

보고서



대학생의 자아정체성 확립을 위한 문답 기록형 플랫폼 telling Me

목차

| | |
|-------------------------|-----------|
| 1. 개요 및 목적 | 3 |
| 1) 프로젝트 개요 | 3 |
| 2) 프로젝트 동기 | 3 |
| 3) 프로젝트 목적 | 3 |
| 2. 배경과 과정 | 4 |
| 1) 프로젝트 배경 | 4 |
| 2) 시장 조사 | 5 |
| 3) 설문 조사 | 6 |
| 3. 기능 요구명세 | 7 |
| 4. 설계 및 구현 | 9 |
| 1) 프로젝트 디자인 패턴 | 9 |
| 2) 프로젝트 아키텍처 | 9 |
| 3) 프로젝트 흐름도 | 10 |
| 5. 실행 결과 | 11 |
| 6. 개선 사항 및 향후 계획 | 15 |
| 1) 개선 사항 | 15 |
| 2) 향후 계획 | 15 |
| 7. 참고자료 | 16 |

1. 개요 및 목적

1) 프로젝트 개요

대학생 시기 진지한 고민들을 이끌어낼 수 있는 질문들을 통해 스스로에 대해 생각할 시간을 제공.

답변을 서로 공유/공감하고 같은 상황에 놓인 이들 간 위로의 소통공간 마련하여 유저간의 공감과 위로를 형성.

자체 제작한 질문콘텐츠로 이용자의 솔직한 생각을 정리하고, 궁극적으로 가치관 탐색에 도움을 주는 서비스임

2) 프로젝트 동기

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

3) 프로젝트 목적

'나를 가장 잘 아는 것은 바로 나' - 자신에 대해 알아갈 시간이 필요한 대학생들에게 자신의 솔직한 생각을 마음껏 발산할 수 있는 유일한 창구가 될 것

상담사가 아니더라도 같은 상황의 누군가가 나의 생각을 들어줬으면 하는 수요를 충족시킬 것(자신과 비슷한 누군가가 들어주고 공감해주기만 해도 고민이 나아질 수 있음)

2. 프로젝트 배경과 과정

1) 프로젝트 배경

- 본인에 대해 잘 알고자 고민하는 20 대의 비율

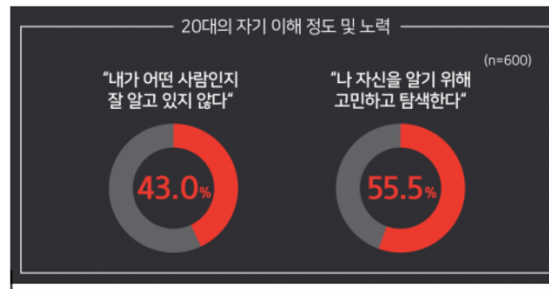


그림 5. [대학내일20대연구소] 20대의 자아에 대한 인식 및 자아 만족 추구 성향 조사 (연구리포트)

- 자존감이 낮은 20 대 중 '자신에 대해 잘 알고 있지 않다'고 응답한 비율은 무려 70%
- 20 대는 스스로를 더욱 잘 알기 위해 고민하고 탐색하는 시간이 부족함

- 대학생 자아탐색 및 심리건강 증진에 대한 대학 및 정부역할의 중요성 강조



- 교육당국및 대학기관이 대학생의 정신건강과 멘탈헬스시장에 많은 관심을 갖고 집중 지원하는 사회적 상황

2) 시장 조사

- Refelctly



- AI 프리미엄 모드를 통한 기분 상관관계 및 그래프와 개인화된 질문 제공

- 하루콩



- 매일의 기분과 활동을 트렌디한 디자인으로 기록 및 분석

- 세줄일기



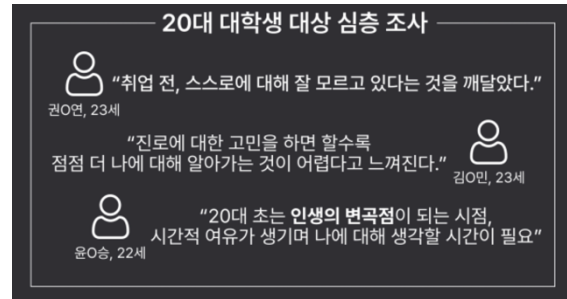
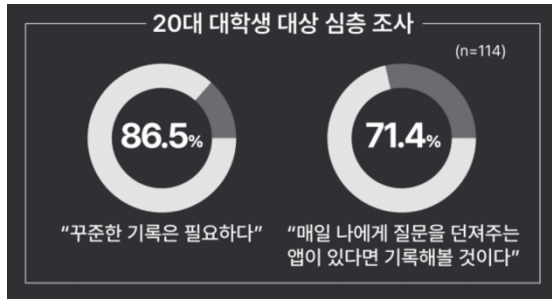
- 세 줄의 글과 한 장의 사진으로 이루어진 일기, 소통공간

