



Demo Page

- '1차 평가'를 위한 Demo Page로, 본 페이지를 기준으로 1차 평가가 진행됩니다.
- 개발하신 서비스를 잘 나타낼 수 있도록 작성해 주세요.

대회 후 한마디

노준호 : 외부에서 하는 첫 해커톤인데 너무 재미있게 하고 갑니다 ~! 😊 (집에 가면 순대국밥 먹어야지)

이동준 : 멘토님들의 멘토링 하에 새로운 프로젝트를 진행할 수 있다는 점이 매우 설레였고, 즐거웠습니다!

손명진 : 6시간 디자인과 코딩의 합작물,,,ㅎㅎ

이유림 : 새로운 기술을 구현하고자 하여 우여곡절이 많았지만, 좋은 팀원들 덕분에 잘 이겨낼 수 있었습니다!! :) 좋은 행사 기획해 주셔서 너무너무 감사드려요!

Background

평소, 다른 개발자나 기업가, VC와 소통하는 과정에서 명함이나 LinkedIn 계정을 빈번히 공유하는 과정이 있었습니다.

명함과 LinkedIn 계정을 공유하는 과정은 크게 다음 두 가지 과정으로 이루어졌습니다.

명함 교환: 지갑에 보관 → 빈번히 잃어버리거나 구겨지는 등의 문제 발생. 또는, 데이터로 저장하는 번거로움 발생

LinkedIn 계정 공유: LinkedIn 계정의 전체 URL 을 외우고, 구두로 알려주는 과정이 있음(전파 오류의 가능성). 또는 타인의 기기를 빌려, 직접 입력해줌.

→ 결정적으로, LinkedIn 계정이 없는 사용자가 많음

크게 위와 같은 **2가지** 불편함에 착안하여, 디지털로 쉽게 명함을 공유하고, 관리 할 수 있는 제품을 만드는 것이 어떤가 고민하게 되었습니다.

이를 해결하기 위한 방안으로 NFC 기술에 집중하여 개발을 진행하였습니다.

Prev App Research

1. remember



종이 명함을 사진으로 찍어 전자 명함으로 저장하는 App

→ 매번 종이 명함을 새롭게 찍어야 한다는 단점 존재

2. 네이미



온라인 상에서 자신의 명함을 만들고, 공유할 수 있는 App

문자를 통해 명함 공유 가능

→ 명함 공유 시, 명함을 보낼 필요가 없는 사람에게도 공유가 된다는 단점 존재

Logo & Meaning



人플 (NFC를 통한 빠르고 간편한 상호 명함 정보 교환 시스템)



人 (사람 인) + people → 사람과 사람이 만나 새로운 가치 창출

in + people → 사람이 가진 정보를 바탕으로 이루어 나가는 서비스

Tech Stack

FrontEnd

→ Flutter

BackEnd

→ Express &
Node.js(yarn)

Database

→ Sqlite3

Slack Bot

→ Flask & Python,
Apache

프로그램 개발 목적

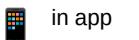
- 평상시 명함 관리와 교환 과정의 번거로움 인지 (종이 형태의 명함 관리의 어려움)
- 휴대폰을 서로 태깅하는 간단한 방식으로 명함 교환 및 연락망을 구축하면 좋겠다고 생각함
- 기존의 전자 명함 앱을 사용하였을 때, 직접 명함을 사진으로 찍어 업로드해야 한다는 단점 발견
- 대면으로 하는 명함 공유의 현장감 등의 장점을 살리면서도 slack bot을 활용한 온라인 공유 모두를 가능하도록 만들고자 함
- 개발자나 회사원 중심으로 주로 진행되는 명함 공유라는 문화를 다른 직업군에게도 확대할 수 있는 가능성 발견

주제와의 관련성

- 우리의 만남은 우연이자 인연이고 필연입니다.
- 소중한 인연을 영원히 기억하기 위해 준비했습니다.
- Business Card 교환 시스템
- 새로운 사람과 사람 사이의 관계를 중점적으로 다루는 프로젝트

Implement Function

App



in app

1. 로딩화면
2. 로그인
3. 홈화면
 - a. 명함 리스트
 - b. 명함에 대한 정보
 - c. 명함 별 간단한 메모 기능(명함 공유 시 느낀 점이나, 적어두어야 할 내용 기록)
4. Slack bot과의 연동

