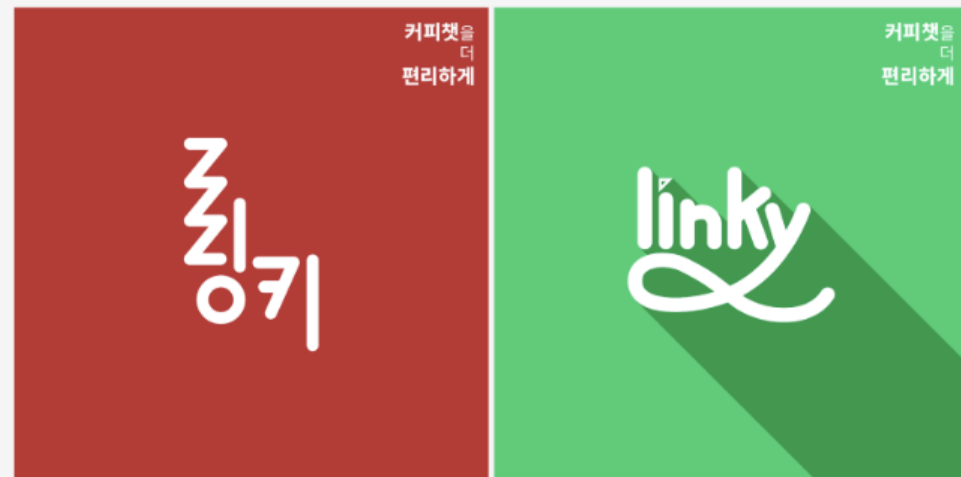
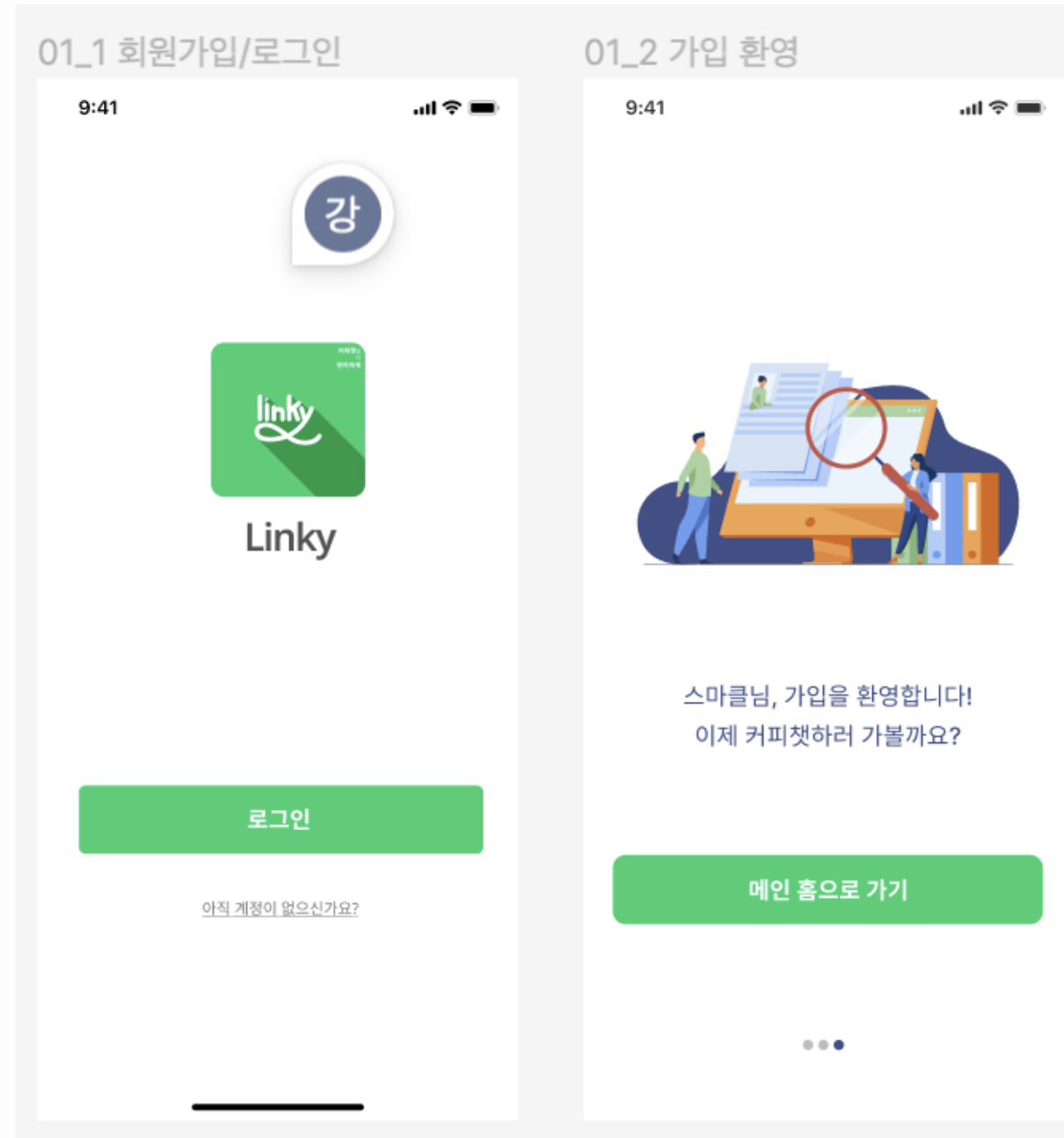