- 타사와의 차별점

- 질문의 일관성 부족, 비속어 다수 발견 등 질적 관리의 부족 -> 자체 질문 제작 및 신고 기능으로 관리
- 나의 답변만 볼 수 있어서 타 유저들과 답변을 공유할 수 없음 -> 소통 공간 마련
- 문장 빈칸에 단어 채우기로 답변의 자율성에 제한이 있음 -> 500 자까지 보장
- 데이터 백업에 대한 제한적 이용 -> 온/오프라인 소장 가능

3) 설문조사

- 설문조사와 심층 인터뷰 결과



문답 기록 플랫폼의 수요를 검증하고, 텔링미의 비전에 동의하는 의견을 다수 확인함

3. 기능 요구 명세

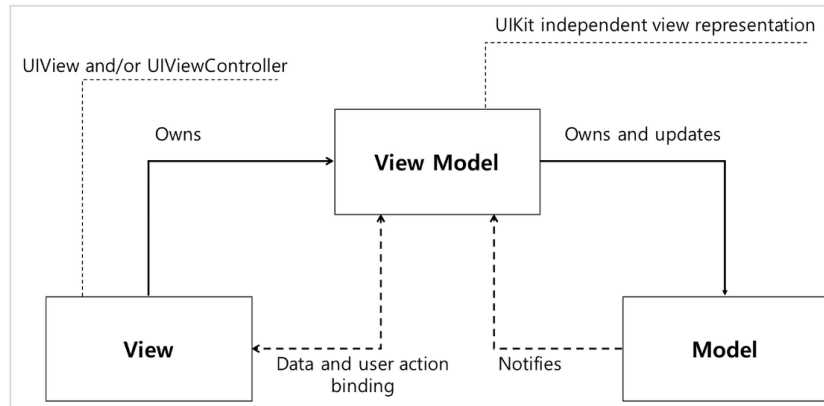
| 화면 | 기능 목록 | 세부사항 |
|--------------|-----------------|--|
| 랜딩 페이지 | 서비스 이용 방법 소개 | |
| 로그인 | 소셜 로그인 | - 카카오 - 애플 |
| 서비스 이용 약관 | 개인정보 처리방침 | |
| 회원 가입 | 유저 정보 저장 | 1) 닉네임 2) 고민 3) 직업 4) 성별 5) 생년월일 6) MBTI 7) 푸쉬알림 동의 |
| 나의 공간 | 나의 답변 리스트 | 1) 리스트 보기 2) 카드섹션 리스트 보기 3) 개별 답변 보기 4) 설정 |
| 홈 | 질문 보기 | 1) 기록 연속 날짜 2) 오늘의 질문 3) 답변 작성하기 버튼 |
| 감정 기록 모달 | 감정 이모지 12 중 택 1 | |
| 답변 | | 1) 답변 작성 2) 답변 저장 3) 감정 변경 가능 4) 공개/비공개 5) 저장 후 수정/삭제 가능 6) 질문 내용 fold 가능 |
| 소통 공간 | | 1) 타인의 답변 목록 2) 정렬 기능 |

| | | |
|----|--|--|
| | | 3) 공감 기능 4) 신고 기능 |
| 설정 | | 1) 개인정보 수정 2) 서비스 이용 약관 3) 로그아웃 4) 회원탈퇴 5) 결제 6) 푸쉬알림 동의 설정 |

