InitialS

Index.css:x

Add rule

\*{

  margin: 0;

}

Hiding Scrollbar with scrolling effect:

.punkList{

    cursor: pointer;

    display: flex;

    overflow: scroll;

}

/\*hiddinfg scrollbar\*/

.punkList::-webkit-scrollbar{

    display: none;

}

To make things center align horizontally and vertically:

.app{

  display: grid;

  place-items: center;

  height: 100vh;

}

Width using flex:

.sidebar{

    display: flex;

    flex-direction: column;

    flex: 0.35; /\*sidebar takes 35% of screen\*/

}

How to break down layout:



Icon use as button:

 <IconButton>

                    <DonutLarge />

                  </IconButton>

Automatic space between contents:

Use justify content:

.sidebar\_\_header{

    display: flex;

    justify-content: space-between;

    padding: 20px;

}

Code to make search bar thing:

<div className="sidebar\_\_search">

                    <div className="sidebar\_\_searchContainer">

                      <SearchOutlined/>

                      <input type="text" placeholder="Search or start new chat"/>

                    </div>

            </div>

.sidebar\_\_search{

    display: flex;

    align-items: center;

    background-color: #f6f6f6;

    padding: 10px;

    height: 39px;

}

.sidebar\_\_searchContainer{

    display: flex;

    align-items: center;

    width: 100%;

    height: 35px;

    border-radius: 20px;

}

.sidebar\_\_searchContainer > .MuiSvgIcon-root{

    color: grey;

    padding: 10px;

}

.sidebar\_\_searchContainer > input{

    border: none;

    margin-left: 10px;

    outline-width: 0;

}

To take full space:

Use flex: 1

For scroll:

Overflow: scroll

Simple card:



        <div className="sidebarChat">

            <Avatar/>

            <div className="sidebarChat\_\_info">

                <h2>Room name</h2>

                <p>This is the last room</p>

            </div>

        </div>

.sidebarChat{

    display:flex;

    padding: 20px;

    cursor: pointer;

    border-bottom: 1px solid #f6f6f6;

}

.sidebarChat:hover{

    background-color: #ebebeb;

}

.sidebar\_\_chats{

    flex:1;

    background-color: white;

    overflow: scroll;

}

.sidebarChat\_\_info{

    margin-left: 15px;

}

.sidebarChat\_\_info > h2{

    font-size: 16px;

    margin-bottom: 8px;

}

To make content full width as possible:

Use flex 1 (to take all possible vertical width)

Example:



.chat\_\_headerInfo{

    flex: 1;

    padding-left: 20px;

}

More than One class:

<p className="chat\_\_message chat\_\_reciever">

Use form where enter is required:



            <div className="chat\_\_footer">

                <InsertEmoticon/>

                <form>

                    <input placeholder="Type a message" type="text"/>

                    <button type="submit">Send a message</button>

                </form>

                <MicNone/>

            </div>

In css, as we only need enter functionality and don’t need to show button, so we hide it

.chat\_\_footer {

    display: flex;

    justify-content: space-between;

    align-items: center;

    height: 60px;

    border-top: 1px solid lightgray;

}

.chat\_\_footer > form {

    flex: 1;

    display: flex;

}

.chat\_\_footer > form > input {

    flex: 1;

    border-radius: 30px;

    padding: 10px;

    border: none;

}

.chat\_\_footer > form > button {

    display: none;

}

Material Ui Icon class:

.sidebar\_\_searchContainer > .MuiSvgIcon-root

MongoDb backend:

Separate folder for backend

Npm init (entry point -> server.js)

Query Selector:



Code:

div.about\_\_text::before{

    content: "<div>";

    color:#515152;

    font-weight: normal;

    font-family: 'La Belle Aurore', cursive;

    font-size: 18px;

    margin-left: -2%;

    z-index: 100;

    line-height: 0px;

}

div.about\_\_text::after{

    content: "</div>";

    color:#515152;

    font-weight: normal;

    font-family: 'La Belle Aurore', cursive;

    font-size: 18px;

    margin-left: -2%;

}

.about\_\_text > h2::before{

    content: "<h2>";

    margin-left: -2%;

    color:#515152;

    font-size: 16px;

    font-weight: normal;

    font-family: 'La Belle Aurore', cursive;

    font-size: 18px;

}

.about\_\_text > h2::after{

    content: "</h2>";

    margin-left: -2%;

    color:#515152;

    font-size: 16px;

    font-weight: normal;

    font-family: 'La Belle Aurore', cursive;

    font-size: 18px;

}

.about\_\_text > p::before{

    content: "<p>";

    margin-left: -2%;

    line-height: 30px;

    color: #515152;

    font-family: 'La Belle Aurore', cursive;

    font-size: 18px;

}

.about\_\_text > p::after{

    content: "</p>";

    margin-left: -2%;

    line-height: 30px;

    color: #515152;

    font-family: 'La Belle Aurore', cursive;

    font-size: 18px;

}

Select Element of Particular Class:

div.about\_\_text{

    content: "<div>";

    color:#515152;

    font-weight: normal;

    font-family: 'La Belle Aurore', cursive;

    font-size: 18px;

    margin-left: -2%;

    z-index: 100;

    line-height: 0px;

}

Selecting div of class about\_\_text

Button:

import { Link } from 'react-router-dom';

                        <Link to="contact">

                            <button id="btn">

                                Contact Us

                            </button>

                        </Link>

Whatsapp mern backend:

In backend folder

Npm init

In package.json -> update it by adding start line in under scripts dict

  "scripts": {

    "test": "echo \"Error: no test specified\" && exit 1",

    "start": "node server.js"

  },

After that, we can either install node modules

Npm install

Or we can directly install dependencies which also install node modules

npm i express mongoose

Then create server.js file

MongoDb:

Reference link:

[Get Started with Atlas — MongoDB Atlas](https://docs.atlas.mongodb.com/getting-started/)

It is a no sql db

After that go to mongodb and create new project and free cluster

Then create api

Structure of server.js

//importing

//app config

//middleware

//DB config

//mongodb stuff

//api routes

//listen

Now to do importing like react in server.js file

Add type:module in package.json

{

  "name": "whatsapp-backend",

  "version": "1.0.0",

  "description": "",

  "main": "server.js",

  "type": "module",

  "scripts": {

    "test": "echo \"Error: no test specified\" && exit 1",

    "start": "node server.js"

  },

First dummy check api on some base url or endpoint (/) and get response hello world when status is ok (200) on port 9000

//importing

import express from 'express'

//app config

const app = express()

const port = process.env.PORT || 9000

//middleware

//DB config

//mongodb stuff

//api routes

app.get('/', (req, res)=>res.status(200).send('hello world'))

//listen

app.listen(port, ()=>console.log(`Listening on localhost: ${port}`))

now to run this server

install nodemon

npm I –g nodemon

Then write

nodemon server.js

If everything goes right, then server will listen on port 9000 and cmd will return

[nodemon] 2.0.8

[nodemon] to restart at any time, enter `rs`

[nodemon] watching path(s): \*.\*

[nodemon] watching extensions: js,mjs,json

[nodemon] starting `node server.js`

(node:1592) ExperimentalWarning: The ESM module loader is experimental.

Listening on localhost: 9000

Now to get hello world, go to postmen, and get this url and it will return hello world

<http://localhost:9000/>

Now again go to mongodb

Now create user by going onto sidepanel under database access. Make sure to copy password if it is autogenerated

Then go to network access and allow access from current ip address or otherwise from any where

Now after that click on connect in cluster-> connect your application

Copy provided link and again come back in the application

Now its time to connect to db

It would be like

mongodb+srv://admin:<password>@cluster0.kgp7k.mongodb.net/myFirstDatabase?retryWrites=true&w=majority

Replace password and if required then dbname

import mongoose from 'mongoose'

//DB config

const connection\_url = 'mongodb+srv://admin:WrKNg2Jfc5vUkVLw@cluster0.kgp7k.mongodb.net/myFirstDatabase?retryWrites=true&w=majority'

mongoose.connect(connection\_url,{

    useCreateIndex:true,

    useNewUrlParser:true,

    useUnifiedTopology:true

})

If want to upload image, then

//db config

const mongoURI = 'mongodb+srv://admin:m7igUKWbGeZoQFat@cluster0.yfh06.mongodb.net/myFirstDatabase?retryWrites=true&w=majority'

mongoose.connect(mongoURI,{

    useCreateIndex:true,

    useNewUrlParser:true,

    useUnifiedTopology:true

})

mongoose.connection.once('open', ()=>{

    console.log('db connected')

})

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*all below work for images \*\*\*\*

const conn = mongoose.createConnection(mongoURI, {

    useCreateIndex:true,

    useNewUrlParser:true,

    useUnifiedTopology:true

})

let gfs

conn.once('open', ()=>{

    console.log('db connected')

    gfs = Grid(conn.db, mongoose.mongo)

    gfs.collection('images')

});

const storage = new GridFsStorage({

    url:mongoURI,

    file: (req, file) => {

        return new Promise((resolve, reject)=> {

            {

                const filename = `image-${Date.now()}${path.extname(file.originalname)}`

                const fileInfo = {

                    filename: filename,

                    bucketName: `images`

                }

                resolve(fileInfo)

            }

        })

    }

})

const upload = multer({ storage })

//api routes

app.get('/', (req, res)=>res.status(200).send('hello world'))

//\*\*\*\*\*\*\*\*\*\*\*getting image\*\*\*\*\*\*\*\*\*

app.get('/retrieve/image/single', (req, res)=>{

    gfs.files.findOne({filename: req.query.name }, (err, file)=>{

        if (err) {

            res.status(500).send(err)

        }else {

            if (!file || file.length == 0) {

                res.status(404).json({err: 'file not found'})

            } else{

                const readstream = gfs.createReadStream(file.filename);

                readstream.pipe(res);

            }

        }

    })

})

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* image post \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

app.post('/upload/image', upload.any('file'), (req, res)=>{

    res.status(201).send(req.file)

})

This is all needed to connect to mongodb

Create another dbMessage.js file. In that we will write schema

import mongoose from 'mongoose'

//defining schema --> defining how data will be build

// 1.message

// 2.name

// 3.timestamp

const whatsappSchema = mongoose.Schema({

    message: String,

    name: String,

    timestamp: String,

recieved: Boolean,

})

export default mongoose.model('messageCollection', whatsappSchema)

As we can have different documents or collection in one cluster, so we provide collection name in mongoose.model. Now after schema, come back to server.js and use this collection to send messages

import Messages from 'dbMessage.js'

//middleware

app.use(express.json()) //To convert string to json

//DB config

const connection\_url = 'mongodb+srv://admin:WrKNg2Jfc5vUkVLw@cluster0.kgp7k.mongodb.net/myFirstDatabase?retryWrites=true&w=majority'

mongoose.connect(connection\_url,{

    useCreateIndex:true,

    useNewUrlParser:true,

    useUnifiedTopology:true

})

//mongodb stuff

//api routes

app.get('/', (req, res)=>res.status(200).send('hello world'))

app.post('/app/v1/messages/new', (req, res)=>{

    const dbMessage = req.body

    Messages.create(dbMessage, (err, data)=>{

        if (err){

            res.status(500).send(err)

        } else {

            res.status(201).send(data)

        }

    })

})

Now send message using postman

<http://localhost:9000/app/v1/messages/new>

Post

Body

Json

{

    "message": "Sending first message to mongodb",

    "name": "muneeb",

    "timestamp": "dummy timestamp",

    "recieved": **false**

}

And in response it will send data as we have mentioned in app.post

{

    "\_id": "60dc592991159b32b8899d78",

    "message": "Sending first message to mongodb",

    "name": "muneeb",

    "timestamp": "dummy timestamp",

    "recieved": **false**,

    "\_\_v": 0

}

When testing, always makesure that your server is running by using command nodemon server.js

Imp status response

Getting data or downloading data or getting ok -> 200

Error -> 500

Creating something in db or post somethinf -> 201

Now signup on pusher:

It makes mongodb realtime and we don’t need to refresh our app for latest feed

Frontend send msg to backend, pusher takes msg from backend and trigger frontend with the msg



When creating channel for project

Frontend -> react

Backend -> node.js

After creating, it will give us the node js code

We will put that code in server.js file in app config

Also we need to install pusher

Npm I pusher

To trigger pusher, we will add change stream in db config which will be triggered when any change in observed in mongodb

mongoose.connect(connection\_url,{

    useCreateIndex:true,

    useNewUrlParser:true,

    useUnifiedTopology:true

})

const db = mongoose.connection;

db.once('open', ()=>{

    console.log("DB is connected")

    const msgCollection = db.collection('messagecollections');//same name for collection

    const changeStream = msgCollection.watch();

    changeStream.on("change", (change)=>{

        console.log("A change occured", change)

    })

})

Store connection const db, when once db gets open we got identified

Now store our collection in msgCollection const

And change stream will watch change in msgCollection or in our collection

When any change happens, it will give us in console

Test it by sending msg in postmen, it will console the data in real time in CMD (console)

Now its time to trigger pusher when change occurs

 changeStream.on("change", (change)=>{

        console.log("A change occured", change)

        if (change.operationType == 'insert'){

            const messageDetails = chnage.fullDocument;

            pusher.trigger('messages', 'inserted', {

                name: messageDetails.name,

                message: messageDetails.message

            })

        } else {

            console.log('Error in triggering pusher')

        }

Adding headers to make app able to deploy on heruko

First way

//middleware

app.use(express.json()) //To convert string to json

app.use((req, res, next)=>{

    //headers for heruko

    //As access from anywhere, so no security in it

    res.setHeader("Access-Control-Allow-Origin", "\*"); //req from any end point

    res.setHeader("Access-Control-Allow-Headers", "\*"); //accept headers from everywhere

    next();

})

Second way:

Npm I cors

import cors from 'cors';

//middleware

app.use(express.json()) //To convert string to json

app.use(cors())//headers for heruko

Backend completed:

Now connect pusher to frontend so it triggers frontend when db changes

UseEffect:

It is used to run code inside it once when the app loads

In app.js, from getting started in pusher, paste this code under useEffect

  useEffect(()=>{

    const pusher = new Pusher('1da9a361a566d7dc83bc', {

      cluster: 'ap2'

    });

    const channel = pusher.subscribe('messages');

    channel.bind('inserted', (data)=> {

      alert(JSON.stringify(data));

    });

  }, [])

Install pusher-js in frontend

Npm I pusher-js

And import it

import Pusher from 'pusher-js'

Now when we post api from postman, we get data in form of alert 

Which means our app is sync with the mongodb backend

Create axios.js file where we place our backend url to fetch data or to use backend apis

Axios is used to fetch or to interact frontend with backend apis

import axios from 'axios'

const instance = axios.create({

    baseURL: "http://localhost:9000",

});

export default instance;

When deploy on heruko, we just need to change baseUrl

Coming back to app.js

import './App.css';

import {useEffect, useState} from 'react'

import Chat from './components/chat/Chat';

import Sidebar from './components/sidebar/Sidebar';

import Pusher from 'pusher-js'

import axios from './axios'

function App() {

  const [messages,setMessages] = useState([]);

  //useEffect for fetching

  useEffect(()=>{

    axios.get('/messages/sync')

      .then(response=>{

        setMessages(response.data)

      })

  }, [])

 useEffect(()=>{

    const pusher = new Pusher('1da9a361a566d7dc83bc', {

      cluster: 'ap2'

    });

    const channel = pusher.subscribe('messages');

    channel.bind('inserted', (newMessage)=> {

      alert(JSON.stringify(newMessage));

      setMessages([...messages, newMessage])

    });

    return ()=>{

      channel.unbind\_all(); // to unbind so it will no listen to all messages everytime, only listen new message

      channel.unsubscribe();

    };

  }, [messages]) //add message here so it updates with the message

  console.log(messages);

First useEffect to fetch messages from api. Now we need to set and update messages. For that useState is used and we update messages everytime by using setMessages.

Second useEffect is for pusher. Pusher give us the changed data which is an object. So what we do is kept all messages […messages] and also add new message and update using setMessage function

Pass message object to chat component so it can be used after destructuring

<Chat messages={messages}/>

In chat,

function Chat({messages}) {

    const [input, setInput] = useState('');

Or

For tsx

    const sendMessage = (e:Event) => {

        e.preventDefault(); //avoiding from refresh

    }

Update chat body div as

            <div className="chat\_\_body">

                {messages.map((message) =>(

                    <p className={`chat\_\_message ${message.recieved && "chat\_\_reciever"}`}>

                        <span className="chat\_\_name">{message.name}</span>

                        {message.message}

                        <span className="chat\_\_timestamp">

                            {message.timestamp }

                        </span>

                    </p>

                ))}

            </div>

Now to post message to api and keep track of send message and input field:

 const [input, setInput] = useState('');

    const sendMessage = async (e) => {

        e.preventDefault(); //avoiding from refresh

    await axios.post('/messages/sync',{

        "message": input,

        "name": "muneeb",

        "timestamp": "Just now!",

        "recieved": true,

    });

    setInput('');

    }

            <div className="chat\_\_footer">

                <InsertEmoticon/>

                <form>

                    <input value={input} onChange={e => setInput(e.target.value)}placeholder="Type a message" type="text"/>

                    <button onClick={sendMessage} type="submit">Send a message</button>

                </form>

                <MicNone/>

            </div>

Use of classes based on condition:

 <p className={`chat\_\_message ${message.recieved && "chat\_\_reciever"}`}>

Use chat\_\_message class, if message.recieved is true, then use chat\_\_reciever class

Using two classes:

<p className="chat\_\_message chat\_\_reciever">

Material ui Icon class:

.header\_\_option > .MuiSvgIcon-root{

    color: gray;

}

.header\_\_option:hover > .MuiSvgIcon-root{

    color: #2e81f4;

}

Active class with bem:

                <div className="header\_\_option header\_\_option--active">

                    <HomeIcon fontSize="large"/>

                </div>

.header\_\_option--active > .MuiSvgIcon-root{

    color: #2e81f4;

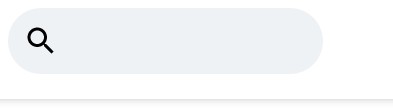
}

.header\_\_option--active{

    border-bottom: 4px solid #2e81f4;

}

Search field:



Code:

                <div className="header\_\_input">

                    <SearchIcon/>

                    <input placeholder="Search Facebook" type="text"/>

                </div>

.header\_\_input{

    display: flex;

    align-items: center;

    background-color: #eff2f5;

    padding: 10px;

    margin-left: 10px;

    border-radius: 999px;

}

.header\_\_input > input{

    outline-width: 0;

    border: none;

    background-color: transparent;

}

React true Condition and component as props:

function Sidebar() {

    return (

        <div className="sidebar">

            <SidebarRow title='Pages'/>

            <SidebarRow title='Friends'/>

            <SidebarRow />

            <SidebarRow />

            <SidebarRow />

            <SidebarRow />

            <SidebarRow />

        </div>

    )

}

We have a sidebar component which use SidebarRow as component and we passed some props to it and use these props in sidebarRow.js. Below is its code

function SidebarRow({title, src, Icon}) {

    return (

        <div className="sidebarRow">

            {src && <Avatar src={src}/>}

            {Icon && <Icon/>}

            {title}

        </div>

    )

}

This is the main thing. Sometimes we need src for avatar and some time we need Icon material ui component, so we destructure all these providing from sidebar.js

Now in first, if src is true then use Avatar component with src

If component is true, then use that Icon component

To pass component as props and use, always make its initial letter capital, like here of Icon

