



최종발표 TELLING ME

대학생의 자아정체성 확립을 위한 문답 기록형 플랫폼



목차

1

개요 및 목적

2

배경과 과정

3

기능 요구명세

4

설계 및 구현

5

실행 결과

6

개선 사항 및 향후 계획

프로젝트 개요

프로젝트 개요

프로젝트 소개

대학생 시기 진지한 고민들을 이끌어낼 수 있는 질문을 통해 스스로에 대해 생각할 시간을 제공합니다.
또한 답변을 서로 공유/공감하여 같은 상황에 놓인 이들 간 위로의 소통공간을 마련합니다.
이용자의 솔직한 생각을 정리하고, 궁극적으로 가치관 탐색에 도움을 주는 서비스입니다.

개발 동기

대학생 시기는 사회인이 되기 직전 단계로, 자아정체성 탐색이 매우 중요한 시기입니다.
하지만 현재 스스로를 제대로 알지 못하고, 혼란을 겪고 고민에 빠지는 대학생이 굉장히
많으므로, 이들의 가치관 형성을 돕는 플랫폼이 필요하다고 생각했습니다.

배경과 과정

배경과 과정

배경

본인에 대해 잘 알고자 고민하는 20대의 비율은 과반수 이상

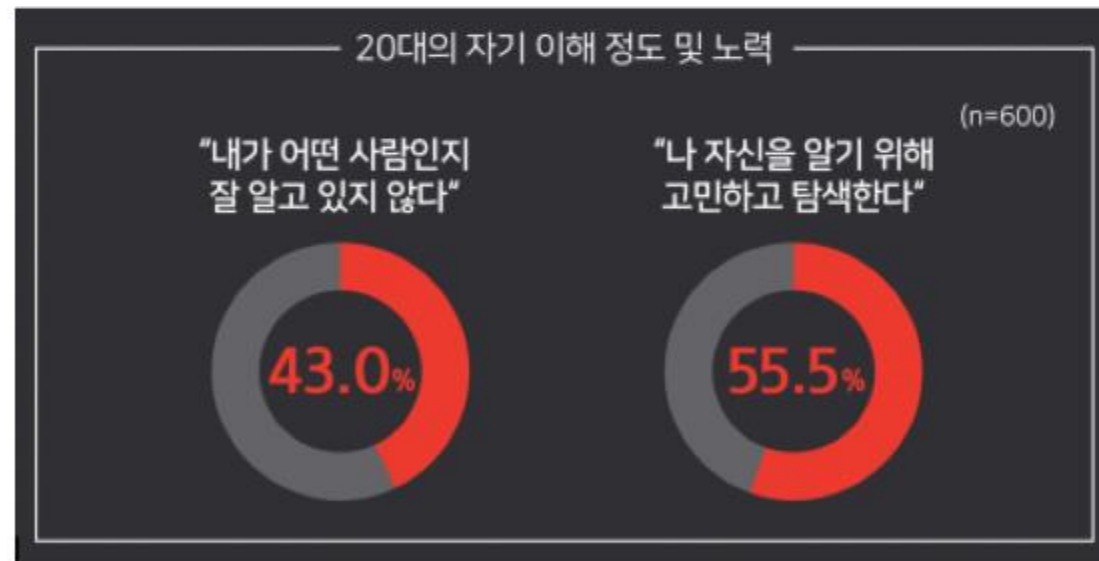


그림 5. [대학내일20대연구소] 20대의 자아에 대한 인식 및 자아 만족 추구 성향 조사 (연구리포트)

배경

대학생 자아탐색 및 심리건강 증진에 대한 대학 및 정부에서 중요성 강조



배경과 과정

질문의 일관성 부족, 비속어 다수 발견 등 질적 관리의 부족
나의 답변만 볼 수 있어서 타 유저들과 답변을 공유할 수 없음
문장 빈칸에 단어 채우기로 답변의 자율성에 제한이 있음
데이터 백업에 대한 제한적 이용

현재 시장

Reflectly



AI 프리미엄 모드를 통한 기분
상관관계 및 그래프와 개인화된
질문 제공

현재 시장

하루콩

하루를 기록하는
가장 간단한 앱, 하루콩!