# 우리의 아이디어는



- 서비스명: 링키 (가명)
- 서비스 로고: 위 이미지
- 메인 폰트: Noto Sans
- 슬로건: link, coffee (ex. 당신 근처의 마켓)
- 색상: #20CE6F

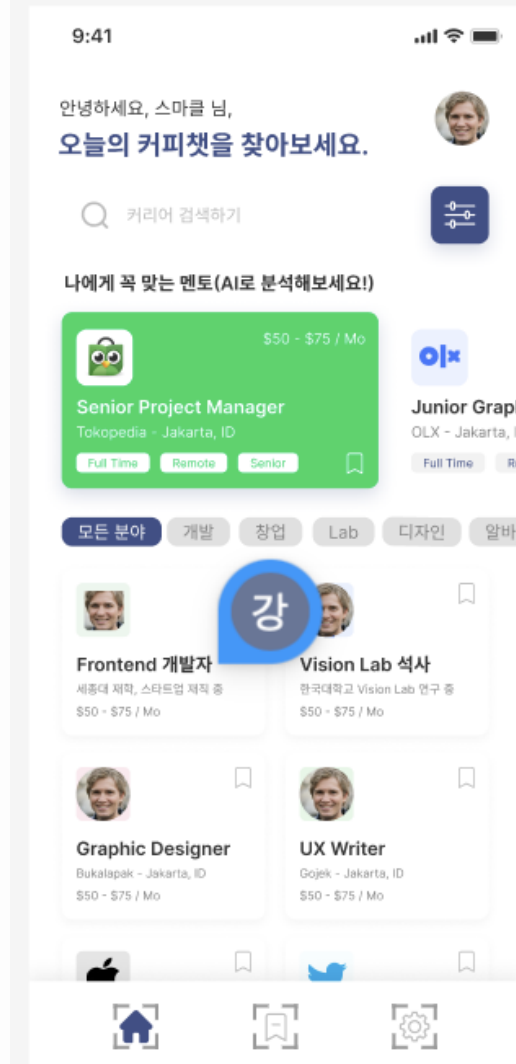
- 팀명: 링커



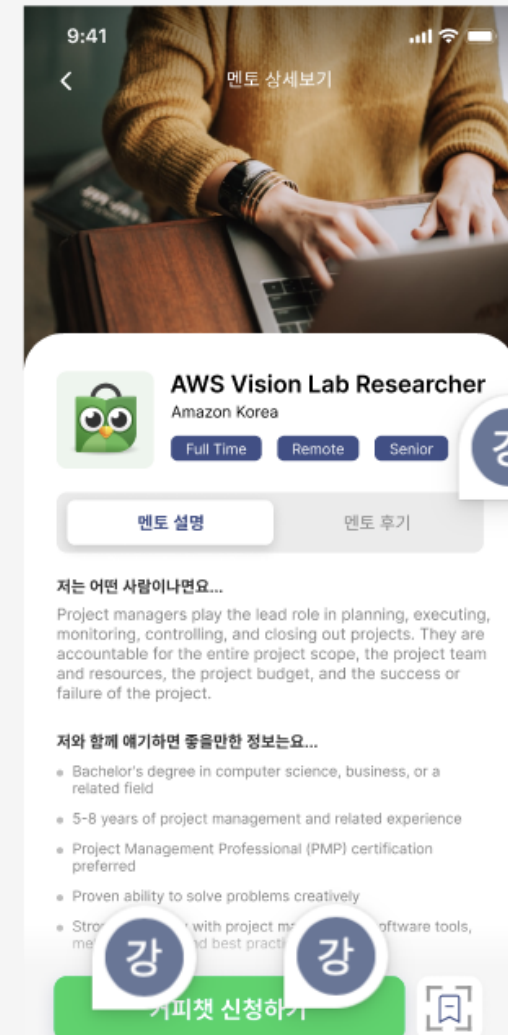
## CONTENTS 01

# 우리의 아이디어는

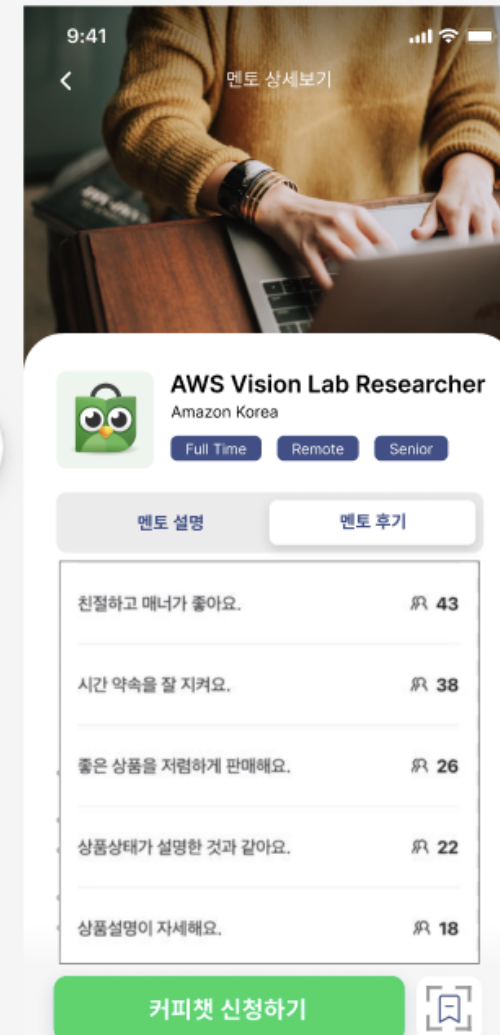
### 02-1 메인 홈 - 멘토 목록 보기



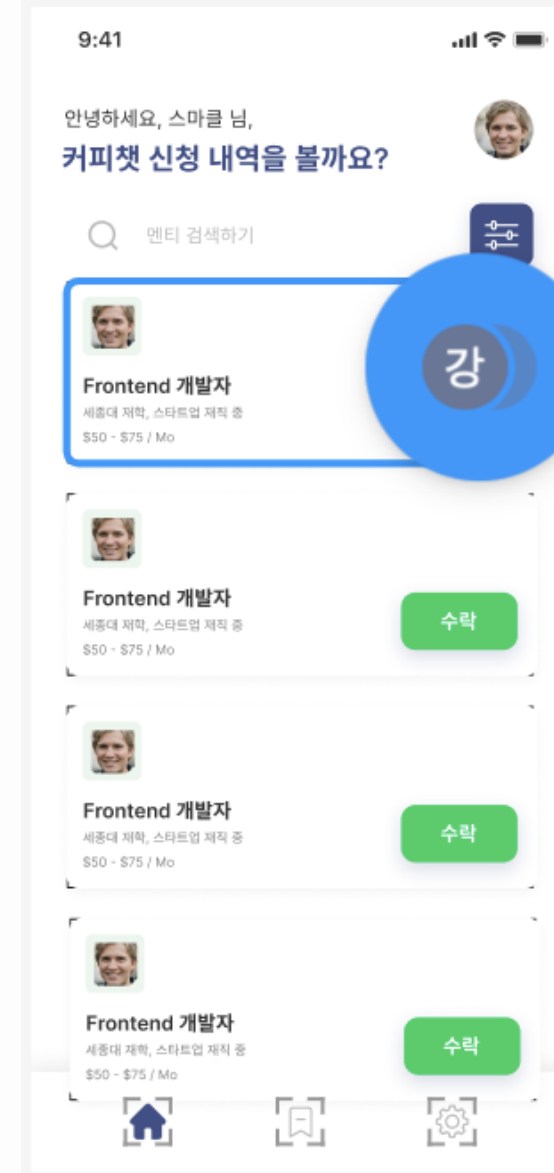
### 02-2 멘토 상세 보기



### 02-2 멘토 상세 보기



### 02-3 멘토가 신청한 멘티의 목록...



# 우리의 아이디어는

API 명세

## API 명세

- 멘토 목록 조회 API (멘티 입장)
- 멘토 상세 조회 API (멘티 입장)
- 커피챗 신청 API (멘티 입장)
- 로그인/회원가입 API (멘토, 멘티 입장)
- 커피챗을 신청한 멘티 목록 조회 API (멘토 입장)
- 커피챗 후기 작성 (멘티 입장)

## 1. 데이터베이스 설계

- ERD Diagram

## 2. API 설계

- 명세에 맞게
  - 멘토 입장 (1명)
  - 멘티 입장 (2명)
- Method, Endpoint(URI), response, request

우리가 연결되는 시간,  
**LINKY.**



# 이런 내용을 담았습니다.

- 01 Linky가 뭔데?
- 02 Linky의 프론트 엔드
- 03 Linky의 백엔드
- 04 어려웠던 부분, 도전했던 부분
- 05 더 시도해 보고 싶은 부분
- 06 시연

LINKY.

