기억해조
(심희수, 최윤석, 최영지

Mill - Planning

언제 어디서든 자유롭게 예약하는 내 손 안의 식당

목차

팀 원 소 개

최 윤 석

Front-End developer

React
Javascript
Html/Css

최 영 재

Lead Developer
Back-End developer

NodeJS
Javascript
React

심 희 수

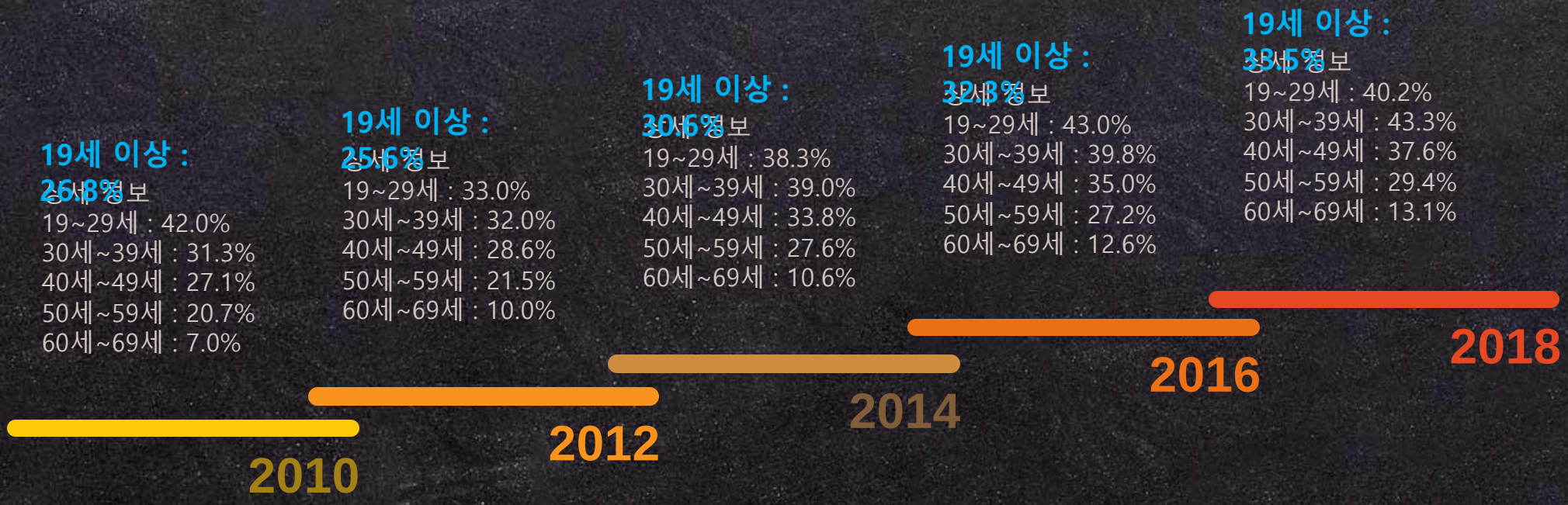
DB Engineer
Back-End developer

NodeJS
MYSQL
Javascript

개 요

웹 어플리케이션 개발 목적

하루 1회 이상 외식률 추이



참조자료 : 국민건강영양조사 (2008~2018)
하루 1회 이상 외식률 추이 (통계청)

언제까지 전화로 예약 해야할까?

번거로운 예약 절차
아직도 여전히 회식, 동창회, 송년회 등 예약 시 대부분 전화이용

업주 또한 번거롭긴 마찬가지
업주들도 바쁜 업무 시간 전화 안내 & 예약 목록 작성의 번거로움

간편하게 예약하고 예약 현황을 확인할 수 있는 앱 필요
Mill-planning 개발 아이디어

자영업자 두 번 울리는 NoShow족

코로나19 사태로 자영업자들의 경영난이 악화하는 가운데 이른바 노쇼 고객들로 인해 자영업자의 고충이 심화하고 있다. 배달이나 주문 예약을 하고서 나타나지 않아 발생하는 모든 피해를 업주들이 떠안는 상황이 빚어지기 때문이다.

이씨는 "과거에는 사전예약시 선입금 또는 선결제가 원칙이었지만 코로나19로 매출이 급감한 상황에서 고객이 케이크를 수령할 때 결제한다는 요청을 거절하기는 어려웠다"며 "주변에 아는 식당에서도 예약 접수가 들어와 음식 준비를 다 끝마친 상태에서 '노쇼'를 당해 전부 버렸다"고 토로했다. 이어 "요즘 같은 시기에 '노쇼'는 살아내려고 안간힘 쓰는 자영업자들을 눈물짓게 만든다"고 하소연했다.

출처 : 이데일리 헤드라인 뉴스
"안 그래도 힘든데".. '노쇼' 고객에 두 번 우는 자영업자
(2021.01.08)

허울 뿐인 예약보증금제도

실효성은 글쎄?