매일의 기분과 활동을 트렌디한
디자인으로 기록 및 분석

현재 시장

세줄일기

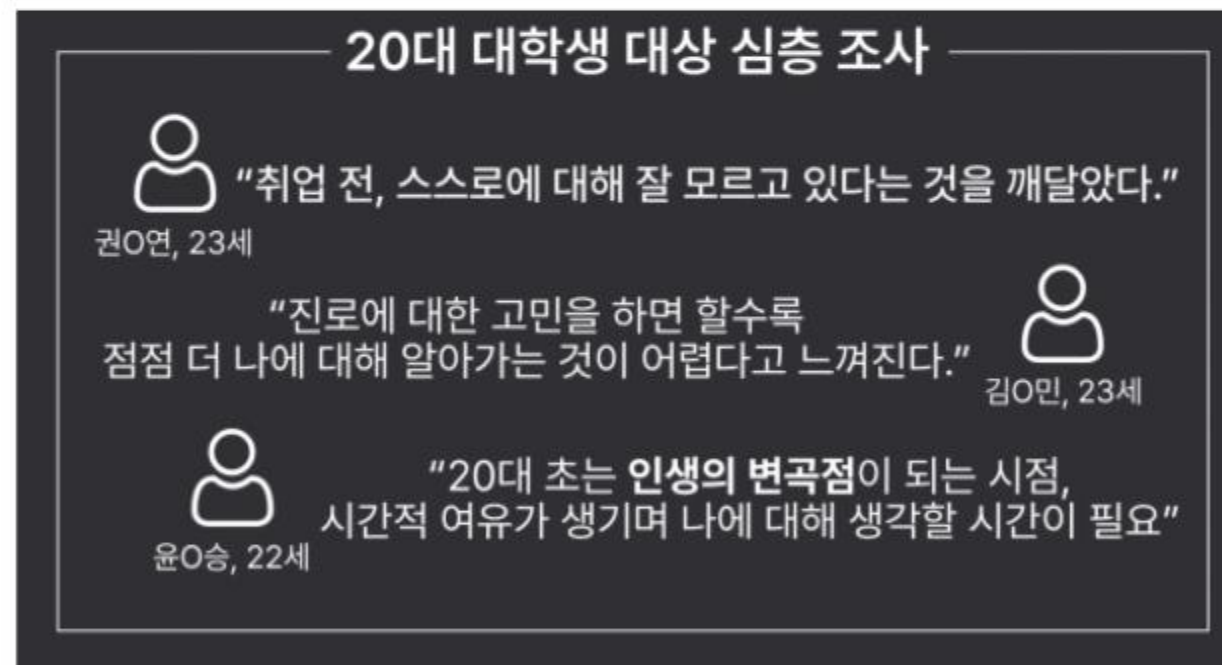
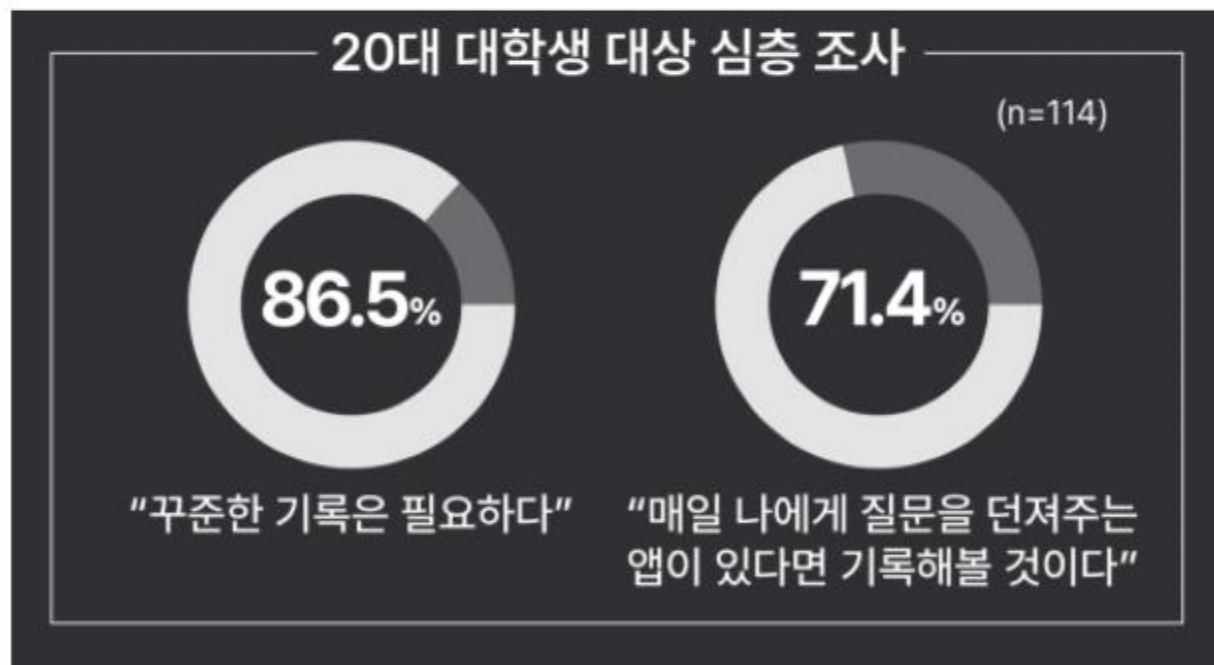


세 줄의 글과 한 장의 사진으로
이루어진 일기, 소통공간

배경과 과정

설문 조사

설문 조사와 심층 인터뷰 결과 (2023.01~2023.02)



기능 요구 명세

기능 요구 명세

| | | | |
|-------------|--|--|--|
| 랜딩 페이지 | | 서비스 이용 방법 소개 | |
| 로그인 | | 소셜 로그인(카카오, 애플) | 1) 애플 개발자 계정 필요 2) 애플 로그인은 https 통신만 가능 |
| 서비스 이용 약관 | | 개인정보 처리방침 | |
| 개인정보 DB | | 1) 닉네임 2) 고민 3) 직업 4) 성별 5) 생년월일 6) MBTI 7) 알림 허용 *iOS고려해서 순서변경 | |
| [왼쪽탭] 나의 공간 | | 1) 리스트 보기 2) 카드섹션 리스트 보기 3) 개별답변 보기 4) 설정 | 1) 드롭박스로 월별 페이징 처리 감정 왼쪽에, 질문제목 오른쪽에 배치 2) 지난 기록 수정/삭제하기 |
| [가운데탭] 홈 | | 1) 기록 연속 날짜 2) 오늘의 질문 3) 답변 작성하기 버튼 | -하루에 한 번, 오전 6시에 질문 업데이트 -푸시알림은 오전 12시 한번 |

