toPangyo
동세권

Project

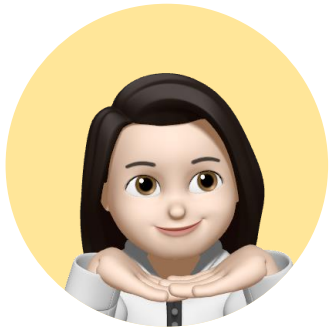
목차

팀원 소개

toPangyo

우명규

Team Leader
&
Front-end

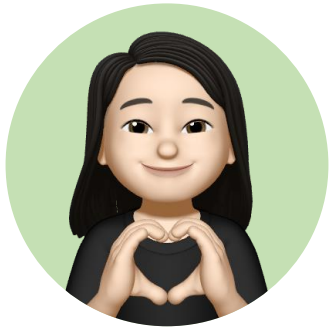


류나연

Front-end

선원종

Back-end



전혜지

Back-end

최성민

Back-end

모두 성공하여 다같이 판교로 가자!

프로젝트 소개 및 기능

프로젝트 주제

은?

플레이스 매칭 커뮤니티 플랫폼

현재 위치를 기반으로
유저들이 다양한 오프라인 장소를 직접 선택하여
함께 즐길 수 있는 플랫폼

프로젝트 주제

이 프로젝트를 선정한 **이유**

프론트엔드

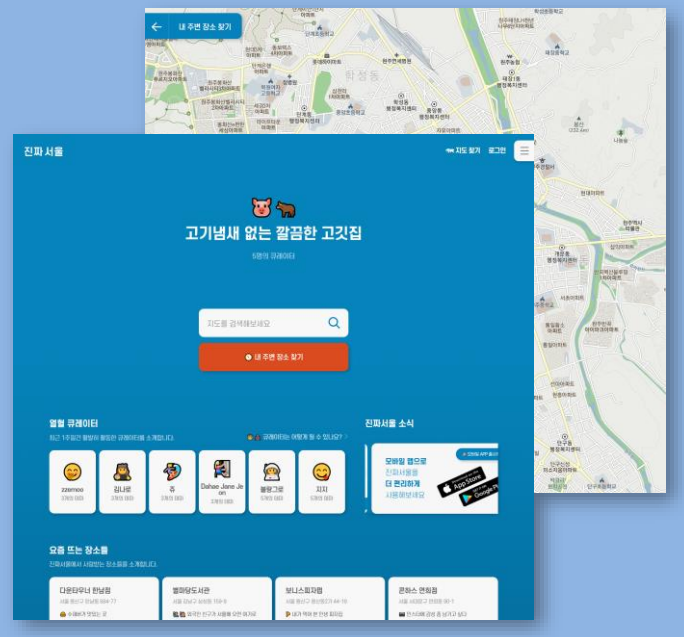
1. 지도 및 위치 기능을 사용하고 가독성 있게 UI를 배치
2. 오류없이 계획대로 기능을 잘 구현하는 것

