

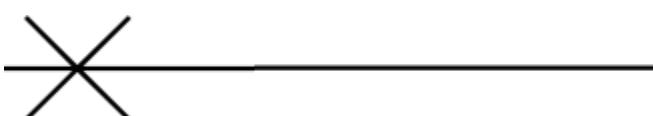
전세사기 예방을 위한

서류 검토 챗봇

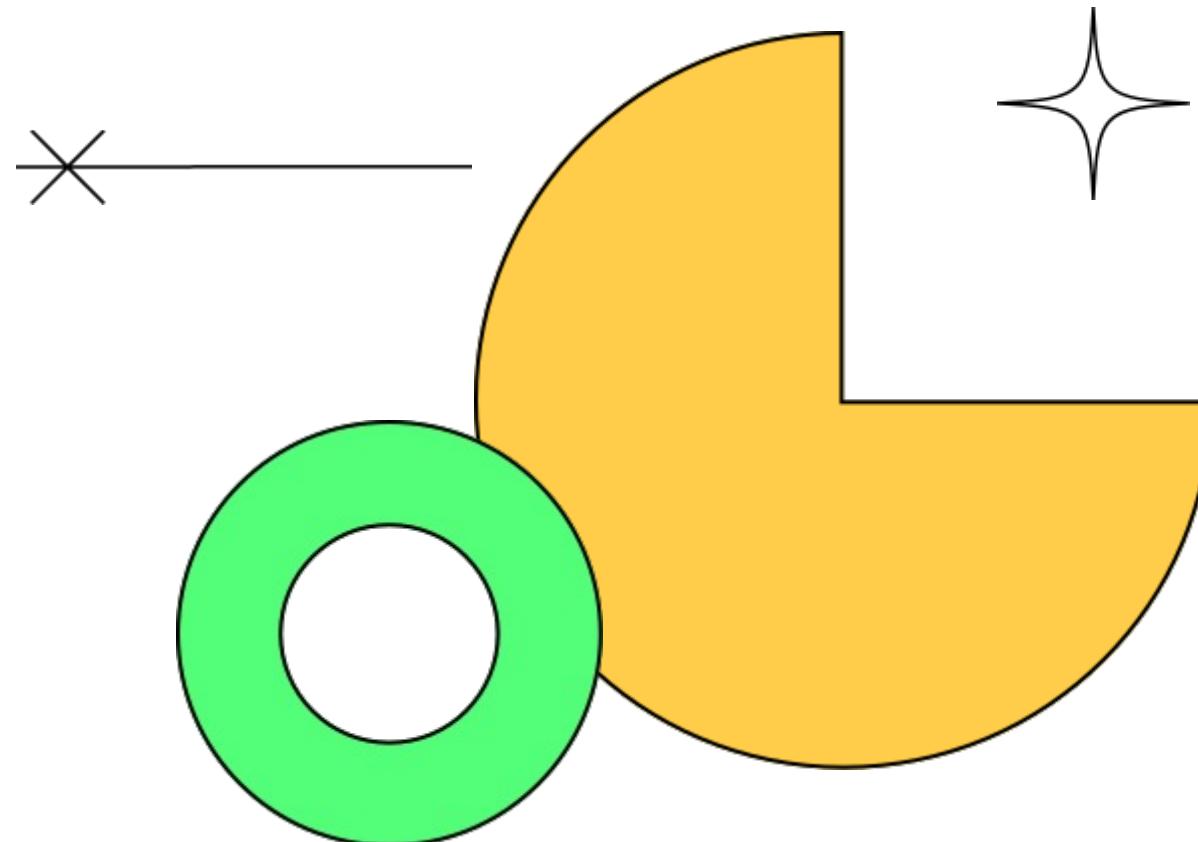
자취 초보생 또는 계약이 처음인 사람들을 위한 사기 방지 챗봇 서비스

팀원 | 최은서 임예은 이선빈 윤현도 김준호

발표자 | 최은서



목 차



01.

프로젝트 주제

전세사기 방지를 위한 챗봇 서비스

02.

팀 소개

최은서 임예은 윤현도 이선빈 김준호

03.

핵심 기능 소개

4가지의 핵심 기능

04.