# 이런 내용을 담았습니다.

- 01 팀원 / 역할 소개
- 02 아이디어 소개
- 03 기술 스택

LINKY.

---

# 우리의 아이디어는

## 서비스 목표(비전)

- 대학생을 위한 최고의 멘토링(커피챗) 솔루션

## 서비스를 사용하는 유저 스토리?

### <멘티>

- 다양한 커리어 진로를 고민하고 있는 유저 (대학원을 갈지? 취업을 할지? 창업을 할지?)
- 커리어는 정했으나 어떤 루트로 나아갈지 고민하는 유저 (어떤 연구실을 갈지?)
- 현직자의 이야기를 궁금하거나, 한번 썰들을 들어보고 싶은 유저

### <멘토>

- 업계의 주니어들과 소통해보고 싶은 유저
- 소소한 부업을 해보고 싶은 유저
- 업계에서 교육이나 멘토링 방면으로 인지도를 쌓고 싶은 유저

# 커피챗을 더욱 간편하게.



## # 간편한 커피챗 신청

멘티의 입장에서, 어떤 멘토가 있는지 편하게 조회하고 신청할 수 있음

## # 간편한 신청 확인

멘토의 입장에서, 어떤 멘티가 커피챗을 신청했는지 확인할 수 있음

## # 간편한 인공지능

gpt API를 사용하여 커피챗 진행에 도움을 줌

**팀원은  
이렇습니다.**

**팀장 / 프론트엔드**

**20학번  
강인영**

**프론트엔드**

**23학번  
김범열**

**백엔드**

**20학번  
이계무**

**백엔드**

**23학번  
김상완**



# 기술스택은?



# 우리의 프로젝트 진행 상황 (사전 개발)

## 프론트엔드

1. html, css, js 로 개발환경 세팅
2. 목 데이터로 커피챗 멘토 목록 보여주기
3. netlify 로 배포 (-> 앱 설치 가능)

<https://linky-smarthon-2024.netlify.app/>.

# 우리의 프로젝트 진행 상황 (사전 개발)

## 프론트엔드






1. html, css, js 로 개발환경 세팅
2. 목 데이터로 커피챗 멘토 목록 보여주기
3. netlify 로 배포 (-> 앱 설치 가능)

<https://linky-smarthon-2024.netlify.app/>.

# 우리의 프로젝트 구현 결과 (API)

API 명세

## API 명세

- 멘토 목록 조회 API (멘티 입장) 
- 멘토 상세 조회 API (멘티 입장) 
- 커피챗 신청 API (멘티 입장) 
- 로그인/회원가입 API (멘토, 멘티 입장) 
- 커피챗을 신청한 멘티 목록 조회 API (멘토 입장) 
- 커피챗 후기 작성 (멘티 입장)

### 1. 데이터베이스 설계

- ERD Diagram

### 2. API 설계

- 명세에 맞게
  - 멘토 입장 (1명)
  - 멘티 입장 (2명)
- Method, Endpoint(URI), response, request

