INSTAGRAM

Creating app:

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

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();

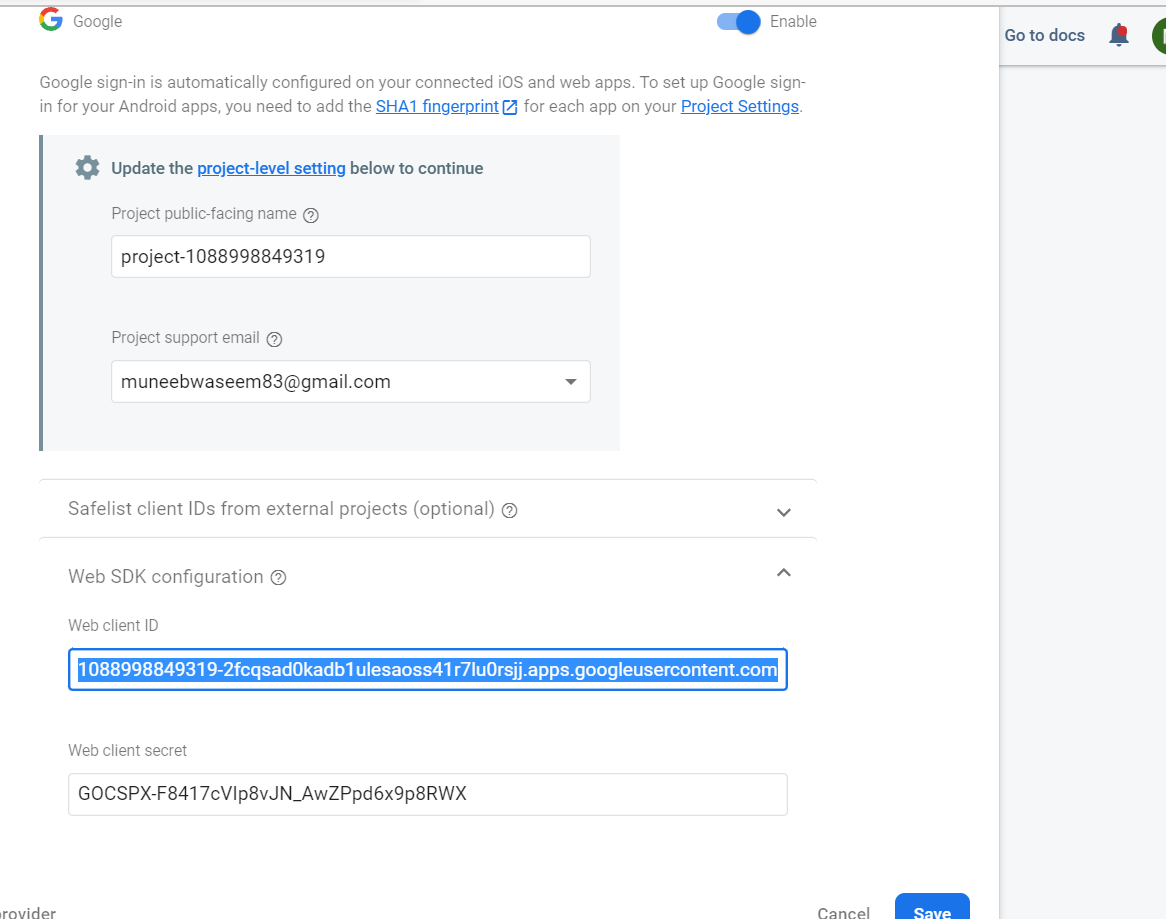
const db = getFirestore();

const storage = getStorage();