Front-End

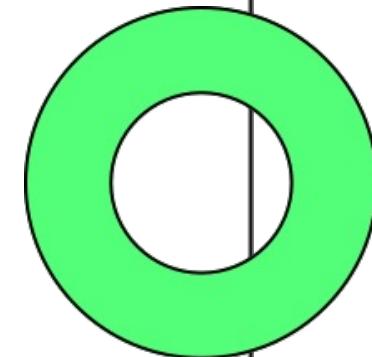
기능 시연 및 React 코드 설명

05.

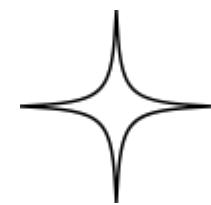
Back-End

AWS Lambda를 활용한 Server 구축

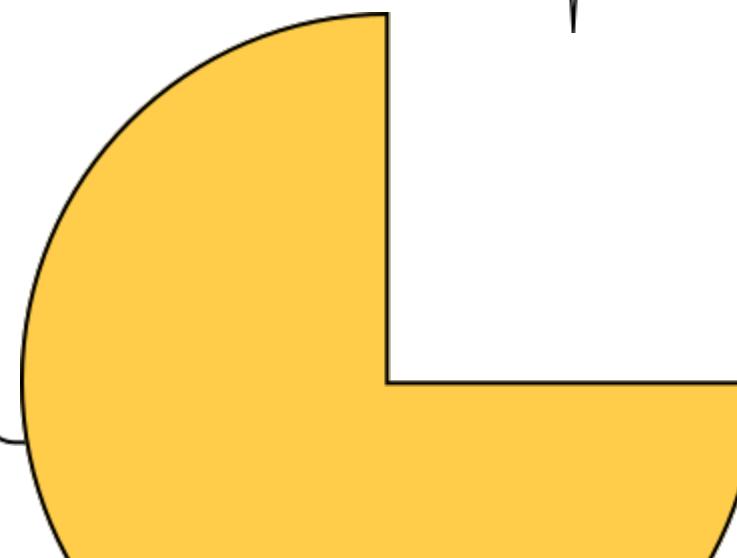
프로젝트 주제



“Cross Check”



전세사기 예방을 위한 서류 검토 챗봇 서비스



팀 및 역할 소개



김준호

백엔드

DB 관리



이선빈

백엔드

LLM 통합



윤현도

백엔드

프로젝트 관리



임예은

프론트엔드

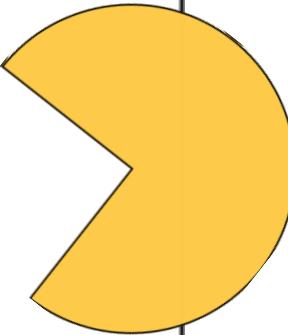
세부 기능



최은서

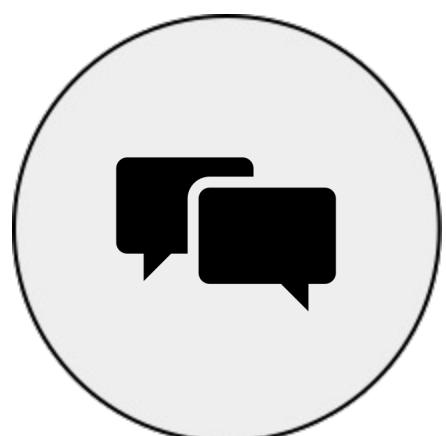
프론트엔드

UX/UI



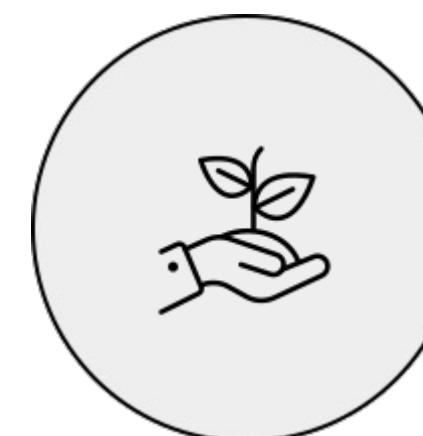
핵심 기능 소개

Chatbot



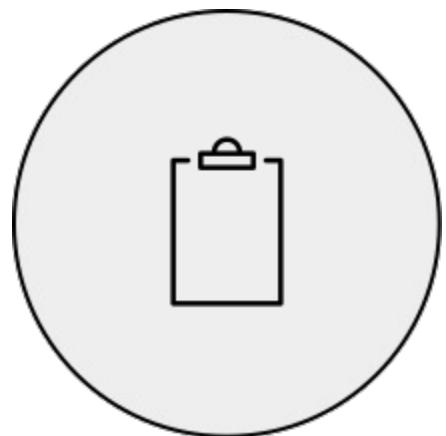
대화 관리 : 챗봇과 사용자 메시지 처리
문서 처리 : PDF 파일 검증 및 업로드
단계별 상담 진행 : 입력한 정보를 바탕으로 상담 진행

