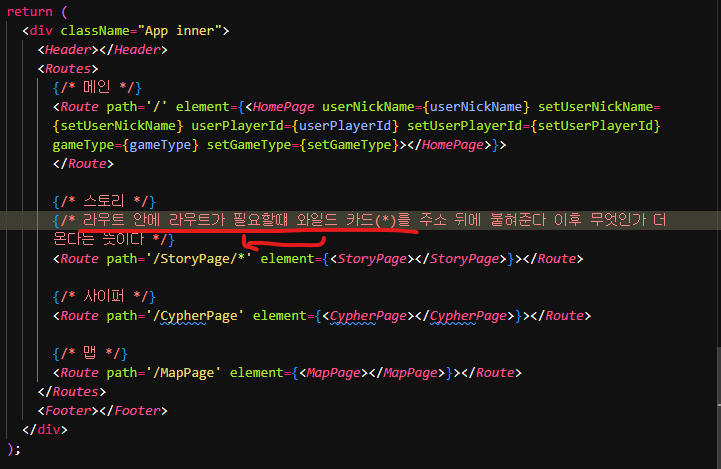
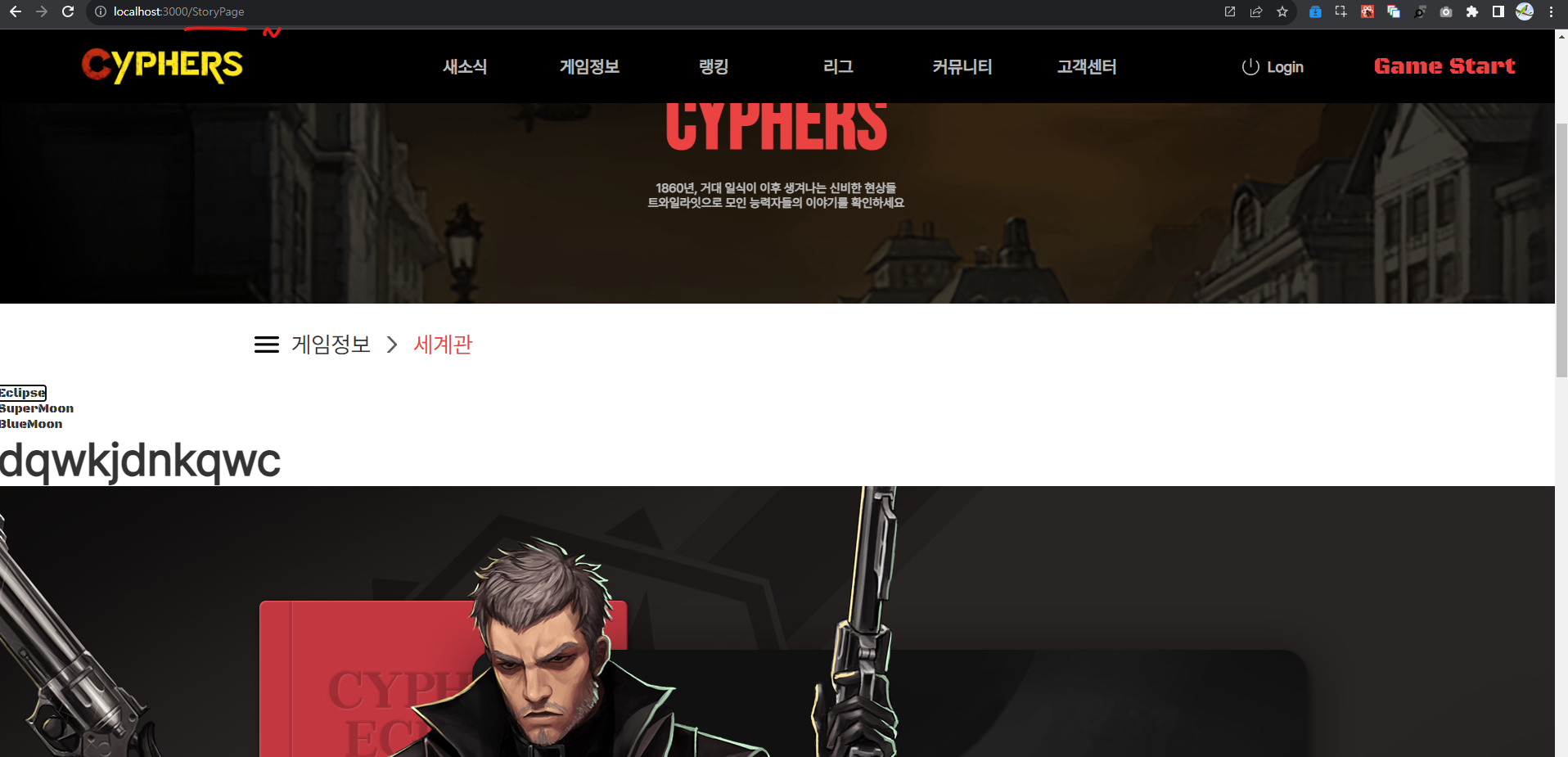
