**준비**

1. home.js 에서 nweet 필드를 text필드로 바꾸자

…

await dbService.collection(‘nweets’).add({

text:nweet,

createdAt: Date:now(),

});

1. 각 필드가 user id를 가질 수 있도록 수정하자. 방법은, App에서 이미 인증상태가 변경되었을 때, user를 받아왔으므로 거기서부터 꺼내오자

//App.js

…

const [userObj, setUserObj] = useState(null);

useEffect(() => {

authService.onAuthStateChanged((user) => {

if(user) {

setIsLoggedIn(true);

setUserObj(user);

}

….

}, []);

…

<AppRouter isLoggedIn={isLoggedIn} userObj={userObj} />

//Router.js

…

<Home userObj={userObj} />

//Home.js

…

const Home = ({ userObj }) => {

…

const onSubmit = async(event) => {

…

createdAt: Date.now(),

createdId: userObj.uid

…

1. 이때 isLoggedIn 대신에 Boolean(userObj) 를 사용한다면, isLoggedIn 관련코드를 다 지울 수 있다.

<AppRouter isLoggedIn={Boolean(userObj)} userObj={userObj} />

**실시간으로 nweet 뜨게하기**

1. 실시간으로 데이터베이스의 상태 변화를 감지하는 메소드 snapshot
2. 기존에 있던 getNweets를 지우고, useEffect() 내부도 정리하자
3. 그다음 useEffect 내부에 다음과 같이 snapshop 사용

 useEffect(() => {

        dbService.collection('nweets').onSnapshot((snapshot) => {

            const nweetArray = snapshot.docs.map((doc) => ({

                id: doc.id,

                ...doc.data(),

            }));

            setNweets(nweetArray);

        });

    }, []);

* DB의 상태가 변화하면, 실시간으로 DB의 ‘nweets’ collection의 내용을 받아와 snapshot에 넣고 함수 시행
* 함수 안에선 각각의 doc 객체에 대해 id는 따로 저장하고 나머지는 그대로 저장하여 nweetArray에 저장
* setNweets로 nweets에 저장
* 리렌더링이 일어나며 실시간으로 nweet가 추가됨

1. snapshop.docs.map 대신 .forEach로 할 수도 있다.

* 하지만 forEach는 계속 리렌더링이 일어나는데 map은 그렇지 않다고 함
* 따라서 map이 더 빠르니 map을 선호한다고 했다