- **코로나 장기화로 어려워진 자영업자**
  NoShow 족은 매출에 더 큰 악영향
- **선 입금 선 결제가 권장될 뿐 시스템이 갖춰져 있지 않음**
  공정거래위원회의 예약보증금제도 유명무실
- **예약과정에서 자연스럽게 선수금을 받을 수 있는 앱 필요**
  Mill-Planning 개발 아이디어

선수금 수익 CYCLE

고객 예약 및 선수금 결제

고객이 식당을 예약하고 선수금을 결제한다

업주 확인 단계

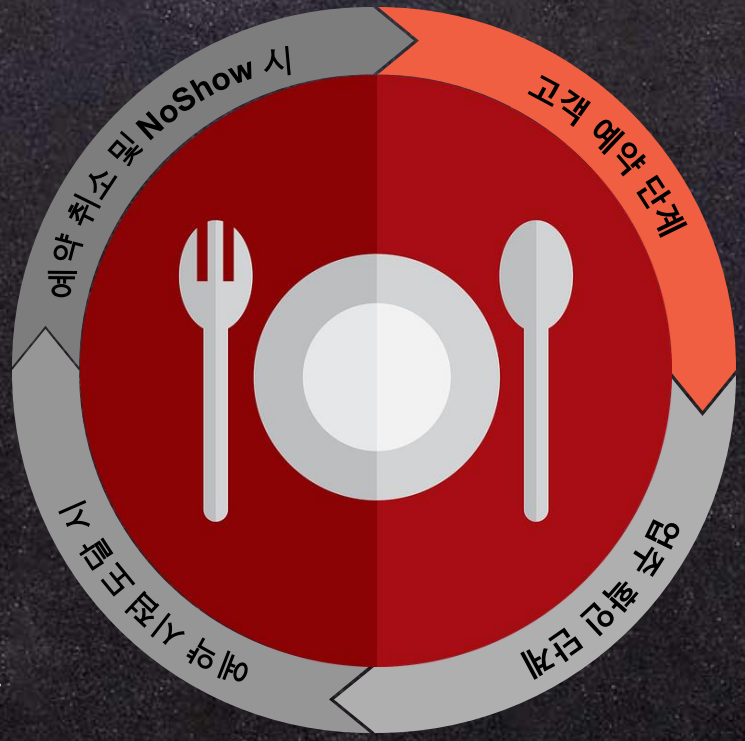
업주가 예약 일정을 확인하고 고객 응대를 준비하는 과정

예약 시점 도달 시

고객이 상점에 방문하여 식사를 정상적으로 마쳤을 시 선수금 중 일정 비율을 수취

예약 취소 및 NoShow 시

선수금을 점주에게 반환

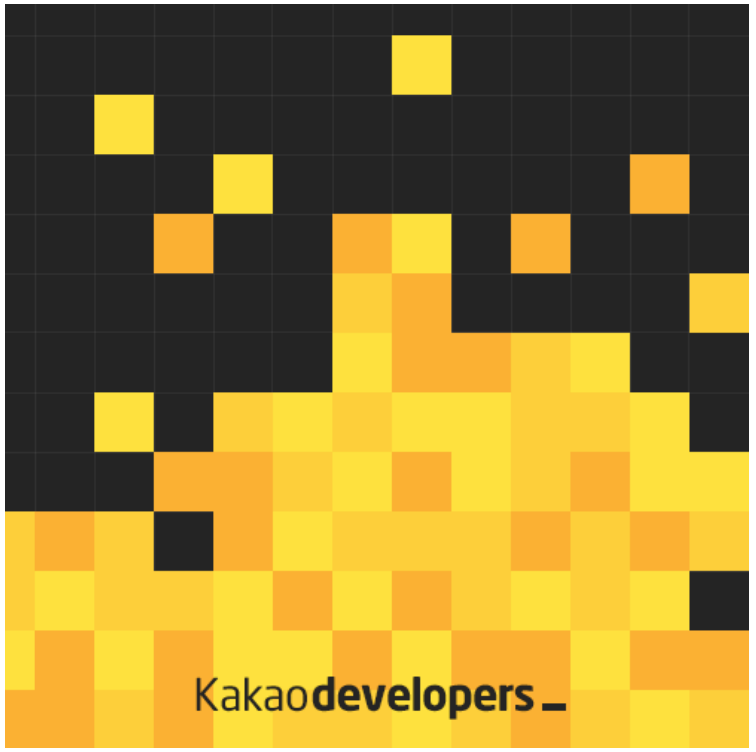


개발 과정

개발 History

Front-End(웹 클라이언트)
사용 기술 및 사용이유

Back-End(웹 서버)
사용 기술 및 사용 이유

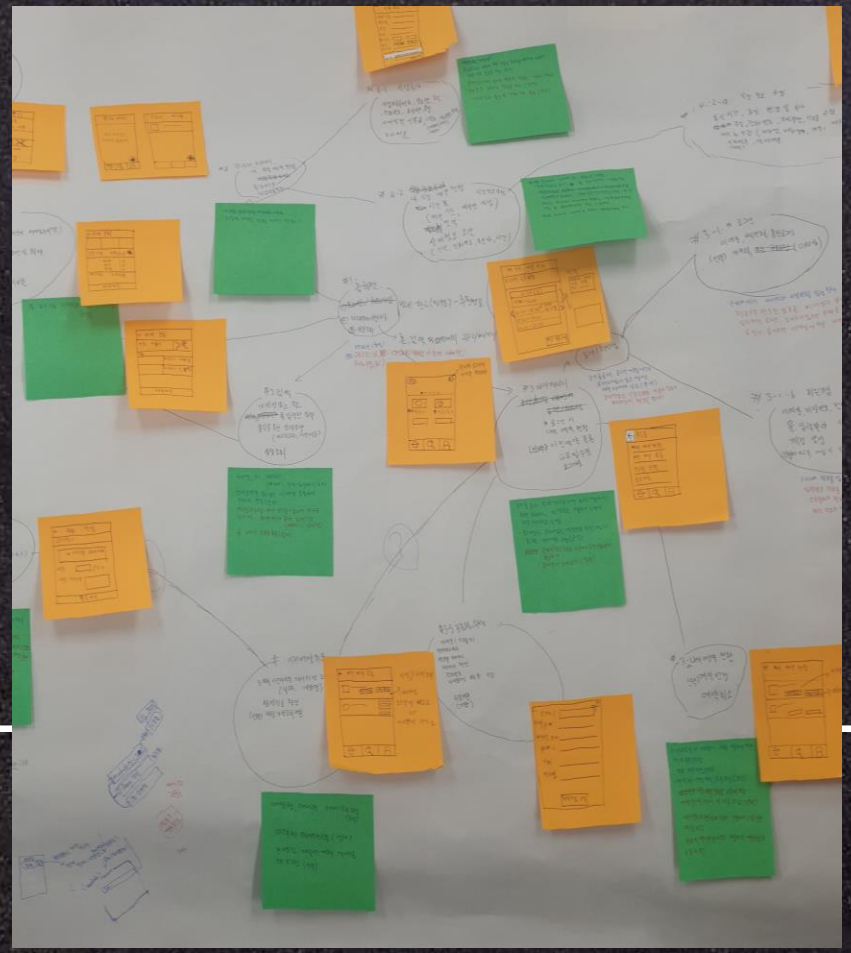


I'mport;

공통 (사용 기술 및 사용 이유)