IMAGE IN BACKGROUND OF CARD OR COMPONENT:

function Story({ image, profileSrc, title}) {

    return (

        <div style={{backgroundImage:`url(${image})`}} className="story">

            <Avatar src={profileSrc}/>

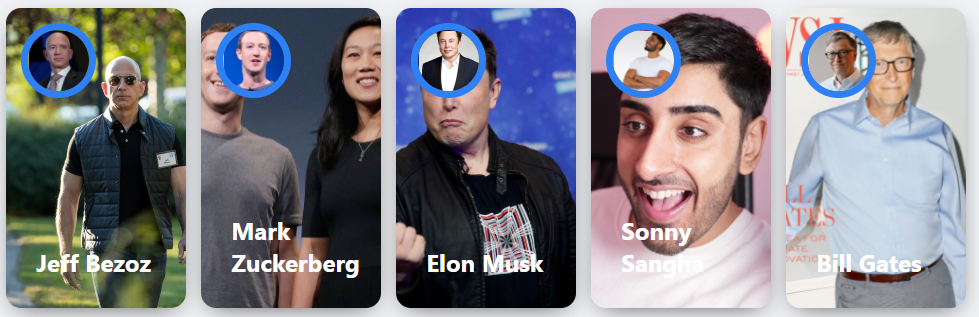
            <h4>{title}</h4>

        </div>

    )

}

Card Design:



import React from 'react'

import './Story.css'

import { Avatar } from '@material-ui/core'

function Story({ image, profileSrc, title}) {

    return (

        <div style={{backgroundImage:`url(${image})`}} className="story">

            <Avatar className="story\_\_avatar" src={profileSrc}/>

            <h4>{title}</h4>

        </div>

    )

}

export default Story

.story{

    position: relative;

    background-position: center center;

    background-size: cover;

    background-repeat: no-repeat;

    width: 100px;

    height: 200px;

    box-shadow: 0px 5px 17px -7px rgba(0,0,0, 0.75);

    border-radius: 10px;

    margin-right: 10px;

    transition: transform 100ms ease-in;

    cursor: pointer;

}

.story:hover{

    transform: scale(1.07);

}

.story\_\_avatar{

    margin: 10px;

    border: 5px solid #2e81f4;

}

.story > h4 {

    position: absolute;

    left: 20px;

    color: white;

    bottom: 20px

}

Calling component with className in CSS:

Example:

<div className="storyReel">

            <Story

                image="https://links.papareact.com/k2j"

                profileSrc="https://links.papareact.com/f0p"

                title="Jeff Bezoz"

            />

Css:

.storyReel\_\_story{

    margin-right: 10px;

}

For middle section when 3 sections:

function Feed() {

    return (

        <div className="feed">

            <StoryReel/>

            {/\* MessageSender \*/}

        </div>

    )

}

.feed{

    flex: 1;

    padding: 30px 150px;

    display: flex;

    flex-direction: column;

    justify-content: center;

    align-items: center;

}

And for left section when 3 sections:

.sidebar{

    flex: 0.33;

    padding: 25px 10px;

}

To hide any tag or content or hide button in form:

Sometimes we only need form input field without button, in that case

<div className="messageSender\_\_top">

                <Avatar/>

                <form>

                    <input className="messageSender\_\_input" placeholder="What's on your mind?"/>

                    <input placeholder="image URL (Optional)"/>

                    <button onClick={handleSubmit} type="submit">Hidden Submit</button>

                </form>

            </div>

.messageSender\_\_top > form > button{

    display: none;

}

After adding button in form, we hide it using css

TRUE FALSE CONDITION IN REACT OR AUTH:

function App() {

  const user = "muneeb"

    return (

    <div className="app">

      {!user ? (

        <h1>Login</h1>

      ):(

        <>

        <Header/>

        <div className="app\_\_body">

          <Sidebar/>

          <Feed/>

          <Widgets />

        </div>

        </>

      )}

    </div>

  );

}

Auth page or place item in center:

Styling material ui button:

 <div className="login">

            <div className="login\_\_logo">

                <img src="https://upload.wikimedia.org/wikipedia/commons/thumb/5/51/Facebook\_f\_logo\_%282019%29.svg/1200px-Facebook\_f\_logo\_%282019%29.svg.png" alt=""/>

                <img src="https://www.logo.wine/a/logo/Facebook/Facebook-Logo.wine.svg" alt=""/>

            </div>

                <Button type="submit" onClick={signIn}>Sign In</Button>

        </div>

.login > button {

    width: 300px;

    background-color: #2e81f4 !important;

    color: #eff3f5;

    font-weight: 800;

}

.login >button:hover{

    background-color: white;

    color: #2e81f4

}

For firebase authentication and db:

1. For authentication, click on authentication in project and select google and enable it
2. For db, click on firestore, get started and select required settings like test mode enable etc
3. Copy the below code in firebase.js file

import firebase from 'firebase';

const firebaseConfig = {

  apiKey: "AIzaSyBDCe6f-cXkeSXyGFmDecXN\_\_jE3-kUVTo",

  authDomain: "facebook-firebase-fullstack.firebaseapp.com",

  projectId: "facebook-firebase-fullstack",

  storageBucket: "facebook-firebase-fullstack.appspot.com",

  messagingSenderId: "663232030936",

  appId: "1:663232030936:web:7fc4d405d249c18c110fad"

};

//initializing app mandatory

const firebaseApp = firebase.initializeApp(firebaseConfig);

//db stuff of firebase

const db = firebaseApp.firestore();

//firebase authentication

const auth = firebase.auth();

const provider = new firebase.auth.GoogleAuthProvider()

export {auth, provider};

export default db;

1. Come to login page. In that, on login button, make sure to add button with onclick having sign in function. In that sign in function, use this function

    const signIn = () => {

        auth

        .signInWithPopup(provider)

        .then((result)=>{

            console.log(result)

        })

        .catch((error)=>alert(error.message))

    };

Here is the complete login page if needed

import { Button } from '@material-ui/core'

import React from 'react'

import './Login.css'

import {auth, provider} from '../../firebase';

function Login() {

    const signIn = () => {

        auth

        .signInWithPopup(provider)

        .then((result)=>{

            console.log(result)

        })

        .catch((error)=>alert(error.message))

    };

    return (

        <div className="login">

            <div className="login\_\_logo">

                <img src="https://upload.wikimedia.org/wikipedia/commons/thumb/5/51/Facebook\_f\_logo\_%282019%29.svg/1200px-Facebook\_f\_logo\_%282019%29.svg.png" alt=""/>

                <img src="https://www.logo.wine/a/logo/Facebook/Facebook-Logo.wine.svg" alt=""/>

            </div>

                <Button type="submit" onClick={signIn}>Sign In</Button>

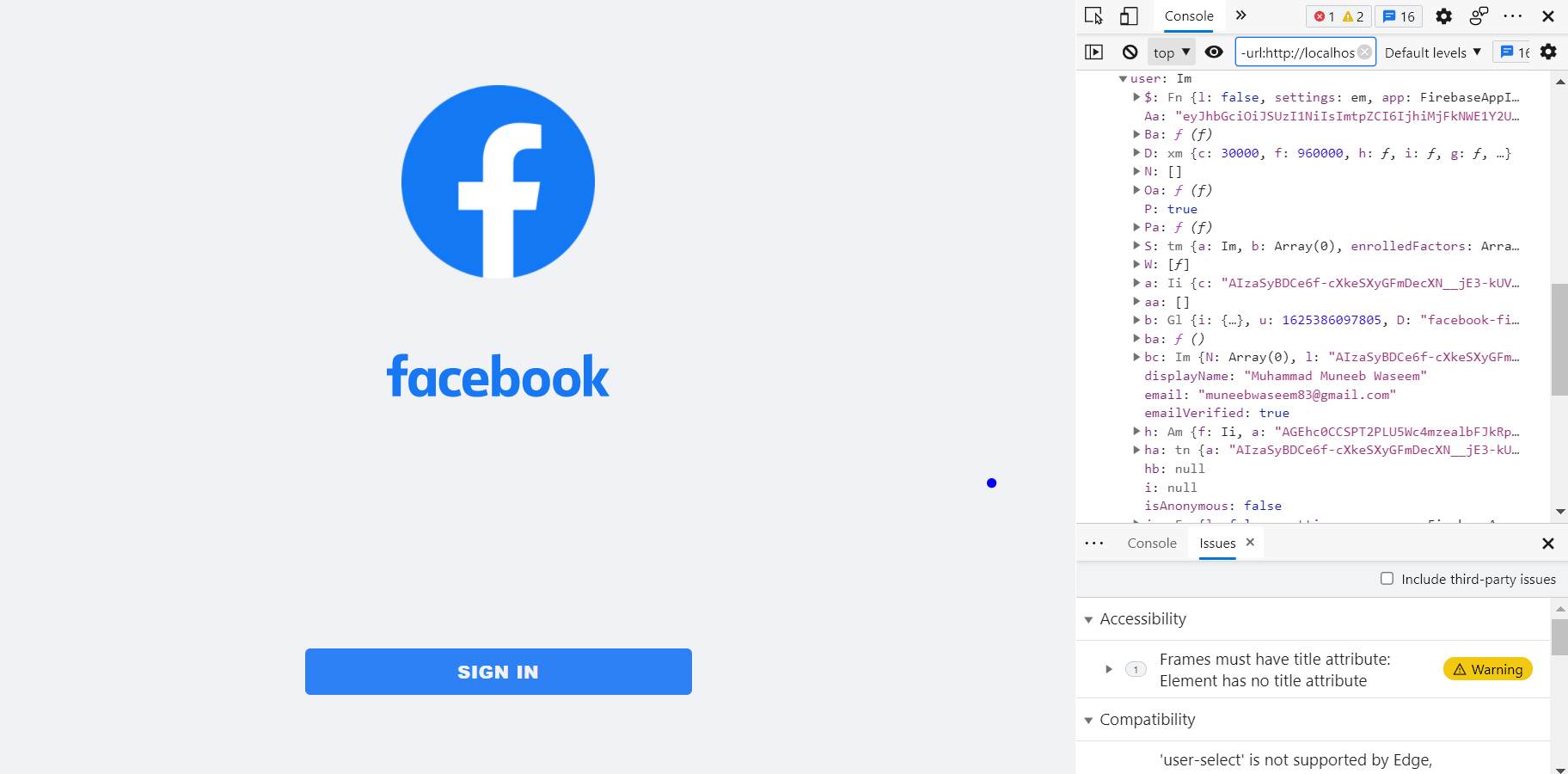
        </div>

    )

}

export default Login

Some info:



When clicking the sign in and selecting the id, it shows details of user in console, not to make it in use, we need to use context api

1. Now to circulate that information in all parent components and their child, context api will come in action. It consist of provider and consumer. Provider is used to provide value to context api component and then usecontext is use to extract that information from context api and able to use it in child components. Remember, place all required all relevant components or parent components under context api provider so that all of their child and child of child can consume that value. And reducer will be used for storing the info

More info: [useState-Context-Reducer/App.js at master · MUHAMMAD-MUNEEB-WASEEM/useState-Context-Reducer (github.com)](https://github.com/MUHAMMAD-MUNEEB-WASEEM/useState-Context-Reducer/blob/master/src/App.js)

1. Now create StateProvider folder, in that create two files stateProvider.js containing context api and reducer.js. They have code of pattern, paste that pattern and wrap whole app in index.js under the state provider

Reducer (with context api) and redux article:

Initially both have two main files.

1. One for the action export containing action type
2. Second is for reducer containing initial state and switch case
3. Main difference is of store and context file. In Redux we provide reducer file to store so all data can be store in the store and wrap app inside index.js under Provider containing store and then use useSelector to get data from store or dispatch to update data in store. Whereas in context file, we use createContext() and pass initial state to it and then pass useReducer to state and dispatch then wrap childer in provider and then useContext to get state or dispatch and from state we can get data and from dispatch we can update data

Article: [Redux and Context API | Codehouse (codehousegroup.com)](https://www.codehousegroup.com/insight-and-inspiration/tech-stream/using-redux-and-context-api)

Code for stateprovider

import React, { createContext, useContext, useReducer } from 'react'

export const StateContext = createContext()

export const StateProvider = ({ reducer, initialState, children }) => (

    <StateContext.Provider value={useReducer(reducer, initialState)}>

        {children}

    </StateContext.Provider>

)

export const useStateValue = () => useContext(StateContext)

Code for reducer.js

export const initialState = {

    user: null,

}

export const actionTypes = {

    SET\_USER: 'SET\_USER'

}

const reducer = (state, action) => {

    console.log(action)

    switch (action.type) {

        case actionTypes.SET\_USER:

            return {

                ...state,

                user: action.user

            }

        default:

            return state

    }

}

export default reducer

Wrapping whole app index:

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

import { StateProvider } from './components/StateProvider/StateProvider';

import reducer, { initialState } from './components/StateProvider/Reducer';

ReactDOM.render(

  <React.StrictMode>

  <StateProvider initialState={initialState} reducer={reducer}>

    <App />

  </StateProvider>

  </React.StrictMode>,

  document.getElementById('root')

);

// If you want to start measuring performance in your app, pass a function

// to log results (for example: reportWebVitals(console.log))

// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals

reportWebVitals();

Now to store data in data layer or reducer store, we use dispatch and passed required user information to it. Again go back to login.js and update code as follows

 const [state, dispatch] = useStateValue();

const signIn = () => {

        auth

        .signInWithPopup(provider)

        .then((result)=>{

            dispatch({

                type: actionTypes.SET\_USER,

                user: result.user

            })

        })

        .catch((error)=>alert(error.message))

    };

What does it do is if action type is SET\_USER, then it will run its corresponding case, and pass user information which we are getting and store it in data layer or store. Now we can use this Information using useContext api in app.js which we have store in useStateValue variable in StateProvider.js

const [{user}, dispatch] = useStateValue();

In this, we are destructuring user from state which stored in data layer and thus our sign in will work now. Now we just need to pass some required user information in our headers sidebar or where we need.

const [{user}, dispatch] = useStateValue();

The above is the most imp line, where we want to use data from data layer using context api, we use this line of code. And then take data from it. Like example of header.js;

const [{user}, dispatch] = useStateValue()

 <div className="header\_\_info">

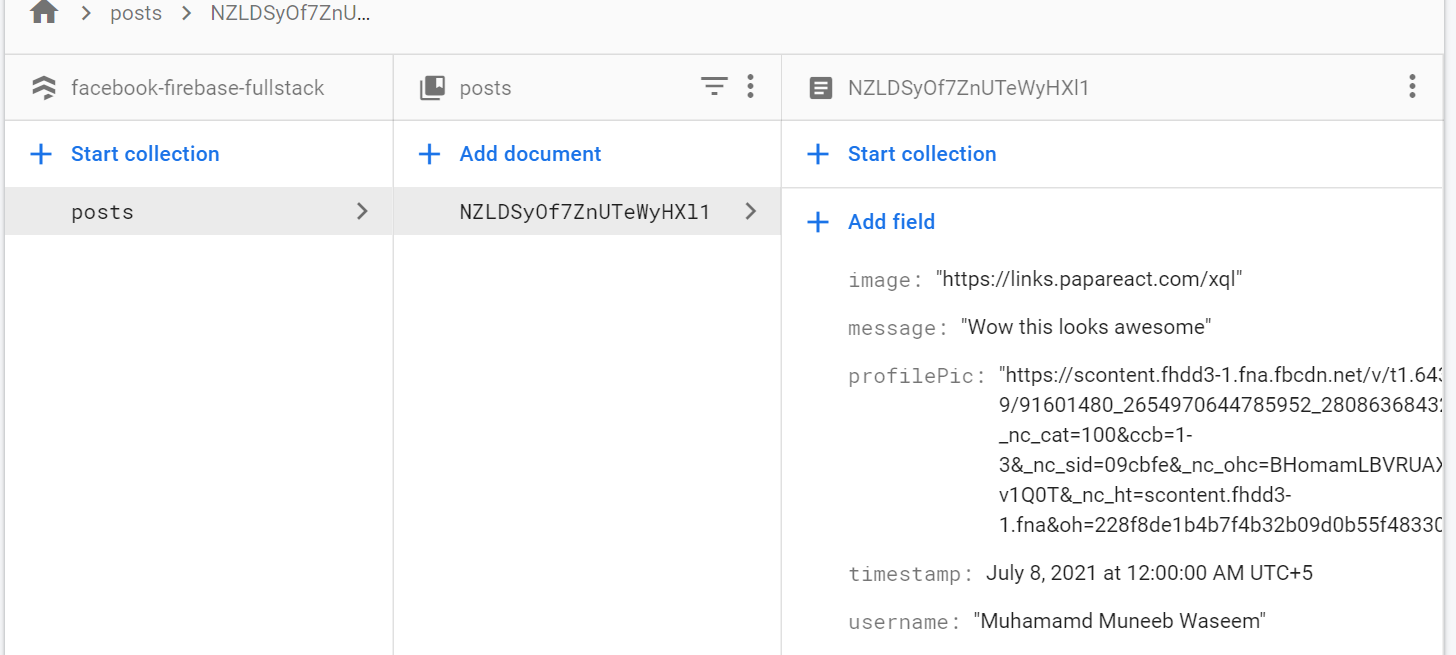
                    <Avatar src={user.photoURL}/>

                    <h4>{user.displayName}</h4>

                </div>

FIRBASE db:

1. firebase.js as done above
2. Now to get the data. First create collection with some name and store data in keys that you required. For example



1. Now after that come back to your application and go to that component where you want to get the data, and use useState which have const to use this data and function to update the data.

Code:

import db from '../../firebase';

import { useEffect } from 'react'

function Feed() {

    const [posts, setPosts] = useState([]);

     useEffect(()=>{

        db.collection('posts').orderBy('timestamp', 'desc').onSnapshot(snapshot => {

            setPosts(snapshot.docs.map(doc => ({ id:doc.id, data: doc.data() })))

        })

    }, [])

    return (

        <div className="feed">

            <StoryReel/>

            <MessageSender/>

        {posts.map((post) => (

            <Post

            key={post.id}

            username={post.data.username}

            message={post.data.message}

            timestamp={post.data.timestamp}

            image={post.data.image}

            profilePic={post.data.profilePic}

            />

        ))}

1. The above code for get always remain same, you just need to change collection name and and properties like username etc as per your requirement
2. Now this point 5 is not imp, I am just showing Post component where we are using these values

function Post({ profilePic, image, username, timestamp, message }) {

    return (

        <div className="post">

            <div className="post\_\_top">

                <Avatar src={profilePic} className="post\_\_avatar"/>

                <div className="post\_\_topInfo">

                    <h3>{username}</h3>

                    <p>{new Date(timestamp?.toDate()).toUTCString()}</p>

                </div>

            </div>

            <div className="post\_\_bottom">

                <p>{message}</p>

            </div>

            <div className="post\_\_image">

                <img src={image} />

            </div>

Remember code for date, it will be always like this. This get code for firebase is available in facebook-firebase-fullstack repo, in feed and post component

DATE for FIREBASE:

                <div className="post\_\_topInfo">

                    <h3>{username}</h3>

                    <p>{new Date(timestamp?.toDate()).toUTCString()}</p>

                </div>

1. Now to post data to firebase, just add this snippet with required properties in the relevant component

import db from '../../../firebase';

import firebase from 'firebase'

function MessageSender() {

    const [input, setInput] = useState('')

    const [imageUrl, setImageUrl] = useState('')

    const [{user}, dispatch] = useStateValue()

    const handleSubmit = e => {

        e.preventDefault();

        db.collection('posts').add({

            message: input,

            timestamp: firebase.firestore.FieldValue.serverTimestamp(),

            profilePic: user.photoURL,

            username: user.displayName,

            image: imageUrl

        })

        setInput('');

        setImageUrl('');

    }

    return (

        <div className="messageSender">

            <div className="messageSender\_\_top">

                <Avatar src={user.photoURL}/>

                <form>

                    <input value={input} onChange={(e) => {setInput(e.target.value)}} className="messageSender\_\_input" placeholder={`What's on your mind, ${user.displayName}`}/>

                    <input value={imageUrl} onChange={(e) => {setImageUrl(e.target.value)}} placeholder="image URL (Optional)"/>

                    <button onClick={handleSubmit} type="submit">Hidden Submit</button>

                </form>

            </div>

That’s it.