# 백엔드 구현결과

/mentor => 멘토 데이터 등록

/mentee => 멘티 데이터 등록

/mentorlist => 전체 멘토 목록 리턴

/menteelist => 전체 멘티 목록 리턴

/mentorname?args1=0 => 0번 멘토 상세정보

/mentorlogin => 이메일 패스워드로 멘토 로그인

/menteelogin => 이메일 패스워드로 멘티 로그인

/waitinglist?args1=0 => waitinglist에서 0번 멘토에게 신청한 사람들을 찾아서  
멘토의 정보의 asked에 최신화함

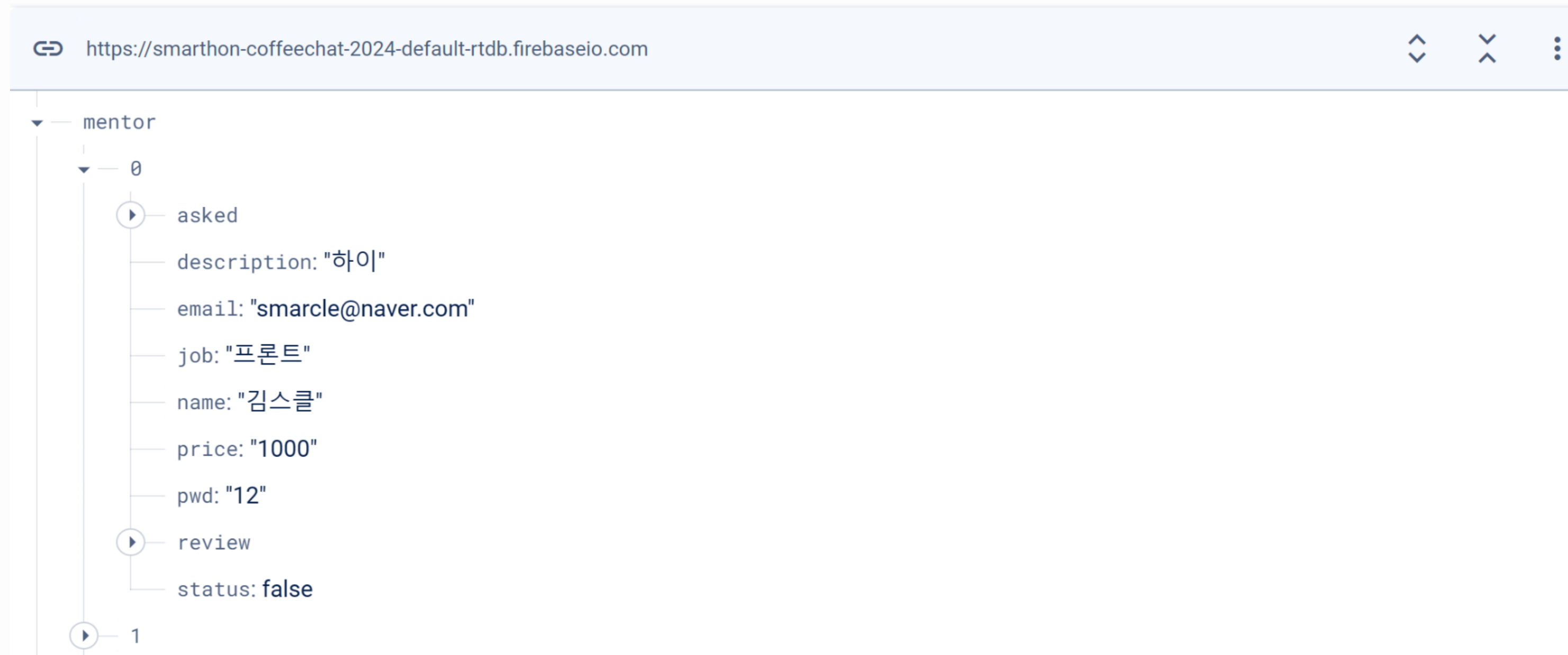
/waitinglistappend?args1=0&args2=1 => 1번 멘토가 멘토리스트 중에서 0번 멘토에게 커피챗을 신청하면,  
1번 멘티를 [0,1]형태로 waitinglist에 append