🤔 **모두의 니즈를 충족?**

백엔드

1. 직접 경험해보지 않았던 통신 소켓 프로젝트 경험
2. 디스코드와 같은 메신저를 만들어보는 것

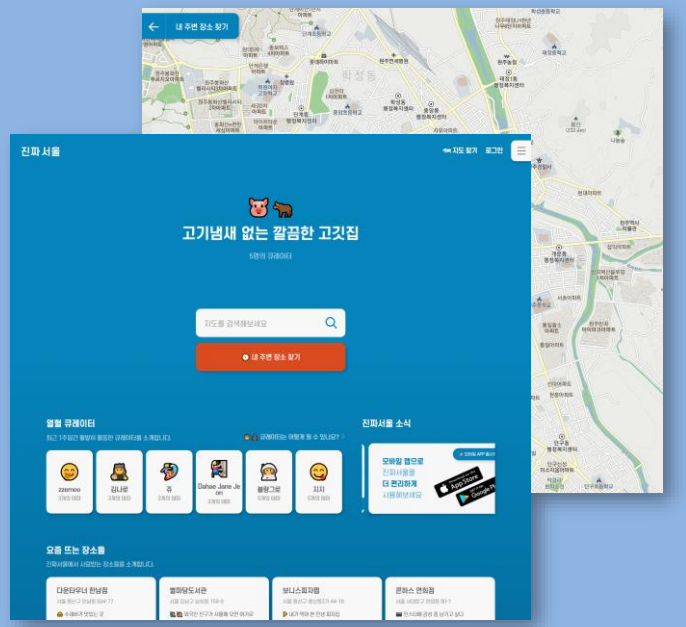
진짜 서울



서울의 여러 오프라인 장소를 추천해주는 플랫폼

프로젝트 주제

진짜 서울



+

채팅 기능

매칭 기능

빠른 성능

=

동세권

양방향 소통의 부재

지연있는 렌더링 성능

프로젝트 기능

매칭
매칭방 생성
매칭방 삭제
카테고리 분류
검색

채팅
메시지 전송
메시지 받기
방장 권한
퇴장 기능

알림
알림 전송
알림 받기

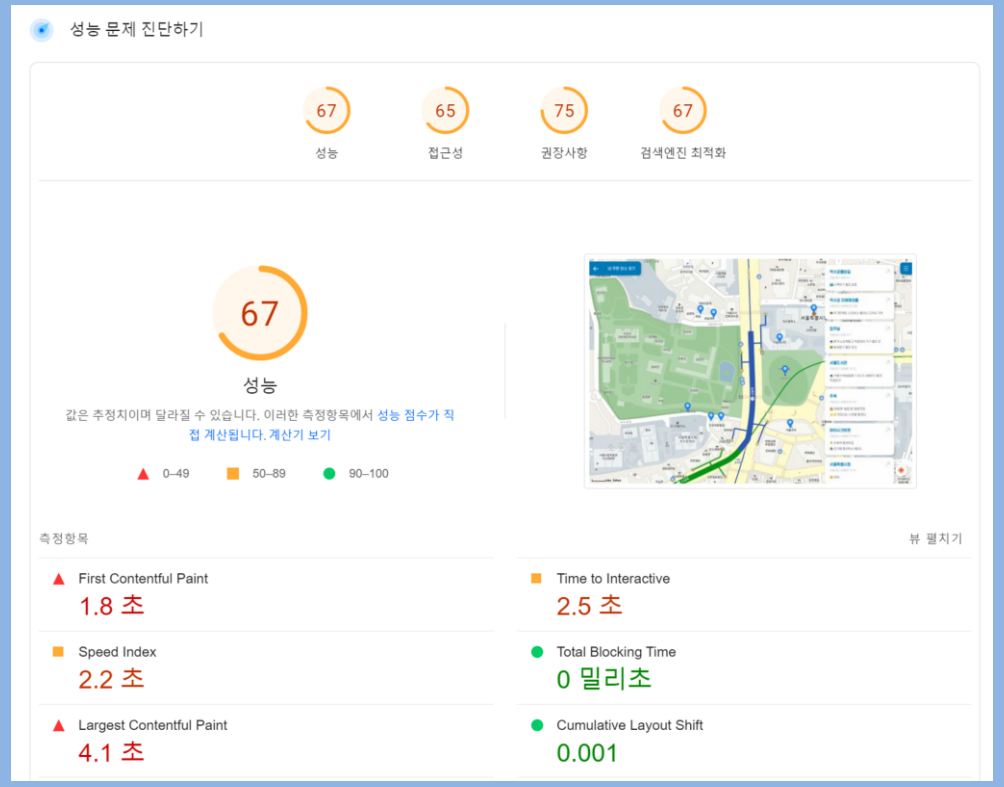
유저
로그인
회원가입
프로필 설정

포스팅
모집글 R CUD
시간 및 장소 설정
신청 기능

팀 목표

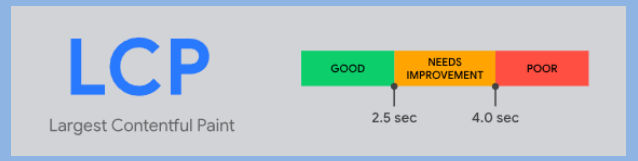
팀 목표

LCP 2.5초 이내로 최적화



By PageSpeed Insights

벤치마킹한 '진짜 서울'의 LCP 성능은 4.1s로 확인
우수한 사용자 경험을 위해서는 2.5s 이내의 LCP여야함

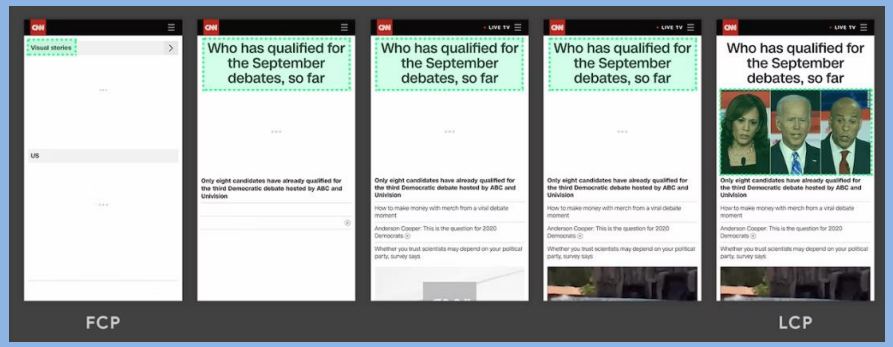


By Chrome Developers

LCP?

Largest Contentful Paint(LCP)의 약자로 Core Web Vitals의 웹사이트 성능 메트릭이며 그 외에도 FID, CLS가 존재

페이지가 처음으로 로드를 시작한 시점을 기준으로 뷰포트 내에 있는 **가장 큰 콘텐츠의 렌더링 시간**



왜?

이전의 성능 메트릭

FCP(First Contentful Paint)
SI(Speed Index)

→

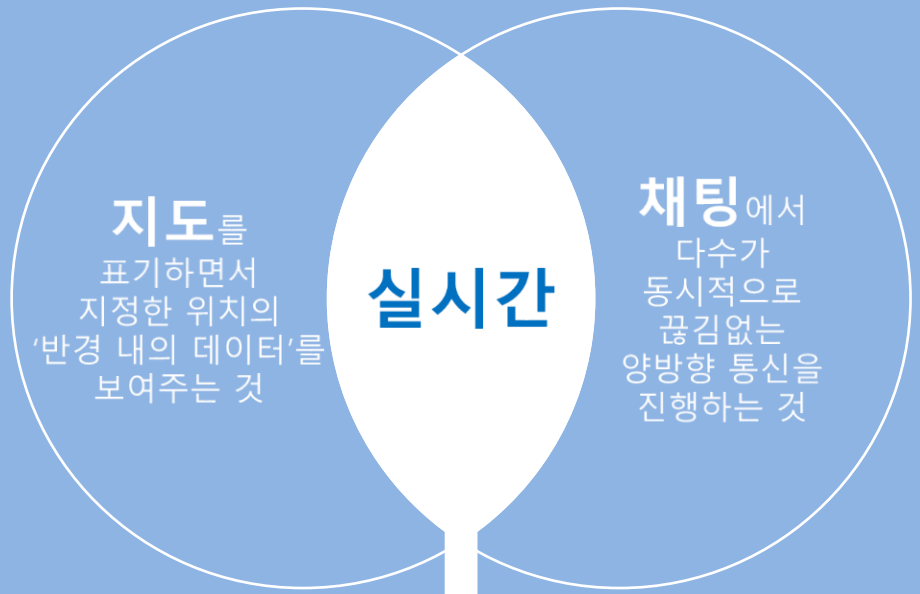
설명이 복잡하고 어려우며
정확한 성능 측정이 불가

→

LCP

사용자가 감지하는 로드 속도를
측정할 수 있는 중요한 사용자 중심 메트릭

왜?



페이지 렌더링 및 로딩속도

→ 페이지를 빠르게 렌더링하거나 새롭게 그려지는 요소들의 연산이 많으면 안됨

데이터 처리의 반응속도

→ 데이터를 빠르게 response 할 수 있어야함

통신의 안정성

→ 소켓 통신을 할 때 끊김없이 상대방과 양방향 통신을 할 수 있어야함

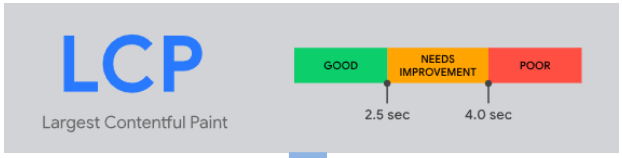
3가지 중 2가지 요소와 실시간이라는 키워드를
고려하였을 때 성능을 측정하기 위한 지표로 LCP를 선정

The probability of bounce increases 32% as page load time goes from 1 second to 3 seconds.

Google/SOASTA Research, 2017.

페이지 로드 시간이 1초에서 3초로 늘어남에 따라
사용자의 페이지 이탈률이 32% 증가된다.

우수한 사용자 경험을 제공하기 위해서는 2.5s의 LCP를 가져야함



LCP
2.5초 이내로
최적화

개인 목표

우명규

1. React의 Virtual DOM만을 활용해 프로젝트를 개발할 것이고 컴포넌트가 여러 번 렌더링되지 않도록 useEffect의 clean-up함수로 초기화 값을 지정하여 메모리 누수 방지하기
2. API를 사용하는 코드는 async & await으로 비동기 처리를 하여 응답 대기시간을 줄이고 문제가 있는 요청이 들어와도 try-catch문으로 사용자에게 정상적인 페이지 응답하기
3. state는 Redux로 관리하여 과도한 props drilling을 방지하고 각 state의 Reducer는 파일을 따로 관리하여 store에 적용시키기

류나연

1. 현재 위치를 받아 온 뒤 해당 위치에 지도의 마커를 찍는 기술을 실패했던 원인을 찾고 이를 해결해 실패의 원인의 해결법을 복기하며 공유하기 위해 문서화하기
2. 매일 to do list 작성으로 체계적인 개발과 회고를 통한 피드백을 하며 매일 문서화 하는 것을 습관화하며 캠프 이후에도 지속적인 문서화로 1일 1포스팅하기