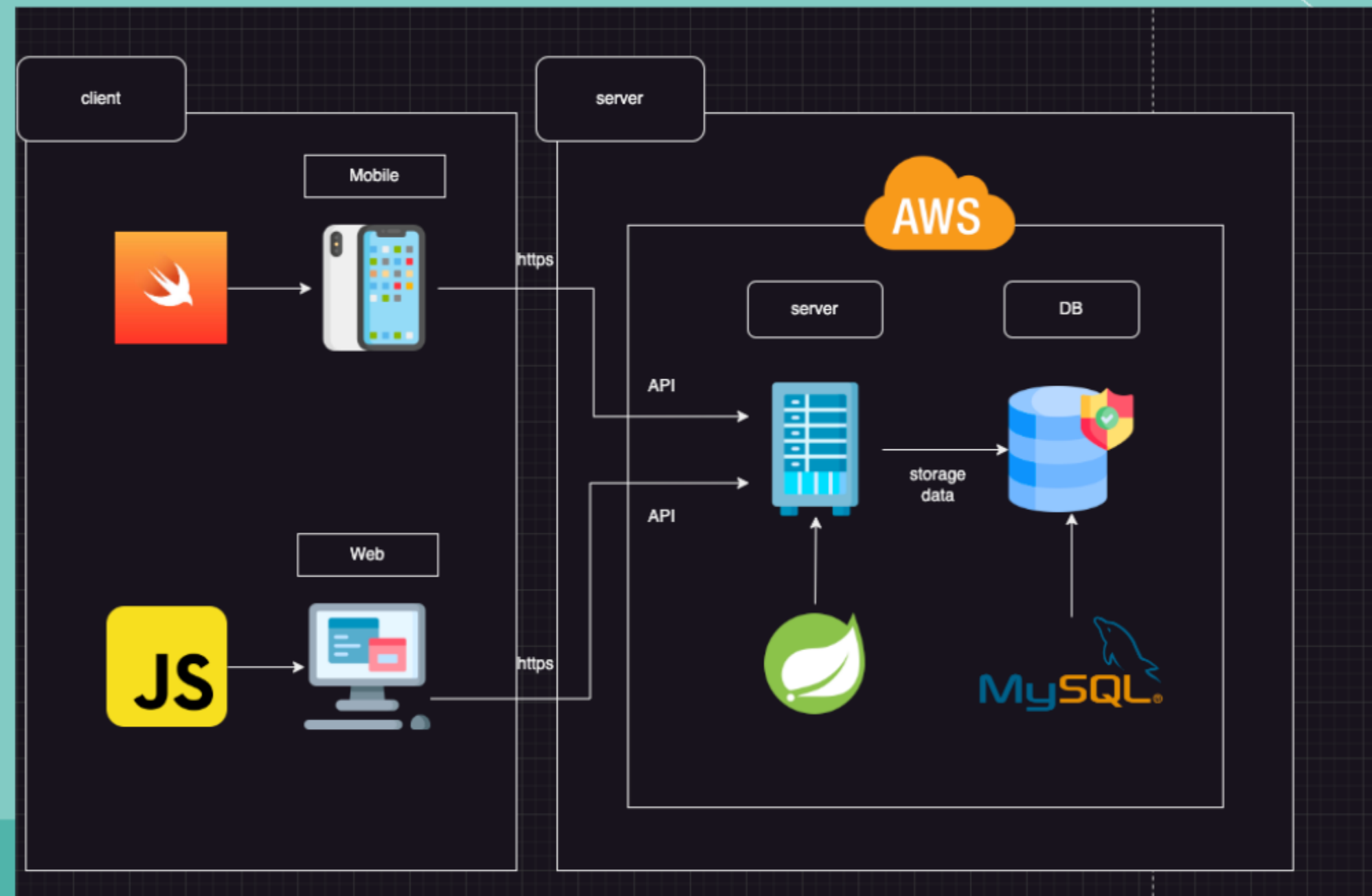
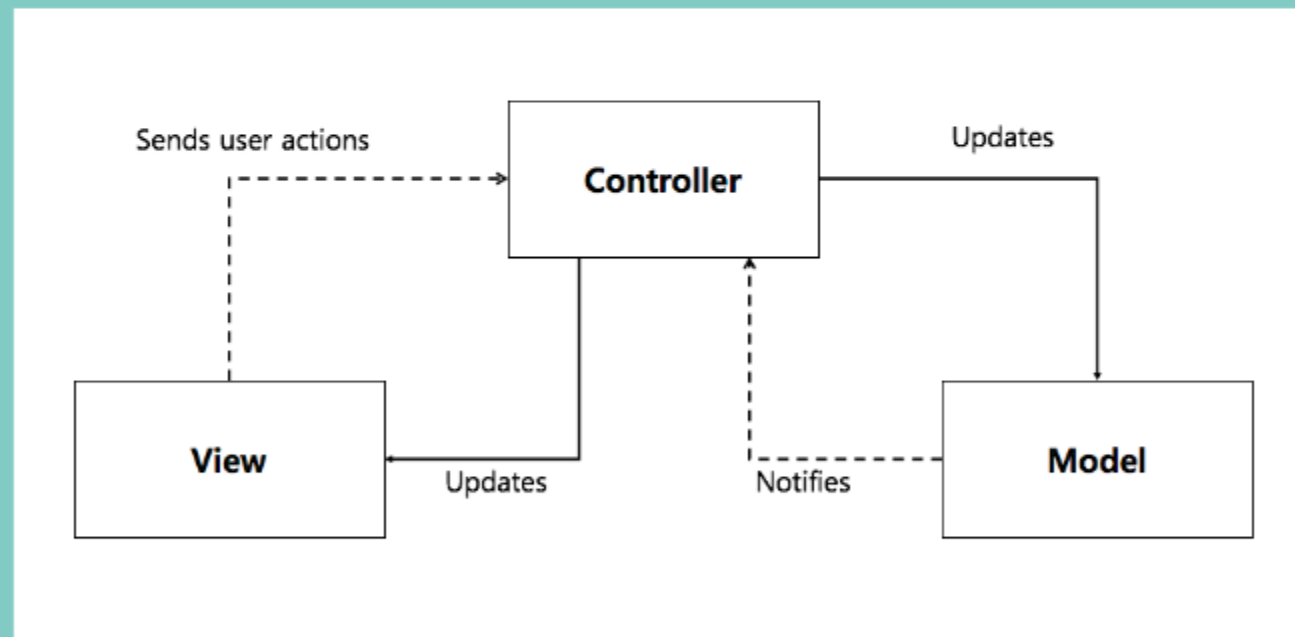
기능 요구 명세