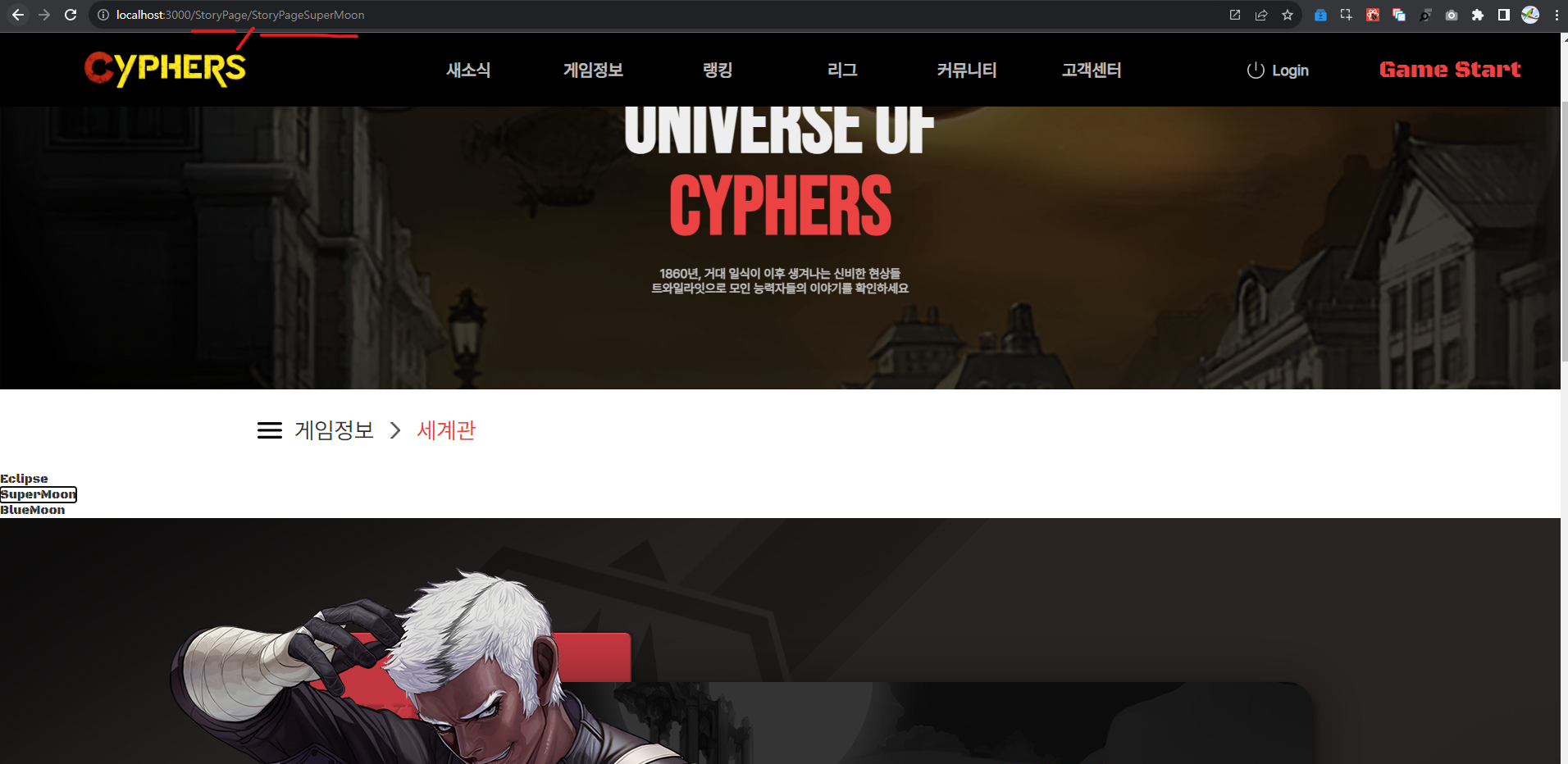
Main -> <Route >