Color Palette

Primary Color

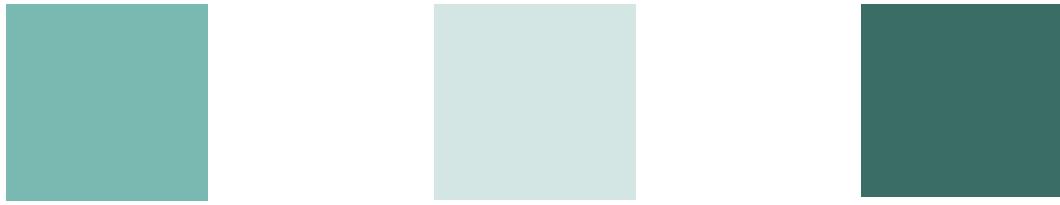
#64bbb1

Secondary Color

#cee7e4

Other

#256F66



: 신뢰감과 편안함을 주는 색상을 혼합하여 해당 앱을 사용하는 사람들로 하여금 앱에 대한 긍정적인 이미지 확립에 도움

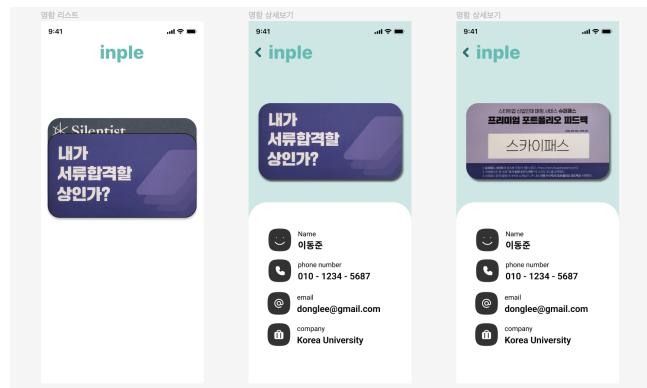
Design View

https://www.figma.com/embed?embed_host=notion&url=https%3A%2F%2Fwww.figma.com%2Ffile%2FIQHyQHBEZICQ0uVM7KPDJa%2FUntitled%3Ftype%3Ddesign%26node-id%3D0-1%26t%3D0o2BMGX8LeNwfkzb-0