History



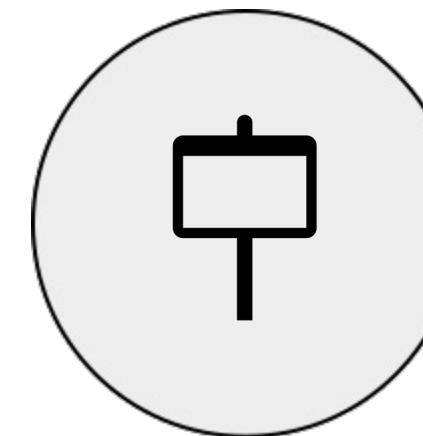
검색 필터 : 실시간 기록 검색
관리 기능 : 기록 편집, 삭제
새 채팅 시작 : 새 채팅 생성 버튼

Legal Brokerage Service

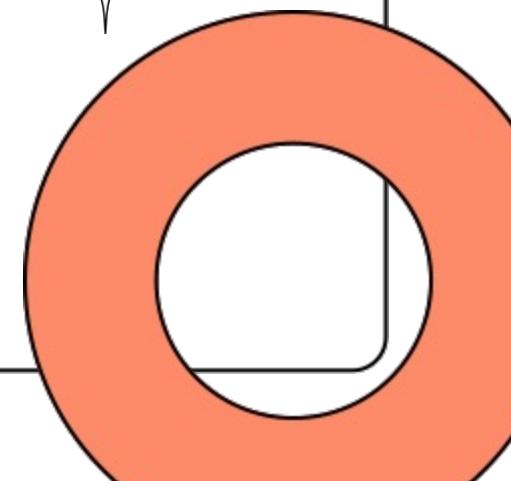
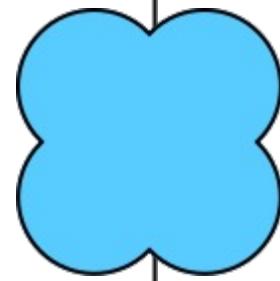
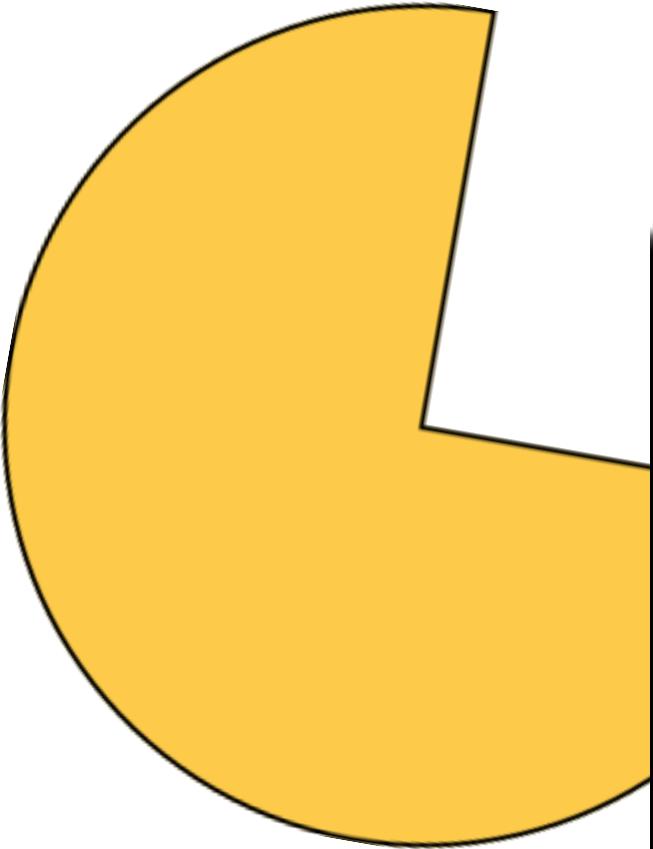