FIREBASE DEPLOYMENT

* Firebase login (where hosting exist)
* Firbase init
* Ready for deployment: Y
* Hosting
* Existing project
* Public directory: build
* Single page app: Yes
* Automatic: No
* Now write: npm run build or yarn build
* Firebase deploy

In order to update firebase deployment:

* Update code
* Delete previous build folder and then yarn or npm build again
* Firebase deploy
* After that go to your project in firbase, click on deployment and in release history, delete previous releases

Facebook mern backend:

Npm init

Npm install

Npm i mongoose express cors multer multer-gridfs-storage@4.2.0 gridfs-stream body-parser path pusher

About library: multer is used to extract gridfs, gridfs is used to store images in storage. If no images, then no need to download multer, gridfs etx

//importing stuff

import express from 'express';

import mongoose from 'mongoose';

import Pusher from 'pusher';

import cors from 'cors';

import path from 'path';

import Grid from 'gridfs-stream';

import multer from 'multer';

import GridFsStorage from 'multer-gridfs-storage'

import bodyParser from 'body-parser';

import mongoPosts from './mongoPosts';

Grid.mongo = mongoose.mongo

// app config

const app = express();

const port = process.env.PORT || 9000;

//middlewares

app.use(express.json()) //To convert string to json

app.use(cors())//headers for heruko

//db config

const mongoURI = 'mongodb+srv://admin:m7igUKWbGeZoQFat@cluster0.yfh06.mongodb.net/myFirstDatabase?retryWrites=true&w=majority'

mongoose.connect(mongoURI,{

    useCreateIndex:true,

    useNewUrlParser:true,

    useUnifiedTopology:true

})

mongoose.connection.once('open', ()=>{

    console.log('db connected')

})

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*all below work for images \*\*\*\*

const conn = mongoose.createConnection(mongoURI, {

    useCreateIndex:true,

    useNewUrlParser:true,

    useUnifiedTopology:true

})

let gfs

conn.once('open', ()=>{

    console.log('db connected')

    gfs = Grid(conn.db, mongoose.mongo)

    gfs.collection('images')

});

const storage = new GridFsStorage({

    url:mongoURI,

    file: (req, file) => {

        return new Promise((resolve, reject)=> {

            {

                const filename = `image-${Date.now()}${path.extname(file.originalname)}`

                const fileInfo = {

                    filename: filename,

                    bucketName: `images`

                }

                resolve(fileInfo)

            }

        })

    }

})

const upload = multer({ storage })

//api routes

app.get('/', (req, res)=>res.status(200).send('hello world'))

//\*\*\*\*\*\*\*\*\*\*\*get and post image\*\*\*\*\*\*\*\*\*

app.get('/retrieve/image/single', (req, res)=>{

    gfs.files.findOne({filename: req.query.name }, (err, file)=>{

        if (err) {

            res.status(500).send(err)

        }else {

            if (!file || file.length == 0) {

                res.status(404).json({err: 'file not found'})

            } else{

                const readstream = gfs.createReadStream(file.filename);

                readstream.pipe(res);

            }

        }

    })

})

app.post('/upload/image', upload.any('file'), (req, res)=>{

    res.status(201).send(req.file)

})

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* get and post data \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

app.get('/retrieve/posts', (req,res)=>{

    mongoPosts.find((err,data)=>{

        if (err){

            res.status(500).send(err)

        }else {

            data.sor((b,a)=>{

                return a.timestamp-b.timestamp

            });

            res.status(200).send(data)

        }

    })

})

app.post('/upload/post', (req, res)=>{

    const dbPost = req.body

    console.log(dbPost)

    mongoPosts.create(dbPost, (err, data)=>{

        if (err) {

            res.status(500).send(err)

        }else {

            res.status(201).send(data)

        }

    })

})

//listener

app.listen(port, console.log(`listening on localhost: ${port}`));

This is the schema with name mongoPost.js

import mongoose  from "mongoose";

const postModel = mongoose.Schema({

    user: String,

    imgName: String,

    text: String,

    avatar: String,

    timestamp: String

})

export default mongoose.model('posts', postModel);

Now connect to frontend:

Install dependencies:

Yarn add pusher-js axios form-data

Update in post.js to retrieve corresponding post image

            {

                imgName ? (

                    <div className="post\_\_image">

                        <img src={`/retrieve/images/single?name=${imgName}`} />

                    </div>

                ) : (

                        console.log('DEBUG >>> no image here')

                    )

            }

Now to get data in feed.js

function Feed() {

    const [profilePic, setProfilePic] = useState('')

    const [postsData, setPostsData] = useState([])

    const syncFeed = () => {

        axios.get('/retreive/posts')

            .then((res)=>{

                console.log(res.data)

                setPostsData(res.data)

            })

    };

    useEffect(()=>{

        syncFeed()

    }, [])

    return (

        <div className="feed">

            <StoryReel/>

            <MessageSender/>

        {postsData.map((entry) => (

            <Post

            profilePic={entry.avatar}

            message={entry.text}

            timestamp={entry.timestamp}

            imgName={entry.imgName}

            username={entry.user}

            />

        ))}

Now to post data along with image in messageSender.js

Backend deployment on heroku:

First create project on heroku

Then come to backend folder

heroku login

git init

heroku git:clone -a facebook-backend-mern-muneeb

The above link will be changed as per project name

git add .

git commit -am "make it better"

git push heroku master

Calling same level tags:

        <div className="dummy">

                <h1 className="heading">Dummy</h1>

                <div className="image">

                    <img  src="https://scontent.fkhi11-1.fna.fbcdn.net/v/t1.18169-9/21751306\_10155724905022838\_7192191338970086519\_n.png?\_nc\_cat=1&amp;ccb=1-3&amp;\_nc\_sid=09cbfe&amp;\_nc\_ohc=IcbUmlfMQ\_4AX8Gd\_cH&amp;\_nc\_ht=scontent.fkhi11-1.fna&amp;oh=476a316551fc8daf192c0acc8392ae30&amp;oe=60E5D3FD"/>

                </div>

        </div>

.dummy > div > img {

    width: 200px;                                                                                                                                                                                                                                                                                                        s

    height: 200px;

    margin-right: 100px;

    visibility: hidden;

}

.dummy > h1:hover ~ div > img{

    visibility: visible;

}

CSS True False property:

backgroundColor:`${isShown1 ? "yellow": "#bf0829"}`

More than 1 true false condition:

 style={{ backgroundColor:`${isShown1 ? "yellow": isShown2 ? "green": "#bf0829"}`}}

Active class CSS:

With class:

|  |
| --- |
| .active{ |
|  | border-bottom: 2px solid #FFA78C; |
|  | border-left: 2px solid #FFA78C; |
|  | border-right: 2px solid #FFA78C; |
|  | border-top: 2px solid #FFA78C; |
|  | pointer-events: none; |
|  | } |

<h2 className="heading"><Link className="effect-2 active" to="/portfolio/">All</Link></h2>

With element and its class:

|  |
| --- |
| a.active{ |
|  | border-bottom: 2px solid #FFA78C; |
|  | border-left: 2px solid #FFA78C; |
|  | border-right: 2px solid #FFA78C; |
|  | border-top: 2px solid #FFA78C; |
|  | pointer-events: none; |
|  | } |

With element only:

|  |
| --- |
| a:active{ |
|  | border-bottom: 2px solid #FFA78C; |
|  | border-left: 2px solid #FFA78C; |
|  | border-right: 2px solid #FFA78C; |
|  | border-top: 2px solid #FFA78C; |
|  | pointer-events: none; |
|  | } |

To navigate Back:

import { Link, useNavigate } from 'react-router-dom'

const navigate = useNavigate();

            <div className="gallery\_\_return">

                <button onClick={() => navigate(-1)}>go back</button>

            </div>

Icon with Link:

                <IconButton>

                    <a href="https://www.facebook.com/groups/swotsrestguide" target="\_blank"><FacebookIcon style={{fontSize: "34px", color: "white"}}/></a>

                </IconButton>

Tag and class of same Element:

                        <Link to={timerapp.name} className="projectlink">

                            Timer App

                        </Link>

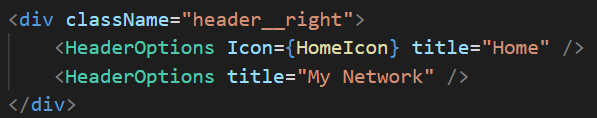
    a.projectlink{

        font-size: 25px;

    }

To pass Icon as props or component:

Make Icon (i) capital as we are passing material-ui Icon



import React from 'react'

import './HeaderOptions.css';

function HeaderOptions({Icon, title}) {

    return (

        <div className="headerOption">

            {Icon && <Icon className="headerOption\_\_icon"/>}

            <h3>{title}</h3>

        </div>

    )

}

export default HeaderOptions

align items work with flexbox, flex direction row or colum

.headerOption{

    display: flex;

    flex-direction: column;

    align-items: center;

    margin-right: 20px;

    color: gray;

    cursor: pointer;

}

Function inside function declare and calling

First:

import { Avatar } from '@material-ui/core'

import React from 'react'

import './Sidebar.css'

function Sidebar() {

    const recentItem = (topic) => (

        <div className="sidebar\_\_recentItem">

            <span className="sidebar\_\_hash">#</span>

            <p>{topic}</p>

        </div>

    )

    return (

        <div className="sidebar">

            <div className="sidebar\_\_bottom">

                <p>Recent</p>

                {recentItem('reactjs')}

                {recentItem('programming')}

                {recentItem('fullstack')}

                {recentItem('design')}

                {recentItem('developer')}

            </div>

        </div>

    )

}

export default Sidebar

Second with also passing separate classNAme type:

function WidgetsLarge() {

    const Button = ({type}) => (

        <button className={"widgetsLarge\_\_button " + type}>{type}</button>

    )

    return (

        <div className="widgetsLarge">

            <h3 className="widgetsLarge\_\_title">Latest Transactions</h3>

            <table className="widgetsLarge\_\_table">

                <tr className="widgetsLarge\_\_tr">

                    <th className="widgetsLarge\_\_th">Customer</th>

                    <th className="widgetsLarge\_\_th">Date</th>

                    <th className="widgetsLarge\_\_th">Amount</th>

                    <th className="widgetsLarge\_\_th">Status</th>

                </tr>

                <tr className="widgetsLarge\_\_tr">

                    <td className="widgetsLarge\_\_user">

                        <img className="widgetsLarge\_\_img" src="https://images.pexels.com/photos/3992656/pexels-photo-3992656.png?auto=compress&cs=tinysrgb&dpr=2&w=500" alt=""/>

                        <span className="widgetsLarge\_\_name">Jackie Jan</span>

                    </td>

                    <td className="widgetsLarge\_\_date">27 Sep 2021</td>

                    <td className="widgetsLarge\_\_date">$122.00</td>

                    <td className="widgetsLarge\_\_date">

                        <Button type="Approved"/>

                    </td>

                </tr>

            </table>

        </div>

    )

}

export default WidgetsLarge

Its CSS with different or separate class will be:

.widgetsLarge\_\_button{

    padding: 5px 7px;

    border: none;

    border-radius: 10px;

}

.widgetsLarge\_\_button.Approved{

    background-color: #e5faf2;

    color: #3bb077;

}

.widgetsLarge\_\_button.Declined{

    background-color: #fff0f1;

    color: #d95087;

}

.widgetsLarge\_\_button.Pending{

    background-color: #ebf1fe;

    color: #2a7ade;

}

To limit text with in area or div or body:

Used overflow-wrap: anywhere;

            <div className="post\_\_body">

                <p>{message}</p>

            </div>

.post\_\_body{

    overflow-wrap: anywhere;

}

Firebase for authentication and db:

import firebase from 'firebase';

const firebaseConfig = {

  apiKey: "AIzaSyDUdfLCHFAHvs9sc5h7O-jJfJ0P8n1U5BY",

  authDomain: "linkedin-311e5.firebaseapp.com",

  projectId: "linkedin-311e5",

  storageBucket: "linkedin-311e5.appspot.com",

  messagingSenderId: "650316104831",

  appId: "1:650316104831:web:55793589eabf36d1b16b4a"

};

// Initialize Firebase

const firebaseApp = firebase.initializeApp(firebaseConfig);

const db = firebaseApp.firestore();

const auth = firebase.auth();

export {db, auth};

Firebase update and collection, adding data to firebase:

Adding data to collection

    const sendPost = (e) => {

        e.preventDefault()

        db.collection('posts').add({

            name: "Muneeb",

            description: "test",

            message: input,

            photourl: "",

            timestamp: firebase.firestore.FieldValue.serverTimestamp(),

        })

    }

Getting and updating data

    useEffect(()=>{

        db.collection('posts').onSnapshot(snapshot=> {

            setPosts(snapshot.docs.map(doc=>(

                {

                    id: doc.id,

                    data: doc.data()

                }

            )))

        })

    }, [])

Now using data from collection

            {posts.map( ({ id, data: { name, description, message, photourl }} )=>{

                <Post

                    key= {id}

                    name={name}

                    description={description}

                    message={message}

                    photourl={photourl}

                />

            })}

Redux:

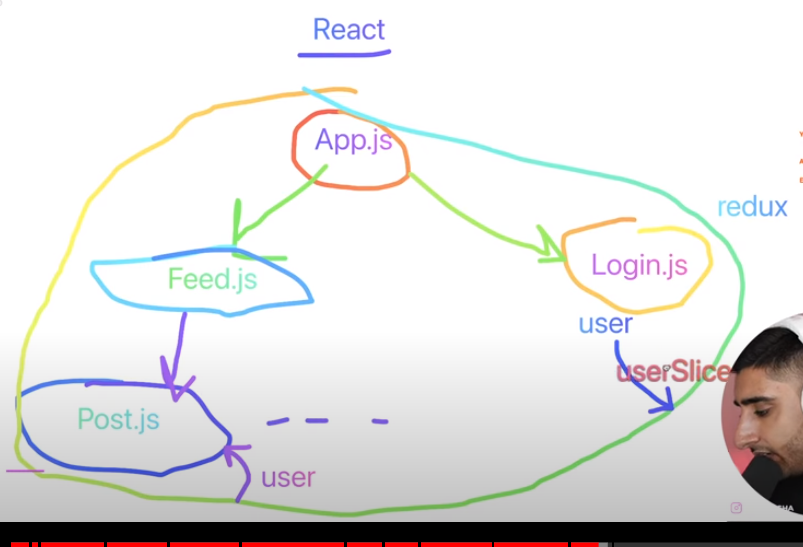


Image css:

.login > img {

    object-fit: contain;

    height: 70px;

    margin-top: 20px;

    margin-bottom: 20px;

}

Signup or Register PAGE:

Repo URL: https://github.com/MUHAMMAD-MUNEEB-WASEEM/linkedin-Social-media.git

REDUX:

Store.js

import { configureStore } from '@reduxjs/toolkit';

import userReducer from './userSlice';

export const store = configureStore({

  reducer: {

    user: userReducer,

  },

});

userSlice.js

import { createSlice } from '@reduxjs/toolkit';

export const userSlice = createSlice({

  name: 'user',

  initialState:{

    user: null

  },

  reducers: {

    login: (state, action) => {

      state.user = action.payload;

    },

    logout: (state) => {

      state.user = null;

    },

  },

});

export const { login, logout} = userSlice.actions;

export const selectCount = (state) => state.user.user;

export default userSlice.reducer;

firebase.js:

import firebase from 'firebase';

const firebaseConfig = {

    apiKey: "AIzaSyDUdfLCHFAHvs9sc5h7O-jJfJ0P8n1U5BY",

    authDomain: "linkedin-311e5.firebaseapp.com",

    projectId: "linkedin-311e5",

    storageBucket: "linkedin-311e5.appspot.com",

    messagingSenderId: "650316104831",

    appId: "1:650316104831:web:55793589eabf36d1b16b4a"

  };

const firebaseApp = firebase.initializeApp(firebaseConfig);

const db = firebaseApp.firestore();

const auth = firebase.auth();

export {auth, db}

App.js

import React, { useEffect } from 'react';

import { useDispatch, useSelector } from 'react-redux';

import './App.css';

import Feed from './components/Feed/Feed';

import Header from './components/Header/Header';

import Login from './components/Login/Login';

import Sidebar from './components/Sidebar/Sidebar';

import { auth } from './firebase';

import { login, logout } from './redux/userSlice';

function App() {

  const dispatch = useDispatch()

  const user = useSelector((state) => state.user.user)

  //to maintain login functionality on page refresh

  useEffect(()=>{

    auth.onAuthStateChanged(userAuth=>{

      if (userAuth){

        //user logged in

        dispatch(login({

          email: userAuth.email,

          uid: userAuth.uid,

          displayName: userAuth.displayName,

          photourl: userAuth.photoURL

        }))

      } else {

        //user is logged out

        dispatch(logout());

      }

    })

  },[])

  return (

    <div className="app">

      <Header/>

      {/\*if user is not present, then show login page8\*/}

      {!user ? (

        <Login/>

      ):(

      <div className="app\_\_body">

        <Sidebar/>

        <Feed/>

      </div>

      )}

    </div>

  );

}

export default App;

Login component:

Login.js

import React, {useState} from 'react'

import { useDispatch } from 'react-redux'

import { auth } from '../../firebase'

import { login } from '../../redux/userSlice';

import './Login.css'

function Login() {

    const dispatch = useDispatch();

     const [email, setEmail] = useState('')

     const [password, setPassword] = useState('')

     const [name, setName] = useState('')

     const [profilePic, setProfilePic] = useState('')

    const register = (e) => {

        if (!name){

            return alert('Please enter a full name')

        }

        auth.createUserWithEmailAndPassword(email, password)

            .then((userAuth)=>{

                userAuth.user.updateProfile({

                    displayName: name,

                    photoURL: profilePic

                })

                .then(()=>{

                    dispatch(login({

                        email: userAuth.user.email,

                        uid: userAuth.user.uid,

                        displayName: name,

                        photourl: profilePic

                    }))

                })

            })

            .catch(error =>{

                alert(error)

            })

    }

    //values taken from redux stored from signup

    const loginToApp = (e) => {

        e.preventDefault();

        auth.signInWithEmailAndPassword(email, password)

            .then(userAuth => {

                dispatch(login({

                    email: userAuth.user.email,

                    uid: userAuth.user.uid,

                    displayName: userAuth.user.displayName,

                    photourl: userAuth.user.photoURL

                }))

            }).catch(error=>{

                alert(error)

            })

    }

    return (

        <div className="login">

            <img src="https://www.seekpng.com/png/full/802-8024287\_advertise-on-linkedin-transparent-linkedin-logo-vector.png" alt="login"/>

            <form>

                <input value={name} onChange={(e)=>setName(e.target.value)} type="text" placeholder="Full name (required if registering)"/>

                <input value={profilePic} onChange={(e)=>setProfilePic(e.target.value)} type="text" placeholder="Profile pic URL (optional)"/>

                <input value={email} onChange={(e)=>setEmail(e.target.value)} type="email" placeholder="Email"/>

                <input value={password} onChange={(e)=>setPassword(e.target.value)} type="password" placeholder="Password"/>

                <button type="submit" onClick={loginToApp}>Sign In</button>

            </form>

            <p>Not a member?{" "}

                <span className="login\_\_register" onClick={register}>Register Now</span>

            </p>

        </div>

    )

}

export default Login

We can find all user in firebase authentication users section

Login.css:

.login{

    display: grid;

    place-items: center;

    padding-top: 10px;

    padding-bottom: 100px;

    margin-left: auto;

    margin-right: auto;

}

.login > img {

    object-fit: contain;

    height: 70px;

    margin-top: 20px;

    margin-bottom: 20px;

}

.login > form {

    display: flex;

    flex-direction: column;

}

.login > form > input {

    width: 350px;

    height: 50px;

    font-size: 20px;

    padding-left: 10px;

    margin-bottom: 10px;

    border-radius: 5px;

}

.login > form > button {

    width: 365px;

    height: 50px;

    font-size: large;

    background-color: #0074b1;

    border-radius: 5px;

    color: #ffffff;

}

.login\_\_register {

    color: #0177b7;

    cursor: pointer;

}

For logout:

import React from 'react'

import './Header.css'

import logo from '../images/linkedin.png'

//material ui

import SearchIcon from '@material-ui/icons/Search';

import HeaderOptions from './HeaderOptions/HeaderOptions';

import HomeIcon from '@material-ui/icons/Home';

import SupervisorAccountIcon from '@material-ui/icons/SupervisorAccount';

import BusinessCenterIcon from '@material-ui/icons/BusinessCenter';

import ChatIcon from '@material-ui/icons/Chat';

import NotificationsIcon from '@material-ui/icons/Notifications';

import { useDispatch, useSelector } from 'react-redux';

import { logout } from '../../redux/userSlice';

import { auth } from '../../firebase';

function Header() {

    const dispatch = useDispatch();

    const logoutToApp = ()=>{

        dispatch(logout())

        auth.signOut()

    }

    return (

        <div className="header">

            <div className="header\_\_left">

                <img src={logo} alt=""/>

                <div className="header\_\_search">

                    <SearchIcon/>

                    <input placeholder="Search" type="text"/>

                </div>

            </div>

            <div className="header\_\_right">

                <HeaderOptions Icon={HomeIcon} title="Home" />

                <HeaderOptions Icon={SupervisorAccountIcon} title="My Network" />

                <HeaderOptions Icon={BusinessCenterIcon} title="Jobs" />

                <HeaderOptions Icon={ChatIcon} title="Messaging" />

                <HeaderOptions Icon={NotificationsIcon} title="Notification" />

                <HeaderOptions

                    avatar={true}

                    onClick = {logoutToApp}

                    title="me"/>

            </div>

        </div>

    )

}

export default Header

HeaderOptions.js:

import React from 'react'

import './HeaderOptions.css';

import { Avatar } from '@material-ui/core';

import { useSelector } from 'react-redux';

function HeaderOptions({Icon, title, avatar, onClick}) {

    const user = useSelector(state => state.user.user);

    return (

        <div onClick={onClick} className="headerOption">

            {Icon && <Icon className="headerOption\_\_icon"/>}

            {avatar &&

                (user?.photourl ?