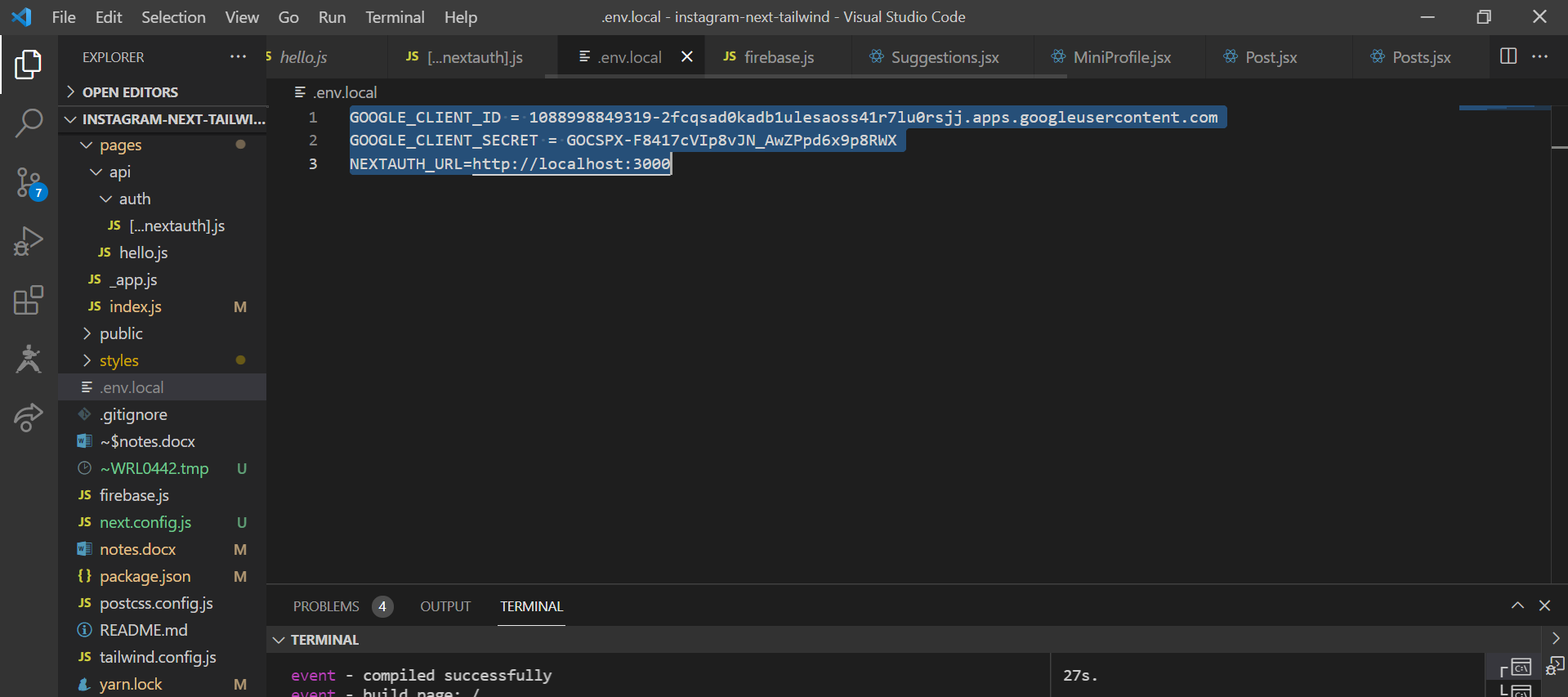
export { db, app, storage }

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

        }

    }

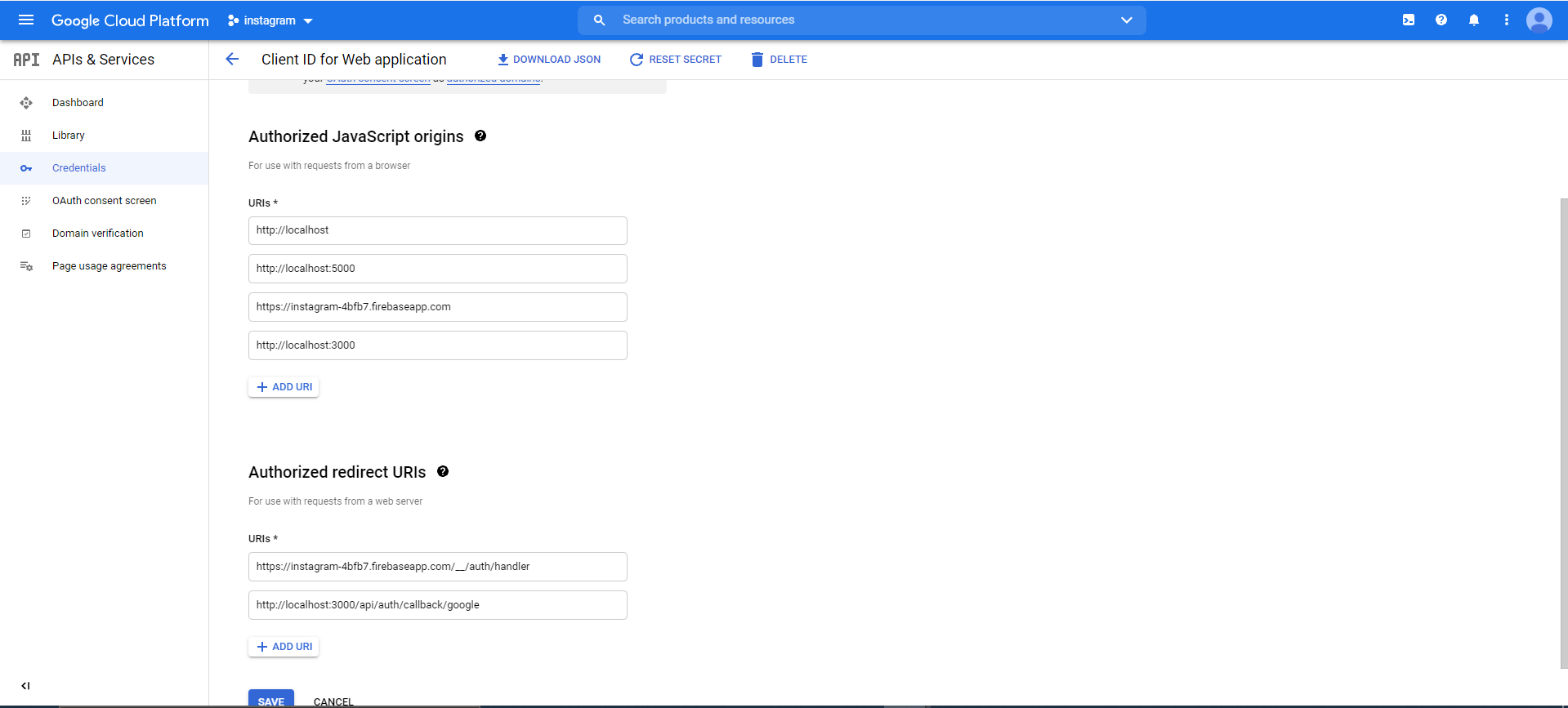
}