법률 예약 : 날짜, 변호사, 상담 내용 입력 및 예약
전문가 리스트 : 변호사 정보 확인 및 선택
연락처 제공 : 법률 중개인 연락처 제공

Community

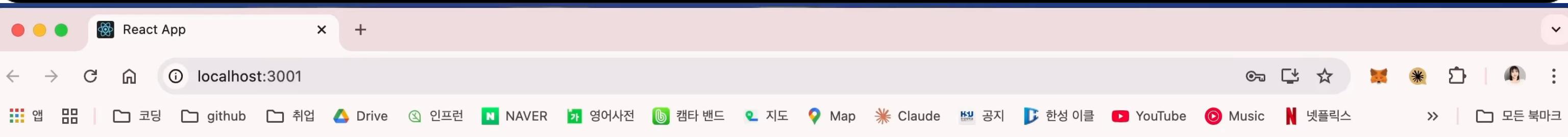


전세 사기 게시판 / 자유 게시판 : 정보 소통, 공유
글쓰기 기능 : 게시글 작성, 공유
댓글 및 좋아요 기능 : 게시글 소통



Front-End

로그인



Cross Check

Login.

아이디

비밀번호

Login

Register

Chat-Bot : 대화 관리

sendToApiGateway()

```
const sendToApiGateway = async (payload) => {
  const url = 'https://qrwrsukdh4.execute-api.ap-northeast-2.amazonaws.com/send_message';
  try {
    console.log('Sending payload to API Gateway:', {
      payloadType: payload.type,
      hasFile: !!payload.file,
      fileName: payload.fileName,
      contentLength: payload.content?.length || 0
    });
    const response = await fetch(url, {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify(payload),
    });
    if (!response.ok) {
      const errorData = await response.json();
      console.error('Server Error:', errorData);
      throw new Error('Server Error');
    }
  } catch (error) {
    console.error('Error sending payload to API Gateway:', error);
  }
};
```

기능

1. API Gateway로 사용자 입력 데이터를 전송하고 응답을 수신
2. 사용자 메시지 또는 파일 데이터를 서버에 전송하고, 서버 응답을 받아 챗봇 메시지로 표시

Chat bot : 문서 처리

convertFileToBase64()

```
const convertFileToBase64 = (file) => {
  return new Promise((resolve, reject) => {
    const reader = new FileReader();
    reader.readAsDataURL(file);
    reader.onload = () => {
      const result = reader.result;
      if (!result.startsWith('data:application/pdf;base64,')) {
        return reject(new Error('Invalid file format or encoding.'));
      }
      const base64Data = result.split(',')[1];
      console.log('File converted to Base64:', {
        fileName: file.name,
        fileType: file.type,
        fileSize: file.size,
        base64Length: base64Data.length
      });
      resolve(base64Data);
    };
    reader.onerror = (error) => reject(error);
  });
};
```

기능

1. PDF 파일을 Base64로 변환하여 서버 전송 준비
2. 업로드된 PDF 파일의 내용을 Base64 포맷으로 인코딩해 서버와 통신에 사용할 수 있도록 처리

Chat bot : 단계별 상담 진행

handleSendMessage()

```
const handleSendMessage = async () => {
  if (!input.trim() && !selectedFile) return;

  const newMessage = { type: 'user', content: input.trim(), align: 'right' };
  setMessages((prev) => [...prev, newMessage]);
  setInput('');
  setSelectedFile(null);

  // 단계별 챗봇 응답 처리
  if (inputStep === 1) {
    setRentalInfo((prev) => ({ ...prev, address: input.trim() })); // 주소 저장
    setMessages(prev => [...prev, {
      type: 'bot',
      content: '집주인의 성함과 주민등록번호를 ,로 구분하여 입력해주세요.',
      align: 'left'
    }]);
    setInputStep(2);
  } else if (inputStep === 2) {
    const [name, idNumber] = input.trim().split(',').map(item => item.trim()); // ,로 나누고 양쪽 공백 제거
    setRentalInfo((prev) => ({
      ...prev,
      landlordName: name || '', // 이름 저장
      landlordIdNumber: idNumber || '' // 주민등록번호 저장
    }));
  }
}
```

기능

1. 사용자 메시지 또는 파일 업로드를 처리하고, 단계별 상담 진행을 관리
2. 사용자 입력이나 파일을 확인하고, 단계(inputStep)에 따라 새로운 메시지를 추가하거나 서버에 데이터를 전송

챗봇 서비스

React App

localhost:3000/Chatbot

Cross Check

+ Start a new chat

Chat Bot

History

Legal Brokerage Service

Community

새로운 채팅방의 제목을 입력해주세요.