| 감정 기록 모달 | | 감정 이모지 12중 택1 | 기본모드 6개 (+프리미엄모드 6개 추가) |
|---------------|--|---|--|
| 답변 | | 1. 질문에 대한 답변 작성하기 2. 답변 저장하기 <ul style="list-style-type: none"> - 답변 작성 중 감정 변경 가능 - 공개 → 소통 공간으로 이동 - 비공개 → 나의 기록으로 이동 - 저장 후 수정/삭제 가능 - 질문 내용 fold 가능 | [기본모드] 공백 제외 300자 제한 [프리미엄모드] 공백 제외 500자 제한 (늘어날 가능성 有) 감정은 한 번 저장하면 변경할 수 없음 답변 수정은 텍스트만 변경 가능 |
| [오른쪽탭] 모두의 공간 | | 1) 타인의 답변 목록 2) 정렬 기능 3) 타인의 답변에 공감 기능 4) 신고기능 | - 최신순, 관련순, 공감순 (기본은 최신순 정렬) - 관련순 <ul style="list-style-type: none"> • 알고리즘 순서 : <ol style="list-style-type: none"> 1. '고민'과 '직업'이 같은 사람 2. '고민'이 같은 사람 3. '직업'이 같은 사람 cf. 고민과 직업 카테고리 모두 '기타'도 동일한 가중치를 두고 질문이 배정됨 |
| 설정 | | 1) 개인정보 수정 2) 서비스 이용 약관 보기 3) 로그아웃 4) 회원 탈퇴 5) 프리미엄 결제 기능 (앱 내 결제) 6) 푸시알림 동의/비동의 | 개인정보 수정 - 닉네임, 고민, 직업, MBTI, 푸시알림 설정 변경 하나씩 누르면 각 처음 설정 페이지 뜨게 함 |