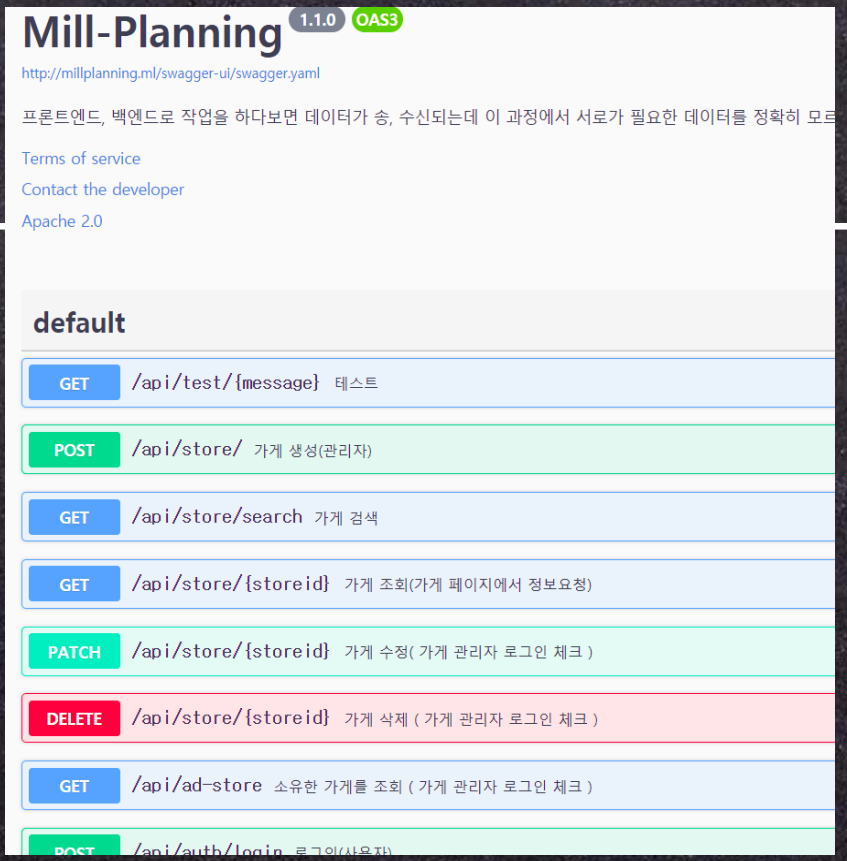
• 개발 과정

Brain Storming 을 통한 Idea 구체화



❖ 전체적인 Layout 및 페이지별 상세 기능을 목록으로 작성

❖ 기능에 따른 역할 분배

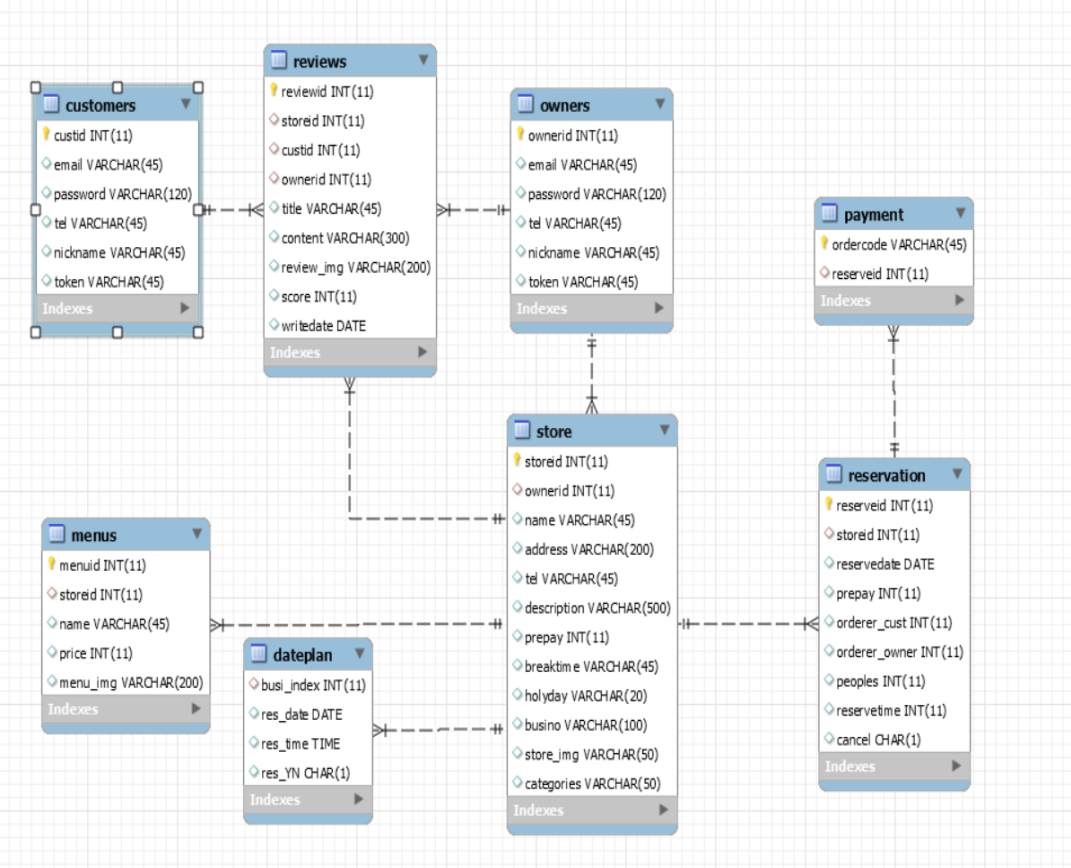


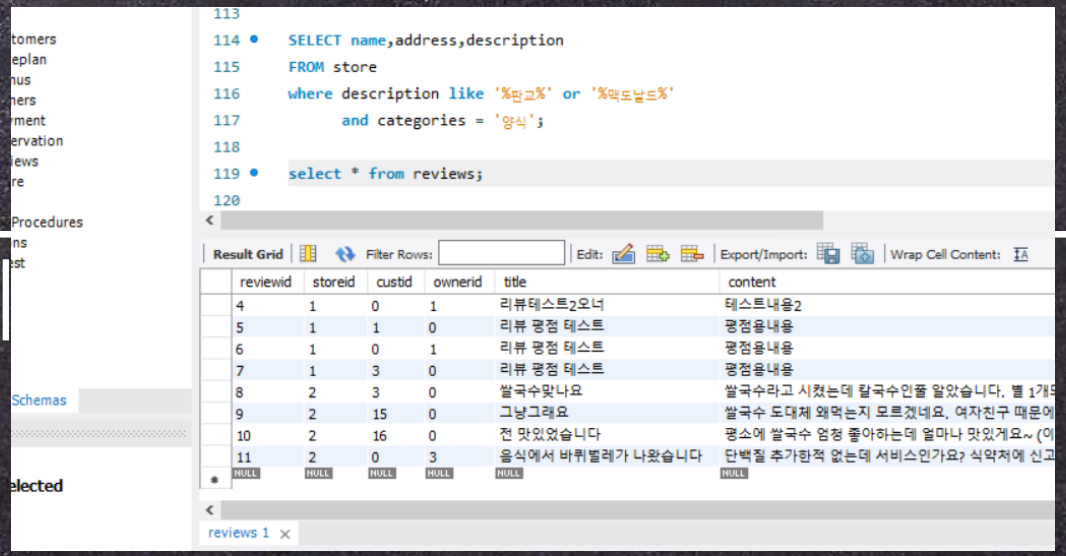
❖ 개발 문서 작성으로 용어 약속 및 혼선 예방

아이디어를 가시화해 개발 로드맵 작성