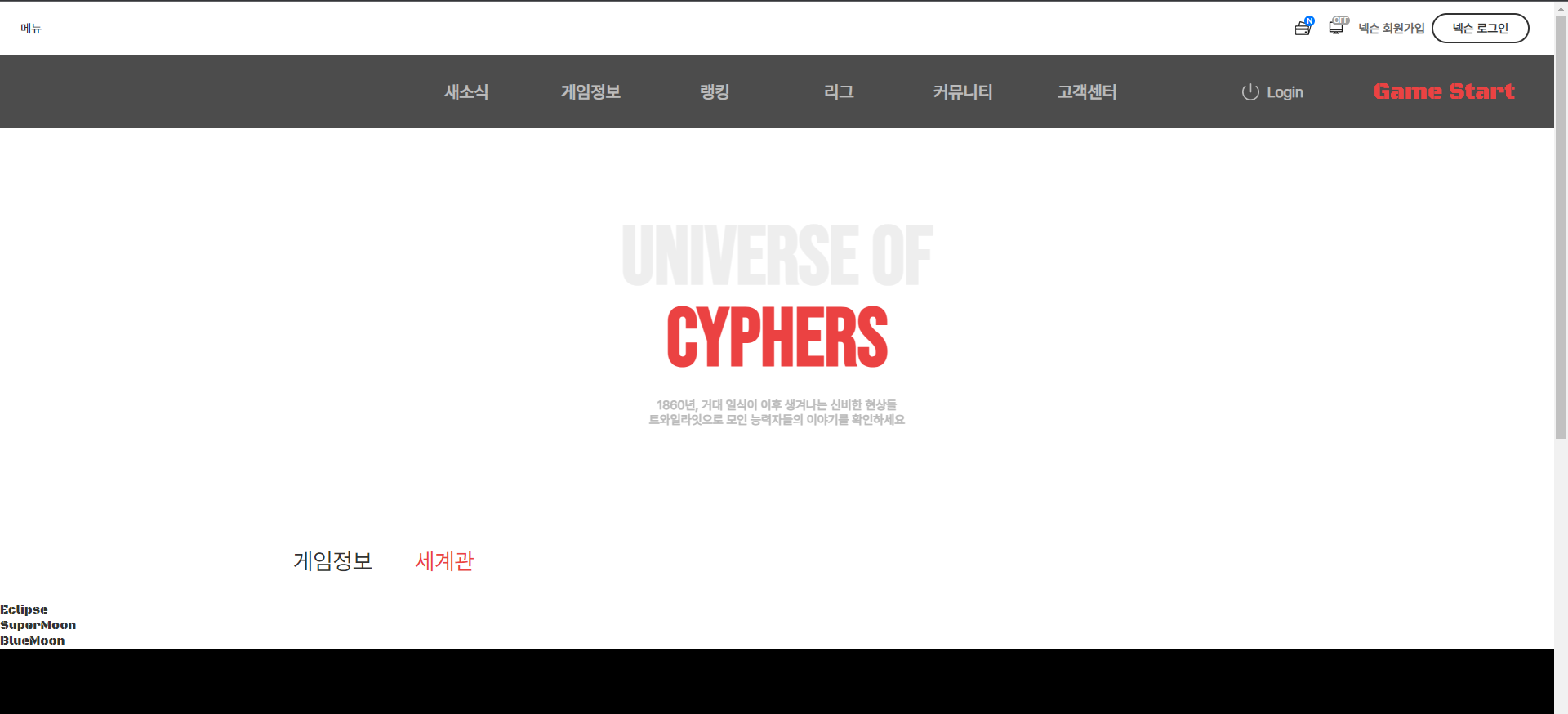
-> <Route>  
이처럼 라우트 안에 라우트를 사용하며 다중 라우트를 사용하려고 한다.  
  