선원종

서버의 부하 테스트시 응답실패율을 없애고 응답시간을 3초 이내로 줄이기

1. Jmeter를 사용하여 최대로 부하가 걸리는 시나리오를 짜보며 커넥션 관리 테스트 진행
2. 채팅방 로그를 계속해서 저장하는 것이 아닌 채팅방이 종료되게 되면 전체 메시지를 DB에 저장하여 I/O줄이기
3. 스케일 아웃시 clustering을 미리 구성해두고 그 데이터 공유 문제를 Redis의 Pub/Sub을 매개체로 구현

전혜지

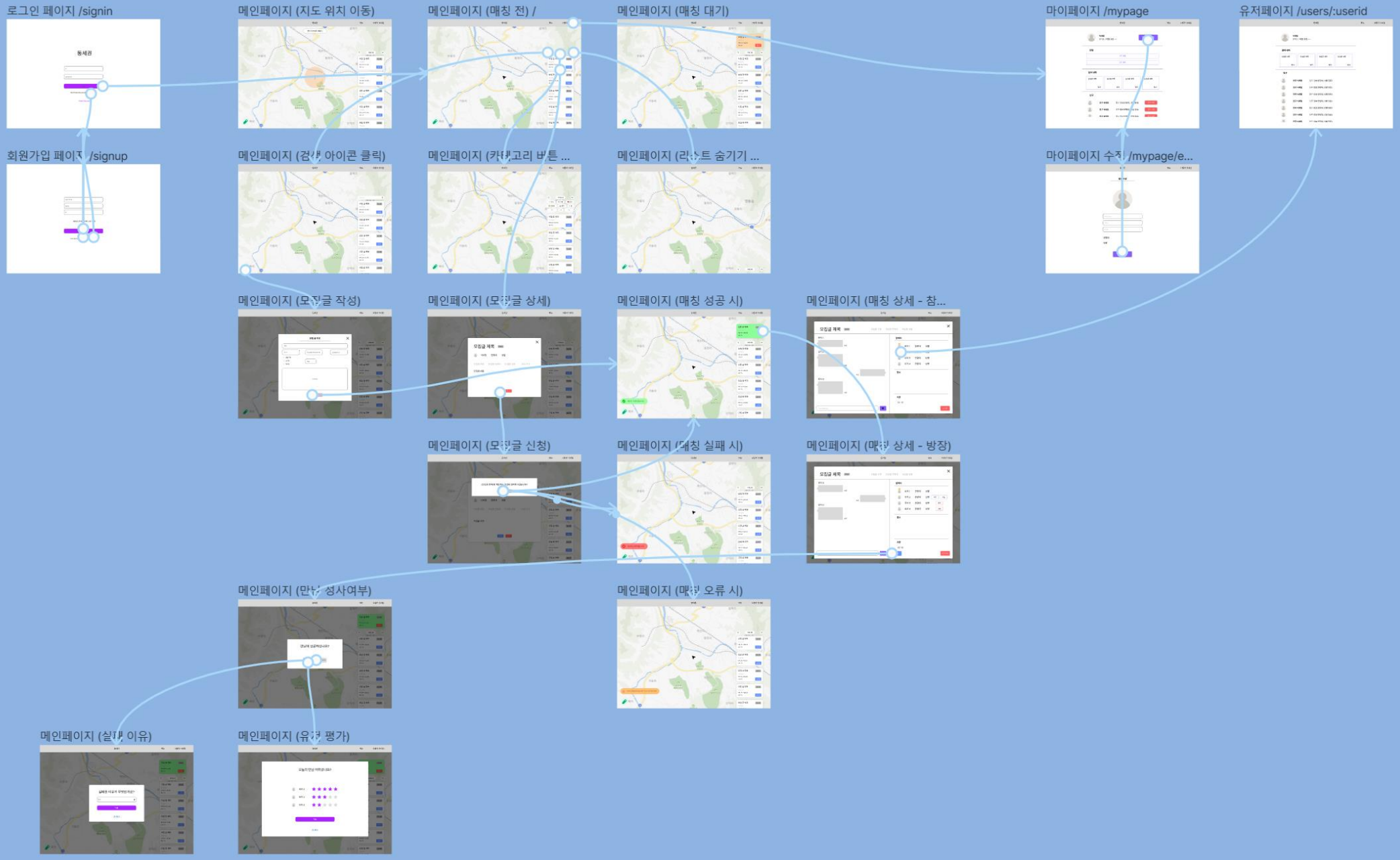
1. DB중복과 무결성 최대한 없이 종속적 규칙을 지키며 설계해보기
2. 쿼리문 작성시 각 절의 순서를 지키고 검색 최소화하여 효율적이고 직관적으로 작성해보기
3. 효율적인 협업을 위해 commit 메시지만을 보고 수정 내용을 알 수 있도록 작성하고, git의 기능 최대한 활용해보기

최성민

1. 서버 호출을 최소화하고 데이터를 효율적으로 읽는 스키마를 설계
   (n+1문제, populate, 적절한 내장 모델등을 사용)
2. RESTful API중심의 서버 구성
   (URI 표현 규칙 통일, 엄격한 HTTP 상태와 method 사용)
3. MSA 아키텍쳐를 통한 서버 구성

프로젝트 구조

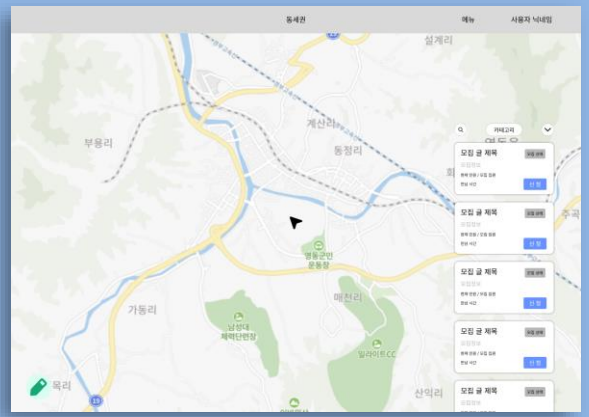
스토리보드


로그인 페이지 /signin
메인페이지 (지도 위치 이동)
메인페이지 (매칭 전) /
메인페이지 (매칭 대기)
마이페이지 /mypage
유저페이지 /users/:userid
회원가입 페이지 /signup
메인페이지 (검색 아이콘 클릭)
메인페이지 (카테고리 버튼 ...
메인페이지 (리스트 숨기기 ...
마이페이지 수정 /mypage/e...
메인페이지 (모집글 작성)
메인페이지 (모집글 상세)
메인페이지 (매칭 성공 시)
메인페이지 (매칭 상세 - 참...
메인페이지 (모집글 신청)
메인페이지 (매칭 실패 시)
메인페이지 (매칭 상세 - 방장)
메인페이지 (만남 성사여부)
메인페이지 (매칭 오류 시)
메인페이지 (실패 이유)
메인페이지 (유저 평가)