설계 및 구현

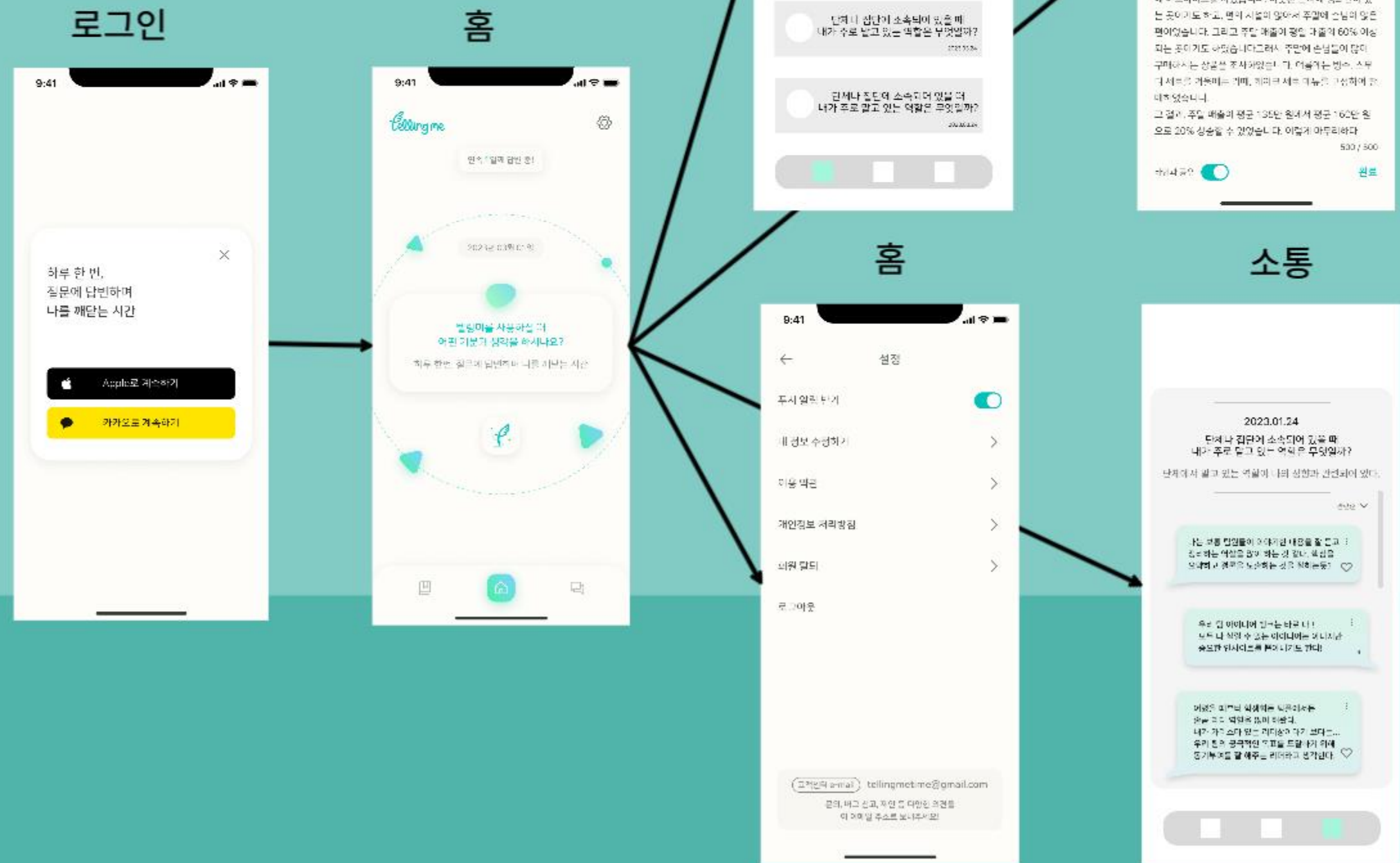
설계 및 구현

App Architecture



설계 및 구현

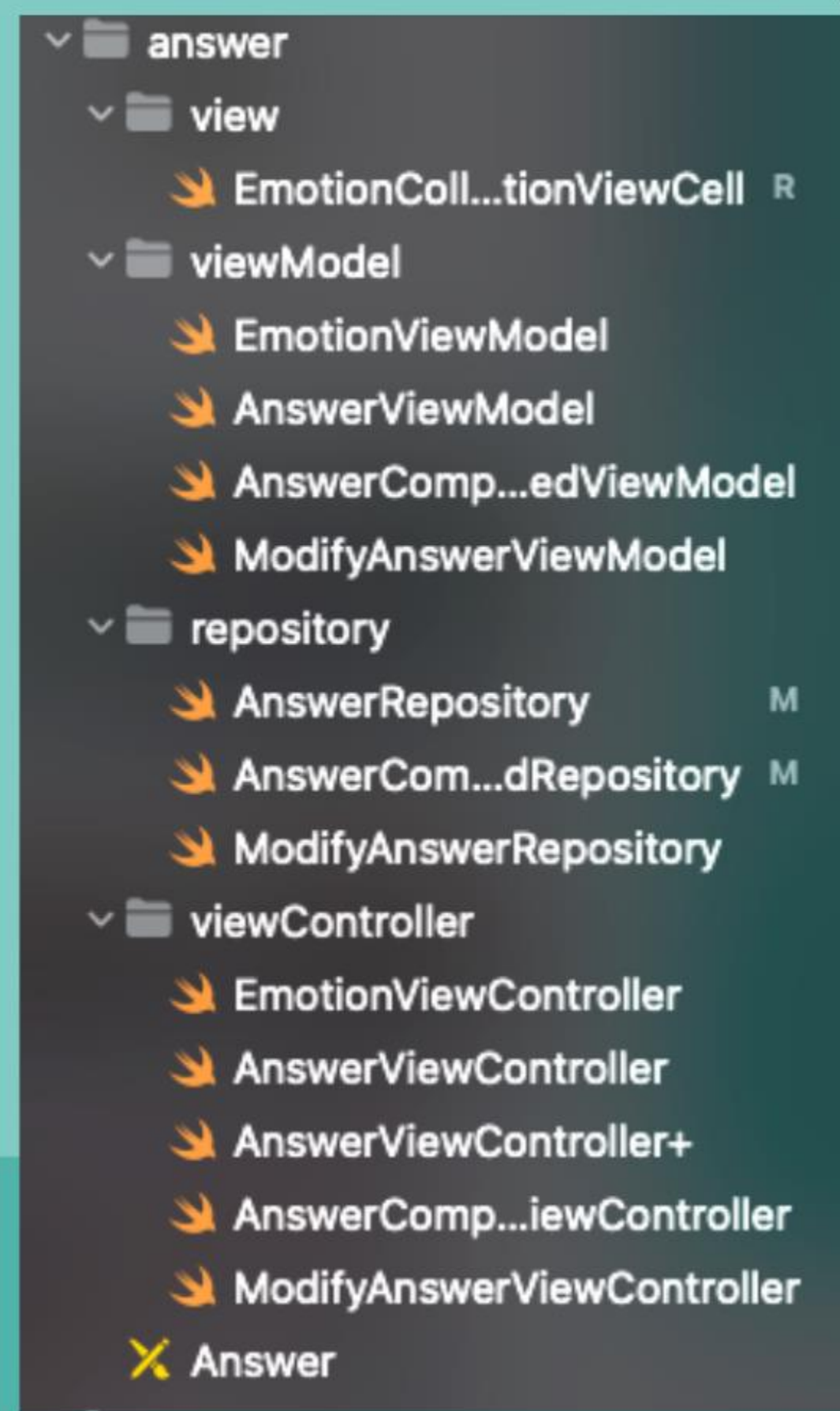