                 (<Avatar className="headerOption\_\_icon" src={user?.photourl}/>)

                 :

                 (<Avatar className="headerOption\_\_icon" >{user?.email[0]}</Avatar>)

                 )}

            <h3 className="headerOption\_\_title">{title}</h3>

        </div>

    )

}

export default HeaderOptions

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Login and logout part ends here \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

User true false parameters, avatar, image:

First method

<HeaderOptions

                    avatar={true}

                    onClick = {logoutToApp}

                    title="me"/>

import React from 'react'

import './HeaderOptions.css';

import { Avatar } from '@material-ui/core';

import { useSelector } from 'react-redux';

function HeaderOptions({Icon, title, avatar, onClick}) {

    const user = useSelector(state => state.user.user);

    return (

        <div onClick={onClick} className="headerOption">

            {Icon && <Icon className="headerOption\_\_icon"/>}

            {avatar &&

                (user?.photourl ?

                 (<Avatar className="headerOption\_\_icon" src={user?.photourl}/>)

                 :

                 (<Avatar className="headerOption\_\_icon" >{user?.email[0]}</Avatar>)

                 )}

            <h3 className="headerOption\_\_title">{title}</h3>

        </div>

    )

}

export default HeaderOptions

Another Example:

           <div className="sidebar\_\_top">

                <img src="https://images.unsplash.com/photo-1560345573-9f453083c335?ixlib=rb-1.2.1&ixid=MXwxMjA3fDB8MHxleHBsb3JlLWZlZWR8NjR8fHxlbnwwfHx8&auto=format&fit=crop&w=500&q=60" alt=""/>

                {user?.photourl ?

                 (<Avatar className="headerOption\_\_icon" src={user?.photourl}/>)

                 :

                 (<Avatar className="headerOption\_\_icon" >{user?.email[0]}</Avatar>)

                 }

                <h2>{user.displayName}</h2>

                <h4>{user.email}</h4>

            </div>

Second Method:

 <Avatar src={photourl} alt="photo">{name[0]}</Avatar>

If photourl exist then show that other wise show first letter of name

Icon and text inside button:

<button className="play">

                        <PlayArrow />

                        <span>Play</span>

                    </button>

Absolute and relative Position:

Top bottom left right used with position absolute. One thing to remember is that parent should always be position relative of position absolute component

To make things in Center in position absolute:

top: 0;

left: 0;

margin: auto (optional)

Video in React:

const trailer = "https://player.vimeo.com/external/371433846.sd.mp4?s=236da2f3c0fd273d2c6d9a064f3ae35579b2bbdf&profile\_id=139&oauth2\_token\_id=57447761"

<video src={trailer} autoPlay={true} loop/>

Second way:

<video

                className="video"

                autoPlay

                progress

                controls

                src="https://vod-progressive.akamaized.net/exp=1624452918~acl=%2Fvimeo-prod-skyfire-std-us%2F01%2F2400%2F14%2F362003850%2F1486625955.mp4~hmac=d6f829e7bb83f1ee6a28047d00aa2c1083c8fe5036c8084a4adf1c3903085856/vimeo-prod-skyfire-std-us/01/2400/14/362003850/1486625955.mp4"

             />

It includes loading progress component with pause play and volume controls

useRef() hook:

Taking input using useRef():

const [email, setEmail] = useState('')

    const [password, setPassword] = useState('')

    const emailRef = useRef();

    const passwordRef = useRef();

    const handleStart = () => {

        setEmail(emailRef.current.value)

    }

    const handleFinish = () => {

        setEmail(passwordRef.current.value)

    }

{!email ? (

                        <div className="input">

                            <input type="email" placeholder="email address" ref={emailRef}/>

                            <button onClick={handleStart} className="register\_\_button">Get started</button>

                         </div>

                    ): (

                        <form className="input">

                            <input type=”password" placeholder="password" ref={passwordRef}/>

                            <button onClick={handleFinish} className="register\_\_button">Start</button>

                        </form>

                    )

                }

To select any container or html element, we use useRef hook and pass ref={ref} in that element. Later we can call that ref and do whatever we want

const listRef = useRef()

 <div className="container" ref={listRef}>

Now using listRef to do work with container

const handleClick = (direction) => {

        let distance = listRef.current.getBoundingClientRect().x - 50;

        if(direction === "left"){

            listRef.current.style.transform = `translateX(${230 + distance}px)`

        }

        if(direction === "right"){

            listRef.current.style.transform = `translateX(${-230 + distance}px)`

        }

        console.log(distance)

    }

.current is very important

Slider Code:

import { ArrowBackIosOutlined, ArrowForwardIosOutlined } from '@material-ui/icons'

import React from 'react'

import { useState } from 'react'

import { useRef } from 'react'

import ListItem from '../listItem/ListItem'

import './List.scss'

function List() {

    const [slideNumber, setSlideNumber] = useState(0)

    const [isMoved, setIsMoved] = useState(false)

    const listRef = useRef()

    const handleClick = (direction) => {

        setIsMoved(true)

        let distance = listRef.current.getBoundingClientRect().x - 50;

        if(direction === "left" && slideNumber > 0){

            setSlideNumber(slideNumber - 1)

            listRef.current.style.transform = `translateX(${230 + distance}px)`

        }

        if(direction === "right" && slideNumber < 5){

            setSlideNumber(slideNumber + 1)

            listRef.current.style.transform = `translateX(${-230 + distance}px)`

        }

        console.log(distance)

    }

    return (

        <div className="list">

            <span className="list\_\_title">Continue to watch</span>

            <div className="wrapper">

                <ArrowBackIosOutlined

                    className="sliderArrow left"

                    onClick={()=>handleClick("left")}

                    style={{display: !isMoved && 'none'}}

                />

                <div className="container" ref={listRef}>

                    <ListItem/>

                    <ListItem/>

                    <ListItem/>

                    <ListItem/>

                    <ListItem/>

                    <ListItem/>

                    <ListItem/>

                    <ListItem/>

                    <ListItem/>

                    <ListItem/>

                </div>

                <ArrowForwardIosOutlined

                    className="sliderArrow right"

                    onClick={()=>handleClick("right")}

                />

            </div>

        </div>

    )

}

export default List

Scss:

.list{

    width: 100%;

    margin-top: 10px;

    .list\_\_title{

        color: white;

        font-size: 20px;

        font-weight: 500;

        margin-left: 50px;

    }

    .wrapper{

        position: relative;

        .container{

            margin-left: 50px;

            display: flex;

            width: max-content;

            margin-top: 10px;

            transform: translateX(0px);

            transition: all 1s ease;

        }

        .sliderArrow{

            width: 50px;

            height: 100%;

            background-color: rgb(22, 22, 22, 0.5);

            color: white;

            position: absolute;

            z-index: 99;

            top: 0;

            bottom: 0;

            margin: auto;

            cursor: pointer;

            &.left{

                left: 0;

            }

            &.right{

                right: 0;

            }

        }

    }

}

ListItem:

import React from 'react'

import './ListItem.scss'

function ListItem() {

    return (

        <div className="list\_\_item">

            Item

        </div>

    )

}

export default ListItem

Scss:

.list\_\_item{

    width: 225px;

    height: 120px;

    background-color: white;

    margin-right: 5px;

}

Hover Container or element:

function ListItem({index}) {

    const [isHovered, setIsHovered] = useState(false);

    const trailer = "https://player.vimeo.com/external/371433846.sd.mp4?s=236da2f3c0fd273d2c6d9a064f3ae35579b2bbdf&profile\_id=139&oauth2\_token\_id=57447761"

    return (

        <div className="list\_\_item"

            style={{left: isHovered && index \* 225 - 50 + index \* 2.5}}

            onMouseEnter={()=>setIsHovered(true)}

            onMouseLeave={()=>setIsHovered(false)}

        >

            <img src="https://occ-0-1723-92.1.nflxso.net/dnm/api/v6/X194eJsgWBDE2aQbaNdmCXGUP-Y/AAAABU7D36jL6KiLG1xI8Xg\_cZK-hYQj1L8yRxbQuB0rcLCnAk8AhEK5EM83QI71bRHUm0qOYxonD88gaThgDaPu7NuUfRg.jpg?r=4ee" alt="" />

            {isHovered && (

            <>

            <video src={trailer} autoPlay={true} loop/>

            <div className="item\_\_info">

                <div className="icons">

                    <PlayArrow className="icon" />

                    <Add className="icon" />

                    <ThumbUpAltOutlined className="icon" />

                    <ThumbDownOutlined className="icon" />

                </div>

                <div className="itemInfo\_\_top">

                    <span>1 hour 14 min</span>

                    <span className="limit">+16</span>

                    <span>1999</span>

                </div>

                <div className="desc">

                    Lorem ipsum dolor, sit amet consectetur adipisicing elit.

                    Praesentium hic rem eveniet error possimus, neque ex doloribus.

                </div>

                <div className="gener">Action</div>

            </div>

            </>

            )}

        </div>

    )

}

Navbar with changing color on scrolling:

If no scroll then transparent color and when scroll then navbar color changes to black or any other color, done using useState and passing className based on condition

import { ArrowDropDown, Notifications, Search } from '@material-ui/icons';

import React from 'react';

import { useState } from 'react';

import './Navbar.scss'

function Navbar() {

    const [isScroll, setIsScroll] = useState(false);

    // console.log(window.scrollY)

    window.onscroll =  () => {

        setIsScroll(window.scrollY === 0 ? false : true)

        return () => (window.onscroll = null)

    }

    // console.log(isScroll)

    return (

        <div className={isScroll ? 'navbar scrolled' : 'navbar'}>

            <div className="container">

                <div className="left">

                    <img

                        src="https://upload.wikimedia.org/wikipedia/commons/thumb/0/08/Netflix\_2015\_logo.svg/2560px-Netflix\_2015\_logo.svg.png"

                        alt=""

                    />

                    <span>Homepage</span>

                    <span>Series</span>

                    <span>Movies</span>

                    <span>New and Popular</span>

                    <span>My list</span>

                </div>

                <div className="right">

                    <Search className="icon" />

                    <span>KID</span>

                    <Notifications className="icon" />

                    <img

                    src="https://images.pexels.com/photos/6899260/pexels-photo-6899260.jpeg?auto=compress&cs=tinysrgb&dpr=2&w=500"

                    alt=""

                    />

                    <div className="profile">

                        <ArrowDropDown className="icon" />

                        <div className="options">

                           <span>Settings</span>

                            <span>Logout</span>

                        </div>

                    </div>

                </div>

            </div>

        </div>

    )

}

export default Navbar

SCSS:

.navbar{

    color: white;

    font-size: 14px;

    position: fixed;

    top: 0;

    width: 100%;

    z-index: 999;

    background: linear-gradient(to top, transparent 0%, rgb(0, 0, 0, 0.3) 50%);

    &.scrolled{

        background-color: var(--main-color);

    }

    .container{

        padding: 0px 50px;

        display: flex;

        align-items: center;

        justify-content: space-between;

        height: 70px;

        img{

            height: 25px;

        }

        .left{

            display: flex;

            align-items: center;

            img{

                height: 25px;

                margin-right: 40px;

            }

            span{

                margin-right: 20px;

                cursor: pointer;

            }

        }

        .right{

            display: flex;

            align-items: center;

            img{

                width: 30px;

                height: 30px;

                border-radius: 5px;

                object-fit: cover;

                cursor: pointer;

            }

            .icon{

                margin: 0 15px;

                cursor: pointer;

            }

            .profile{

                .options{

                    display: none;

                    background-color: var(--main-color);

                    border-radius: 5px;

                    span{

                        padding: 10px;

                        cursor: pointer;

                    }

                }

                &:hover{

                    .options{

                        display: flex;

                        flex-direction: column;

                        position: absolute;

                    }

                }

            }

        }

    }

}

SCSS:

Netflix Repo Link: <https://github.com/MUHAMMAD-MUNEEB-WASEEM/netflix-react.git>

Creating variable in app.scss to use in whole app where needed

//accessible in all files

:root{

  --main-color: #0b0b0b

}

Now using in home.scss

.home{

    background-color: var(--main-color);

}

Call same level classes or same level element and class in scss:

Same level Classes:

<div className={isScroll ? 'navbar scrolled' : 'navbar'}>

.navbar{

    color: white;

    font-size: 14px;

    position: fixed;

    top: 0;

    width: 100%;

    z-index: 999;

    background: linear-gradient(to top, transparent 0%, rgb(0, 0, 0, 0.3) 50%);

    &.scrolled{

        background-color: var(--main-color);

    }

Same level Element and class:

<button className="play">

                        <PlayArrow />

                        <span>Play</span>

                    </button>

                    <button className="more">

                        <InfoOutlined />

                        <span>Info</span>

                    </button>

button{

                padding: 10px 20px;

                border: none;

                border-radius: 5px;

                display: flex;

                align-items: center;

                justify-content: center;

                font-size: 18px;

                font-weight: 500;

                margin-right: 10px;

                cursor: pointer;

                &.play{

                    background-color: white;

                    color: var(--main-color)

                }

                &.more{

                    background-color: gray;

                    color: white;

                }

                span{

                    margin-left: 5px;

                }

NEXT JS AND TAILWIND CSS:

Creating app:

npx create-next-app -e with-tailwindcss instagram-next-tailwind

main:

npx create-next-app --example with-tailwindcss

Starting app:

In development

Yarn dev or npm run dev

Main

Yarn start or npm run start

Dependencies:

Yarn add faker (to get fake data for stories)

yarn add @tailwindcss/forms

yarn add @heroicons/react

yarn add tailwind-scrollbar

yarn add tailwind-scrollbar-hide

yarn add next-auth@beta

yarn add firebase

yarn add recoil

yarn add @headlessui/react

yarn add react-moment (to get time like about 10 mins ago etc on comments or post)

Changing Head:

import Head from 'next/head'

export default function Home() {

  return (

    <div className="flex flex-col items-center justify-center min-h-screen py-2">

      <Head>

        <title>Instagram</title>

        <link rel="icon" href="/favicon.ico" />

      </Head>

      <h1>This is the instagram</h1>

    </div>

  )

}

next.config.js:

Server for middle ground

Tailwind:

Inline styling for responsiveness

In tailwind, it is always mobile first

Example:

We made first image hidden by default (keeping mobile in mind), it will only visible in larger screen

Second div visible on small screen and hidden on lg screen

<div>

            <div className="flex justify-between bg-white max-w-6xl">

                 {/\*Left\*/}

                <div className="relative hidden lg:inline-grid h-24 w-24 cursor-pointer">

                    <Image

                        src="https://links.papareact.com/ocw"

                        layout="fill"

                        objectFit="contain"

                    />

                </div>

                <div className="relative w-10 h-10 lg:hidden flex-shrink-0 cursor-pointer">

                    <Image

                        src="https://links.papareact.com/jjm"

                        layout="fill"

                        objectFit="contain"

                    />

                </div>

                {/\*Middle\*/}

                {/\*Right\*/}

            </div>

        </div>

Tailwind Icon:

heroIcons.com

npm install @heroicons/react

To use input field and forms:

yarn add @tailwindcss/forms

then update plugins in tailwindcss.config.js

module.exports = {

  mode: 'jit',

  purge: ['./pages/\*\*/\*.{js,ts,jsx,tsx}', './components/\*\*/\*.{js,ts,jsx,tsx}'],

  darkMode: false, // or 'media' or 'class'

  theme: {

    extend: {},

  },

  variants: {

    extend: {},

  },

  plugins: [require('@tailwindcss/forms')],

}

Custom Classes in tailwind:

* Create styles folder in root, and then create globals.css file in it
* Creating navBtn class

@tailwind base;

@tailwind components;

@tailwind utilities;

@layer components {

    .navBtn {

        @apply

        hidden h-6

        md:inline-flex

        cursor-pointer

        hover:scale-125

        transition-all

        duration-150

        ease-out

    }

}

Update \_app.js

import '../styles/globals.css'

function MyApp({ Component, pageProps }) {

  return <Component {...pageProps} />

}

export default MyApp

Using Custom Class

       <div className="flex items-center justify-end space-x-4">

                    <HomeIcon className="navBtn"/>

                    <MenuIcon className="h-6 md:hidden cursor-pointer"/>

                    <PaperAirplaneIcon className="navBtn"/>

                </div>

Getting Fake Data from faker

import faker from 'faker'

import { useEffect, useState } from "react"

import Story from './Story'

function Stories() {

    const [suggestions, setSuggestions] = useState([])

    useEffect(()=>{

        //here suggestions is dictionary of fake data

        const suggestions = [...Array(20)].map((\_,i)=>(

            {

                ...faker.helpers.contextualCard(),

                id:i

            }

        ))

        console.log(suggestions)

        setSuggestions(suggestions)

    }, [])

    return (

        <div>

            {suggestions.map((profile)=>(

                <Story

                key={profile.id}

                img={profile.avatar}

                username={profile.username}

                />

            ))}

        </div>

    )