• 개발 과정

MySQL 을 이용한 DataBase 설계





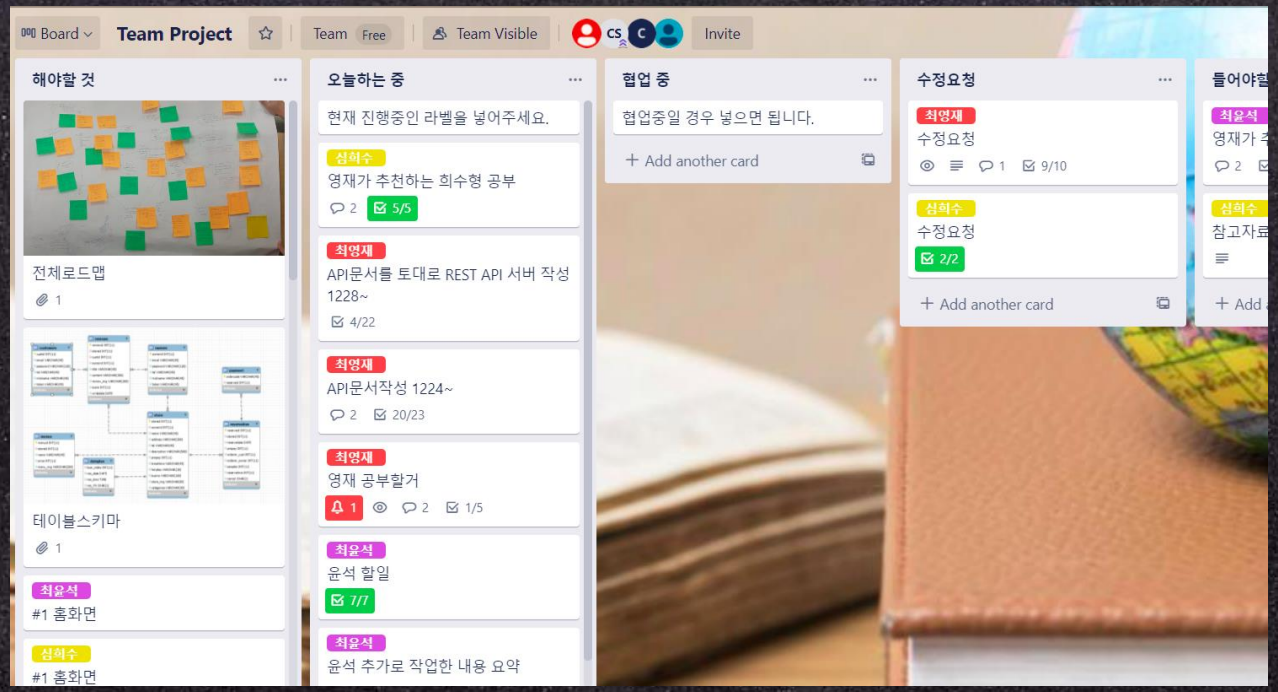
- 협업 Tool로 MySQL Workbench 사용

- 검색 효율을 최우선으로 테이블의 간소화
- 칼럼명은 직관적으로 통일

서버개발자,DB개발자,프론트개발자 각각의 스키마로 분리 운용

• 개발 과정

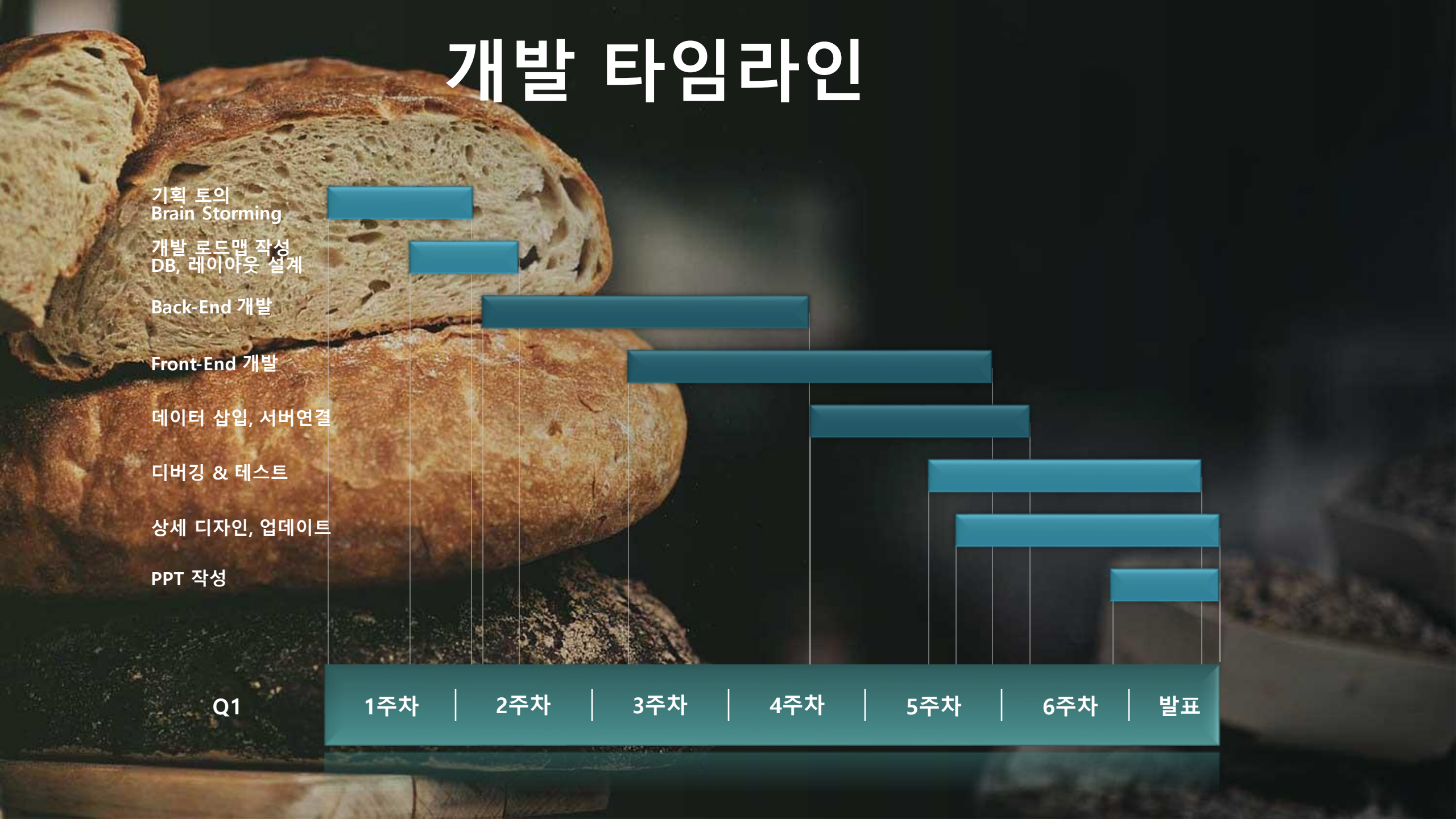
Agile을 활용하기 위해 Trello 사용



- ❖ NameTag를 이용해 각자의 역할 분배
- ❖ HotFix 등의 중요 Issue를 처리하기 위한 List 사용

개발 과정은 실시간으로 분배 및 확인

개발 타임라인


기획 토의
Brain Storming
개발 로드맵 작성
DB, 레이아웃 설계
Back-End 개발
Front-End 개발
데이터 삽입, 서버연결
디버깅 & 테스트
상세 디자인, 업데이트
PPT 작성
Q1
1주차
2주차
3주차
4주차
5주차
6주차
발표

기능설명

웹 어플리케이션 기능 상세

기능소개(소비자)