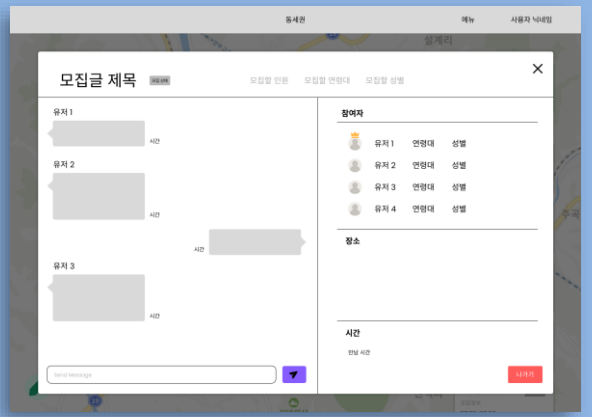
주요 화면 구성

로그인

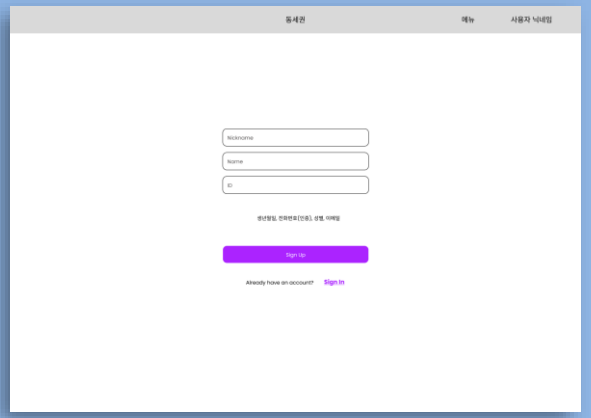
메인



매칭

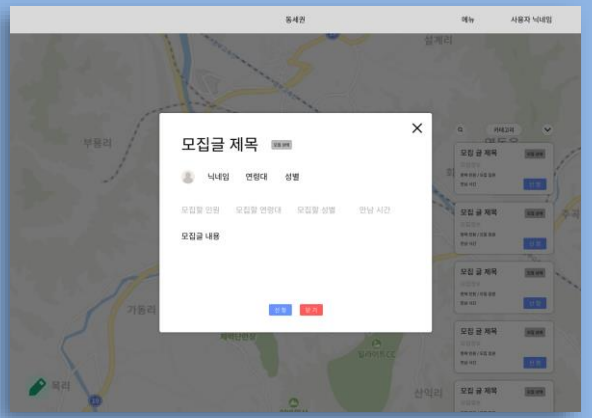


회원가입



마이

모집 글



주요 컴포넌트 구성

<Navbar>

- 현재 로그인한 사용자의 닉네임을 표시해주고 동세권 로고, 여러 메뉴 표시

/signin **<SignInPage>**

- <SignInBox> : 사용자의 id, password를 입력

/signup **<SignUpPage>**

- <SignUpBox> : 사용자의 닉네임, id, password, 생년월일, 전화번호, 성별, 이메일의 정보를 입력

/account **<Account>**

- <Profile> : 나의 프로필사진, 닉네임 등의 정보를 확인할 수 있으며 회원정보를 수정할 수 있는 버튼을 표시
- <Alert> : 매칭성공/실패여부, 채팅 알림을 확인
- <History> : 내가 매칭되었던 내역을 확인

/account/change **<ChangeAccount>**

- 프로필사진과 유저의 정보를 수정할 수 있는 컴포넌트

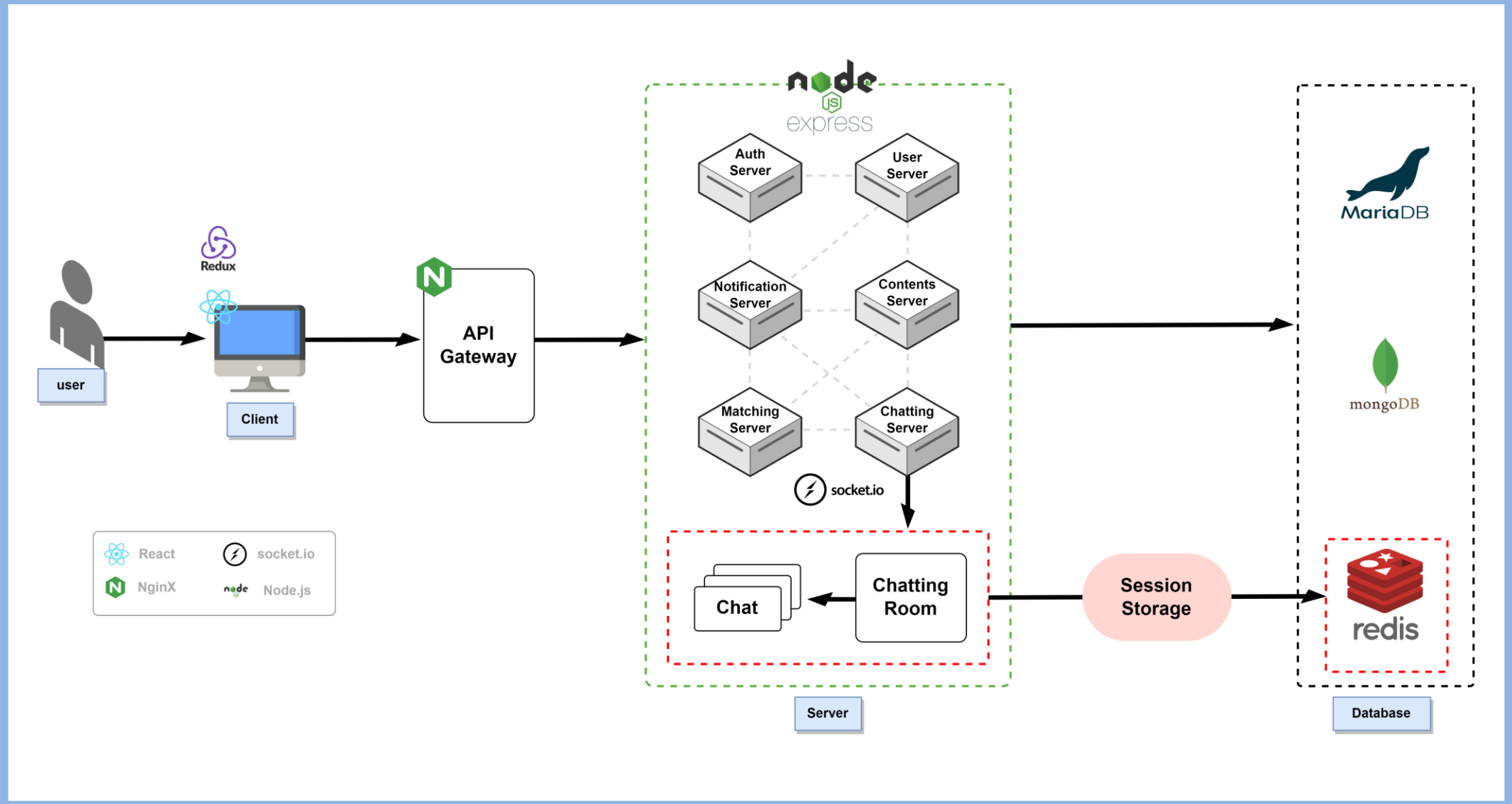
/ **<HomePage>**

- <Map> : 지도와 유저의 현재위치를 표시
- <MatchingPost> : 신청한 모집글이 대기 중인지 매칭완료된건지 표시
- <Toolbar> : 검색, 카테고리 찾기, 모집글 리스트 숨기기 기능
- <PostList> : 모집 글들을 리스트로 표시
  - <Post> : 모집글 제목, 내용, 인원, 시간, 모집상태 등 정보를 카드형태로 표시
- <WritePost> : 모집글을 작성
- <DetailPost> : 모집글 카드클릭 시 모집글의 상세정보 표시
- <ApplyPost> : 신청버튼 클릭시 신청여부 확인
- <MatchingDetail> : 매칭이 완료되어 매칭된 사람들과 채팅, 유저확인, 시간/장소 설정(방장-참여자 간 표시되는게 다르게)
  - <MatchingChat> : 매칭완료된 사람들과 채팅
  - <MatchingUser> : 매칭완료된 사람들을 확인
  - <MatchingPlace> : 매칭완료된 사람들과 장소를 결정
  - <MatchingTime> : 매칭완료된 사람들과 시간을 결정
- <CheckGather> : 매칭완료된 유저들이 만났는지 여부 확인
  - <UserRating> : 모임 이후 유저들 평가
  - <FailReason> : 모임에 실패하였을 때 이유 확인