나의경우 App.js에서 Route로 HomePage를 구현, Homepage에서 Story 섹션으로 이동했을 때   
Story 섹션 안에서 또 Route로 3가지의 페이지를 구현하려하는데  
경로의 문제인지 제대로 페이지가 표시되지 않았다.

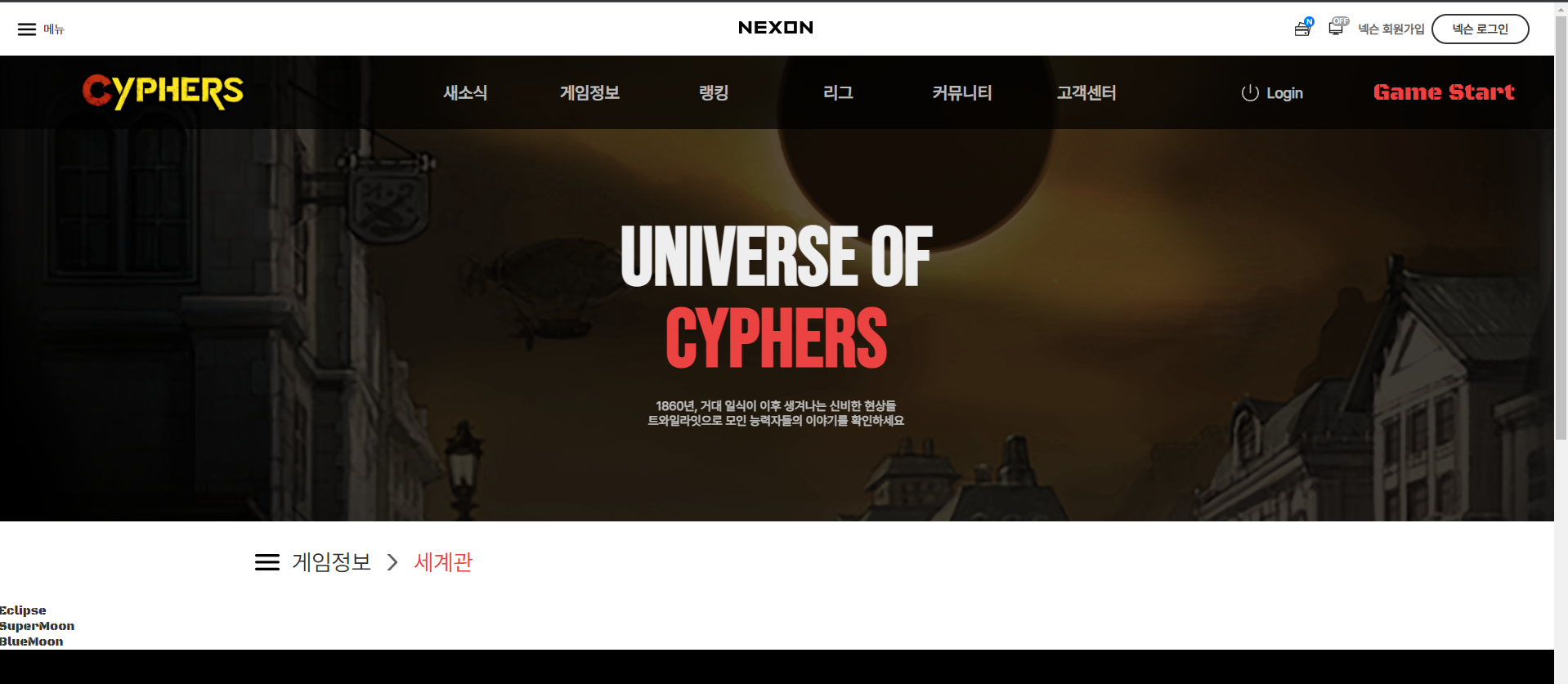
이렇게 다중 라우트를 구현하려면 2가지의 코드작업이 필요하다

1.  
  
  
라우트가 본인도 라우트를 가진다는 것을 브라우저에게 알려주기 위해 위와같이 pat의 뒤에 와일드 카드” /\* “ 를 붙여줘야 브라우저가 알 수 있다.  
2. 다중 라우트가 되는 라우트의 path 앞에는 “ / “ 을 붙히지 않고!! path를 만들어준다!!  
  