카카오 로그인

사용자 정보는 서버의 세션에
프론트에는 세션을 조회할 수 있는 쿠키를
전달

카카오 로그인
기능설명

예약하기
예약확인 및 취소

리뷰쓰기
기능설명

기능소개(소비자)

	imp_uid(클릭하면 영수증 보기) merchant_uid	결제금액 (취소금액)	구분 (클릭)	PG사 (상점아이디)
38	imp_499225738964 MILLP1-1611649224508 웹훅로그 부가 데이터	결제완료 ₩3,000 (₩0)	pc 체크카드	이니시스(웹표준) (INIpayTest)
37	imp_383213673094 MILLP64-1611648105639 웹훅로그 부가 데이터	결제실패 ₩18,000 (₩0)	pc 신용카드	이니시스(웹표준) (INIpayTest)
36	imp_049138580393 MILLP1-1611649136208 웹훅로그 부가 데이터	결제완료 ₩3,000 (₩0)	mobile 체크카드	이니시스(웹표준) (INIpayTest)

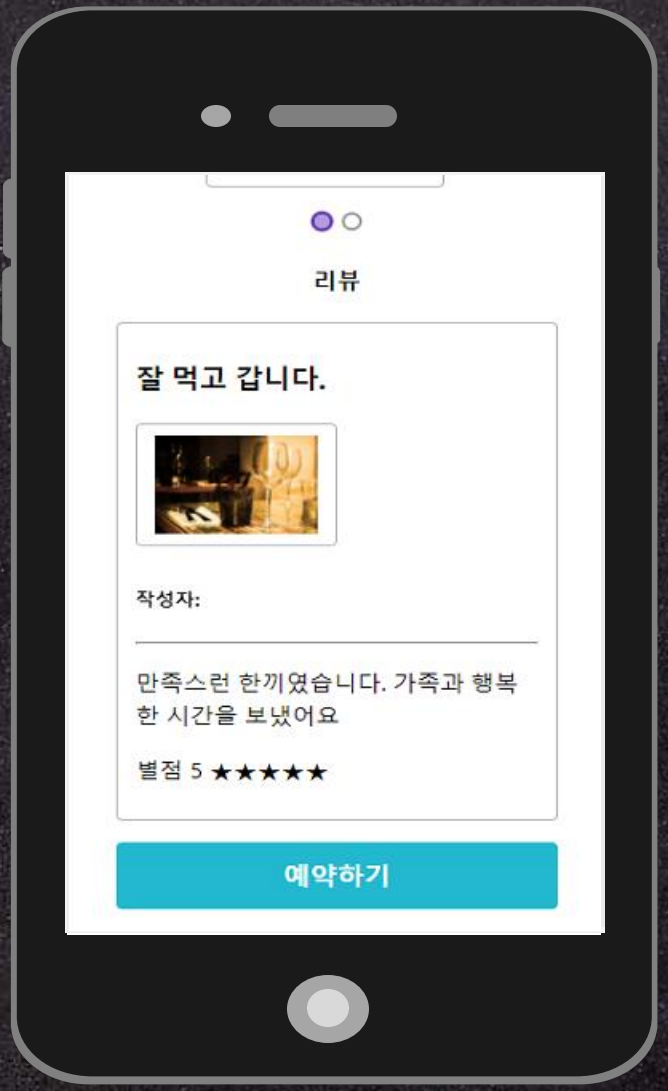
결제 모듈 사용,
결제 모듈에 결제된 결제코드를 기반으로
데이터베이스에 있는 예약 정보 조회

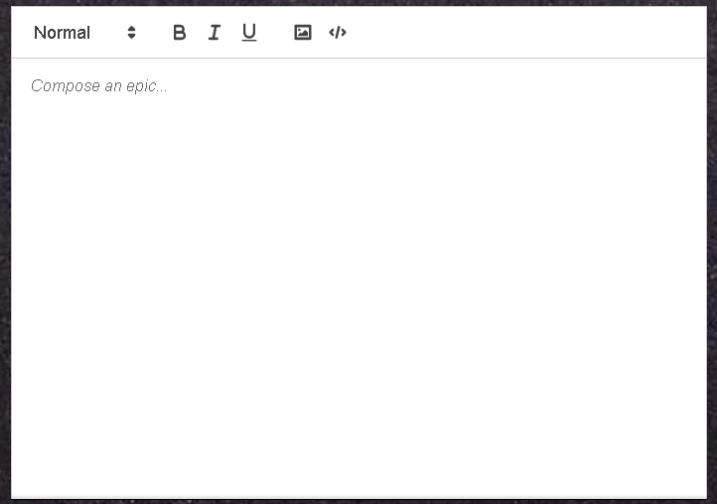
☑ **카카오 로그인**
기능설명

☑ **예약하기**
예약확인 및 취소

☑ **리뷰쓰기**
기능설명

기능소개(소비자)





예약한 가게에 한해 리뷰작성 가능
게시판 에디터 (quill) 적용
리뷰와 함께 별점 적용

카카오 로그인
기능설명

예약하기
예약확인 및 취소

리뷰쓰기
기능설명

기능소개(관리자)

가게 등록
기능설명

예약자 관리
예약확인 및 취소

관리자 로그인시 홈페이지에 관리자 버튼 노출
관리자는 자신이 관리할 가게를 생성 가능
카테고리 설정으로 검색노출 활성화

기능소개(관리자)