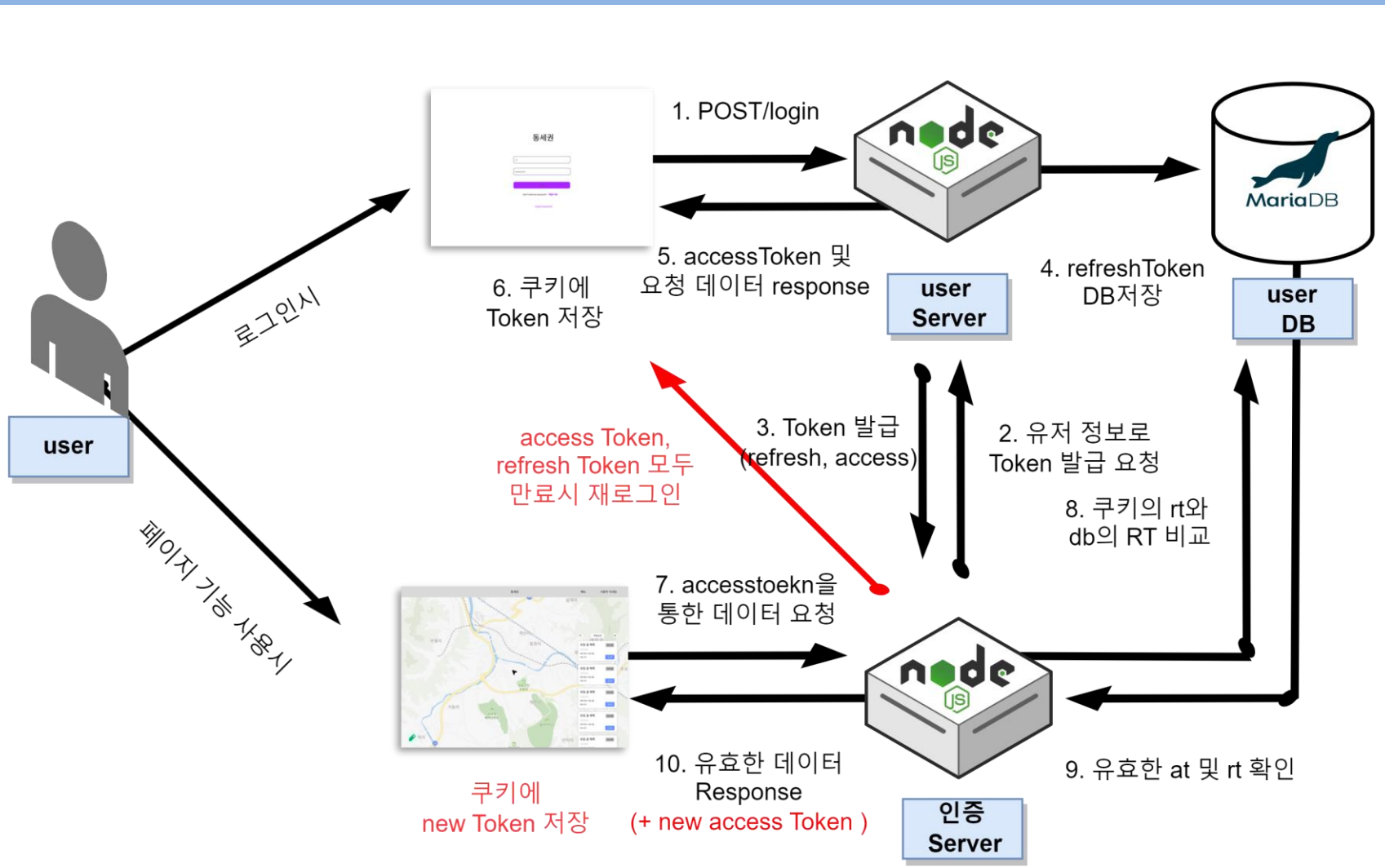
/users/{userId} **<UserPage>**

- <Profile> : id에 맞는 유저의 프로필사진, 닉네임 등 유저에 대한 정보를 확인
- <History> : 해당 유저가 매칭되었던 내역을 확인