4. 설계 및 구현

1) 프로젝트 디자인패턴

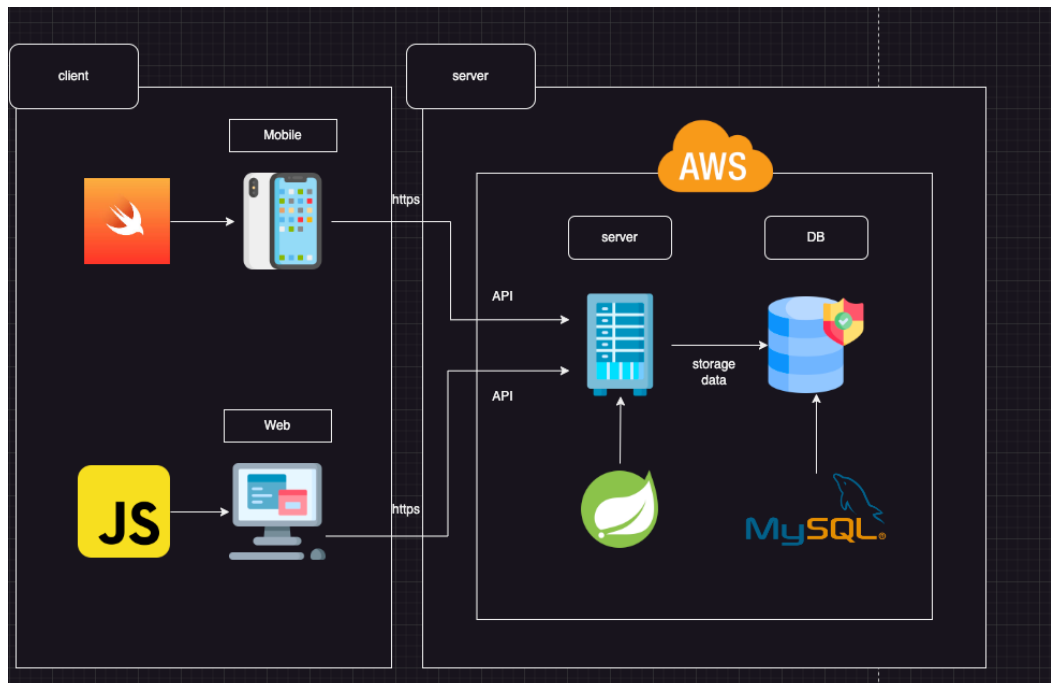
MVVM



MVVM 패턴

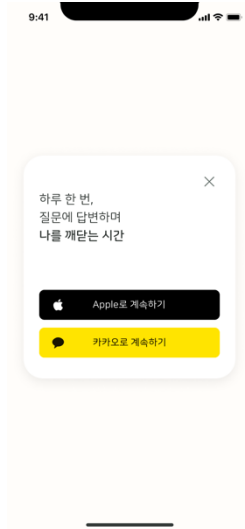
MVVM 패턴을 사용하여 프로젝트를 관리한다.

2) 프로젝트 아키텍처(도식화)



3) 프로젝트 흐름도

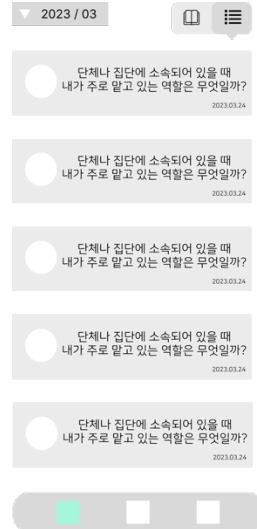
[로그인]



[홈]



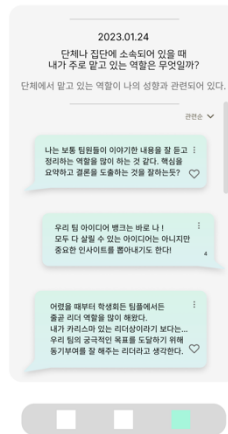
[답변 리스트]



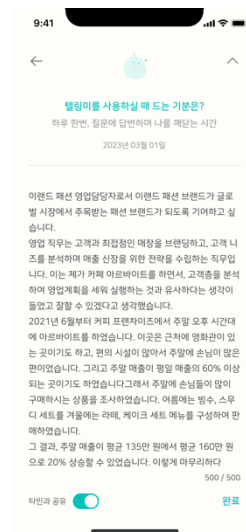
[설정]



[소통]