※일반 유저 MyPage

관리자 로그인시 MyPage - 내 가게 관리 탭 노출
현재 들어온 예약 목록 확인
클릭시 상세 예약 정보 Modal로 확인가능

가게 등록
기능설명

예약자 관리
예약확인 및 취소

시연

웹 어플리케이션 기술 시연

시연(데이터 베이스)

```
//3개월 이전의 데이터까지
rsv_3month_before: async function (rsv_data) {
    const { custid, ownerid } = rsv_data;

    let userid = custid;

    let query = 'SELECT res.reserveid as reserveid, ' +
        'res.storeid as storeid, ' +
        'store.name as name, ' +
        'res.prepay as prepay, ' +
        'res.reservedate as reservedate, ' +
        'res.reservetime as reservetime ' +
        'FROM reservation as res, store as store ' +
        'WHERE orderer_cust = ? AND ' +
        'reservedate < NOW() - INTERVAL 1 DAY AND ' +
        'reservedate > NOW() - INTERVAL 3 MONTH AND ' +
        'res.storeid = store.storeid ' +
        'ORDER BY reservedate asc';

    if (custid <= 0 && ownerid <= 0) return { errno: "잘못된 데이터정보입니다." }

    //관리자로 로그인된 사용자가 조회를 요구하였을 경우
    if (custid <= 0) {
        query = query.replace('orderer_cust', 'orderer_owner');
        userid = ownerid
    }

    const connection = await connect();
    if (connection.error) return { errno: "connection failed" };

    try {
        const [rows] = await connection.query(query, [userid]);

        return rows;
```

❖ 기간 별 예약 데이터 조회 모듈

시연(데이터 베이스)

```
insert_rsv: async function (store) {
    if (!store) return { errno: "store is null" };

    const connection = await connect();
    if (connection.error) return { errno: "connection failed" };

    try {
        const query =
            'INSERT INTO reservation(' +
            'storeid,' +
            'reservedate,' +
            'prepay,' +
            'orderer_cust,' +
            'orderer_owner,' +
            'peoples,' +
            'reservetime) ' +
            'VALUES (?,?,?,?,?,?,?)';

        const [rows] = await connection.query(
            query,
            [store.storeid,
            store.reservedate,
            store.prepay,
            store.orderer_cust,//custid
            store.orderer_owner,//ownerid
            store.peoples,
            store.reservetime
            ]
        );

        const lastIndex = await connection.query('SELECT LAST_INSERT_ID() as insertid')

        return lastIndex[0][0].insertid;

    } catch (error) {
```

❖ 예약정보 입력

시연(백엔드)

```
exports.Kakao = async function (req, res) {
    const { admin } = req.params

    if(admin) {
        req.session.admin = true;
        return res.redirect('/api/auth/oauth/kakao');
    }
    passport.authenticate('kakao', (err, account) => {
        if (err) return res.status(500).json(err);
        const redirectPage =
            '<meta http-equiv="refresh" content="1;url=http://millplanning.ml/MyPage"></meta>'

        req.login(account, (error) => {
            if (error) return res.status(500).json(error);
            const token = jwt.sign({
                custid: account.custid,
                ownerid: account.ownerid,
                oauth_token: account.oauth_token,
                admin: account.admin,
            },
                global.secret,
                { expiresIn: 60 * 60 * 24 }
            )

            if(!token || '') return res.send(redirectPage + '<div>로그인에 실패하셨습니다</div>')
        })


        res.send(redirectPage + '<div>잠시 후 페이지로 넘어갑니다</div>')
    })(req, res);
}
```

❖ 카카오 계정 로그인

시연(백엔드)

```
const bcrypt = require('bcrypt')
const saltRounds = 12

/**
 * 비밀번호를 암호화하여 salt값과 함께 반환
 */
exports.genEncryption = (password) => {
    const salt = bcrypt.genSaltSync(saltRounds)
    const hash = bcrypt.hashSync(password, salt)

    return { salt, hash }
}

/**
 * 평문과 암호화된 비밀번호를 비교하여 올바른 패스워드인지 확인
 */
exports.verifiEncrypt = (password, saltedPassword) =>
                bcrypt.compareSync(password, saltedPassword)
```

❖ 패스워드 암호화

시연(프론트엔드)

```
<Iamport
  identificationCode={importConfig}
  params={{
    pg: "html5_inicis",
    pay_method: "card",
    merchant_uid: `MILLP${storeid}-${new Date().getTime()}`,
    name: `${findStoreInfo.store.name}`,
    amount: findStoreInfo.store.prepay,
    buyer_email: `${!account.account || '' ? '' : account.account.email}`,
    buyer_tel: `${!account.account || '' ? '' : account.account.tel}`,
    buyer_name: `${account.account.nickname}`,
    m_redirect_url: `${domain.default}api/customers/mobile/reserve/${storeid}`
  }}
  onFailed={err => {
    props.history.push(`/StorePage/${storeid}`)
  }}
  onSuccess={res => {
    const resData = {
      imp_uid: res.imp_uid,
      merchant_uid: res.merchant_uid,
      name: res.name,
      amount: res.amount,
      buyer_name: res.buyer_name,
      buyer_tel: res.buyer_tel,
      storeid: storeid,
      reservedate: new Date(Date.now() + (date * 86400 * 1000)).format('yyyyMMdd'),
      prepay: findStoreInfo.store.prepay,
      peoples: 1,
      reservetime: time * 10000
    }
    dispatch(resAction.doReserve(resData)).then((data) => {
      props.history.push(`/CompletePage/${data.payload.data.insert_reserve}`)
    })
  }}
```

❖ 결제 모듈 연결

시연(프론트엔드)

```
function Base64toServerImage() {
    const changeStr = value.split('>').map(v => {
      if(v.includes("<p")) {
       return v + '>'
     } else if(v.includes("</p")) {
     return v + '>'
     } else if(v.includes("<img")) {
       return false
     } else {
     return false
     } } ).filter(v => v !== false).join('')

     return changeStr
  } // <p><img ~~~/></p> => <p></p>

const DataURIToBlob = function (dataURI) {
    const splitDataURI = dataURI.split(',')
    const byteString = splitDataURI[0].indexOf('base64') >= 0 ? atob(splitDataURI[1]) : decodeURI(splitDataURI[1])
    const mimeString = splitDataURI[0].split(':')[1].split(';')[0]
    const ia = new Uint8Array(byteString.length)
    for (let i = 0; i < byteString.length; i++)
        ia[i] = byteString.charCodeAt(i)

    return new Blob([ia], { type: mimeString })
}

const searchSrc = () => {
    const arr1 = value.split('img').map(v => v.includes('src') === true && v.split("src="));
    const arr2 = arr1.map(v => v && v[1]?.split("></p"))
    return arr2.map(v => v && v[0].slice(1, v[0]?.length - 1)).filter(v => v !== false);
  } // "data:image/png;base64~~~"
```