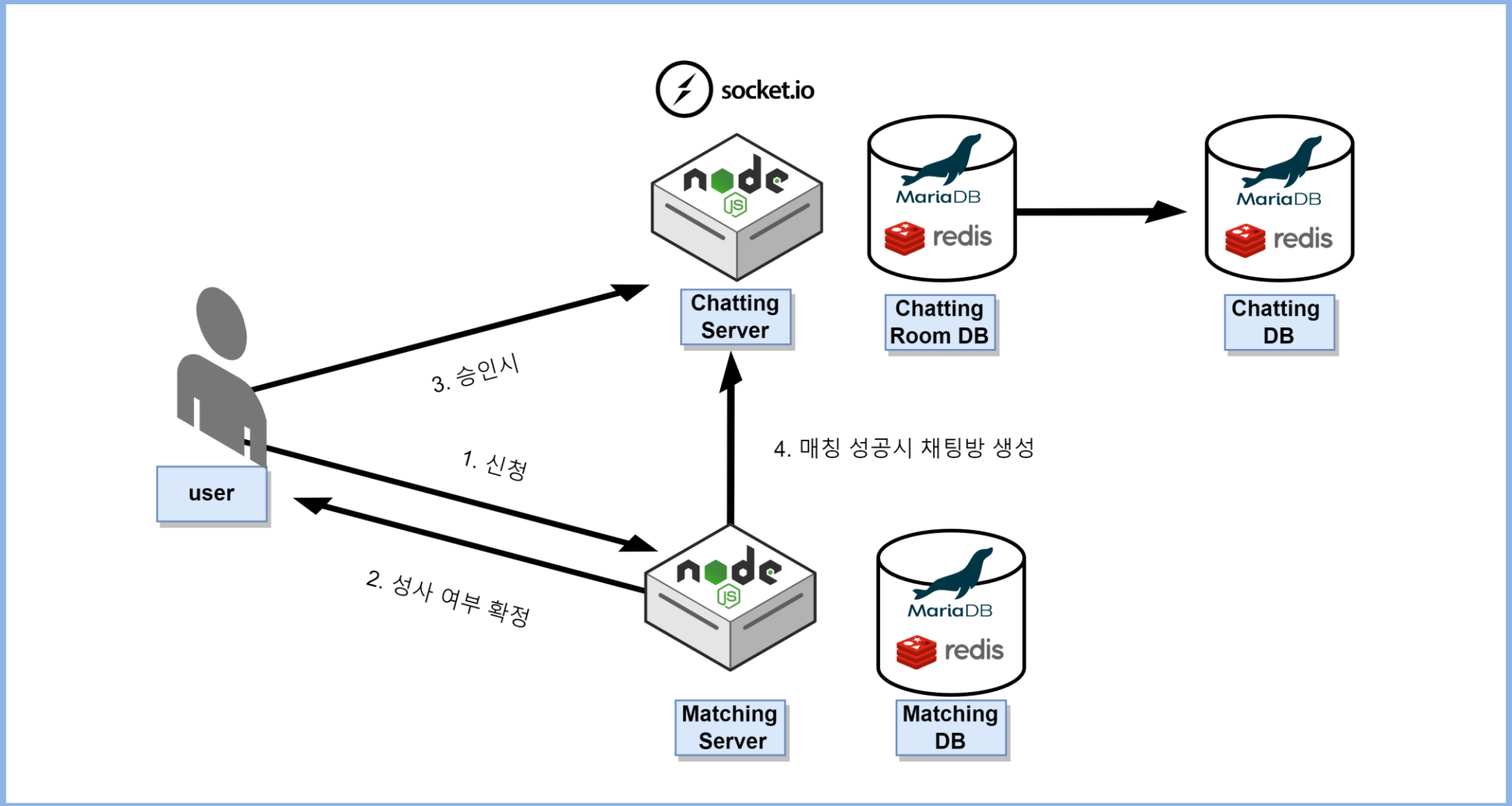
전체 아키텍쳐


Redux
user
Client
API Gateway
node
express
Auth Server
User Server
Notification Server
Contents Server
Matching Server
Chatting Server
socket.io
Chat
Chatting Room
Server
MariaDB
mongoDB
Session Storage
redis
Database
React
socket.io
NginX
Node.js