/waitinglistreturn?args1=0 => 0번 멘토에게 커피챗을 신청한 1,3번 멘티 {"list" : [1,3]} json 반환

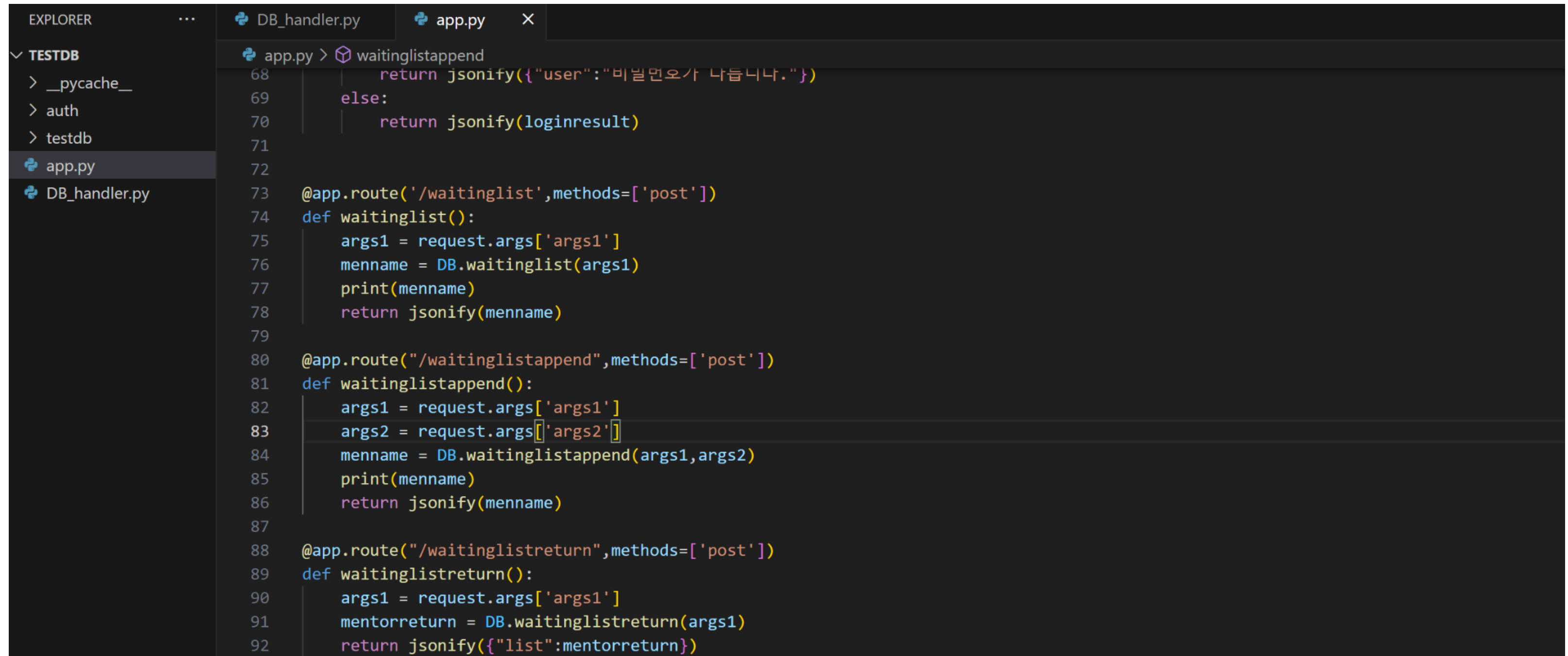
# 백엔드 구현결과



# 백엔드 구현결과



# 백엔드 구현결과

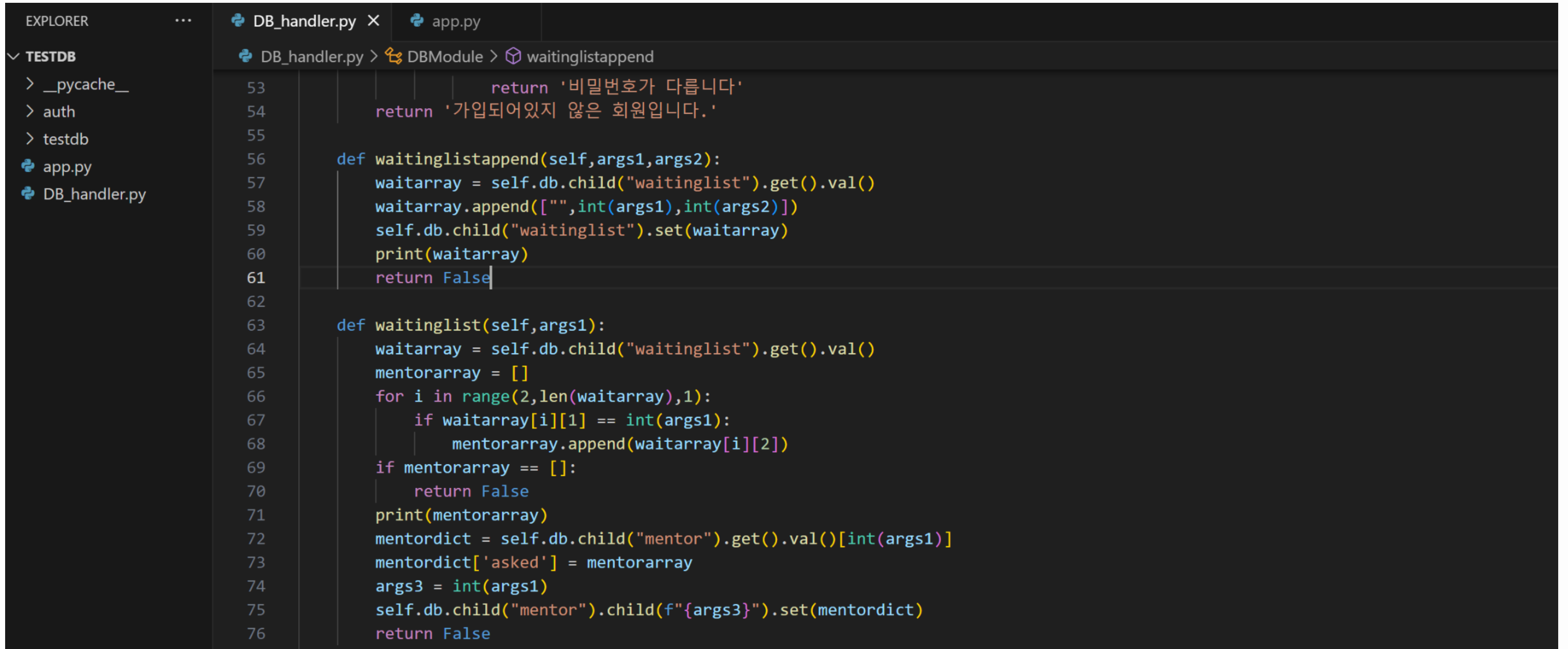


```
EXPLORER
  ...
  ▾ TESTDB
    > __pycache__
    > auth
    > testdb
    + app.py
    + DB_handler.py

app.py > waitinglistappend
68     return jsonify({"user": "비밀번호가 나옵니다."})
69     else:
70         return jsonify(loginresult)
71
72
73 @app.route('/waitinglist', methods=['post'])
74 def waitinglist():
75     args1 = request.args['args1']
76     menname = DB.waitinglist(args1)
77     print(menname)
78     return jsonify(menname)
79
80 @app.route("/waitinglistappend", methods=['post'])
81 def waitinglistappend():
82     args1 = request.args['args1']
83     args2 = request.args['args2']
84     menname = DB.waitinglistappend(args1, args2)
85     print(menname)
86     return jsonify(menname)
87
88 @app.route("/waitinglistreturn", methods=['post'])
89 def waitinglistreturn():
90     args1 = request.args['args1']
91     mentorreturn = DB.waitinglistreturn(args1)
92     return jsonify({"list": mentorreturn})
```