❖ 리뷰 이미지 파일 변환

구현하지 못한 것들

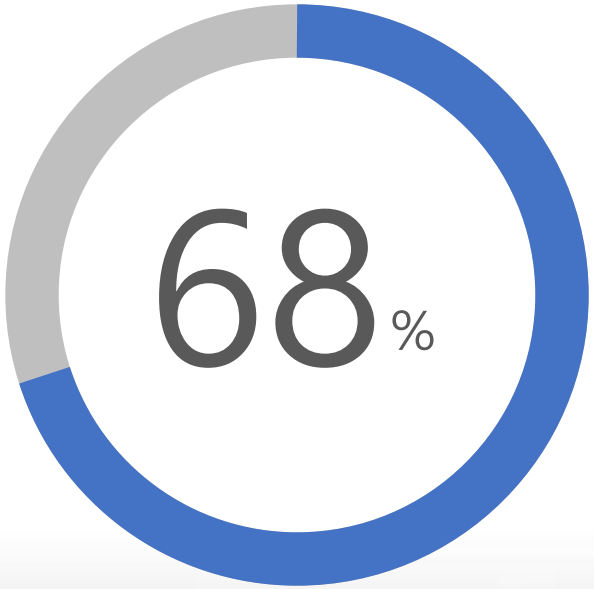
- 예약한 손님의 NoShow 비율을 가게에 공지하여 관리자가 NoShow율이 높은 손님의 예약을 거절하거나 차등적인 선수금을 부과할 수 있도록 하는 기능
- 관리자의 가게 정보 수정(등록된 선수금이나 휴일, BreakTime 등을 변경하는 기능)
- 카카오에서 지원하는 카카오채널 또는 알림톡 기능을 사용하여 이용자가 예약을 진행하였을 때 해당하는 정보를 이용자의 핸드폰에 전송

향후 추가 기능

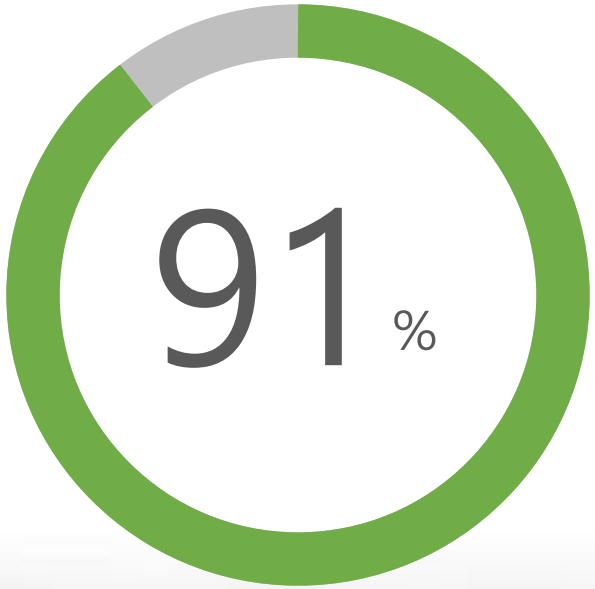
- 주소 API를 이용하여, 가게 생성시 주소를 정확하게 검색할 수 있도록 하는 기능
- 지도 API를 이용하여 예약시간에 실제로 고객이 방문을 하였는지를 판단하는 기능 추가
- 문서 자동화 세팅으로 개발 시 클라이언트와 서버 간의 데이터 응답과 처리의 혼동이 없게끔 설정
- 휴대폰 SMS 인증서비스로 이용자의 실제 휴대폰을 통해 인증을 요구하여 무분별한 가입을 제한하는 기능
- 현재 웹으로 작동을 하여 모바일 웹에서 작동 수행이 가능하게 되어있으나, 차후 모바일 앱에서 작동할 수 있게끔 방향성을 변경

자체 평가

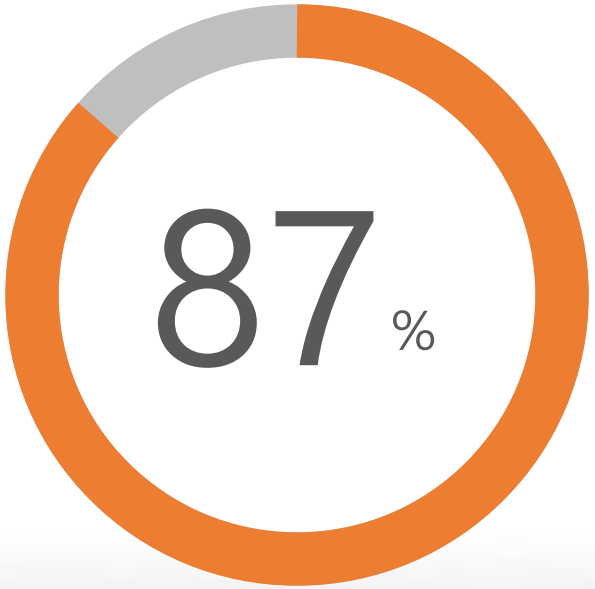
프로젝트 달성률



Front-end



Back-end



Server


THANK YOU
Insert the Sub Title of Your Presentation