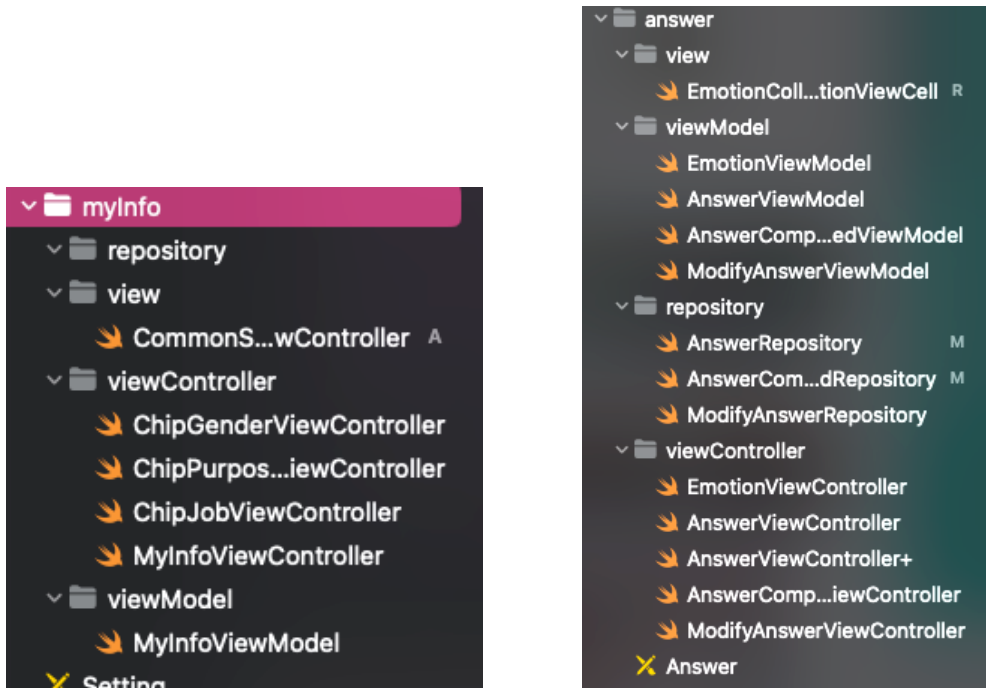


[답변 작성]



5. 실행 결과

- MVVM 패턴 적용



- 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)
                    }
                }
            }
        }
    }
}
```

• 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)
                        }
                    }
                }
            }
        }
    }
}
```

• 소셜로그인 - 실행결과

```
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":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI4NThlMzZmMmM1ZjM5LjYzUWYyYUQyNC1mZTMxNmE4ODQ0NDgiLCJpYXQiOiJlZDQ0UWMTkzOTIsImV4cCI6MTY4ODc0MDU1MX0.1EQy-gjvVaEAsqTs6fyQi6yCWltZMFb-T6rnStYLD4M","refreshToken":"eyJhbGciOiJIUzI1NiJ9.eyJpYXQiOiJlZDQ0UWMTkzOTIsImV4cCI6MTY4ODc0MDU1MX0.uE93He8eHQR-ZP4hOpsb8A3get1cyIPZ0Ip6fE3i284"}
[HTTP Response End]
```