  
이 Route를 연결해주는 것은 버튼클릭에 따른 useNavigate를 이용했는데 path에서 지정한 대로 / 를 뺀 주소를 사용해준다.  


이렇게 해주면 브라우저가 다중 라우트 확인하고 구현할 수 있게된다.  
  
path를 “” 로 지정해준 라우트 는 / 가 붙지 않고 라우트를 선언한 부모 라우트 주소에서 바로 나온다.  
  
  


path를 “PageSuperMoon”로 지정해준 라우트 는 본인의 주소를 부모 라우트 주소에 / 를 붙힌채로 나오게된다.  
  
이처럼 이중라우트를 구현할때는 위와같은 설정이 필요하다.

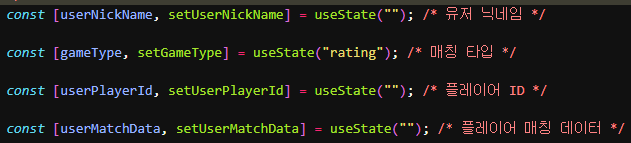
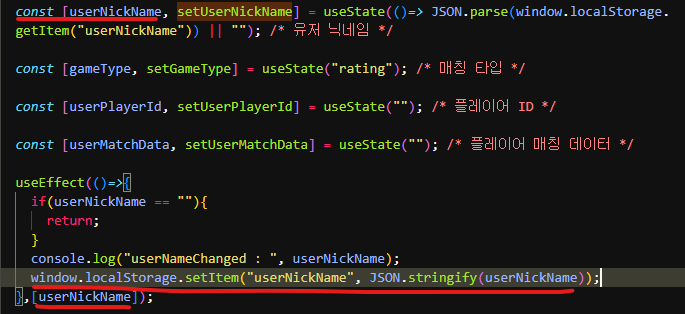
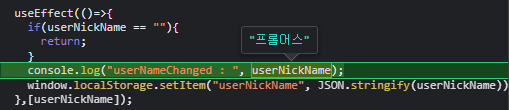
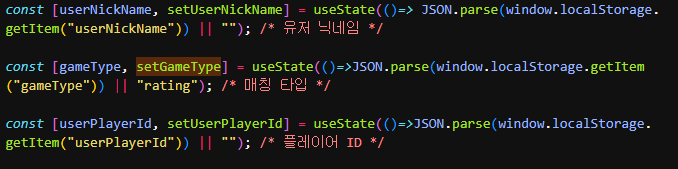
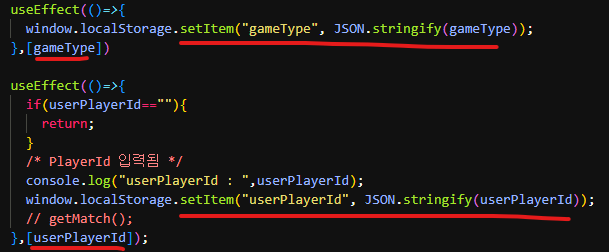
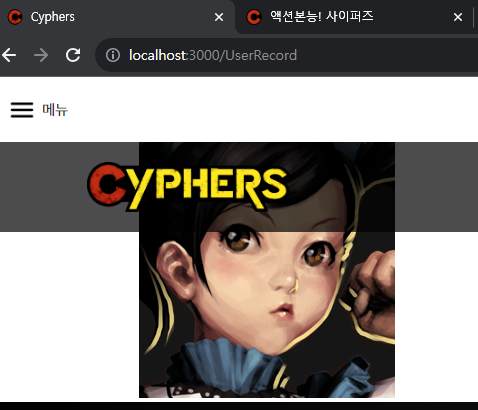
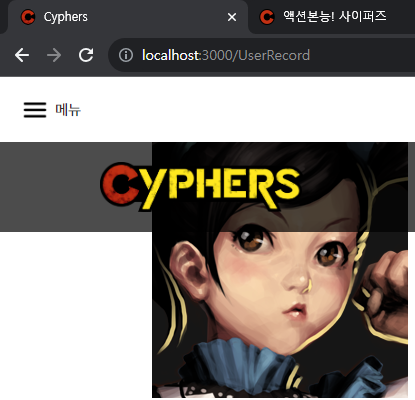
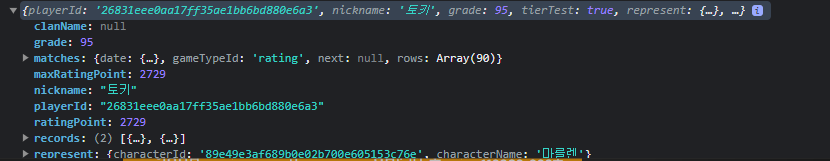
또한가지 문제가 생겼다.  
다중 라우트 안에서 이미지가 가져와지지 않았다.  
  