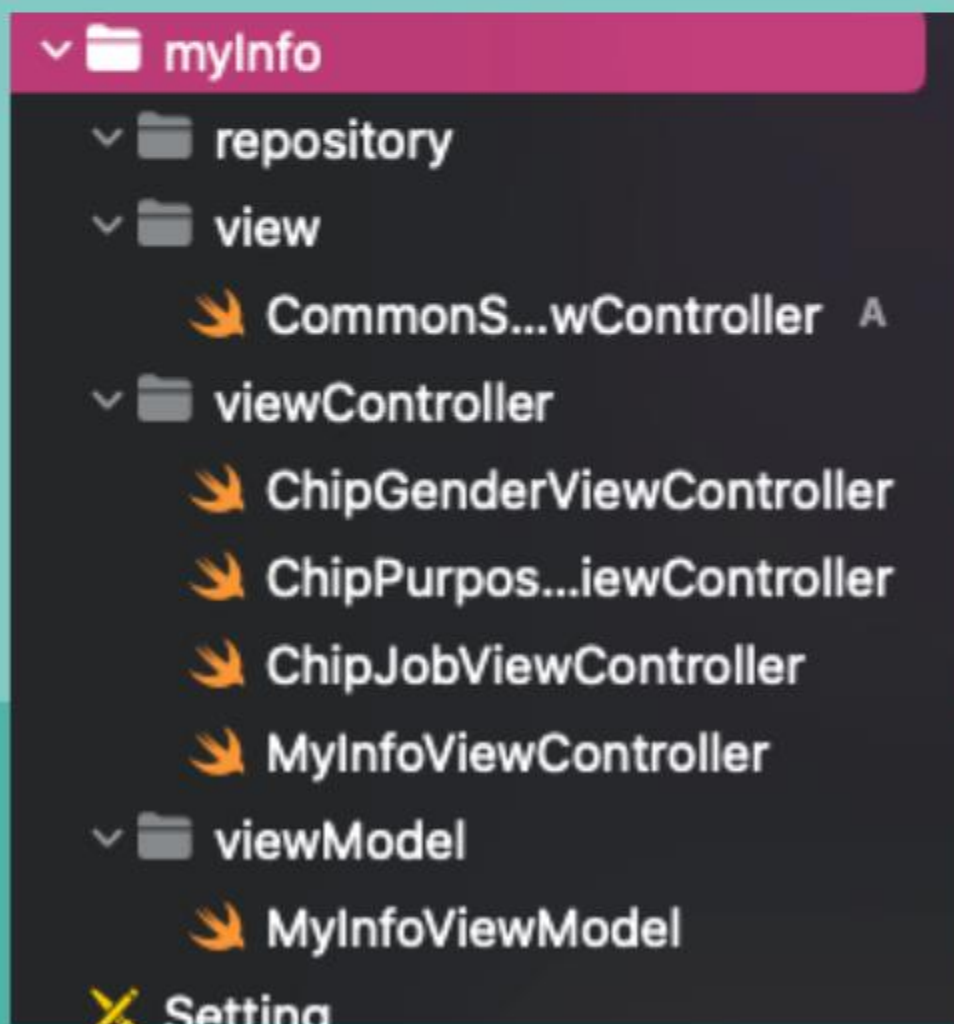
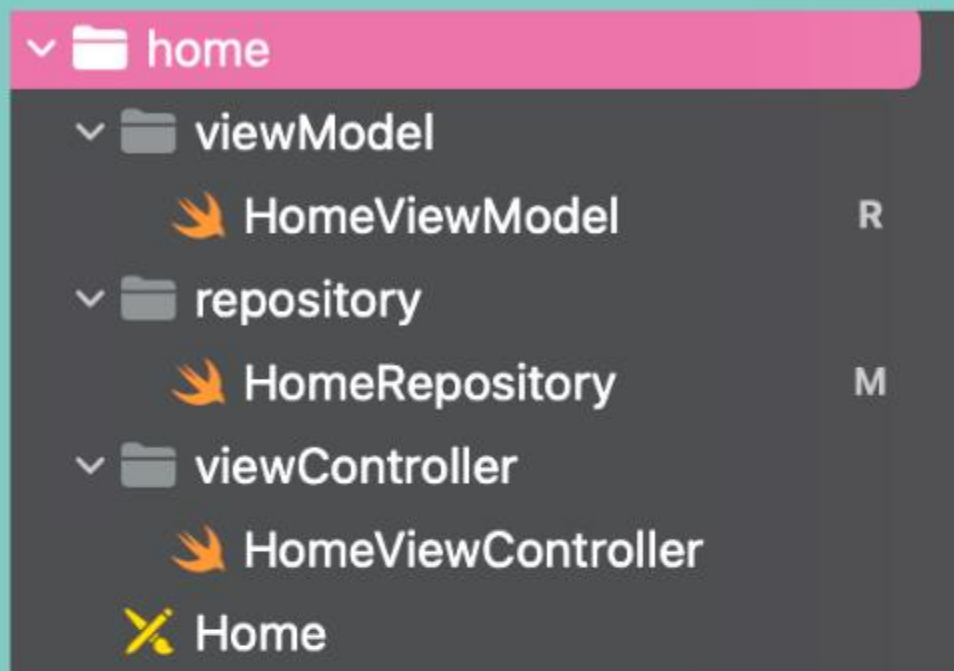
App Flow