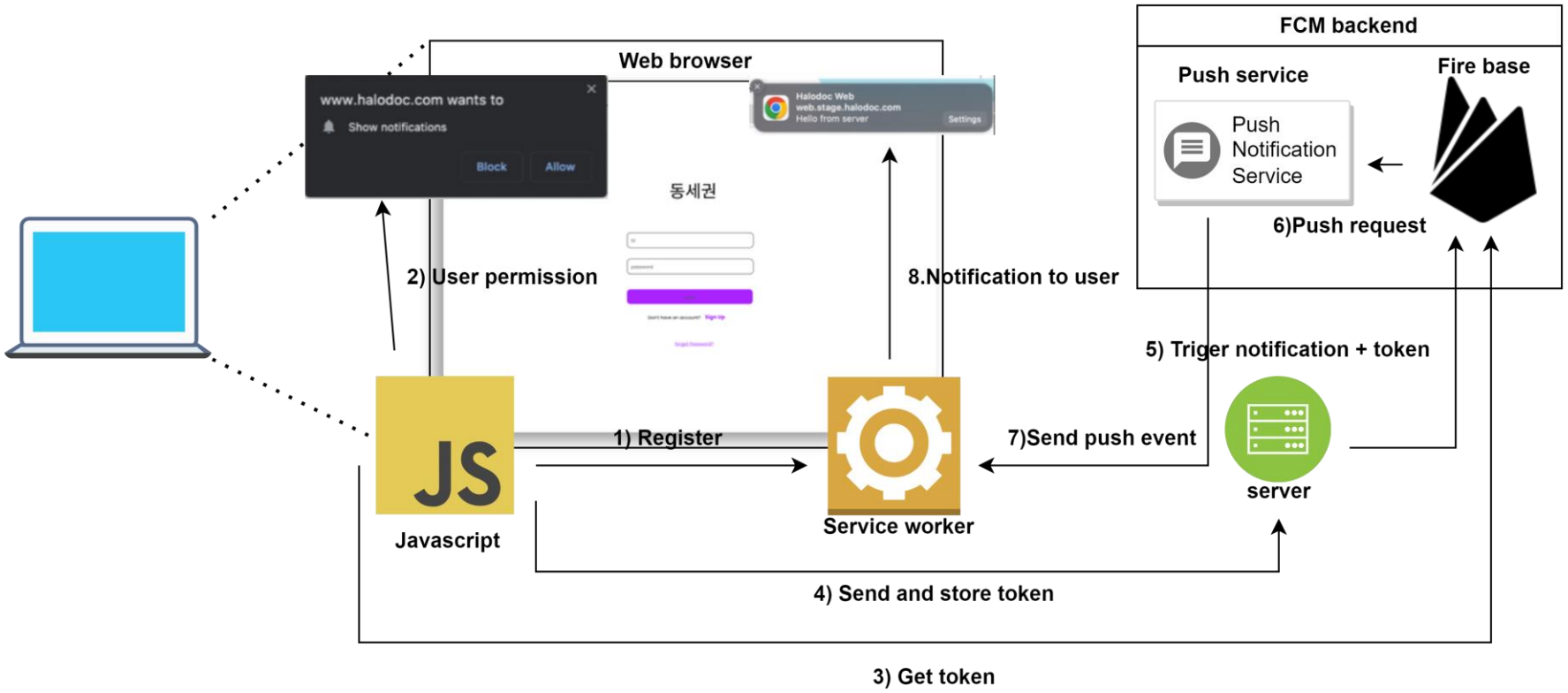
주요 서버 아키텍쳐 – User(주요기능) & Auth server



주요 서버 아키텍쳐 – Chatting & Matching server

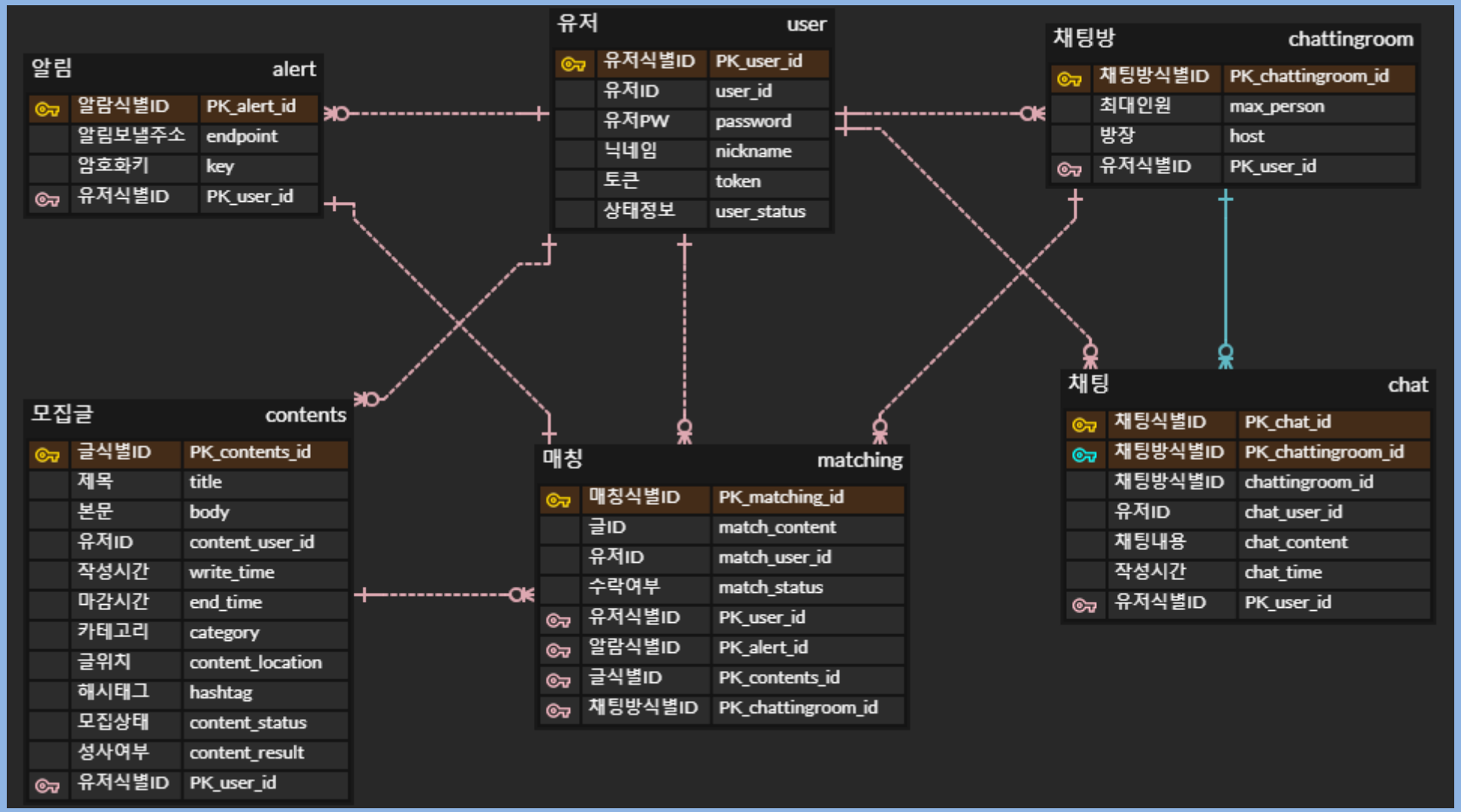


주요 서버 아키텍쳐 – Notification server



Notification architecture

ERD


알림 alert
알람식별ID PK_alert_id
알림보낼주소 endpoint
암호화키 key
유저식별ID PK_user_id
유저 user
유저식별ID PK_user_id
유저ID user_id
유저PW password
닉네임 nickname
토큰 token
상태정보 user_status
채팅방 chattingroom
채팅방식별ID PK_chattingroom_id
최대인원 max_person
방장 host
유저식별ID PK_user_id
모집글 contents
글식별ID PK_contents_id
제목 title
본문 body
유저ID content_user_id
작성시간 write_time
마감시간 end_time
카테고리 category
글위치 content_location
해시태그 hashtag
모집상태 content_status
성사여부 content_result
유저식별ID PK_user_id
매칭 matching
매칭식별ID PK_matching_id
글ID match_content
유저ID match_user_id
수락여부 match_status
유저식별ID PK_user_id
알람식별ID PK_alert_id
글식별ID PK_contents_id
채팅방식별ID PK_chattingroom_id
채팅 chat
채팅식별ID PK_chat_id
채팅방식별ID PK_chattingroom_id
채팅방식별ID chattingroom_id
유저ID chat_user_id
채팅내용 chat_content
작성시간 chat_time
유저식별ID PK_user_id

API 명세서

POST	/users/login	로그인
POST	/users/register	회원가입
POST	/users/logout	로그아웃
GET	/users/{userId}	유저의 정보 가져오기
GET	/users/{userId}/mypage	현재 로그인 된 사용자 정보 가져오기
PUT	/users/{userId}/mypage	현재 로그인 된 사용자 정보 저장하기
GET	/api/post	모집글 가져오기
POST	/api/post	모집글 작성하기
DELETE	/api/post	모집글 삭제하기
GET	/api/post/{postId}	모집글 상세 정보 가져오기
PUT	/api/post/{postId}/apply	모집글 신청하기
GET	/api/post/search	맵에 지정한 좌표계 위치에서 재검색하여 모집글 가져오기
DELETE	/api/post/{postId}/kick/{userId}	참여자 강퇴
PUT	/api/post/{postId}/accessed	참여자 참여 수락
PUT	/api/post/{postId}/denied	참여자 참여 거절
PUT	/api/post/{postId}/success-matching	매칭 확정
GET	/api/{alertId}/{userId}	알림 가져오기
DELETE	/api/{alertId}/{userId}	알림 삭제하기

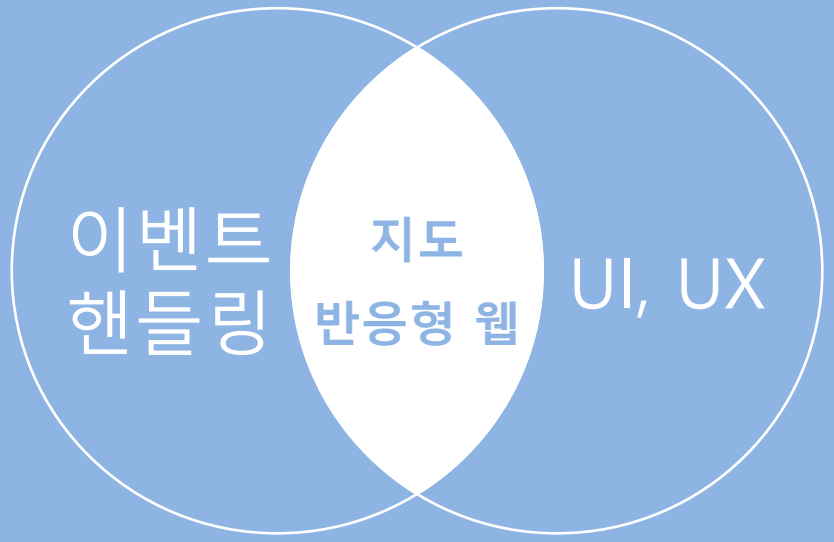
200 - OK	데이터 요청 정상
400 - Bad Request	요청이 비정상이므로 응답 불가
401 - Unauthorized	올바른 API가 아님
402 - Request Failed	어떠한 요인으로 요청 실패
403 - Forbidden	API를 요청을 수행할 수 있는 권한이 없음
404 - Not Found	요청한 리소스가 없음
429 - Too Many Requests	너무 많은 API요청이 있는 것
500, 502, 503, 504 - Server Errors	서버에 비정상적인 에러