- 답변 작성 – 500 자 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(location: range.location, length: range.length)) 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)"
    }
}
```

- 테스트 (testflight 등록 완료)
 - 팀원들끼리 피드백 주고 받을 수 있음

 **텔링미** ▾ [App Store](#) [서비스](#) [TestFlight](#) [Xcode Cloud](#)

빌드


iOS

피드백

충돌

스크린샷

내부 테스트 +

 tellingUs

외부 테스트 +

일반 정보

모든 테스트(9/10,100)



테스트 정보

TestFlight 데이터 정보 ?

iOS 빌드

테스트가 가능한 빌드입니다. 빌드 상태 및 지표에 관한 추가 정보

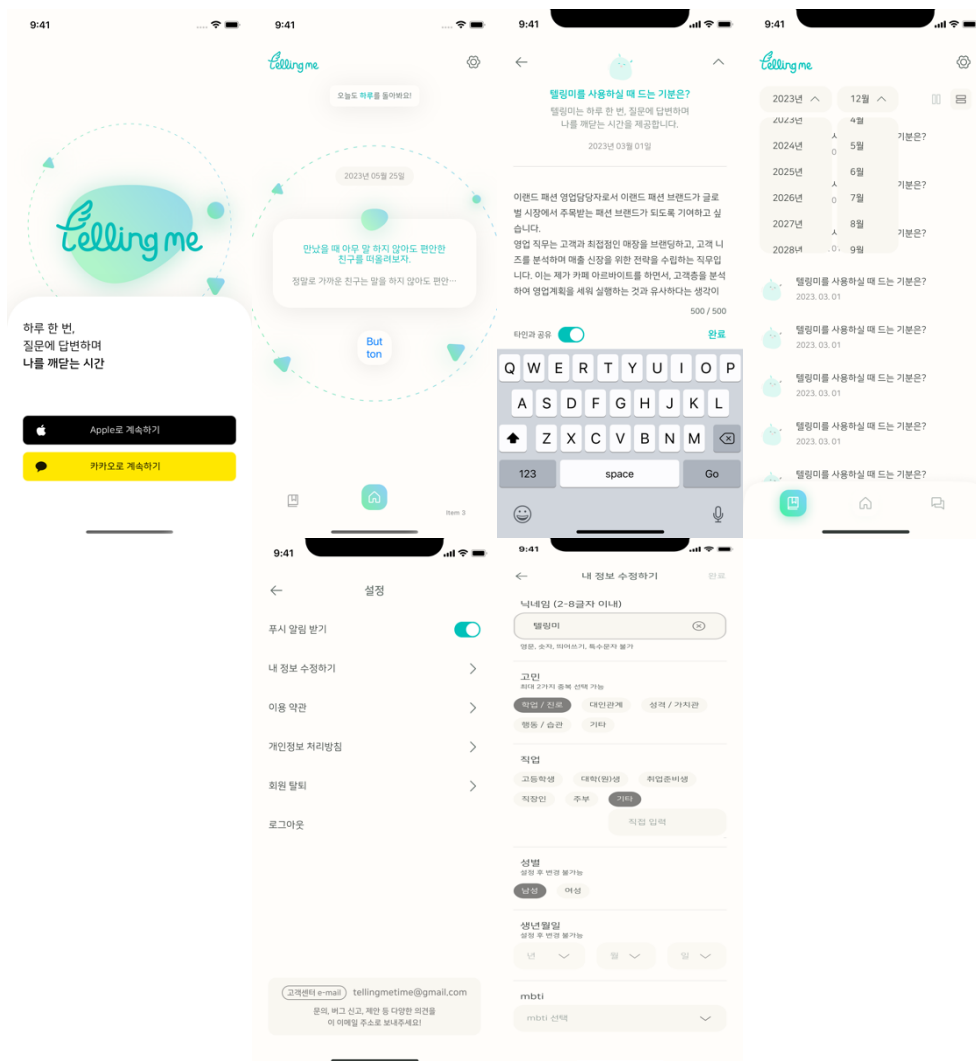
▼ 버전 1.2

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

> 버전 1.1

> 버전 1.0

- 현재까지 완료된 부분
 - 로그인
 - 회원가입
 - 나의 공간(답변 리스트)
 - 홈
 - 감정 기록
 - 답변
 - 설정



6. 개선 사항 및 향후 계획

1) 개선 사항

- a. 답변 관리
 - i. 현재 신고기능을 이용하여 답변을 관리할 예정
 - ii. 네이버의 클린봇 같이 비속어 등을 포함한 글을 필터링 하는 기능을 추가 개선
- b. 답변 분석
 - i. 현재는 답변 월별 보기 기능과 연속 장성일 수 정도만 포함
 - ii. 답변을 분석한 인사이트 페이지를 추가 개선

2) 향후 계획

- a. 2023 년 6 월 중
 - i. 푸쉬 알림
 - ii. 에러 핸들링
 - iii. 전체적으로 마무리
- b. 2023 년 7 월 중
 - i. 소통 페이지
 - ii. 마이페이지
- c. 2023 년 9 월 중
 - i. 결제 기능
 - ii. 서재 기능
- d. 2023 년 10 월 ~ 12 월 중
 - i. 클리닝 봇 도입

7. 참고자료

<https://jhtop0419.tistory.com/m/21> (MVVM 이미지)

<https://www.nextunicorn.kr/company/deabed8666255947> (세줄일기)

<https://apps.apple.com/kr/app/%ED%95%98%EB%A3%A8%EC%BD%A9/id1553223828>
(하루콩)

<https://reflectly.app/> (reflectly)

<https://www.slideshare.net/20slab/20-20-201711> (대학내일 20 대연구소)

<http://happyjalip.com/notice/?q=YToyOntzOjEyOiJrZXI3b3JkX3R5cGUiO3M6MzoiYWxsljtzOjQ6InBhZ2UiO2k6Mjt9&bmode=view&idx=13148561&t=board> (청년마음건강 지원사업)

<http://his.pusan.ac.kr/bbs/nursing/2584/936650/artclView.do> (마음+ 캠페인 2 차)