  
이는 다중 라우트를 사용할때 다중 라우트가 되는 주소에서는 이미지의 경로를 읽을 때

src = “img/~ “ 로 읽을 수 없기 떄문이다.  
  
이를 해결하는 방법으론 정말 간단한데 img 경로앞에 / 슬래시 하나만 붙여주면된다.  
  
이런식으로 / 만 붙혀주면되는데  
  
이는 라우트가 아닌곳에서도 똑같이 보이니 이중 라우트를 작업한다면 img경로를 위와같이 해주는게 좋겠다.  


그런데 이렇게 이중라우트로 구현하고나니까 그냥 탭메뉴 처럼 만드는게 좋다고생각된다.  
라우트는 결국 재 렌더링 할때 이미지를 다시 깔아주기때문에 깜빡 깜빡하는게 있는데  
이게 굉장히 거슬린다.

에러는 아닌데... useState가 새로고침시 초기화되는것 때문에 Api 사용하는 userRecord 페이지가 문제가있다..  
  
전적검색을 위해 닉네임을 입력하고 그에따른 id 값들을 useState로 가져가 리액트에서 렌더링 해주는데  
여기서 새로고침을 하게되면 이 useState가 초기화되며 “” 빈 값이 되버리는것이다...  
  
이때 필요한것이 웹 스토리지 라고한다.  
구글링해서 본 자료는 다음과같이 예시를 들며 설명하고있다.  


즉, 렌더링 될때마다 useEffect(()=>{ ,[목표]}) window.localStorage.setItem(“저장이름”, JSON,stringify(저장대상));   
을 이용하여 웹 스토리지에 저장하는듯 하다.  
그리고 useState를 사용할때 useState(“”) 로 선언하는게아닌 화살표함수를 사용해

( () => JSON.parse(window.localStorage.getItem(“저장이름”)) || “” ); 으로 선언하면 왼쪽이 참이면 왼쪽이, 아니라면 오른쪽이 저장될 것이다.  
  
이를 반영해보자   
  
이렇게 사용중인 useState를 바꿔줘야한다  
  
  
나는 userNickName에 의해 다른 useState가 결정되니까 이렇게 userNickName만 적용해봤다.  
  
  
  
  
우선 입력했을때 확인해보니 잘 들어간모습을 볼 수 있었다.   
새로고침을 한 후 다시 확인해보면?  
  
[새로고침 F5 후]  
  
맙소사.. 정말 useState가 유지됨을 볼 수있다.  
정확히는 웹스토리지에 저장된 값을 다시금 불러온것이다.  
  
--> userName만 저장되면 되는게아니였다.   
userPlayerId, gameType 도 저장해줘야됐다 !   
  
  
matchId 는 어차피 userRecord.jsx 가 렌더링 될때 가져오니까 냅둬보고 이 3가지만 세팅해주자  
  
useEffect를 이용할때 window.localStorage.setItem 해주었다.  
  
  
만세!! 새로고침해도 이전의 내용을 불러오는것을 확인할 수 있었다.  
 --> 새로고침  
똑같아 보이지만 오른쪽은 새로고침한 상황이고  
  
위와같이 useState를 props로 받아온 userRecord.jsx 에서 새로고침을 해도 문제없이 이전 matchData를 가져올 수 있었다 !!  


Netlify 에서 새로고침 할 시 404 – not Found 가 되는 오류   
  
**원인**

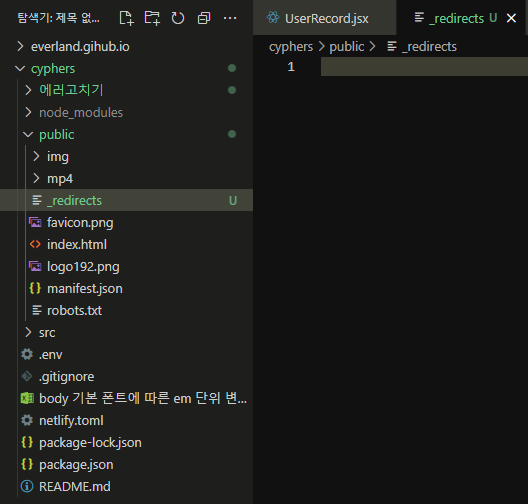
내 설정에서 리액트가 최초 접속할 때의 url 은 / 이다.  
그리고 이 때 index.html 을 읽는다. 즉 해당 url 에 대한 자원은 index.html 로 정해져 있다.