실행 결과

실행 결과

MVVM



실행 결과

소셜로그인

oauth-controller

Oauth Controller

▼

POST

/api/oauth/{loginType}

소셜로그인 API

🔒

Parameters

Try it out

| Name | Description |
|---|-----------------|
| idToken string (header) | idToken |
| loginType ★ required string (path) | loginType |
| oauthRequestDto (body) | oauthRequestDto |

Example Value

Model

```
{  "socialId": "string"} 
```

Parameter content type

application/json ▼

실행 결과

소셜로그인 - 애플

```
func authorizationController(controller: ASAuthorizationController, didCompleteWithAuthorization authorization: ASAuthorization) {
    switch authorization.credential {
    case let appleIDCredential as ASAuthorizationAppleIDCredential:
        if let identityToken = appleIDCredential.identityToken,
           let tokenString = String(data: identityToken, encoding: .utf8) {
            KeychainManager.shared.save(tokenString, key: "appleAccessToken")
            LoginAPI.postAppleOauth(type: "apple", token: tokenString, request: OAuthRequest(socialId: "")) { result in
                switch result {
                case .success(let response):
                    KeychainManager.shared.save(response!.accessToken, key: "accessToken")
                    KeychainManager.shared.save(response!.refreshToken, key: "refreshToken")
                    self.pushHome()
                case .failure(let error):
                    switch error {
                    case .errorData(let errorData):
                        self.showToast(message: errorData.message)
                    case .notJoin(let errorResponse):
                        KeychainManager.shared.save(errorResponse.socialId, key: "socialId")
                        KeychainManager.shared.save("kakao", key: "socialLoginType")
                        self.pushSignUp()
                    default:
                        print(error)
                    }
                }
            }
        }
    }
}
```


실행 결과

소셜로그인 - 카카오

```
func getUserInfo(oauthToken: OAuthToken) {  
    UserApi.shared.me() {(user, error) in  
        if let error = error {  
            print("\(error) 사용자 정보 가져오기 실패")  
        } else {  
            print("사용자 정보 가져오기 성공")  
            guard let user_data = user else { return }  
            let request = OAuthRequest(socialId: String(user_data.id!))  
            LoginAPI.postKakaoOAuth(type: "kakao", request: request) { result in  
                switch result {  
                    case .success(let response):  
                        KeychainManager.shared.save(response!.accessToken, key: "accessToken")  
                        KeychainManager.shared.save(response!.refreshToken, key: "refreshToken")  
                        self.pushHome()  
                    case .failure(let error):  
                        switch error {  
                            case .errorData(let errorData):  
                                self.showToast(message: errorData.message)  
                            case .notJoin(let ErrorResponse):  
                                KeychainManager.shared.save(errorResponse.socialId, key: "socialId")  
                                KeychainManager.shared.save("kakao", key: "socialLoginType")  
                                self.pushSignUp()  
                            default:  
                                print(error)  
                        }  
                    }  
                }  
            }  
        }  
    }  
}
```

```
사용자 정보 가져오기 성공  
[HTTP Request]  
URL: https://tellingme.co.kr:8080/api/oauth/kakao  
TARGET: kakao(loginType: "kakao", body: tellingMe.OAuthRequest(socialId: "2709650727"))  
METHOD: POST  
HEADER: [  
    Content-Type: application/json  
]  
BODY:  
{ "socialId": "2709650727" }  
[HTTP Request End]  
[HTTP Response]  
TARGET: kakao(loginType: "kakao", body: tellingMe.OAuthRequest(socialId: "2709650727"))  
URL: https://tellingme.co.kr:8080/api/oauth/kakao  
STATUS CODE: 200  
HEADER: [  
    Cache-Control: no-cache, no-store, max-age=0, must-revalidate  
    Content-Type: application/json  
    x-content-type-options: nosniff  
    x-frame-options: DENY  
    Strict-Transport-Security: max-age=31536000 ; includeSubDomains  
    Expires: 0  
    Vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers  
    x-xss-protection: 1; mode=block  
    Date: Thu, 25 May 2023 12:56:32 GMT  
    Pragma: no-cache  
]  
RESPONSE DATA:  
{ "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleQy-gjvVaEAsqTs6fyQic.eyJleQy-gjvVaEAsqTs6fyQic.eyJleQy-gjvVaEAsqTs6fyQic"  
[HTTP Response End]
```