}

export default Stories

To Use pixels or specific value in tailwind

className="h-14 w-14 rounded-full p-[1.5px]

NEXT AUTH:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*START\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Under Api folder:

Create auth folder and create […nextauth.js] file:

import NextAuth from "next-auth"

import GoogleProvider from "next-auth/providers/google"

export default NextAuth({

  // Configure one or more authentication providers

  providers: [

    GoogleProvider({

      clientId: process.env.GOOGLE\_CLIENT\_ID,

      clientSecret: process.env.GOOGLE\_CLIENT\_SECRET,

    }),

    // ...add more providers here

  ],

})

Firebase.js file at root:

// Import the functions you need from the SDKs you need

import { initializeApp, getApp, getApps } from "firebase/app";

import { getStorage } from 'firebase/storage'

// TODO: Add SDKs for Firebase products that you want to use

// https://firebase.google.com/docs/web/setup#available-libraries

// Your web app's Firebase configuration

const firebaseConfig = {

  apiKey: "AIzaSyDv\_izhE9oWcdFQ4qjo9u9uXaXepyIJGC8",

  authDomain: "instagram-4bfb7.firebaseapp.com",

  projectId: "instagram-4bfb7",

  storageBucket: "instagram-4bfb7.appspot.com",

  messagingSenderId: "1088998849319",

  appId: "1:1088998849319:web:7cc26389f0dc3a6a428702"

};

// Initialize Firebase

const app = !getApps().length ? initializeApp(firebaseConfig) : getApp();

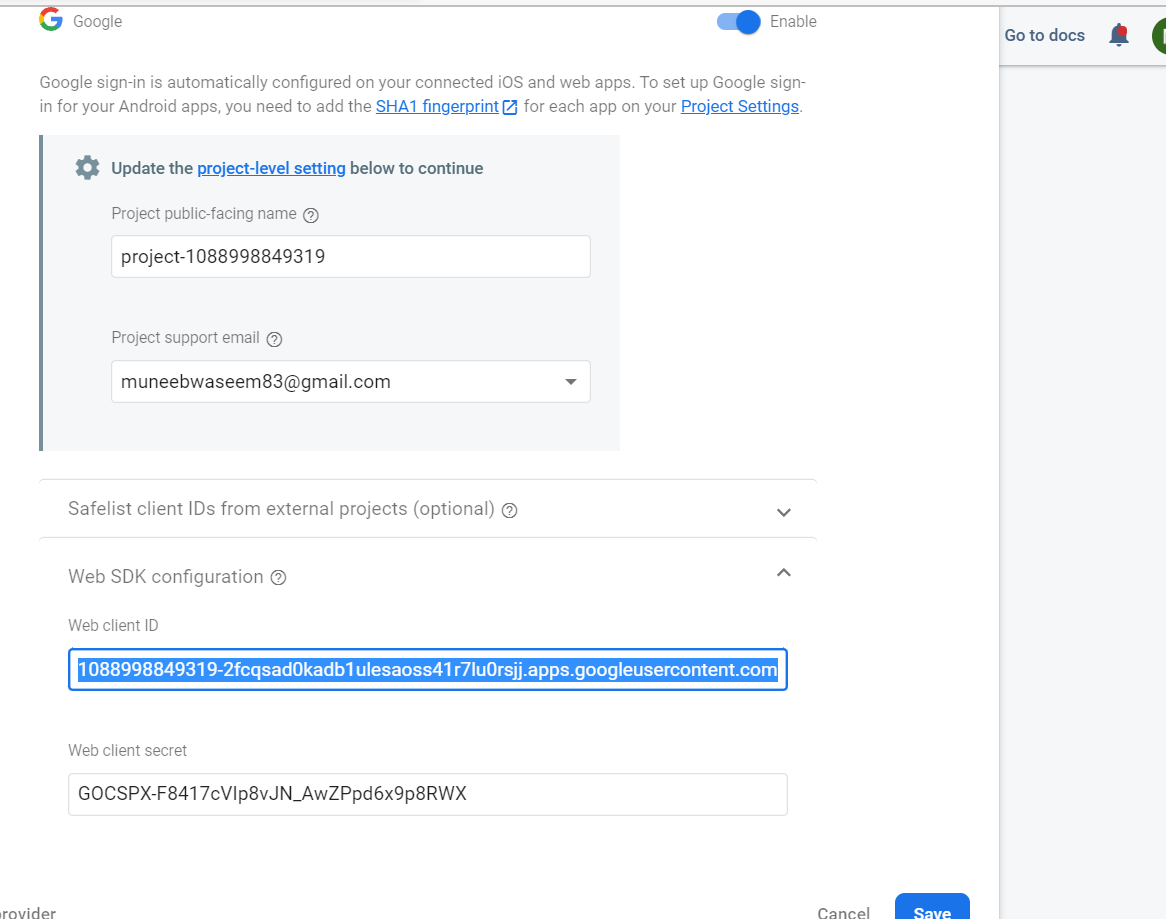
const db = getFirestore();

const storage = getStorage();

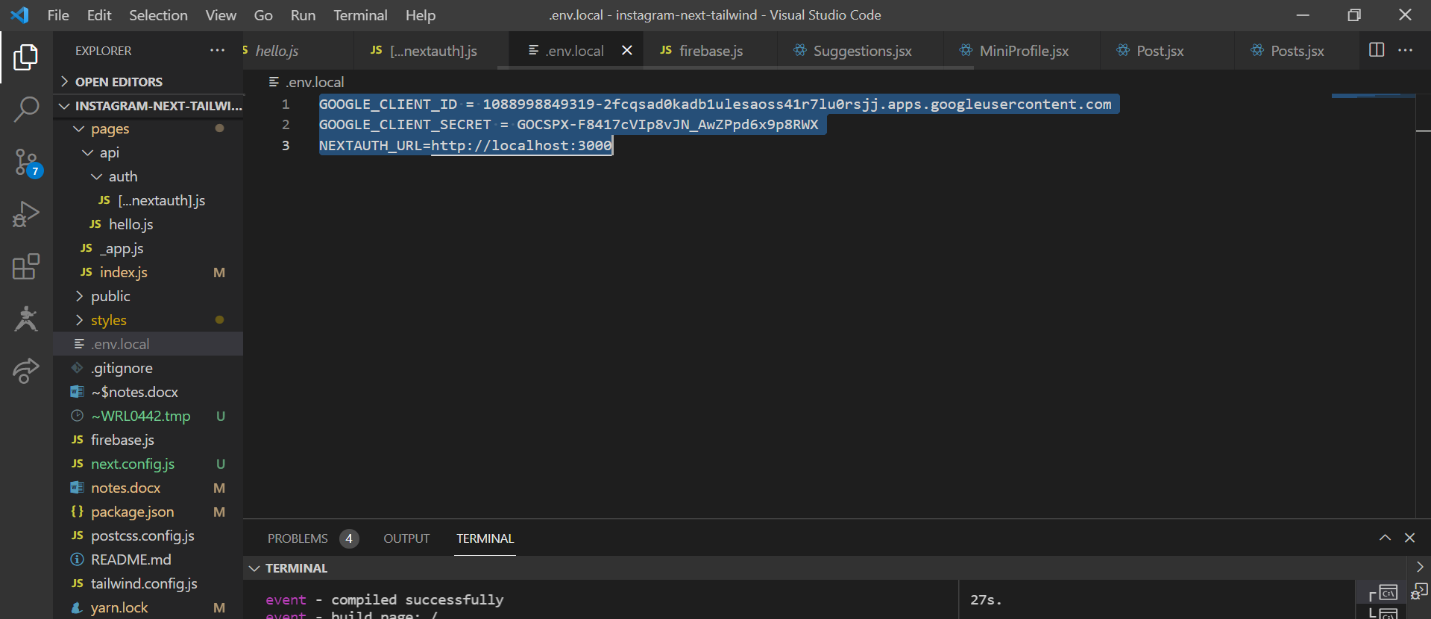
export { db, app, storage }

Then go to firebase, setup app, then go to authentication, select google, enable it, write email, and then save

Now to use client id and client secret in next auth, click on web sdk configuration after writing email on google provide on firebase and copy values from their



Now create .env.local file at root and pass client id and secret and auth url



Update nextauth.js file to have signin page:

1. Default next signin page code:

export default NextAuth({

  // Configure one or more authentication providers

  providers: [

    GoogleProvider({

      clientId: process.env.GOOGLE\_CLIENT\_ID,

      clientSecret: process.env.GOOGLE\_CLIENT\_SECRET,

    }),

    // ...add more providers here

  ],

  //code to get by default login page from auth

  theme:{

    logo: "http://links.papareact.com/sq0",

    brandColor: "#F13287",

    colorScheme: "auto"

  }

})

1. Customize signin page:

import NextAuth from "next-auth"

import GoogleProvider from "next-auth/providers/google"

import MiniProfile from "../../../components/Feed/FeedComponents/MiniProfile/MiniProfile";

export default NextAuth({

  // Configure one or more authentication providers

  providers: [

    GoogleProvider({

      clientId: process.env.GOOGLE\_CLIENT\_ID,

      clientSecret: process.env.GOOGLE\_CLIENT\_SECRET,

    }),

    // ...add more providers here

  ],

  //code to get by default login page from auth

  // theme:{

  //   logo: "http://links.papareact.com/sq0",

  //   brandColor: "#F13287",

  //   colorScheme: "auto"

  // }

  pages:{

    signIn: '/auth/signin',

  },

  callbacks:{

    async session({session, token, user}){

      session.user.username = session.user.name

      .split(' ')

      .join("")

      .toLocaleLowerCase();MiniProfile

      //Muhammad Muneeb Waseem = muhammadmuneebwaseem

      session.user.uid = token.sub;

      return session;

    }

  }

})

Creating auth folder and signin.js file as shown in above link under pages folder:

import { getProviders, signIn as SignIntoProvider } from 'next-auth/react'

import Header from '../../components/Header/Header';

//Browsers... or client

function signIn({providers}) {

    return (

    <>

        <Header/>

        <div className="flex flex-col items-center justify-center min-h-screen py-2 -mt-20 px-14 text-center">

            <img className="w-80" src="https://links.papareact.com/ocw" alt=""/>

            <p className="font-xs italic">This is not a REAL  app, it built for practice and eductational purposes only</p>

            <div className="mt-40">

                {Object.values(providers).map((provider) => (

                    <div key={provider.name}>

                    <button

                    className="p-3 bg-blue-500 rounded-lg text-white"

                    onClick={() => SignIntoProvider(provider.id, {callbackUrl: '/'})}>

                        Sign in with {provider.name}

                    </button>

                    </div>

                ))}

            </div>

        </div>

    </>

    )

}

//Server Side rendering

export async function getServerSideProps(){

    const providers = await getProviders();

    return {

        props: {

            providers

        }

    }

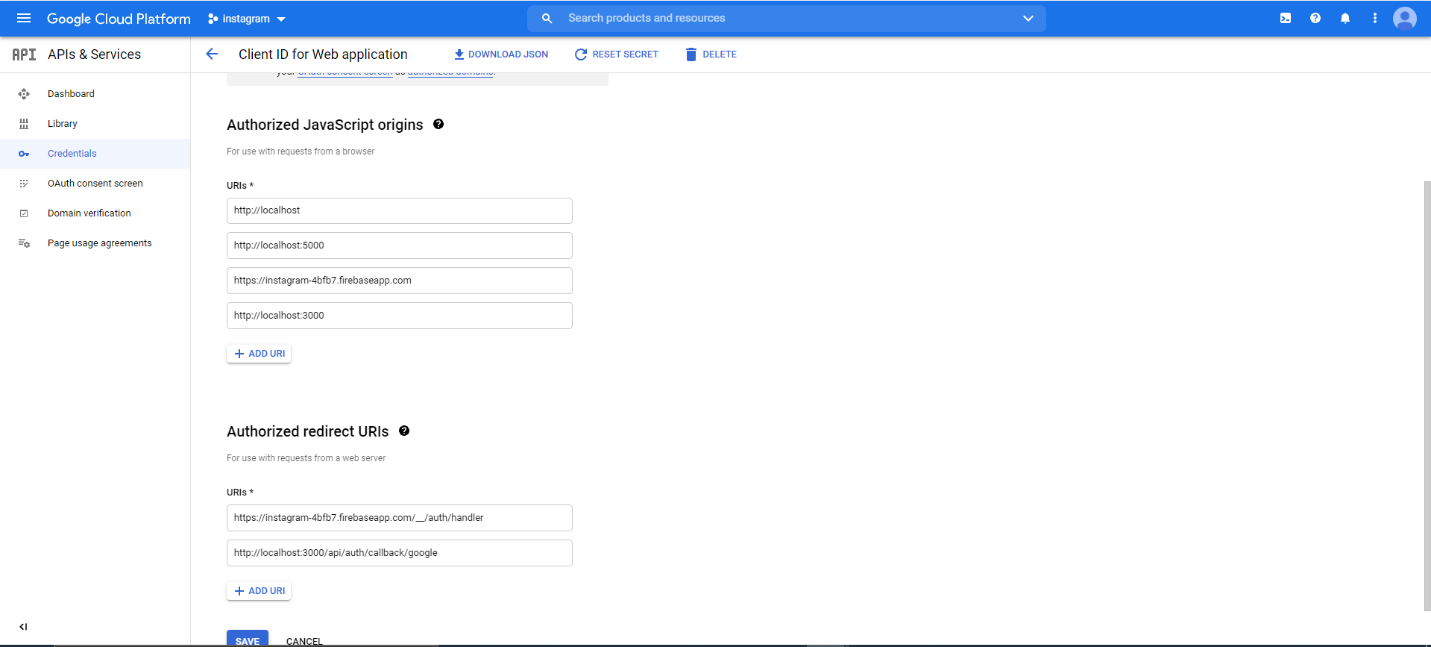
}

export default signIn

Now go to below link with the same google account as use for registering app on firebase

[Getting started – instagram – Google Cloud Platform](https://console.cloud.google.com/apis/credentials?authuser=6&project=instagram-4bfb7)

Go to oAuth 2.0 webclient and click on pencil icon to update it

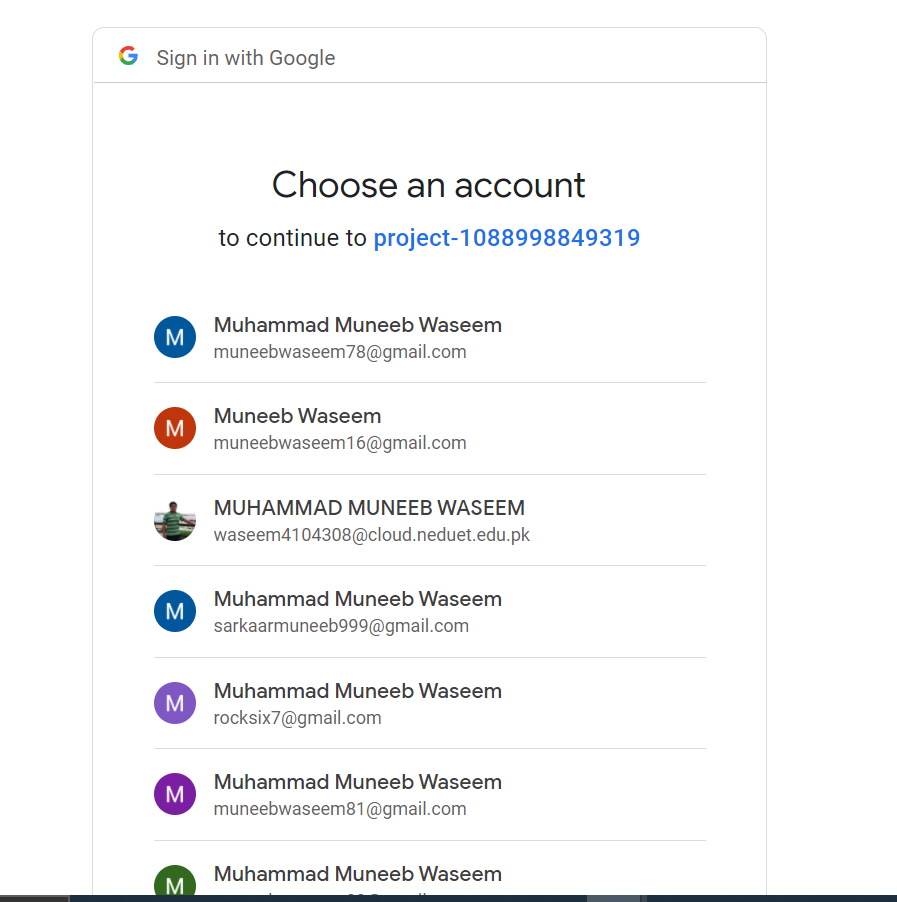


This update is for localhost, when we deploy on somehosting or vercel, we need to update localhost3000 links to that link

Now we got authorize

Sigin Link:

localhost:3000/auth/signin



Update app.js:

import '../styles/globals.css'

import { SessionaProvider } from 'next-auth/react'

function MyApp({ Component, pageProps: {session, ...pageProps} }) {

  return (

    <SessionaProvider session={session}>

      <Component {...pageProps} />

    </SessionaProvider>

  )

}

export default MyApp

Now to signout or show username image etc from session:

Const {data:session} = useSesssion() -> very imp

import { signOut, useSession } from 'next-auth/react';

function MiniProfile() {

    const { data: session } = useSession()

    console.log(session)

    return (

        <div className="flex items-center justify-between mt-14 ml-10">

            <img

             src={session?.user?.image}

             alt=""

             className="w-16 h-16 rounded-full border p-[2px]"

             />

             <div className="flex-1 mx-4">

                 <h2 className="font-bold">{session?.user?.username}</h2>

                 <h3 className="text-sm text-gray-400">Welcome to Instagram</h3>

             </div>

             <button

             className="text-blue-400 text-sm font-semibold"

             onClick={signOut}>

                Sign Out

            </button>

        </div>

    )

}

export default MiniProfile

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Next Routing:

const router = useRouter();

* First Example:

<div onClick={()=>router.push('/')}  className="relative w-10 lg:hidden flex-shrink-0 cursor-pointer">

                    <Image

                        src="https://links.papareact.com/jjm"

                        layout="fill"

                        objectFit="contain"

                    />

                </div>

* Second Example:

 <HomeIcon onClick={()=>router.push('/')} className="navBtn"/>

Recoil:

Uploading image on clicking + icon at header:

It changes in global store, so we require global store. One option is redux and other one is recoil

Recoil is fast, and very light weight and by default a global store

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* MAIN RECOIL START \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Surrounding the app.js as global store

<RecoilRoot>

        <Component {...pageProps} />

      </RecoilRoot>

1. Atom file just like slice in redux (create atom folder and root and modalAtom.js file in it):

import { atom } from "recoil";

export const modalState = atom({

    key: 'modalState',

    default: false,

})

1. Getting or updating data in global store using recoil

/getting and updating data in global store using recoil

    const [open, setOpen] = useRecoilState(modalState);

    //To read only from global store using recoil

    const open = useRecoilValue(modalState)

In redux, we need to use useSelector for getting data from store, and useDispatch to update data in store, but in recoil it is very simple just like simple useState

* Now using setOpen function to update state in the global store

 <PlusCircleIcon onClick={()=>setOpen(true)} className="navBtn"/>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* MAIN RECOIL END \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Now till above main recoil end, we have put logic in plus icon when someone click on it, open value becomes true and this value is stored in the global store. Now we do this to show some component when open gets true. When open gets true, we show them post upload component which Modal component in this case. So I am creating this component right now which only visible when open is true or when person click on plus icon