1. 로딩화면 & 로그인 화면



2. 명함 리스트 화면 및 명함 정보 화면



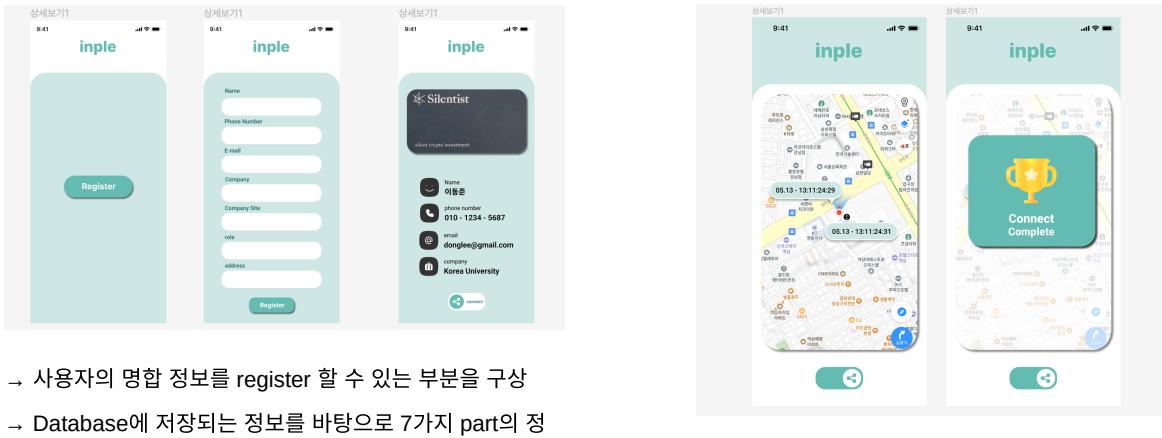
→ 명함을 슬라이드 형식으로 넘겨볼 수 있도록 구상

→ 해당 명함 선택 시, 명함에 저장된 정보를 한 눈에 알아보기 편하도록 구상

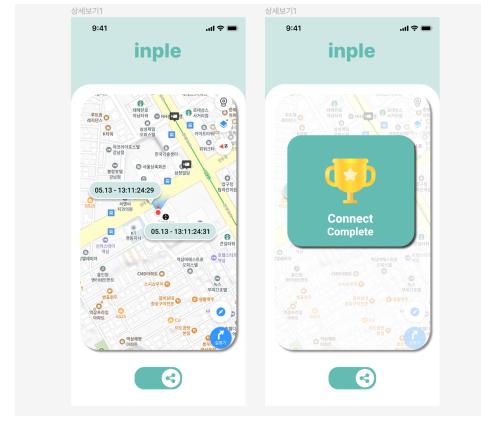
→ 명함 이미지를 앞, 뒤로도 볼 수 있도록 표현

3. 자신만의 명함 등록

4. NFC tagging을 통한 명함 교환



- 사용자의 명함 정보를 register 할 수 있는 부분을 구상
- Database에 저장되는 정보를 바탕으로 7가지 part의 정보들을 저장할 수 있도록 표현
- register 이후, 자신의 명함이 나타나도록 구상



- 위치 정보와 태깅 시간을 통하여 특정 오차 범위 이내에서 인식될 경우, connect될 수 있도록 구상
- 명함 교류가 이루어지는 모습 표현

5. 추가 메모 작성 화면



- 해당 명함을 받은 시간과 장소, 해당 사람에 대한 인상을 알기 쉽게 정리하는 방법을 각각의 명함에 추가 메모를 작성하는 방법을 통해 표현

Simulation View

SlackBot

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/ce91f55c-a4cb-4816-8ead-69241ee58293/Slack_Bot.mov

```

ldj6192 오전 5:54
@inple_Bot my_info user1@gmail.com
password12312

inple_Bot 웹 오전 5:54
[+] fail :{

ldj6192 오전 5:54
@inple_Bot card_delete user1@gmail.com
password

inple_Bot 웹 오전 5:54
[+] success

ldj6192 오전 5:54
@inple_Bot my_info user1@gmail.com password

inple_Bot 웹 오전 5:54
[+] fail :{

ldj6192 오전 5:54
@inple_Bot register user2@gmail.com password
세 향독

inple_Bot 웹 오전 5:54
[+] success

```



```

inple_Bot 웹 오전 5:54
command not found :(
@inple_Bot help

ldj6192 오전 5:53
@inple_Bot help

inple_Bot 웹 오전 5:54
[usage]
- my_info [your_mail] [your_pw]
- add_friend [your_mail] [your_pw] [target_token]
- del_friend [your_mail] [your_pw] [target_token]
- register [your_mail] [your_pw]
- card_add [your_main] [your_pw] [email2] [name]
[company] [phone] [address] [site] [role]
- card_delete [your_mail] [your_pw]

ldj6192 오전 5:54
@inple_Bot register user1@gmail.com password

inple_Bot 웹 오전 5:54
[+] success

ldj6192 오전 5:54
@inple_Bot card_add user1@gmail.com
password user1@korea.ac.kr 유저1 korea_univ
010-4615-6192 성북구안암동3가 naver.com 학부
생

```

Server Logging

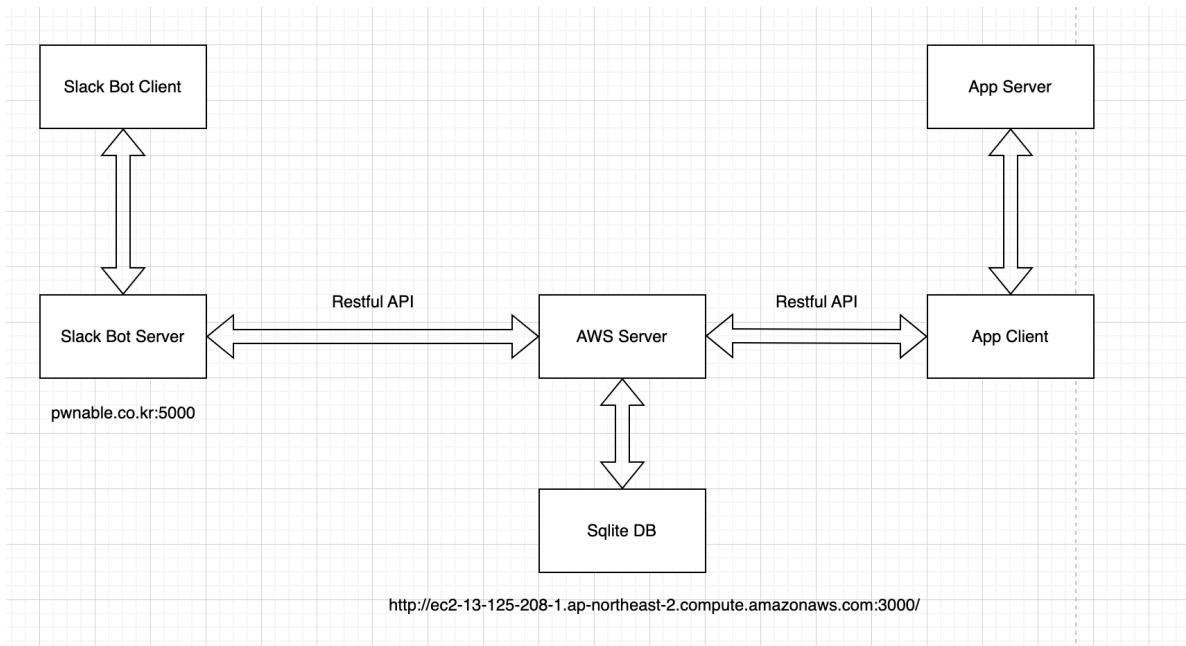
https://s3-us-west-2.amazonaws.com/secure.notion-static.com/ed885d6d-33aa-42ab-917e-fc9b17e34038/IMG_9499.mov