하지만 화면을 넘기면서 라우팅된 url 이 예를 들어 /a 혹은 /b 가 됐을 땐, 각 url 에 맞는 자원이 존재하지 않는다.  
index.html 처럼 정해진 자원이 없다. (react 는 single page application 이기 때문이다)

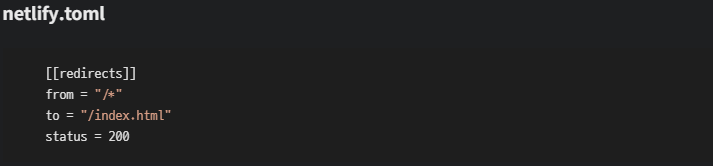
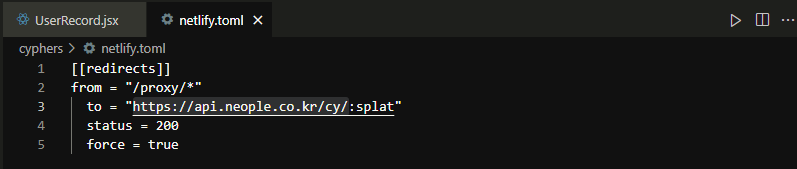
즉 /a , /b 라는 url 에서 새로고침을 하면, / 와 달리 곧바로 내려줄 자원이 존재하지 않기 때문에 404 에러가 발생하는 것이다.

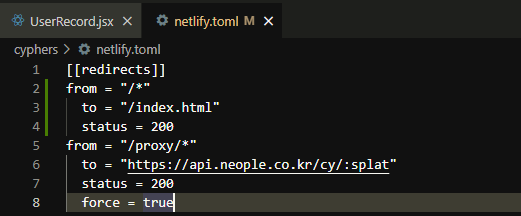
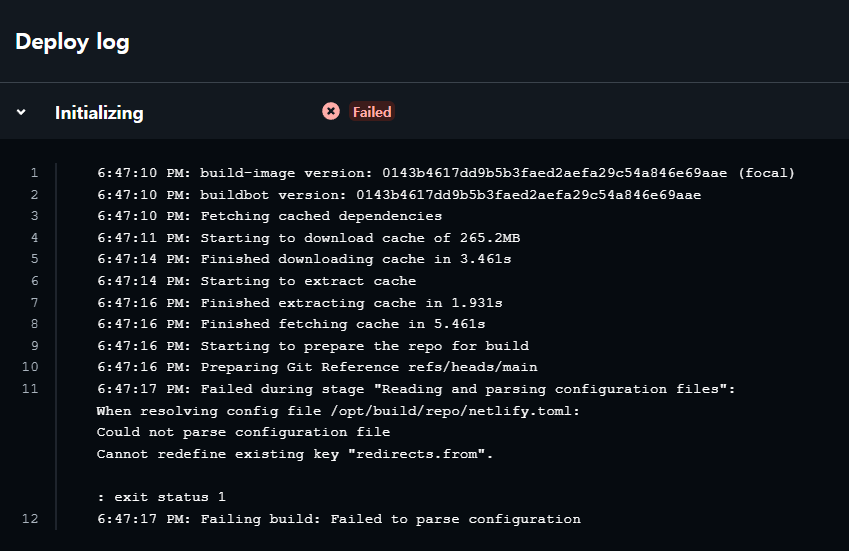
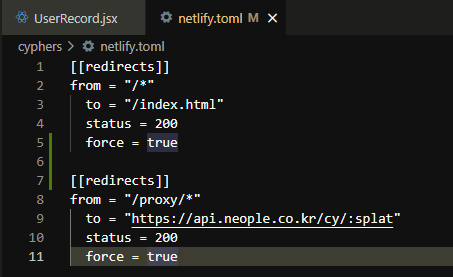
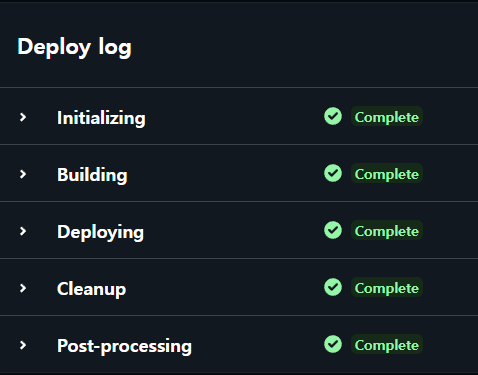
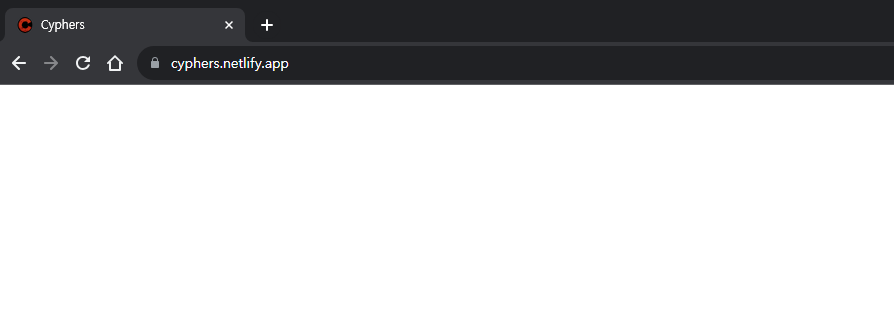
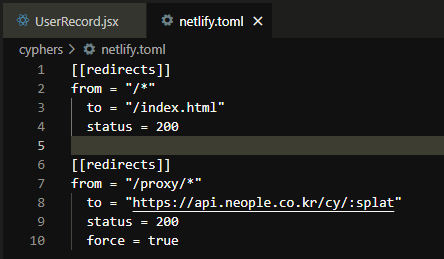
**해결**