Update index.js to render modal component

import Head from 'next/head'

import Feed from '../components/Feed/Feed'

import Header from '../components/Header/Header'

import Modal from '../components/Modal/Modal'

export default function Home() {

  return (

    <div className="bg-gray-50 h-screen overflow-y-scroll scrollbar-auto scrollbar-hide">

      <Head>

        <title>Instagram</title>

        <link rel="icon" href="/favicon.ico" />

      </Head>

      <Header/>

      <Feed/>

      {/\*Modal\*/}

      <Modal/>

    </div>

  )

}

Creating Modal folder and Modal.js file init and using useRecoilState to get or update the data

Basic:

import { useRecoilState } from "recoil"

import { modalState } from "../../atoms/modalAtome"

function Modal() {

    const [open, setOpen] = useRecoilState(modalState)

    return (

        <div>

            {open && (

                <p>I am modal</p>

            )}

        </div>

    )

}

export default Modal

Complete Component:

import { Fragment, useCallback, useRef, useState } from 'react';

import { useRecoilState } from 'recoil';

import { Dialog, Transition } from '@headlessui/react';

import {

  addDoc,

  collection,

  serverTimestamp,

  updateDoc,

  doc,

} from '@firebase/firestore';

import { ref, uploadString, getDownloadURL } from '@firebase/storage';

import { modalState } from '../../atoms/modalAtome';

import { CameraIcon } from '@heroicons/react/outline';

import { db, storage } from '../../firebase';

import { useSession } from 'next-auth/react';

function Modal() {

  const { data: session } = useSession();

  const [open, setOpen] = useRecoilState(modalState);

  const filePickerRef = useRef(null);

  const captionRef = useRef(null);

  const [selectedFile, setSelectedFile] = useState(null);

  const [loading, setLoading] = useState(false);

  const uploadPost = useCallback(async () => {

    if (loading) return;

    setLoading(true);

    // 1)create a post and add to firestore 'posts' collection

    // 2) get the post Id for the newly created post

    // 3) upload the image to firebase storage with the post Id

    // 4) get a download URL from fb storage and update the oriiginal post iwht image

    const post = await addDoc(collection(db, 'posts'), {

      username: session.user?.username,

      caption: captionRef.current.value,

      profileImg: session.user?.image,

      timestamp: serverTimestamp(),

    });

    const imageRef = ref(storage, `posts/${post.id}/image`);

    await uploadString(imageRef, selectedFile, 'data\_url').then(

      async (snapshot) => {

        const downloadUrl = await getDownloadURL(imageRef);

        await updateDoc(doc(db, 'posts', post.id), { image: downloadUrl });

      }

    );

    setLoading(false);

    setOpen(false);

    setSelectedFile(null);

  }, [selectedFile, session]);

  const addImageToPost = useCallback((e) => {

    const reader = new FileReader();

    if (e.target.files[0]) {

      reader.readAsDataURL(e.target.files[0]);

    }

    reader.onload = (readerEvent) => {

      setSelectedFile(readerEvent.target.result);

    };

  }, []);

  if (!open) return null;

  return (

    <Transition.Root show={open} as={Fragment}>

      <Dialog

        as='div'

        className='fixed z-10 inset-0 overflow-y-auto'

        onClose={setOpen}

      >

        <div className='flex items-center justify-center min-h-[800px] sm:min-h-screen pt-4 px-4 pb-20 text-center sm:block sm:p-0'>

          <Transition.Child

            as={Fragment}

            enter='ease-out duration-200'

            enterFrom='opacity-0'

            enterTo='opacity-100'

            leave='ease-in duration-200'

            leaveFrom='opacity-100'

            leaveTo='opacity-0'

          >

            <Dialog.Overlay className='fixed inset-0 bg-gray-500 bg-opacity-75 transition-opacity' />

          </Transition.Child>

          {/\* trick to the browser into centering modal contents. \*/}

          <span className='hidden sm:inline-block sm:align-middle sm:h-screen'>

            &#8203;

          </span>

          <Transition.Child

            as={Fragment}

            enter='ease-out duration-200'

            enterFrom='opacity-0 translate-y-4 sm:translate-y-0 sm:scale-95'

            enterTo='opacity-100 translate-y-0 sm:scale-100'

            leave='ease-in duration-200'

            leaveFrom='opacity-100 translate-y-0 sm:scale-100'

            leaveTo='opacity-0 translate-y-4 sm:translate-y-0 sm:scale-95'

          >

            <div className='inline-block align-bottom bg-white rounded-lg px-4 pt-5 pb-4 text-left overflow-hidden shadow-xl transform transition-all sm:my-8 sm:align-middle sm:max-w-sm sm:w-full sm:p-6'>

              <div>

                {selectedFile ? (

                  <img

                    src={selectedFile}

                    onClick={() => setSelectedFile(null)}

                    alt=''

                  />

                ) : (

                  <div

                    onClick={() => filePickerRef.current.click()}

                    className='mx-auto flex items-center justify-center h-12 w-12 rounded-full bg-red-100 cursor-pointer'

                  >

                    <CameraIcon

                      className='h-6 w-6 text-red-600'

                      aria-hidden='true'

                    />

                  </div>

                )}

                <div>

                  <div className='mt-3 text-center sm:mt-5'>

                    <Dialog.Title

                      as='h3'

                      className='text-lg leading-6 font-medium text-gray-900'

                    >

                      Upload a photo

                    </Dialog.Title>

                    <div>

                      <input

                        ref={filePickerRef}

                        type='file'

                        hidden

                        onChange={addImageToPost}

                      />

                    </div>

                    <div>

                      <input

                        className='border-none focus:ring-0 w-full text-center'

                        ref={captionRef}

                        type='text'

                        placeholder='Please enter a caption...'

                      />

                    </div>

                  </div>

                </div>

                <div className='mt-5 sm:mt-6'>

                  <button

                    type='button'

                    disabled={!selectedFile}

                    className='inline-flex justify-center w-full rounded-md border border-transparent shadow-sm px-4 py-2 bg-red-600 text-base font-medium text-white hover:bg-red-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-red-500 sm:text-sm disabled:bg-gray-300 disabled:cursor-not-allowed disabled:hover:bg-gray-300'

                    onClick={uploadPost}

                  >

                    {loading ? 'Uploading...' : 'Upload Post'}

                  </button>

                </div>

              </div>

            </div>

          </Transition.Child>

        </div>

      </Dialog>

    </Transition.Root>

  );

}

export default Modal;

Adding data with image in firebase/Post data to firebase:

We use firebase firestore as db to store our text related data like posts. And we use firebase storage as storage to store images related to that post as can be seen in firebase.js file

Steps:

// 1)create a post and add to firestore 'posts' collection

    // 2) get the post Id for the newly created post

    // 3) upload the image to firebase storage with the post Id

    // 4) get a download URL from fb storage and update the oriiginal post iwht image

1. Create a post and add to firestore posts collection

const post = await addDoc(collection(db, 'posts'), {

      username: session.user?.username,

      caption: captionRef.current.value,

      profileImg: session.user?.image,

      timestamp: serverTimestamp(),

    });

* 1. Get the post Id for the newly created post and upload image to firebase storage:

const imageRef = ref(storage, `posts/${post.id}/image`);

1. get a download url from firebase storage and update original post with the image

await uploadString(imageRef, selectedFile, 'data\_url').then(

      async (snapshot) => {

        const downloadUrl = await getDownloadURL(imageRef);

        await updateDoc(doc(db, 'posts', post.id), { image: downloadUrl });

      }

    );

Another Example of adding text only, same as above without image:

1. Defining states to be used or update:

const [comment, setComment] = useState('')

    const [comments, setComments] = useState([])

1. Function that works onClick and add data to specified collection

const sendComment = async (e) => {

        e.preventDefault();

        const commentToSend = comment;

        setComment('');

        //adding comment to backend posts collection

        await addDoc(collection(db, 'posts', id, 'comments'), {

            comment: commentToSend,

            username:session.user.username,

            userImage: session.user.image,

            timestamp: serverTimestamp(),

        })

    }

Here, it first goes to db, then goes to posts collection, and watch id and pass comment to specified id (as per comment on specified post)

1. Send Comment method call on api

<button

                    type='submit'

                    onClick={sendComment}

                    disabled={!comment.trim()}

                    className="font-semibold text-blue-400">

                    Post

                    </button>

Conclusion:

First post goes to firestore post collection, after that image goes to firebase storage, now that image linked with the corresponding post through post.id

* Complete code (Modal.js):

import { Fragment, useCallback, useRef, useState } from 'react';

import { useRecoilState } from 'recoil';

import { Dialog, Transition } from '@headlessui/react';

import {

  addDoc,

  collection,

  serverTimestamp,

  updateDoc,

  doc,

} from '@firebase/firestore';

import { ref, uploadString, getDownloadURL } from '@firebase/storage';

import { modalState } from '../../atoms/modalAtome';

import { CameraIcon } from '@heroicons/react/outline';

import { db, storage } from '../../firebase';

import { useSession } from 'next-auth/react';

function Modal() {

  const { data: session } = useSession();

  const [open, setOpen] = useRecoilState(modalState);

  const filePickerRef = useRef(null);

  const captionRef = useRef(null);

  const [selectedFile, setSelectedFile] = useState(null);

  const [loading, setLoading] = useState(false);

  const uploadPost = useCallback(async () => {

    if (loading) return;

    setLoading(true);

    // 1)create a post and add to firestore 'posts' collection

    // 2) get the post Id for the newly created post

    // 3) upload the image to firebase storage with the post Id

    // 4) get a download URL from fb storage and update the oriiginal post iwht image

    const post = await addDoc(collection(db, 'posts'), {

      username: session.user?.username,

      caption: captionRef.current.value,

      profileImg: session.user?.image,

      timestamp: serverTimestamp(),

    });

    const imageRef = ref(storage, `posts/${post.id}/image`);

    await uploadString(imageRef, selectedFile, 'data\_url').then(

      async (snapshot) => {

        const downloadUrl = await getDownloadURL(imageRef);

        await updateDoc(doc(db, 'posts', post.id), { image: downloadUrl });

      }

    );

    setLoading(false);

    setOpen(false);

    setSelectedFile(null);

  }, [selectedFile, session]);

  const addImageToPost = useCallback((e) => {

    const reader = new FileReader();

    if (e.target.files[0]) {

      reader.readAsDataURL(e.target.files[0]);

    }

    reader.onload = (readerEvent) => {

      setSelectedFile(readerEvent.target.result);

    };

  }, []);

  if (!open) return null;

  return (

    <Transition.Root show={open} as={Fragment}>

      <Dialog

        as='div'

        className='fixed z-10 inset-0 overflow-y-auto'

        onClose={setOpen}

      >

        <div className='flex items-center justify-center min-h-[800px] sm:min-h-screen pt-4 px-4 pb-20 text-center sm:block sm:p-0'>

          <Transition.Child

            as={Fragment}

            enter='ease-out duration-200'

            enterFrom='opacity-0'

            enterTo='opacity-100'

            leave='ease-in duration-200'

            leaveFrom='opacity-100'

            leaveTo='opacity-0'

          >

            <Dialog.Overlay className='fixed inset-0 bg-gray-500 bg-opacity-75 transition-opacity' />

          </Transition.Child>

          {/\* trick to the browser into centering modal contents. \*/}

          <span className='hidden sm:inline-block sm:align-middle sm:h-screen'>

            &#8203;

          </span>

          <Transition.Child

            as={Fragment}

            enter='ease-out duration-200'

            enterFrom='opacity-0 translate-y-4 sm:translate-y-0 sm:scale-95'

            enterTo='opacity-100 translate-y-0 sm:scale-100'

            leave='ease-in duration-200'

            leaveFrom='opacity-100 translate-y-0 sm:scale-100'

            leaveTo='opacity-0 translate-y-4 sm:translate-y-0 sm:scale-95'

          >

            <div className='inline-block align-bottom bg-white rounded-lg px-4 pt-5 pb-4 text-left overflow-hidden shadow-xl transform transition-all sm:my-8 sm:align-middle sm:max-w-sm sm:w-full sm:p-6'>

              <div>

                {selectedFile ? (

                  <img

                    src={selectedFile}

                    onClick={() => setSelectedFile(null)}

                    alt=''

                  />

                ) : (

                  <div

                    onClick={() => filePickerRef.current.click()}

                    className='mx-auto flex items-center justify-center h-12 w-12 rounded-full bg-red-100 cursor-pointer'

                  >

                    <CameraIcon

                      className='h-6 w-6 text-red-600'

                      aria-hidden='true'

                    />

                  </div>

                )}

                <div>

                  <div className='mt-3 text-center sm:mt-5'>

                    <Dialog.Title

                      as='h3'

                      className='text-lg leading-6 font-medium text-gray-900'

                    >

                      Upload a photo

                    </Dialog.Title>

                    <div>

                      <input

                        ref={filePickerRef}

                        type='file'

                        hidden

                        onChange={addImageToPost}

                      />

                    </div>

                    <div>

                      <input

                        className='border-none focus:ring-0 w-full text-center'

                        ref={captionRef}

                        type='text'

                        placeholder='Please enter a caption...'

                      />

                    </div>

                  </div>

                </div>

                <div className='mt-5 sm:mt-6'>

                  <button

                    type='button'

                    disabled={!selectedFile}

                    className='inline-flex justify-center w-full rounded-md border border-transparent shadow-sm px-4 py-2 bg-red-600 text-base font-medium text-white hover:bg-red-700 focus:outline-none focus:ring-2 focus:ring-offset-2 focus:ring-red-500 sm:text-sm disabled:bg-gray-300 disabled:cursor-not-allowed disabled:hover:bg-gray-300'

                    onClick={uploadPost}

                  >

                    {loading ? 'Uploading...' : 'Upload Post'}

                  </button>

                </div>

              </div>

            </div>

          </Transition.Child>

        </div>

      </Dialog>

    </Transition.Root>

  );

}

export default Modal;

Get data from firebase:

import Post from "./Post"

import {

    collection,

    doc,

    onSnapshot,

    orderBy,

    query,

  } from '@firebase/firestore';

import { useEffect, useState } from 'react';

import { db } from '../../../../firebase';

function Posts() {

    const [posts, setPosts] = useState([]);

    useEffect(() => {

        const unsuscribe = onSnapshot(

          query(collection(db, 'posts'), orderBy('timestamp', 'desc')),

          (snapshot) => {

            setPosts(snapshot.docs);

          }

        );

        return unsuscribe; // clean up

      }, [db]);

      console.log(posts)

    return (

        <div>

        {posts.map((post)=>(

            <Post

            key={post.id}

            username={post.data().username}

            userImg={post.data().profileImg}

            id={post.id}

            img={post.data().image}

            caption={post.data().caption}

            />

        ))}

        </div>

    )

}

export default Posts

Another example of getting data, getting comments:

1. UseEffect that works on page loading and set state so it can be used where we want to show comments or any variable:

Defined state:

const [comments, setComments] = useState([])

useEffect(()=>

        onSnapshot(

            query(

                collection(db, 'posts', id, 'comments'),

                orderBy('timestamp', 'desc')),

                (snapshot)=>setComments(snapshot.docs)

        ),[db])

1. Using Comments for display purposes:

/\*Comments\*/}

            {comments.length > 0 && (

                <div className='ml-10 mt-4 h-20 overflow-y-scroll scrollbar-thumb-black scrollbar-thin'>

                    {comments.map((comment) => (

                        <div

                        key={comment.data().id}

                        className='flex items-center truncate mb-3 space-x-2'

                        >

                        {comment.data().userImage ? (

                            <img

                            src={comment.data().userImage}

                            alt='User comment image'

                            className='h-7 rounded-full'

                            />

                        ) : (

                            <span className='h-7 w-7 rounded-full bg-gray-600 text-white text-center'>

                            {comment.data().username[0].toUpperCase()}

                            </span>

                        )}

                        <p className='text-sm flex-1'>

                            <span className='font-bold mr-2'>{comment.data().username}</span>

                            {comment.data().comment}

                        </p>

                        <Moment fromNow className='pr-5 text-xs text-gray-400'>

                            {comment.data().timestamp?.toDate()}

                        </Moment>

                        </div>

                    ))}

                </div>

      )}

Disable button when input field is empty or only empty space in it:

<form className="flex items-center p-4">

                    <EmojiHappyIcon className="h-7"/>

                    <input className="flex-1 border-none focus:ring-0 outline-none"

                    type="text"

                    placeholder="Add a Comment..."

                    value={comment}

                    onChange={(e)=>setComment(e.target.value)}

                    type='submit'

                    />

                    <button

                    type='submit'

                    disabled={!comment.trim()}

                    className="font-semibold text-blue-400">

                    Post

                    </button>

                </form>

To get time (about mins ago):

Yarn add react-moment

import Moment from 'react-moment';

<Moment fromNow className='pr-5 text-xs text-gray-400'>

                            {comment.data().timestamp?.toDate()}

                        </Moment>

Vercel Deployment:

1. Push final code to github
2. Sign in to vercel using github, import project you want to deploy from github
3. Skip team
4. Add environment variables (if depending on deployed link, then first add localhost url, and deploy that will give you deployed url, after that take that url and update environment variable as per requirement and then redeploy again to get better result)
5. Deploy

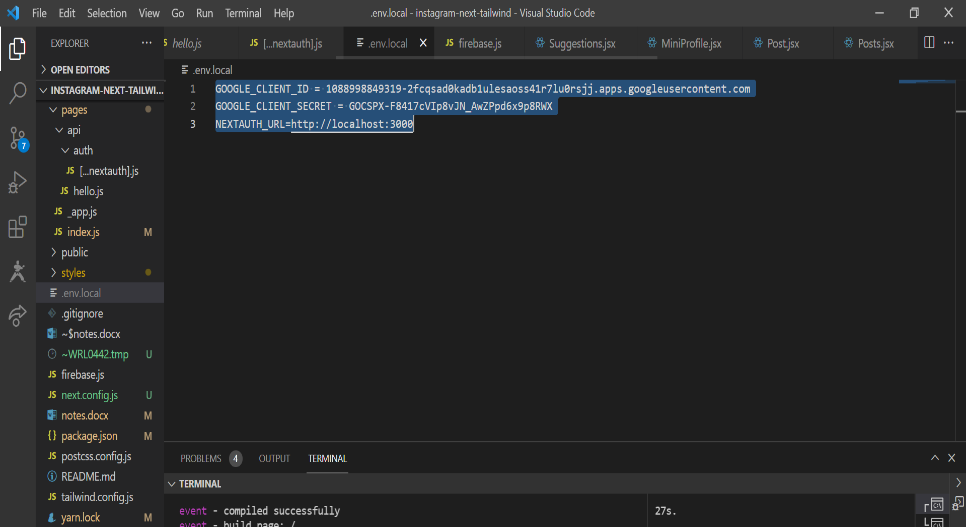
For Nextjs:

Visit next.config.js file of avocado app, paths and loader are important if not important. If using firebase, then put && next export under build key in package.json

Imp: When update environment variable, always redeploy your app from vercel to get better results

Vercel Deployment in this build:

* Steps 1-3 like above
* Pass environment variable from below pic to vercel



* Don’t worry for localhost url, after deployment, we get new vercel url, we update our NEXTAUTH\_URL with that link and then redeploy app again

Avocado Agency Website (Nextjs):

Image component:

<div className={styles.card}>

                <Image

                    src='/img/Avocado.png'

                    width="100%"

                    height="100%"

                    layout='fill'

                    objectFit="cover"

                    alt=""/>

            </div>

Layout fill will ultimately takes all space, to get provide space, make it parent component which is card relative

card{

    flex: 0.5;

    padding: 100px;

    padding-right: 0;

    display: flex;

    flex-direction: column;

    justify-content: space-around;

    position: relative; /\*for image fill\*/

}

Using Static Data in Nextjs:

import Head from 'next/head'

import Image from 'next/image'

import Footer from '../components/Footer'

import Intro from '../components/Intro'

import Navbar from '../components/Navbar'

import Services from '../components/Services'

import { data } from '../data'

import styles from '../styles/Home.module.css'

export default function Home({services}) {

  //console.log(services)

  return (

    <div>

      <Head>

        <title>Agency Website</title>

        <meta name="description" content="Made using Nextjs" />

      </Head>

      <Intro/>

      <Services services={services}/>

    </div>

  )

}

//Fetching data before rendering above components

//Below part should be done on that page where we

//want to show that data, here we show cards on services page

export const getStaticProps = async () => {

  // if api

  //  const services =  await axios.get('/', req,res,)=>{

  // }

  //Static data from file

  const services = data

  return {

    props : { services }

  }

//Now we use these props immediately in this page as shown in home function above

//Pass these services props in services components and thus we can use this in services component

}

To hover some inside class when hovering parent class:

.service:hover .media{

    transform: scale(1.1);

}

Showing text or desc when hovering over image or div:

div className={styles.service}>

                            <div className={styles.desc}>{service.desc}</div>

                            <span className={styles.cat}>

                                {service.title}

                            </span>

                            <div className={styles.media}>

                                {service.video ? (

                                    <video

                                    src={`/img/${service.video}`}

                                    autoPlay

                                    loop

                                    className={styles.video}

                                    />

                                ):(

                                    <Image

                                    src={`/img/${service.photo}`}

                                    width="100%"

                                    height="100%"

                                    layout="responsive"

                                    objectFit="cover"

                                    alt=""

                                    />

                                )}

                            </div>

                        </div>

.service{

    width: 500px;

    height: 500px;

    display: flex;

    align-items: center;

    justify-content: center;

    padding: 10px;

    margin-bottom: 50px;

    position: relative;

    background-color: whitesmoke;

}

.cat{

   position: absolute;

   top: 10px;

   left: 10px;

   color: gray;

   letter-spacing: 5px;

   font-weight: 300;

   text-transform: uppercase;

}

.service:hover .media{

    transform: scale(1.1);

}

.desc{

    position: absolute;

    width: 60%;

    text-align: center;

    z-index: 999;

    background-color: rgba(189, 183, 183, 0.575);

    border-radius: 10px;

    padding: 20px;

    opacity: 0;

    transition: all 1s ease;

}

.service:hover .desc{

    opacity: 1;

}

.media{

    width: 70%;

    height: 70%;

    border-radius: 50%;

    overflow: hidden;

    transition: all 1s ease;

}

.video{

    width: 100%;

    height: 100%;

    object-fit: cover;

}

Main point:

By default, text or description opacity will be 0, on hovering over service component, its opacity will become 1

Center Item:

With flex:

display:flex

align-items:center

justify-content: center

With absolute:

top: 0 //can also pass some value in top right left, but other remains zero

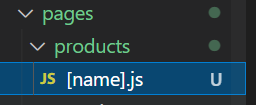
right: 0

left: 0

margin: auto

How to create subpaths

Localhost:3000/products/adasda



Fetching static data:

Using name as path

import { data } from "../../data"

import styles from '../../styles/Product.module.css'

function Product({product}) {

    return (

        <div className={styles.container}>

        </div>

    )

}

export const getStaticPaths = async () => {

    const products = data

    //we can use it as props or path under return

    //we choose name as path ->> /products/name

     const paths = products.map((item) => {

        return {

        params: { name: item.name },

        };

  });

  //name of all products has been saved in params

  //curly bracket with return as used as dict to destructure easily

  return {

      paths,

      fallback:false,

  }

}

export const getStaticProps = async (ctx) => {

    //ctx : { params: { name: item.name } },

    const name = ctx.params.name;

    const product = data.filter(item=>item.name===name)[0]; //single product

    return {

        props: { product }

    }

}

//first we select name as path

//we use that path(name) in getStaticProps to fetch single product from data

//as use that single product as props

export default Product

Real Liver NEXTJS:

<https://realliver.vercel.app/>

Create .babelrc:

|  |
| --- |
| { |
|  | "presets": ["next/babel"], |
|  | "plugins": [] |
|  | } |

Dependencies Used:

npm i

1. @chakra-ui/react @emotion/react @emotion/styled (Chakra UI pear dependencies)
2. axios
3. framer-motion
4. mollify (large numbers into human readable form)
5. nprogress (progress bar)
6. react-horizontal-scrolling-menu (for images)
7. react-icons

ChakraUi

<Flex> -> flex

<Box> -> div

ChakraUI:

It is used to avoid writing css expilicitly. To make sure it works fine, we need to wrap our app inside ChakraProvider

import '../styles/globals.css'

import Router from 'next/router'

import Head from 'next/head'

import NProgress from 'nprogress'

import { ChakraProvider} from '@chakra-ui/react'

import Layout from '../components/Layout'

function MyApp({ Component, pageProps }) {

  return(

    <>

    <Head>

    </Head>

    <ChakraProvider>

      <Layout>

          <Component {...pageProps} />

        </Layout>

    </ChakraProvider>

    </>

  )

}

export default MyApp

Using or fetching data from Api:

Create utils folder at root:

import axios from 'axios';

export const baseUrl = 'https://bayut.p.rapidapi.com'

export const fetchApi = async (url) => {

    const {data} = await axios.get(url, {

        headers: {

        'x-rapidapi-host': 'bayut.p.rapidapi.com',

        'x-rapidapi-key': '530cf11fabmsh79e60c0e05c7b69p139c05jsna753fd49ae56'

        }

    })

    return data;

}

These headers are provided by rapidApi. Now we can call fetchApi function anywhere we want in getStaticProps function (as it is nextJs) and pass url and it will fetch data and we can map on that data and extract useful information from it.

Now extracting data from fetchApi function in index.js:

At the bottom,

export async function getStaticProps () {

  const propertyForSale = await fetchApi(`${baseUrl}/properties/list?locationExternalIDs=5002&purpose=for-sale&hitsPerPage=6`);

  const propertyForRent = await fetchApi(`${baseUrl}/properties/list?locationExternalIDs=5002&purpose=for-rent&hitsPerPage=6`);

  return {

    props: {

      propertiesForSale: propertyForSale?.hits,

      propertiesForRent: propertyForRent?.hits,

    }

  }

}

Now we can use these props and map over them.

Thus index.js becomes

import Link from 'next/link';

import Image from 'next/image';

import { Flex, Box, Text, Button } from '@chakra-ui/react';

import {baseUrl, fetchApi} from '../utils/fetchApi';

import Property from '../components/Property';

export const Banner = ({purpose, title1, title2, desc1, desc2, buttonText, linkName, imageUrl}) => (

  <Flex flexWrap='wrap' justifyContent="center" alignItems='center' m='10'>

    <Image src={imageUrl} width="500" height="300" alt="banner"/>

    <Box p='5'>

      <Text color="gray.500" fontSize="sm" fontWeight="medium">{purpose}</Text>

      <Text fontSize="3xl" fontWeight="bold">{title1}<br/>{title2}</Text>

      <Text fontSize="lg" paddingTop="3" paddingBottom="3" color="gray.700">{desc1}<br/>{desc2}</Text>

      <Button fontSize="xl">

        <Link href={linkName}>{buttonText}</Link>

      </Button>

    </Box>

  </Flex>

)

export default function Home({propertiesForRent, propertiesForSale}) {

  return (

    <Box>

      <Banner

      purpose="RENT A HOME"

      title1="Rental Home for"

      title2="Everyone"

      desc1="Expolre Apartments, Villaas, Homes"

      desc2="and more"

      buttonText="Explore Renting"

      linkName="/search?purpose-for-rent"

      imageUrl="https://bayut-production.s3.eu-central-1.amazonaws.com/image/145426814/33973352624c48628e41f2ec460faba4"

      />

      <Flex flexWrap="wrap">

        {propertiesForRent.map(property => (

          <Property property={property} key={property.id}/>

        ))}

      </Flex>

       <Banner

      purpose="BUY A HOME"

      title1="Find, Buy & Own Your"

      title2="Dream Home"

      desc1="Expolre Apartments, Villaas, Homes"

      desc2="and more"

      buttonText="Explore Buying"

      linkName="/search?purpose-for-sale"

      imageUrl="https://bayut-production.s3.eu-central-1.amazonaws.com/image/110993385/6a070e8e1bae4f7d8c1429bc303d2008"

      />

      <Flex flexWrap="wrap">

        {propertiesForSale.map(property => (

            <Property property={property} key={property.id}/>

        ))}

      </Flex>

    </Box>

  )

}

export async function getStaticProps () {

  const propertyForSale = await fetchApi(`${baseUrl}/properties/list?locationExternalIDs=5002&purpose=for-sale&hitsPerPage=6`);

  const propertyForRent = await fetchApi(`${baseUrl}/properties/list?locationExternalIDs=5002&purpose=for-rent&hitsPerPage=6`);

  return {

    props: {

      propertiesForSale: propertyForSale?.hits,

      propertiesForRent: propertyForRent?.hits,

    }

  }

}

getStaticProps: fetch data at build time

getServerSideProps: fetch data live on each request

Sizes to Image tag in nextjs:

<Image

                    alt="property"

                    placeholder="blur"

                    blurDataURL={item.url}

                    src={item.url}

                    width={1000} height={500}

                    sizes="(max-width: 500px) 100px, (max-width: 1023px) 400px, 1000px"

                    />

BlurImage on while page is loading:

<Image

                    alt="property"

                    placeholder="blur"

                    blurDataURL={item.url}

                    src={item.url}

                    width={1000} height={500}

                    sizes="(max-width: 500px) 100px, (max-width: 1023px) 400px, 1000px"

                    />

Adding loading progress bar on NextJs:

Under \_app.js

import '../styles/globals.css'

import Router from 'next/router'

import Head from 'next/head'

import NProgress from 'nprogress'

import { ChakraProvider} from '@chakra-ui/react'

import Layout from '../components/Layout'

function MyApp({ Component, pageProps }) {

  NProgress.configure({ showSpinner: false})

  Router.events.on('routeChangeStart', ()=>{

    NProgress.start();

  });

  Router.events.on('routeChangeComplete', ()=>{

    NProgress.done();

  });

  return(

    <>

    <Head>

      <link rel='stylesheet' href='https://cdnjs.cloudflare.com/ajax/libs/nprogress/0.2.0/nprogress.min.css' integrity='sha512-42kB9yDlYiCEfx2xVwq0q7hT4uf26FUgSIZBK8uiaEnTdShXjwr8Ip1V4xGJMg3mHkUt9nNuTDxunHF0/EgxLQ==' crossOrigin='anonymous' referrerPolicy='no-referrer' />

    </Head>

    <ChakraProvider>

      <Layout>

          <Component {...pageProps} />

        </Layout>

    </ChakraProvider>

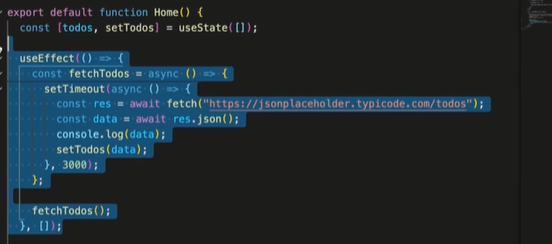
    </>

  )

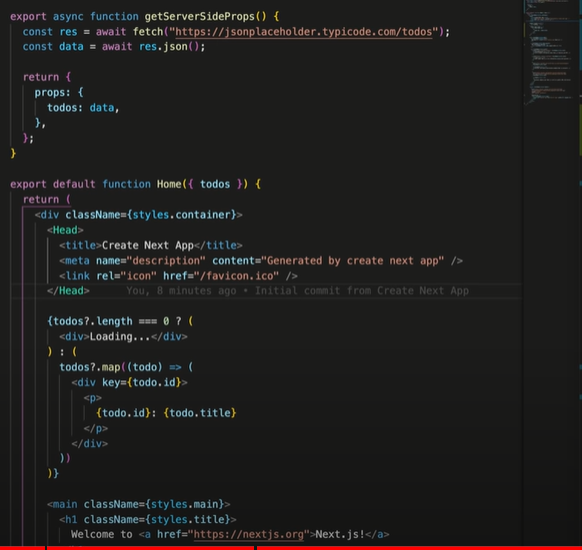
}

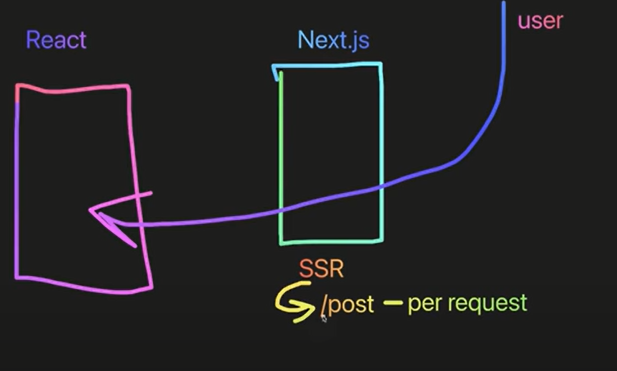
export default MyApp

setTimeout (Timeout)



getServerSideProps:

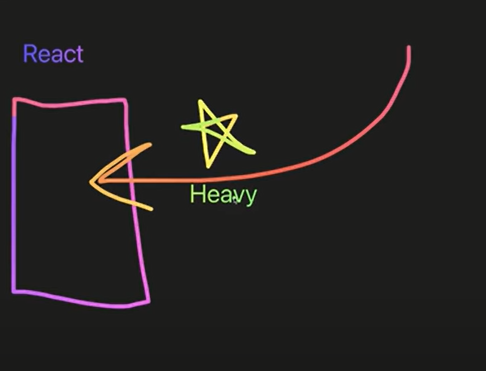




SSR is very fast. It create page in server and handle per request which is very fast whereas get static (static side rendering) is prebuild which is slower as compared to SSR

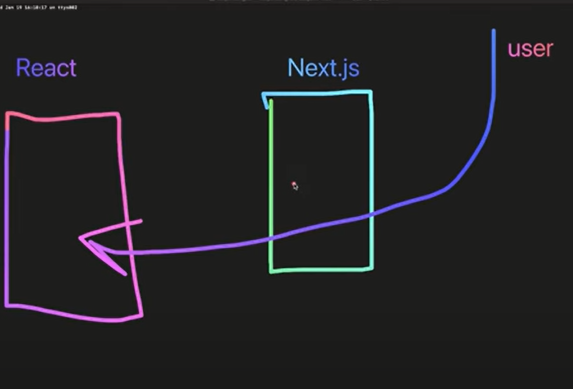
REACT VS REACT + NEXTJS:

Without NEXTJS



WITH NEXTJS

User not get complete it, instead it get pages which he need which increases speed. For this we use server side rendering (SSR)



Medium typescript tailwind sanity:

npx create-next-app --example with-tailwindcss medium-typescript-tailwind-sanity

yarn add react-hook-form

yarn add react-portable-text

yarn add @sanity/client

**For Sanity**

**npm i next-sanity**

npm install -g @sanity/cli

yarn add @sanity/image-url

[Sonny Sangha (sanity.io)](https://www.sanity.io/sonny)

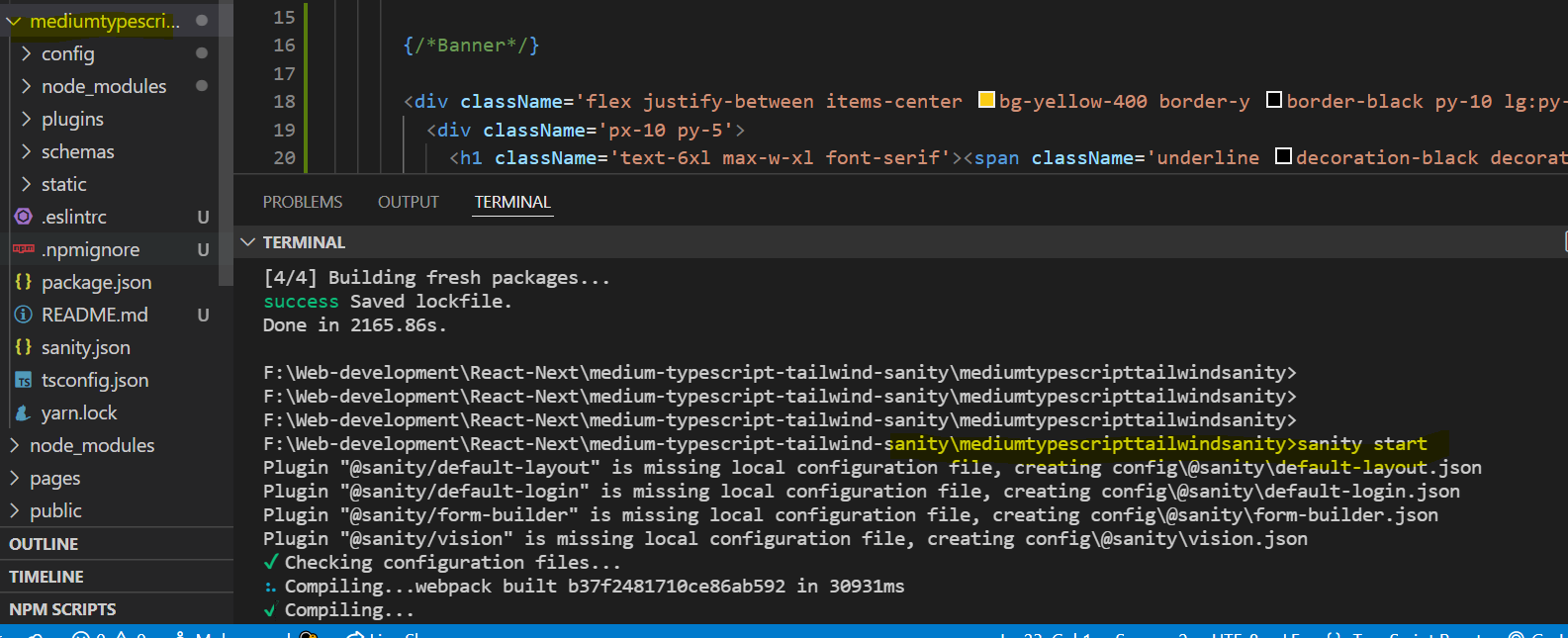
1. sanity login
2. sanity init --coupon sonny2022
3. dataset -> y
4. blog