inple APP

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/fac1d1ae-5071-42d9-8e2a-d42fe515e9cf/client_2023-05-14_05-58-03.mp4

Development View

DataFlow Chart



Flutter (Frontend)

in front

- Develop with Flutter
- Design with Figma

RestFul API(Backend)

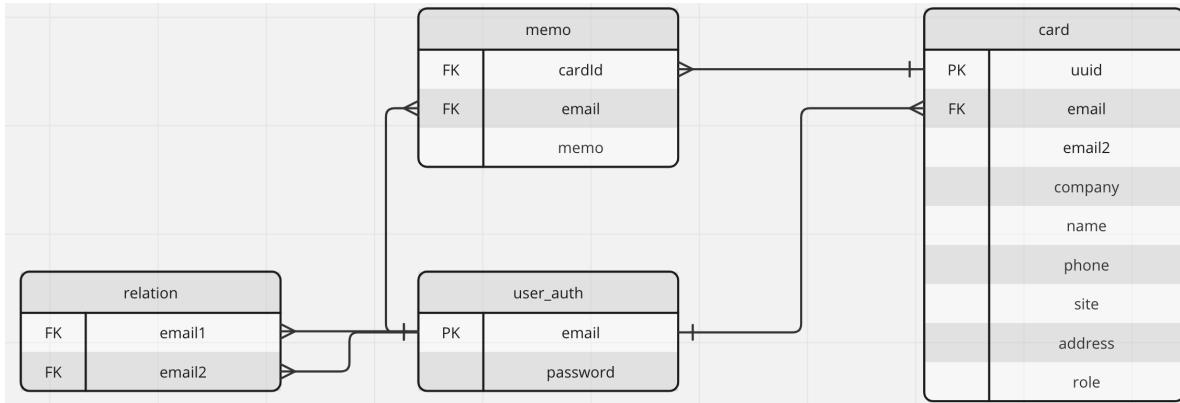
in back

- Develop with node.js Express
- JWT Auth token

 [\(Old\)rokdev-API Request](#)

Sql Database

using miro.com



Server Setting

in server

- AWS EC2
- EC2 기반 Express backend server 구축
- Database: sqlite3
- JWT 기반 auth 구축 / middleware를 통한 인증
- ~~TODO: Github action & S3 & codedeploy 를 통한 배포 자동화~~ → pm2 manage tool & 수동 관리로 대체

Slack Bot

using slack tech.

- Python Slack bot
- Connection with Api for Auth
- Auth : xoxb-5256809882501-5283461542848-hvbzzfHL8lc0flkXtYYZrfOr
- Hostring with : pwnable.co.kr:5000
- Apache server

Optimization

- 비동기 처리: 비동기적으로 작업을 처리하여 블로킹을 최소화 (Async 사용)
- 반복되는 루틴에 대한 Class 생성을 통한 별도의 로직으로 처리
- 추후 코드 번들링, 캐싱, 프로파일링에 대한 작업 추가 예정.