그렇다면 각 /a, /b 이란 url 에 곧바로 접근하더라도, 보여줄 자원만 있으면 404 에러를 피할 수 있다.  
바로 index.html 을 보여주면 된다.  
우선 public 폴더 내부에 \_redirects 파일을 생성한다.

  
그리고 해당 파일 내부에 아래 코드를 작성한다.  
  
  
이는 어떠한 경로로 들어와도 ( /\* ) ,  
200 상태코드를 내려주며, index.html 파일을 렌더링 하겠다는 의미이다.

이렇게 설정하고 배포한뒤 다시 테스트를 해보면 새로고침을 하여도 NotFound 에러가 안나타나는 것을 확인할 수 있다.   
하지만!!!!!!! **이방법으로 하면 API를 사용하는데 문제가 생겼다**

그래서 다른 방법을 찾아보았다.  
위의 \_redirects 는 **삭제**하고   
이전에 netlify에서 PROXY 설정을 위해 만든 netlify.toml 을 이용한다.  
  
루트 디렉토리에 **netlify.toml**파일을 만들고 안의 내용을 아래와 같이 작성해주면 된다고 한다.  
  
  
API를 위해 나는 아래와같이 작성해 놓은 상황이다.  


여기서 from어떻게 바꿀지 잘 모르겠지만… 위의 해결책의 form와 to를 한번 추가해보자   
  
  
  
이렇게 commit-push 해보니 netlify에서 에러를 반환한다.  
  
해석해보면 수정한 netlify.toml 를 해석할 수 없다 !!  
라는 뜻이다..  
  
구분을 지어줘야하는걸까? 위처럼 해보았다.   
  
이렇게 해보니 netlify 에서 오류를 발견하지 못했다.  
  
즉, [[redirects]]는 이렇게 구분지어 사용해야하는 것 같다.  
  
하지만 이랬더니 아~~~무것도 나오지않는다… 구성요소도 하나도 없다  
혹시 force=true 를 둘다 가지고있어서 그런가?   
  
force 하나를 없애봤다.

이번에는 페이지가 보이긴 했지만 다시 API가 먹통이다…………………………..  
그냥 netlify에서는 새로고침을 포기하고 빌드하고 github을 도메인으로 사용하자