실행 결과

답변 작성 - 500자 작성 제한

```
func textView(_ textView: UITextView, shouldChangeTextIn range: NSRange, replacementText text: String) -> Bool {
    //이전 글자 - 선택된 글자 + 새로운 글자(대체될 글자)
    let newLength = textView.text.count - range.length + text.count
    let koreanMaxCount = 500 + 1
    //글자수가 초과 된 경우 or 초과되지 않은 경우
    if newLength > koreanMaxCount { //11글자
        let overflow = newLength - koreanMaxCount //초과된 글자수
        if text.count < overflow {
            self.countTextLabel.text = String(newLength)
            return true
        }
        let index = text.index(text.endIndex, offsetBy: -overflow)
        let newText = text[..<index]
        guard let startPosition = textView.position(from: textView.beginningOfDocument, offset: range.location) else { return false }
        guard let endPosition = textView.position(from: textView.beginningOfDocument, offset: NSRange(range)) else { return false }
        guard let textRange = textView.textRange(from: startPosition, to: endPosition) else { return false }

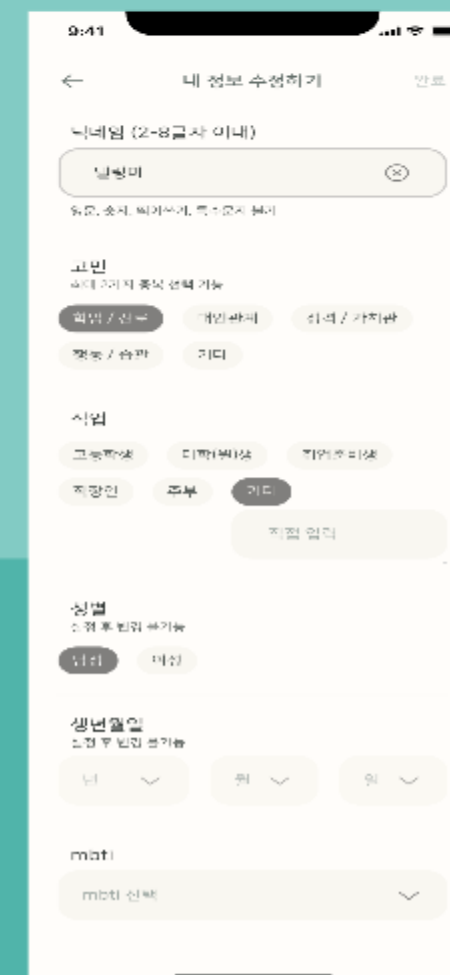
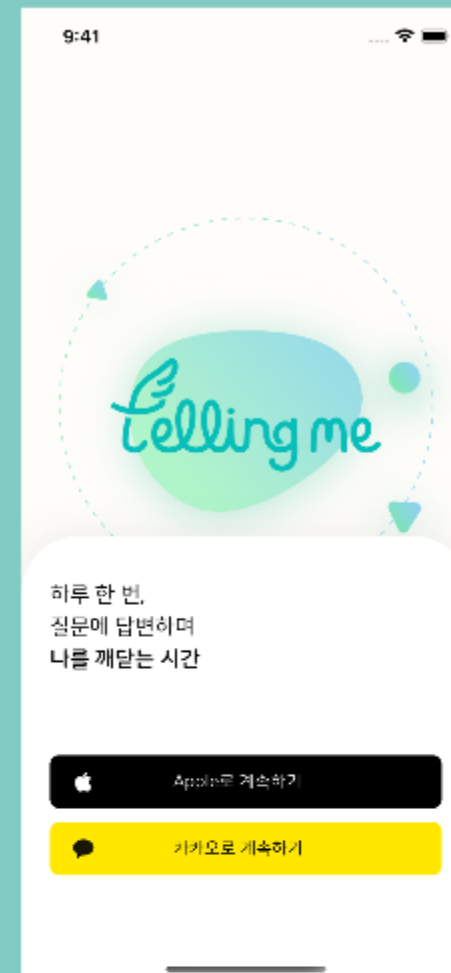
        textView.replace(textRange, withText: String(newText))

        return false
    }
    self.countTextLabel.text = String(newLength)
    return true
}

func textViewDidEndEditing(_ textView: UITextView) {
    if textView.text.count > 500 {
        //글자수 제한에 걸리면 마지막 글자를 삭제함.
        textView.text.removeLast()
        countTextLabel.text = "\(500)"
    }
}
```




실행 결과

- 로그인
- 회원가입
- 나의 공간 (답변 리스트 보기)
- 홈
- 감정 기록
- 답변
- 설정



실행 결과

테스트

 텔링미 

[App Store](#) [서비스](#) [TestFlight](#) [Xcode Cloud](#)


빌드


iOS


피드백

충돌

스크린샷


내부 테스트 


 tellingUs

외부 테스트 

일반 정보


모든 테스터(9/10,100)





테스트 정보 


TestFlight 데이터 정보 


iOS 빌드

테스트가 가능한 빌드입니다. [빌드 상태 및 지표에 관한 추가 정보](#)

 버전 1.2

| 빌드  | 진행 상태 | 그룹 | 초대 | 설치 | 세션 | 충돌 | 피드백 |
|---|---|---|----|----|----|----|-----|
|  1 |  제출 준비 완료 70일 후 무효화 |  | 8 | 5 | — | 4 | 1 |

 버전 1.1

 버전 1.0

15

개선 사항 및 향후 계획

개선 사항 및 향후 계획

개선 사항

답변 관리

- 현재 신고기능을 이용하여 답변을 관리할 예정
- 네이버의 클린봇 같이 비속어 등을 포함한 글을 필터링 하는 기능을 추가하면 좋을 것 같다.

답변 분석

- 현재는 답변 월별 보기 기능과 연속 작성일 수만 볼 수 있다.
- 답변을 분석한 인사이트 페이지를 만들면 좋을 것 같다.
예를 들어, 가장 길게 작성한 답변은?, 최대 연속 작성일은? 등

개선 사항 및 향후 계획

향후 계획

1

푸쉬 알림
에러 핸들링

2023년 6월 중

2

소통
마이페이지

2023년 7월 중

3

결제
서재

2023년 9월 중

4

클리닝 봇 도입

2023년 10월
~ 2023년 12월

감사합니다