팀원 간 업무 분담 현황

이유림(@yurimn)
[Front]

- Flutter
- NFC read/write

손명진(@thsaudwls)
[Front]

- Flutter
- UI/UX Design

노준호(@d2n0s4ur)
[Back]

- AWS 인스턴스 관리
- Express Server

이동준(@m0nd2y)
[Back]

- Slack Server Bot
- DevOps
- ☕ Coffee maker

향후 product 개발 방향성

사업성

- TAM-SAM-SOM
 - SOM: 국내 경제활동인구 중 신규 기술기반 창업 (약 29만 기업, 2022년 기준, e-나라지표)
 - SAM: 국내 경제활동인구 중 관리자 및 창업 종사자 (약 50만명, 2023년 4월 기준, 통계청)
 - TAM: 국내 경제활동인구 중 관리자, 전문가 및 관련 종사자 (약 680만명, 2023년 4월 기준, 통계청)
- BM

- NFC 명함 공유 기술의 지적재산권화 및 저작료 수입
- 업계 표준화 기술 개발을 통한 B2B 형태의 업무 협약 수익

The Business Model Canvas



⚖️ 확장성

- 기술적 측면
 - 발전된 정밀한 **NFC Tagging 기술**을 통한 간단한 명함 교환
 - 하나의 표준을 만들어(현재 linkedin이 우리에게 인식되는 영향) 일종의 CV 또는 자기소개의 새로운 품으로 표준화
 - user relation big dataset을 통한 새로운 feature 발굴, 라벨링
- 사업적 측면
 - **NFC 데이터 공유 기술**을 기반한 사업 확장(송금, ...)
 - 명함 공유 시 Slack을 활용하여 우수한 **네트워크 형성**
 - 유저간 relation 및 데이터 분석을 통한 유저 value 측정 및 추천 이웃 기능 제공
 - 새로운 feature 발굴, 라벨링을 통한 데이터 사업화?

Appendix: Github & EC2

- Frontend / Backend / SlackBot 크게 3가지 영역으로 나누어 작업 진행
- logging과 debugging을 통한 버그 탐지 및 수정

The screenshot shows the GitHub organization page for 'skycc-rokDev'. It lists three repositories: 'frontend' (Public, C++, 0 stars, 0 issues, updated 1 hour ago), 'slack_bot' (Public, Python, 0 stars, 0 issues, updated 5 hours ago), and 'backend' (Public, JavaScript, 0 stars, 0 issues, updated 6 hours ago). The 'Repositories' tab is selected.

Commit Count

Frontend : 7
Backend : 9
Slack-bot : 15

- pm2를 활용한 backend server 상시 실행
- 데이터베이스 logging
- Nohub을 이용한 Flask 서버 지속 실행
- 서버 호스팅을 통한 원격 접속 가능

The screenshot shows the pm2 process list and metrics for the 'backend' application. The process list shows a single instance of 'index' with various logs. The metrics section shows custom metrics like Used Heap Size (18.60 MiB), Heap Usage (90.73 %), and Event Loop Latency p95 (0.32 ms).

Frontend

GitHub - skycc-rokDev/frontend: SKYCC hackathon rokDev Frontend
SKYCC hackathon rokDev Frontend. Contribute to skycc-rokDev/frontend development by creating an account on GitHub.

<https://github.com/skycc-rokDev/frontend>

skycc-rokDev/
frontend



SKYCC hackathon rokDev Frontend

1 Contributor 0 Issues 0 Stars 0 Forks



Backend

GitHub - skycc-rokDev/backend: SKYCC hackathon rokDev Backend
SKYCC hackathon rokDev Backend. Contribute to skycc-rokDev/backend development by creating an account on GitHub.

<https://github.com/skycc-rokDev/backend>

skycc-rokDev/
backend



SKYCC hackathon rokDev Backend

1 Contributor 0 Issues 0 Stars 0 Forks



SlackBot

GitHub - skycc-rokDev/slack_bot
Contribute to skycc-rokDev/slack_bot development by creating an account on GitHub.