# 백엔드 구현결과



```
EXPLORER  ...  DB_handler.py  app.py

TESTDB
  > __pycache__
  > auth
  > testdb
  app.py
  DB_handler.py

DB_handler.py > DBModule > waitinglistappend

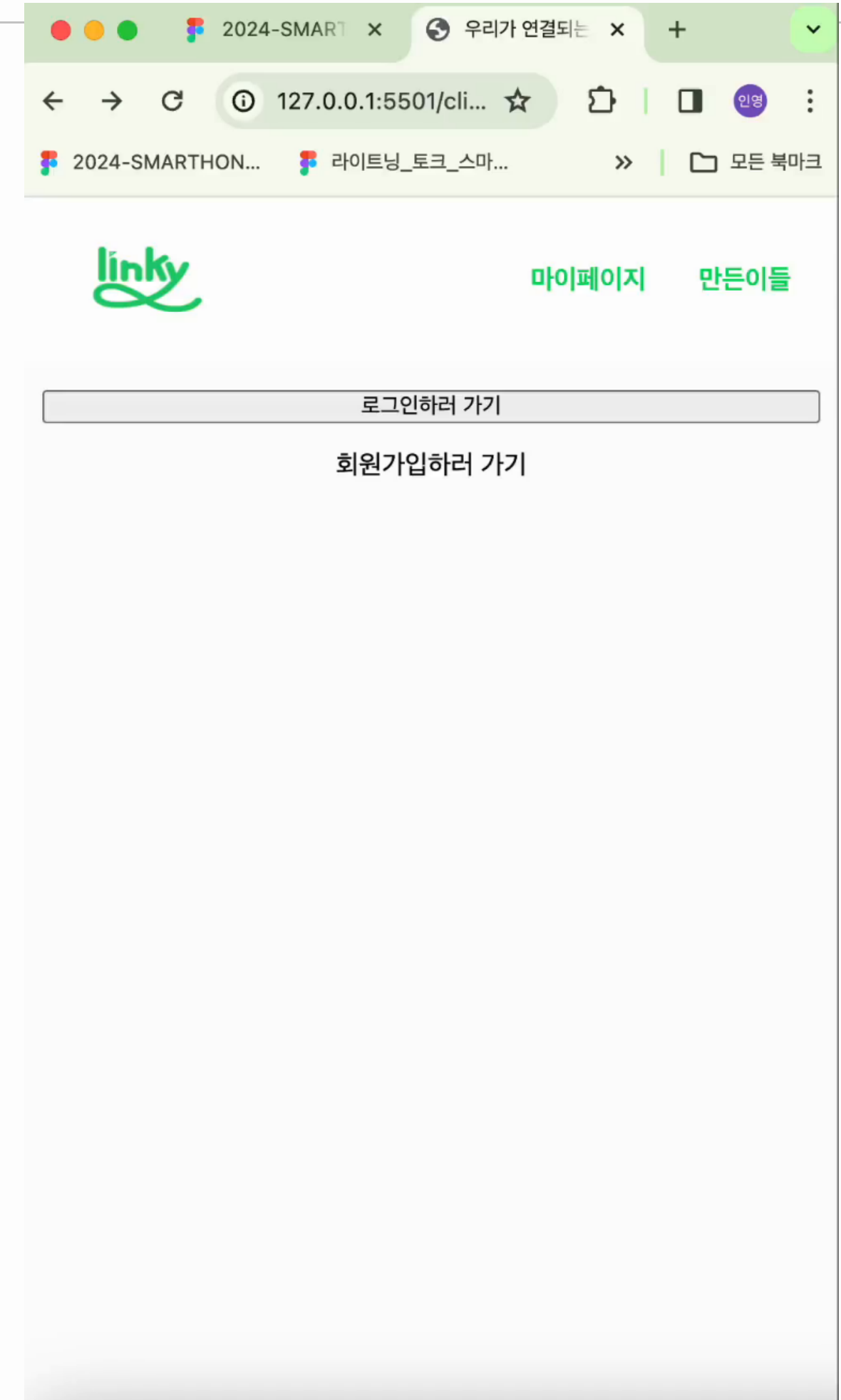
53         return '비밀번호가 다릅니다'
54         return '가입되어있지 않은 회원입니다.'
55
56     def waitinglistappend(self, args1, args2):
57         waitarray = self.db.child("waitinglist").get().val()
58         waitarray.append(["", int(args1), int(args2)])
59         self.db.child("waitinglist").set(waitarray)
60         print(waitarray)
61         return False
62
63     def waitinglist(self, args1):
64         waitarray = self.db.child("waitinglist").get().val()
65         mentorarray = []
66         for i in range(2, len(waitarray), 1):
67             if waitarray[i][1] == int(args1):
68                 mentorarray.append(waitarray[i][2])
69         if mentorarray == []:
70             return False
71         print(mentorarray)
72         mentordict = self.db.child("mentor").get().val()[int(args1)]
73         mentordict['asked'] = mentorarray
74         args3 = int(args1)
75         self.db.child("mentor").child(f"{args3}").set(mentordict)
76         return False
```