업무 분담

업무 분담

프론트엔드

우명규 류나연

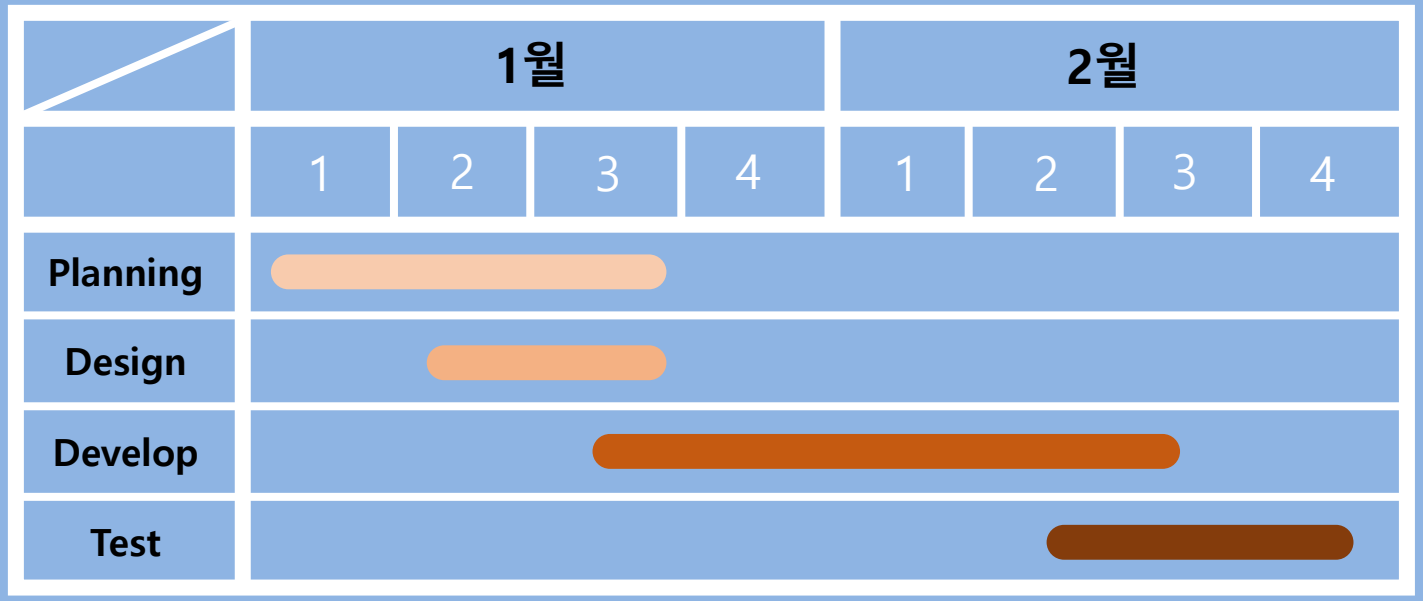


백엔드

전혜지
선원종

최성민

개발 계획



Planning (1/1 ~ 1/19)

PMP 작성

아키텍쳐 문서 작성

중간발표 준비

Design (1/9 ~ 1/19)

DB Schema 작성

화면 및 컴포넌트 구성

전체 및 개별 아키텍쳐 설계

API 명세서 정의

Develop (1/19 ~ 2/15)

유저, 인증 → 포스트, 알림 → 매칭 → 채팅, 채팅방 순서로 서버 구현

API gateway

레이아웃 틀 잡기 → 컴포넌트 기능 구현 → UI/UX로 FE 구현

REST API 프론트-백 간 연결

Test (2/15 ~ 2/21)

LCP 최적화

Jmeter의 TPS성능 측정

코드 피드백 및 리팩토링

최종발표 준비

ALL FE BE

Daily Scrum

Development

workflow

Step by Step

Weekly Meeting

Ground rules

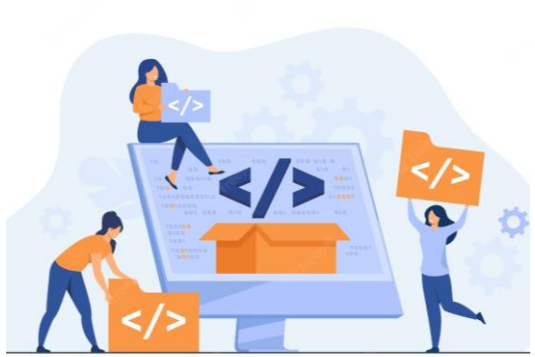
Daily Scrum

- 매일 평일 오전 11시에 Discord에서 모닝 스크럼 20분 진행

  팀원들의 건강 상태 체크하기
  → 프로젝트 진행에 영향을 끼칠만큼의 상태가 생길 경우 팀원들과 일정을 조율하여 to do 계획

  긍정적인 멘트 하나씩 말하며 팀원들간 격려하기
  → 프로젝트를 진행하면서 힘들고 지치겠지만, 서로간의 트러블을 줄이기 위해 멘트 하나씩 생각하기

  각자의 to do plan을 발표하는 시간 가지기
  → 작성한 to do plan이 팀목표/개인목표에 다가가기 위한 계획인지 되새기는 시간을 가지고 피드백도 주고받기



Development

- 순수 개발하는 시간인 CoreTime 최소 3시간 갖기

  당일 개발한 내용은 사소한거라도 항상 To do list에 기록하기
  → 프로젝트를 진행하면서 언제 어떻게 작업을 했는지 알아야하고, 그 내용이 언젠간 다시 필요해지는 순간이 오기때문에 기록 잘하기

  이슈에 대해 최소 1시간 이상 고민해봤음에도 해결되지 않을 시 팀원들에게 공유하기
  → 프로젝트를 하다가 문제가 생기는 부분이 반드시 올 것이지만
  먼저 혼자서 해결해보는 것이 중요하고 그래도 안되면 슬랙이나 카톡에 공유하기

  모르는 내용이라도 읽씹보다는 읽었다는 반응과 함께 문제 해결해보기
  → 읽씹을 하는 경우에는 사람들간의 소통이 되지않아 진척관리를 할 수 없다.
  또, 어떤 문제로 인해 계획에 차질이 생기는 것을 방지하기 위해 모두가 함께 문제를 해결하기

Ground rules

Weekly Meeting

- 매주 금요일 밤 10시에 Discord에서 정기회의 진행

  현재까지의 진행상황을 보고하고, 다음 1주간의 계획을 회의하기
  → 프로젝트를 정해진 시간 내에 마무리하기 위해서는 계획을 철저히 세운 후 그 계획에 맞게 구현을 해내는 것이 중요하기에
  목표에 다가가기 위해서는 프론트엔드-백엔드 간 어떤 계획을 세울 건지 회의 후 보고하기

  의견에 대한 의사결정을 할 때에는 뒷받침할 수 있는 근거와 투표로 결정하기
  → 그런 의사결정을 따랐을 때에 문제가 생겨도 선택한 결정에 불평하지 않고 해결방법을 우선으로 생각하기

  한 주간 개발된 내용이 목표에 근접하기 위한 내용인지 회고하고, 파트 간 피드백하기
  → 현재 진행되고 있는 내용이 목표를 달성하기 위한 것인지 다시 한번 생각하고
  프론트-백엔드 간의 교류를 하면서 기술의 이해도와 코드에 대한 피드백을 받는 시간 가지기

Step by Step

- 위와 같은 workflow와 Ground rules로 목표를 향해 차근차근 다가가기

  성능을 높이기 위한 테스트를 진행하기 위해서는
  개발 일정을 타이트하게 가져가야 했고
  빠르게 기능을 구현함으로써 성능 테스트에 좀 더 치중을 두기 위해
  위와 같은 개발계획과 workflow, Ground rules를 작성

toPangyo하기위해

포기하지 말고

프로젝트를 잘 완성해보자!

감사합니다