Then go to sanity website and your backend will be created there

Later go in sanity folder as of same name as provided under project while running

sanity init --coupon sonny2022

Under cmd, switch or cd to that folder and run sanity start

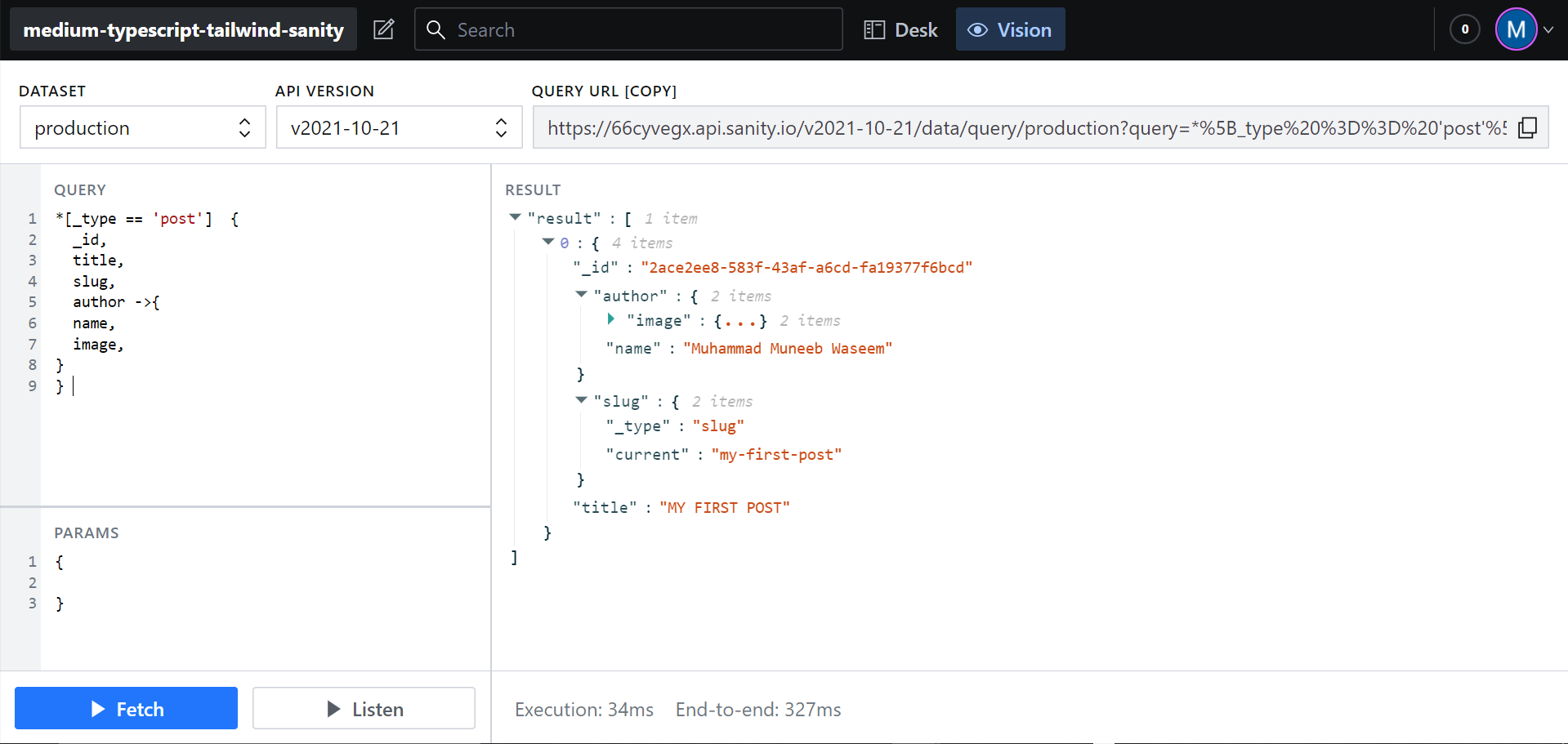


If error, then delete nodemodules under sanity named folder and yarn install and then run sanity start

Now create post in sanity

**To query data from sanity In frontend:**

It uses GROQ (graph relational object query) to query data from sanity backend api to frontend client



**In root folder:**

**npm i next-sanity**

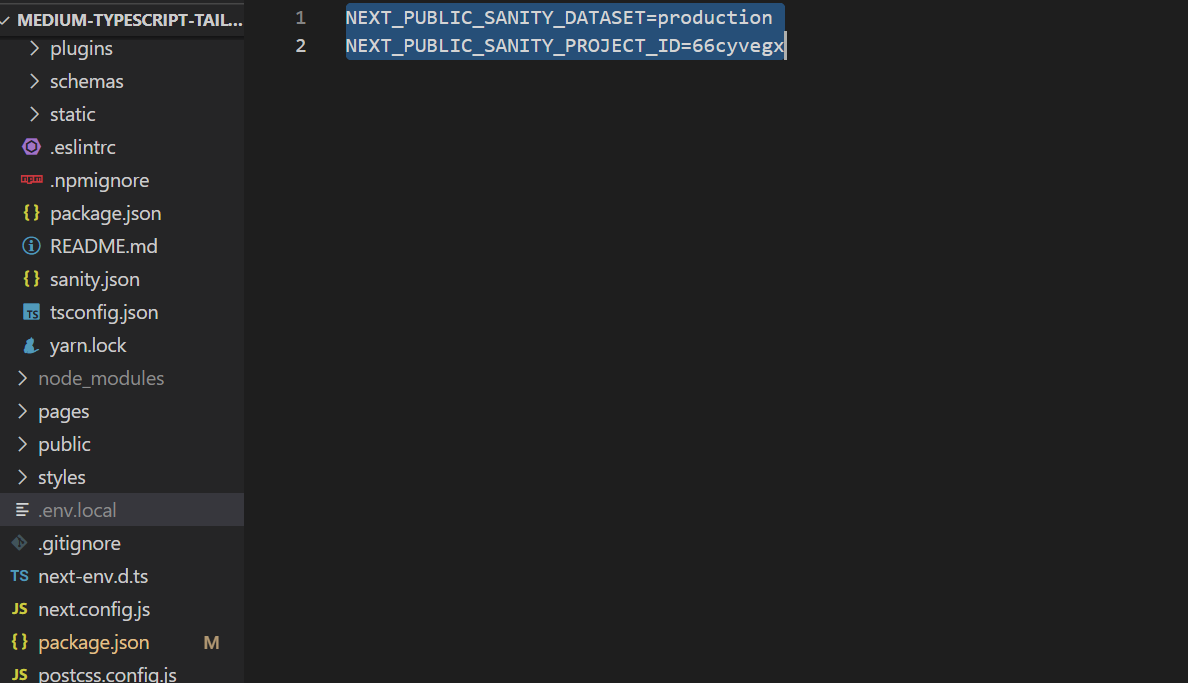
**Create sanity.js file**

|  |
| --- |
| import { |
|  | createImageUrlBuilder, |
|  | createCurrentUserHook, |
|  | createClient, |
|  | } from 'next-sanity'; |
|  |  |
|  | export const config = { |
|  | /\*\* |
|  | \* Find your project ID and dataset in `sanity.json` in your studio project. |
|  | \* These are considered “public”, but you can use environment variables |
|  | \* if you want differ between local dev and production. |
|  | \* |
|  | \* https://nextjs.org/docs/basic-features/environment-variables |
|  | \*\*/ |
|  | dataset: process.env.NEXT\_PUBLIC\_SANITY\_DATASET || 'production', |
|  | projectId: process.env.NEXT\_PUBLIC\_SANITY\_PROJECT\_ID, |
|  | apiVersion: '2021-08-11', // or today's date for latest |
|  | /\*\* |
|  | \* Set useCdn to `false` if your application require the freshest possible |
|  | \* data always (potentially slightly slower and a bit more expensive). |
|  | \* Authenticated request (like preview) will always bypass the CDN |
|  | \*\*/ |
|  | useCdn: process.env.NODE\_ENV === 'production', |
|  | }; |
|  |  |
|  | export const sanityClient = createClient(config); |
|  |  |
|  | /\*\* |
|  | \* Set up a helper function for generating Image URLs with only the asset reference data in your documents. |
|  | \* Read more: <https://www.sanity.io/docs/image-url> |
|  | \*\*/ |
|  | export const urlFor = (source) => createImageUrlBuilder(config).image(source); |
|  |  |
|  | // Helper function for using the current logged in user account |
|  | export const useCurrentUser = createCurrentUserHook(config); |

Now create .env.local file to pass projectId and dataset

NEXT\_PUBLIC\_SANITY\_DATASET=production

NEXT\_PUBLIC\_SANITY\_PROJECT\_ID=66cyvegx



These projectId and dataset can be found from sanity named folder under sanity.json file



How to create custom schema in sanity (like comment schema)?

1. Create api of comment under api folder -> /api/createComment

import type { NextApiRequest, NextApiResponse } from 'next';

import sanityClient from '@sanity/client';

export const config = {

  dataset: process.env.NEXT\_PUBLIC\_SANITY\_DATASET || 'production',

  projectId: process.env.NEXT\_PUBLIC\_SANITY\_PROJECT\_ID,

  useCdn: process.env.NODE\_ENV === 'production',

  token: process.env.SANITY\_API\_TOKEN,

};

const client = sanityClient(config);

export default async function createComment(

  req: NextApiRequest,

  res: NextApiResponse

) {

  const { \_id, name, email, comment } = JSON.parse(req.body);

  try {

    await client.create({

      \_type: 'comment',

      post: {

        \_type: 'reference',

        \_ref: \_id,

      },

      name,

      email,

      comment,

    });

  } catch (err) {

    return res.status(500).json({ message: "Couldn't submit comment", err });

  }

  res.status(200).json({ message: 'Comment submitted' });

}

1. Comment.js file in schema folder

export default {

    name: 'comment',

    type: 'document',

    title: 'Comment',

    fields: [

      {

        name: 'name',

        type: 'string',

      },

      {

        title: 'Approved',

        name: 'approved',

        type: 'boolean',

        description: "Comments won't show on the site without approval",

      },

      {

        name: 'email',

        type: 'string',

      },

      {

        name: 'comment',

        type: 'text',

      },

      {

        name: 'post',

        type: 'reference',

        to: [{ type: 'post' }],

      },

    ],

  };

1. Import comment and pass in schema.js file

// First, we must import the schema creator

import createSchema from 'part:@sanity/base/schema-creator'

// Then import schema types from any plugins that might expose them

import schemaTypes from 'all:part:@sanity/base/schema-type'

// We import object and document schemas

import blockContent from './blockContent'

import category from './category'

import post from './post'

import author from './author'

import comment from './comment'

// Then we give our schema to the builder and provide the result to Sanity

export default createSchema({

  // We name our schema

  name: 'default',

  // Then proceed to concatenate our document type

  // to the ones provided by any plugins that are installed

  types: schemaTypes.concat([

    // The following are document types which will appear

    // in the studio.

    post,

    author,

    category,

    // When added to this list, object types can be used as

    // { type: 'typename' } in other document schemas

    blockContent,

    comment,

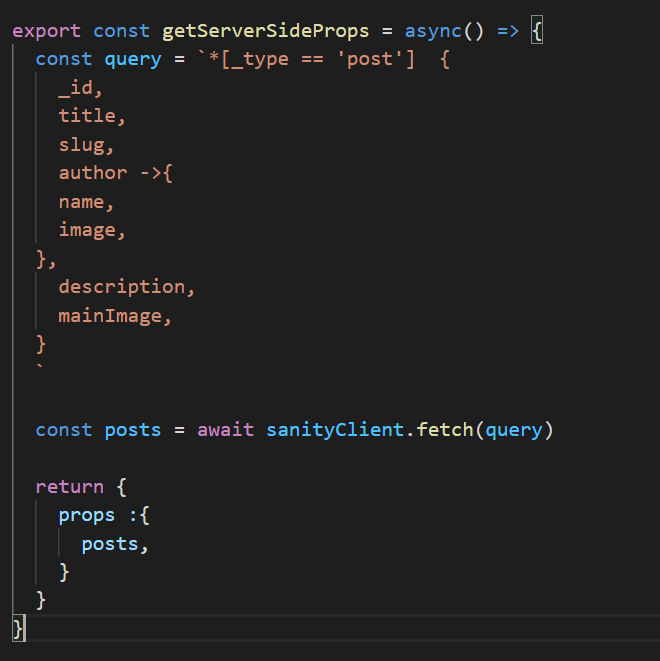
  ]),

})

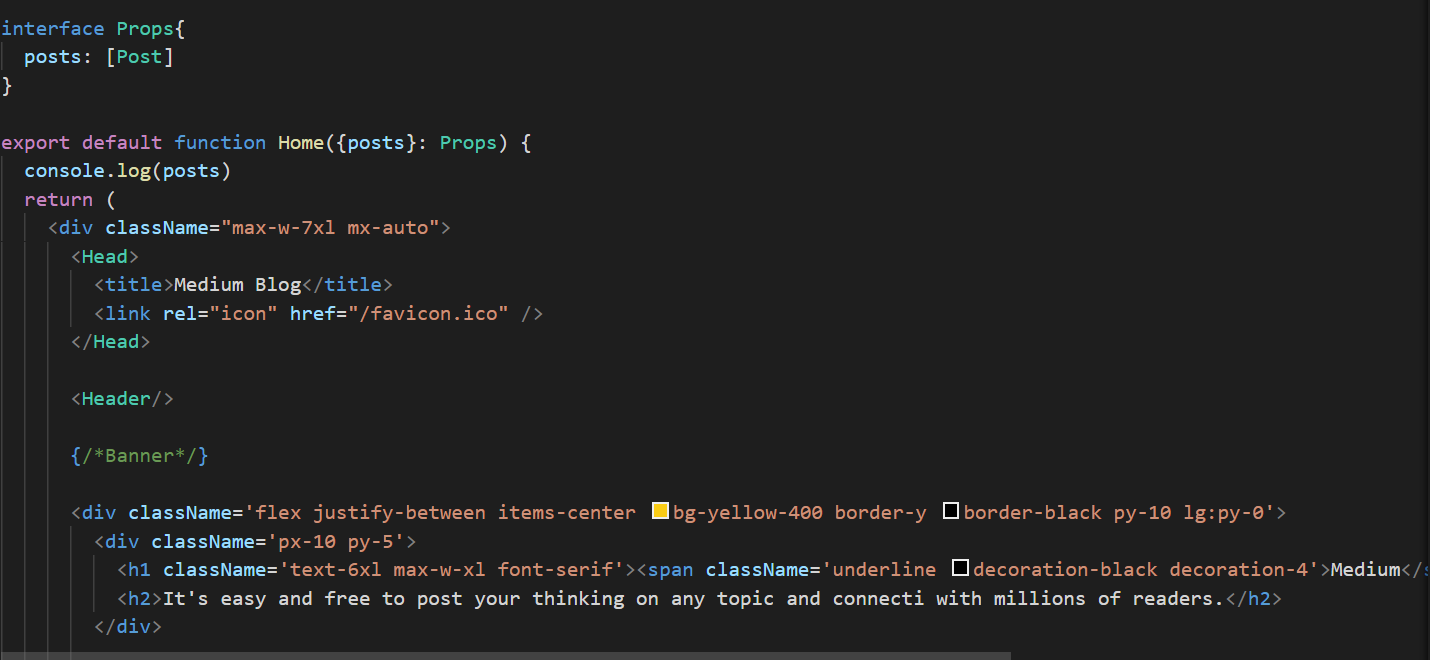
**SSR vs SSG or (server site rendering vs static site generation):**

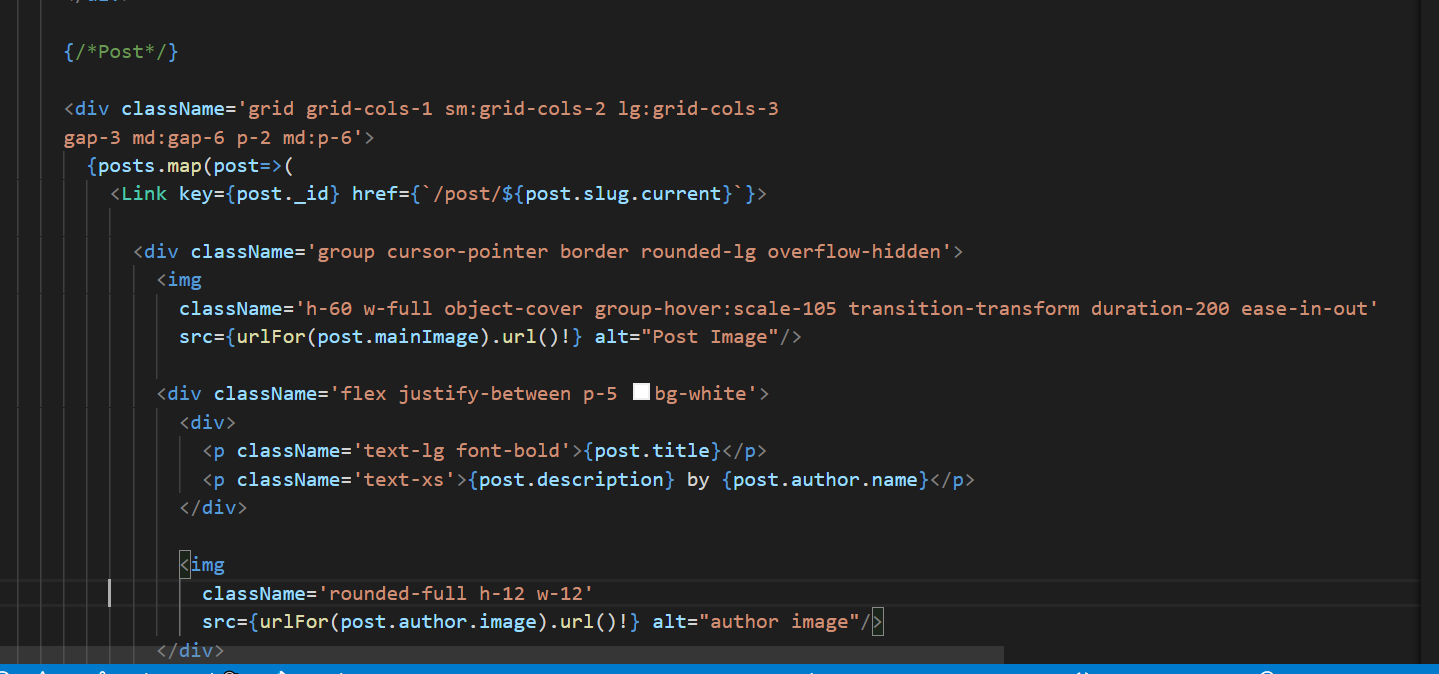
SSR:

server side rendering SSR used to show data on pages at a server which is very fast like data from api etc is fetch under SSR



It will be used here:





Static site generation (SSG):

Similary pages are pre-build as cache and update every 60s so to get dynamic data also. So for this, static side rendering is used. It is used to build route pre-build or in advance. But to update data, we used revalidate and pass time to it, this is called ISR (incremental static generation)

Example

post/my-first-post

For this, create folder name post under pages section and create file name [slug].tsx

1. getStaticPath (post/my-first-post)
2. getStaticProps now used to get data based on path, like get data of my-first-post. Thus my-first-post page gets created. It is called ISR as we have pass time under revalidate

EXAMPLE

export const getStaticPaths = async () => {

    // to make paths

    // post/my-first-post or post/slug

    //So we need to query slug

    const query = `\*[\_type == 'post']  {

        \_id,

        slug,

      }

      `

    const posts = await sanityClient.fetch(query);

    //we can use it as props or path under return

    //we choose name as path ->> /post/slug

     const paths = posts.map((post:Post) => {

        return {

        params: { slug: post.slug.current },

        };

  });

  //slug of all posts has been saved in params

  //curly bracket with return as used as dict to destructure easily

  return {

      paths,

      fallback: 'blocking',

  }

}

//first we create path for post page

//Now we need to get data on that post page to make it pre-build,

//for that we use get static props

//when to fetch post page

//use getstatic paths with getstaticprops

export const getStaticProps = async (ctx:any) => {

    //ctx : { params: { name: item.name } },

    const slug = ctx.params?.slug;

    const query = `\*[\_type == "post" && slug.current == $slug][0] {

        \_id,

        \_createdAt,

        title,

        author -> {

          name,

          image

        },

        "comments": \*[

          \_type == "comment" &&

          post.\_ref == ^.\_id &&

          approved == true

        ],

        description,

        mainImage,

        slug,

        body

      }`

      const post = await sanityClient.fetch(query, {

          slug: slug

      });

      if(!post){

          return {

              notFound:true

          }

      }

    return {

        props: {

            post

        }

revalidate: 60, //after 60 seconds, update old cache

    }

}

//first we select name as path

//we use that path(name) in getStaticProps to fetch single product from data

//as use that single product as props

Now use post in Post function as props with type PROPS