https://github.com/skycc-rokDev/slack_bot

skycc-rokDev/
slack_bot



1 Contributor 0 Issues 0 Stars 0 Forks



Appendix: API Docs

API 명세서

Aa 기능	HTTP 메서드	API Path	Request	Response
회원가입	POST	/auth/register	UserAuthRequest	UserAuthResponse
로그인	POST	/auth/login	UserAuthRequest	UserAuthResponse
내 명함 추가	POST	/card/add	AddMyCardRequest	DefaultResponse
내 이웃 명함 목록 조회	POST	/card/list	DefaultRequest	CardListResponse
내 명함 조회	GET	/card/mycard	DefaultRequest	CardResponse
내 명함 삭제	DELETE	/card/delete	DefaultRequest	DefaultResponse
서로 이웃 추가 하기	POST	/relation/add	AddRelationRequest	DefaultResponse
명함의 메모 추가	POST	/memo/add	AddMemoRequest	DefaultResponse
명함의 메모 조회	GET	/memo/get	ViewMemoRequest	ViewMemoResponse

DTO(Data Transfer Object)

Aa 이름	API 분류	태그
AddMyCardRequest	API Request	[명함] 명함 정보
UserAuthRequest	API Request	[인증] 계정
UserAuthResponse	API Response	[인증] 계정
CardListResponse	API Response	[명함] 명함 정보
CardResponse	API Response	[명함] 명함 정보
AddMemoRequest	API Request	[명함] 명함 정보
ViewMemoRequest	API Request	[명함] 명함 정보
ViewMemoResponse	API Response	[명함] 명함 정보
DefaultResponse	API Response	[기타] 통합 DTO
DefaultRequest	API Request	[기타] 통합 DTO
AddRelationRequest	API Request	[소셜] 이웃

Appendix: Key Code

Server

- JWT 토큰 인증 관련 미들웨어

```

const jwt = require('../modules/jwt');
const MSG = require('../modules/responseMessage');
const CODE = require('../modules/statusCode');

const TOKEN_EXPIRED = -3;
const TOKEN_INVALID = -2;

// jwt 기능 사용을 위하여 async function으로 작성함.
const authUtil = {
  checkToken : (async (req, res, next) => { // request의 token이 있는 경우 파싱, 그렇지 않으면 파싱하지 않고 이어서 작업
    var token = req.headers.token;
    if (!token)
    {
      req.email = undefined;
      next();
    }
    // decode
    const user = await jwt.verify(token);
    // 유효기간 만료된 jwt -> 토큰 오류 메시지 및 권한 없음 status 코드 제공
    if (user === TOKEN_EXPIRED)
      return res.send({ "code": CODE.UNAUTHORIZED, "message": MSG.INVALID_TOKEN });
    // 유효하지 않는 토큰(client 단에서 변조되었을 확률이 높음)
    if (user === TOKEN_INVALID)
      return res.send({ "code": CODE.UNAUTHORIZED, "message": MSG.INVALID_TOKEN });
    // decode 된 jwt이나, email 필드가 없는 경우
    if (user.email === undefined)
      return res.send({ "code": CODE.UNAUTHORIZED, "message": MSG.INVALID_TOKEN });
    // req의 email에 파싱한 이메일 값 저장
    req.email = user.email;
    // promise await
    return await next();
  })
}

module.exports = authUtil

```

SlackBot

- Slack API 사용 핵심 코드

```

# 이벤트 핸들러 함수 정의
def event_handler(event_type, slack_event):
    channel = slack_event["event"]["channel"] # Slack 이벤트에서 채널 정보 추출
    string_slack_event = str(slack_event) # Slack 이벤트를 문자열로 변환

    # 맨션으로 호출된 경우 처리
    if string_slack_event.find("{'type': 'user', 'user_id': ") != -1:
        try:
            user_query = slack_event['event']['blocks'][0]['elements'][0]['elements']
            # 사용자 쿼리 추출
            answer = get_answer(user_query[1]['text'][1:], user_query) # 답변 가져오기
            post_message(token, channel, answer) # 답변을 Slack에 전송
            return make_response("ok", 200) # HTTP 응답 생성하여 반환
        except IndexError:
            pass

    message = "[%s] cannot find event handler" % event_type # 이벤트 핸들러를 찾을 수 없는 경우 에러 메시지 생성
    return make_response(message, 200, {"X-Slack-No-Retry": 1}) # HTTP 응답 생성하여 반환

# '/' 경로에 대한 POST 메서드 핸들러 함수 정의
@app.route('/', methods=['POST'])
def hello_there():
    slack_event = json.loads(request.data) # 전달받은 요청 데이터를 JSON으로 로드하여 slack_event 변수에 저장

    if "challenge" in slack_event:
        # Slack 이벤트 유효성 검증을 위한 challenge 값이 포함된 경우, 해당 값을 반환
        return make_response(slack_event["challenge"], 200, {"content_type": "application/json"})

    if "event" in slack_event:

```

```

event_type = slack_event["event"]["type"] # Slack 이벤트 유형 추출
return event_handler(event_type, slack_event) # 이벤트 핸들러 함수 호출

return make_response("There are no slack request events", 404, {"X-Slack-No-Retry": 1}) # Slack 요청 이벤트가 없는 경우 에러 메시지 반환

```

- try, except 를 통한 에러제어

```

# 개인 정보 조회 함수 정의
def func_my_info(full_query):
    try:
        email = full_query[2]['url'].split("mailto:")[1] # 이메일 추출
        pw = full_query[3]['text'][1:] # 비밀번호 추출
        token = get_my_token(email, pw) # 이메일과 비밀번호로 토큰 생성
        print(token) # 토큰 출력 (디버깅용)

        url = host + "/card/mycard" # 조회할 URL 생성
        data = {"data": "data"} # 전송할 데이터 생성

        status, response = get_request_token(url, token) # 토큰을 사용하여 요청 전송
        if status == 200: # 요청이 성공한 경우
            response = json.loads(response) # 응답을 JSON 형식으로 로드
            print(response) # 응답 출력 (디버깅용)

            string = ""
            # 응답에서 필요한 정보 추출하여 문자열 생성
            string += "email : " + response['data']['email'] + "\n"
            string += "email2 : " + response['data']['email2'] + "\n"
            string += "company : " + response['data']['company'] + "\n"
            string += "name : " + response['data']['name'] + "\n"
            string += "phone : " + response['data']['phone'] + "\n"
            string += "site : " + response['data']['site'] + "\n"
            string += "address : " + response['data']['address'] + "\n"

            return string # 생성한 문자열 반환

        return "[-] fail :(" # 요청이 실패한 경우 에러 메시지 반환
    except:
        return "[-] fail :(" # 예외가 발생한 경우 에러 메시지 반환

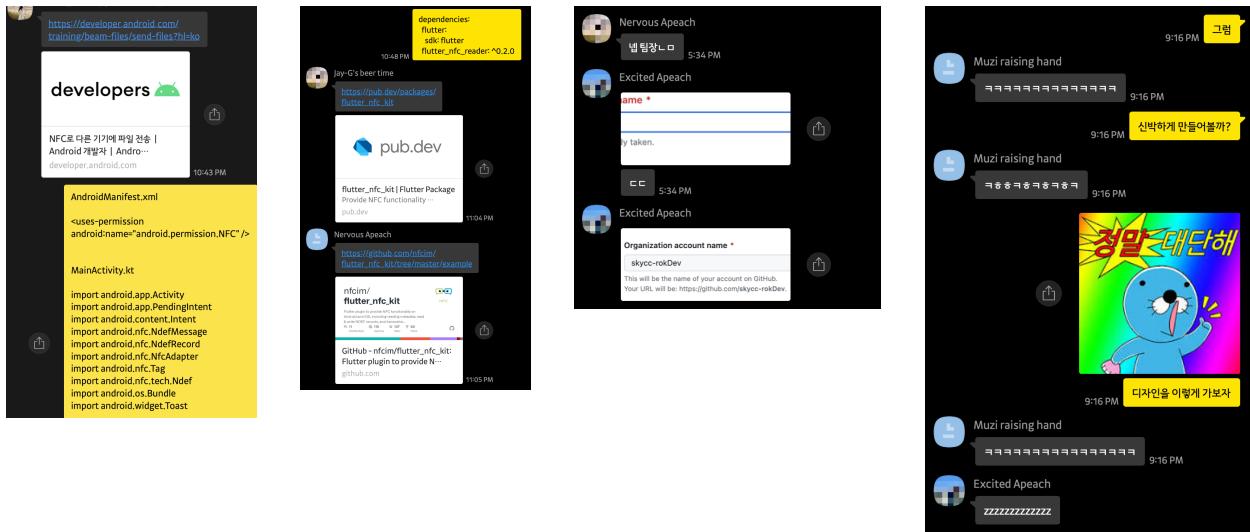
```

Appendix: Communication

[Slack]



[KakaoTalk]



Appendix: Presentation

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/27525f76-cf6d-4623-807f-33c60bac8cb7/202305140946_1%E1%84%8B%E1%85%A1%E1%86%AB.pdf