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

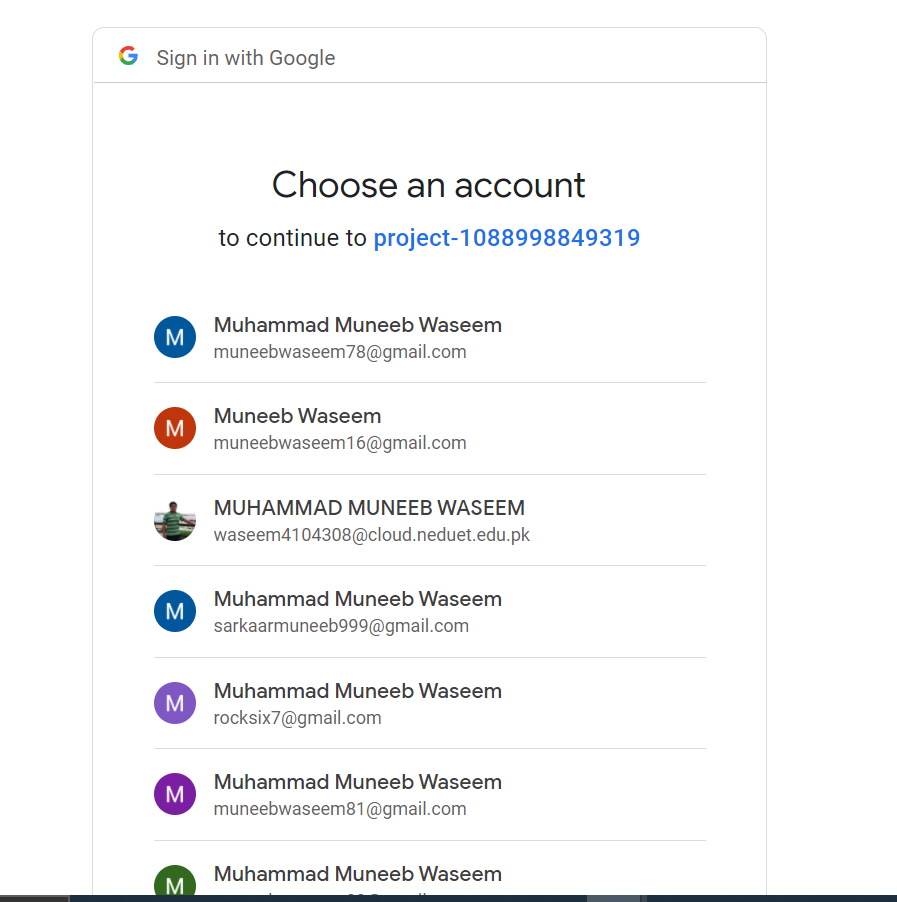


This update is for localhost, when we deploy on somehosting or vercel, we need to update localhost3000 links to that link

