
const mailOptions = {
    from: 'kuldeep@gmail.com'
    to: email
    subject: ———
    html:

}
```

(8) -> Sending mail

```
const mailResponse = await transport.sendMail
                     (mailOptions);
```



src
↳ app
↳ api -> user
↳ verifyEmail (POST)
↳ route.ts

(1) -> get Token from req body

(2) -> Find user using Token & (Find One)

Note: -> verifyTokenExpiry: {$gt: Date.now()}

This thing should be greater than Date.now

23/9/25

(3) -> Update user.isVerified = true
user.verifyToken = undefined
user.verifyTokenExpiry = undefined

(4) -> Return Response

Email Sending ---> Signup (before return response)
api/users/signup/route.ts
await sendEmail ({ email, emailType: 'verify', userId: user._id }) (req body)

return response

Note -> Will Recieve Mail at Mailtrap Inbox

Frontend Verify Email page

- Src
  - app -> verifyemail -> page.tsx ("use client")

(1) -> use state --> token, setToken -> ''
false <-[-> error, setError
-> verified, setVerified

23/9/23

③ -> const verifyUserEmail = async () => {
try {
await axios.post('api/users/verifyemail', {token},
setVerified(true)
}

④ -> use useEffect

useEffect(() => {

const token = window.location.search.split("=")[1]

Set token(token || '')

}, [])

for protection => code na jate

[0] : [1]

http://localhost/verifyemail?token=abcdef

split

search

window.location

☆ useEffect(() => {

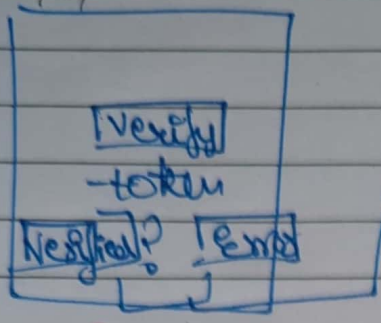
If (token.length > 0) {

verifyUserEmail();

}, [token]);

Good Write

23/9/23



→ Condition Rendering

★ => Also add verify email in Middleware Matcher and in isPublicPaths

Assignment =>

1. => Improve the UI of the application
2. => Add feature of forgot password

=> Finally Deploy on vercel
↳ Add env also