CONTENTS 01

# 우리의 프로젝트 구현 결과 (UI/UX)

추가 구현할 부분

- 커피챗 후기 작성 API 연결
- 멘토에게 신청된 커피챗 목록 조회 API 연결



# 어려워던 부분

김범열: 멘토의 정보에서 hover를 사용하여 경력 세부사항을 띄우는 것

chat gpi api를 java scriptdp 연결하여 사용할 수 있도록 하는 부분

김상완: 백엔드가 처음이어서 논리구조를 다루는 것이 어려웠고, FLASK도 파이썬 기반이긴했지만 FLASK 내장 함수를 다루는 것이 어려웠다.

백엔드 협업도 처음이어서 어떻게 해야할지 몰랐다.

강인영: chat gpt api를 연동하는 것과, 답변(응답으로 오는 text)이 잘리지 않게 API를 사용하는 것 & flask 서버를 로컬에서 실행하는 것(버전 문제)

이계무: Firebase가 NoSQL 기반이라서 DB가 늘어날수록 구조가 복잡해지고, DB를 이용하기에 불편함이 있었고, 구조를 짜기 어려웠다.

파이썬 버전이 달라 가상환경을 설정하고 백엔드 코드를 배포해야해서 하지 못했다.

# 더 시도해보고 싶은 부분

김범열: 외부에서 멘토와 멘티를 연결하여 학교 생활이나 진로에 관한 것을 묻고 답하는 프로젝트를 기획했는데, Linky를 더 발전시켜 그곳에서 쓰도록 배포해보고 싶다.

강인영: 남아있는 API 통신을 성공하고, openai fine-tuning을 통해 키워드와 멘티의 상황에 맞는 답변을 도출해보고 싶다.

이계무: 백엔드 코드를 배포해서 웹주소에서 API 요청을 주고받아보고 싶고, NoSQL이 아닌 SQL 기반의 데이터베이스를 다뤄보고 싶다.

김상완: 혼자서 논리를 짜서 코드 시행착오도 해보고, 혼자서 해내고 싶다. 이미 도착했을 땐 여러 얘기가 오간 상태였고, 그 지식이 나에게겐 없었기에 참여가 힘들었던 것 같다.

---

# 감사합니다.