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>

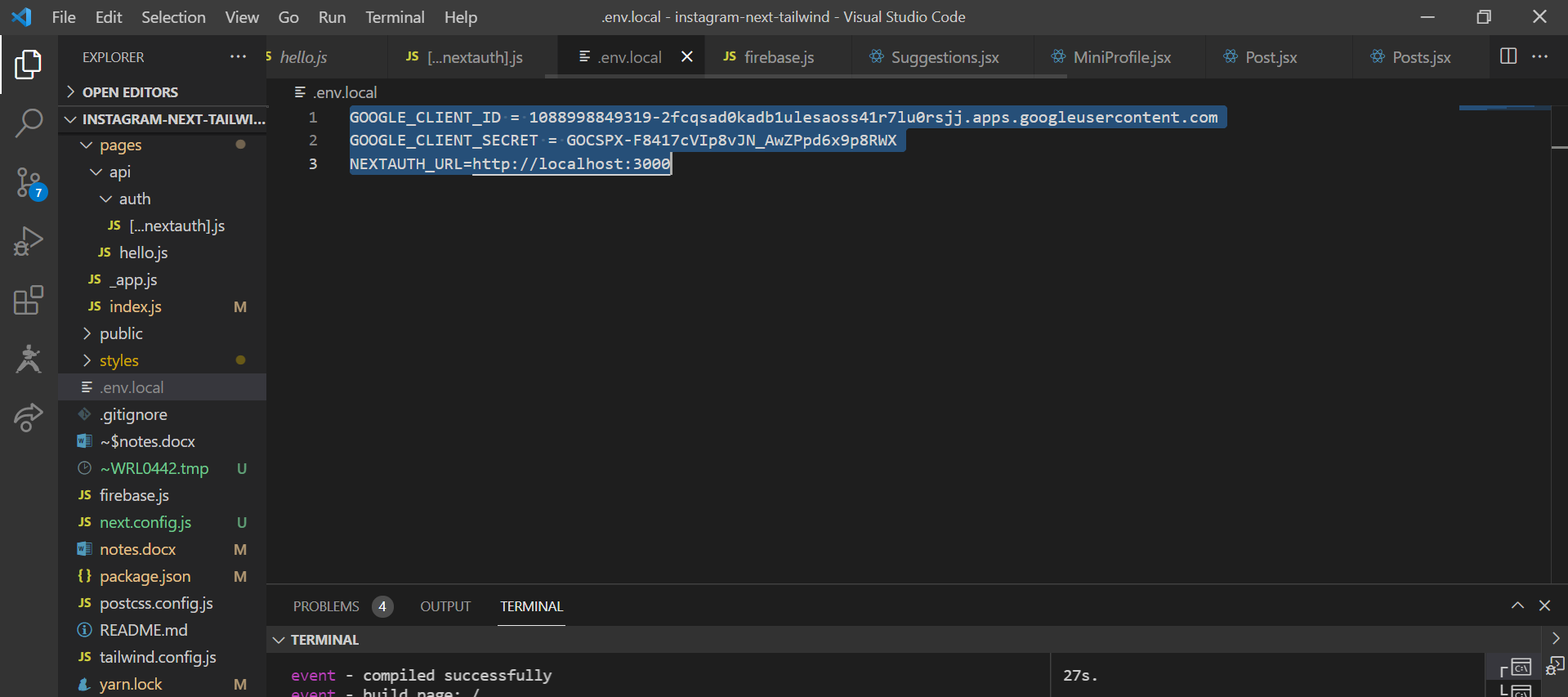
Vercel Deployment:

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

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

Reference Link

[nextjs-instagram/Posts.js at main · jpaddeo/nextjs-instagram (github.com)](https://github.com/jpaddeo/nextjs-instagram/blob/main/components/Posts.js)s