제목 입력하기...

채팅방 생성

History : 검색 기능

handleSearchChange ()

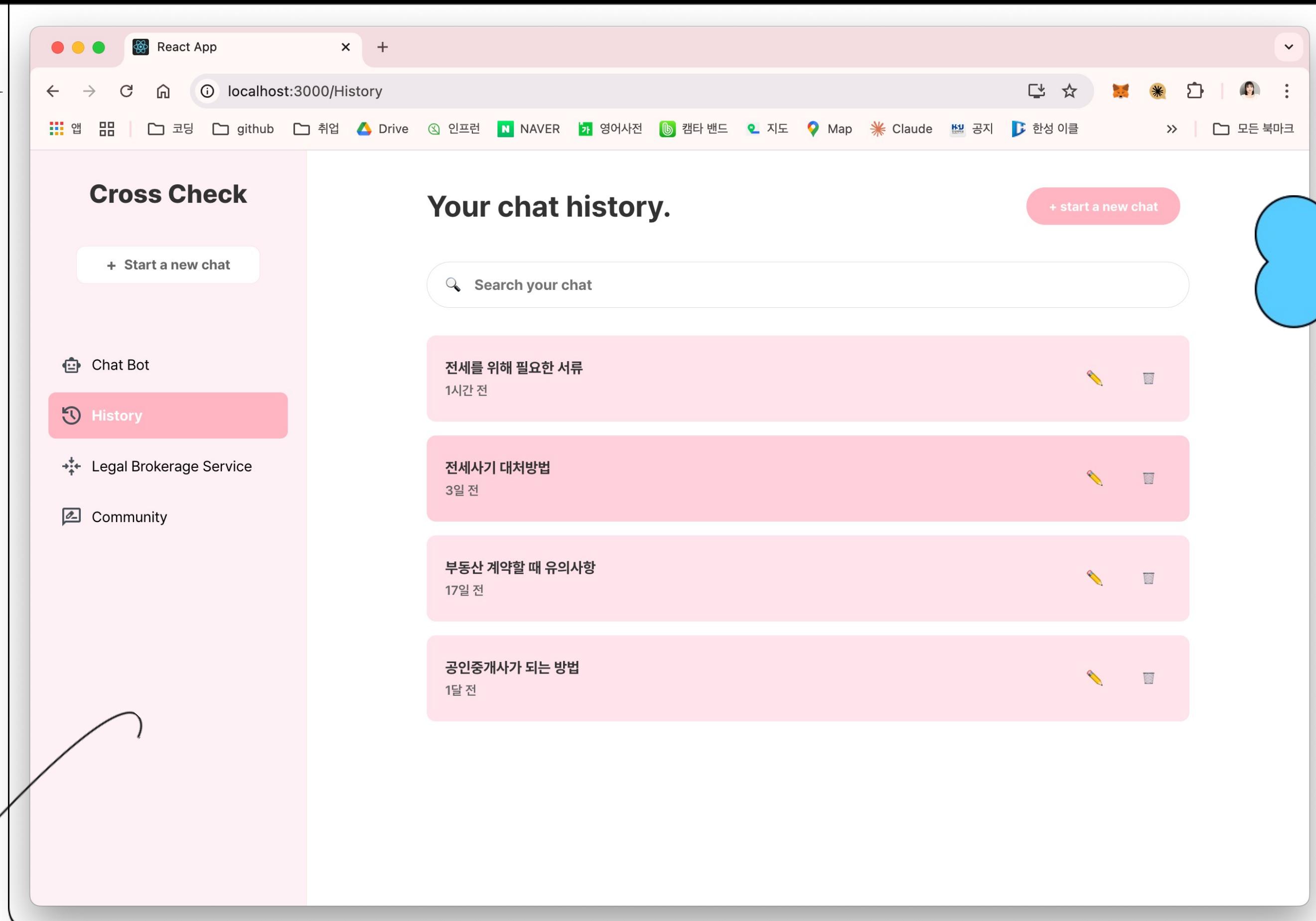
```
// 검색어 변경 핸들러
const handleSearchChange = (e) => {
  const searchValue = e.target.value.toLowerCase(); // 소문자로 변환
  setSearchTerm(searchValue);

  // 검색어에 따라 히스토리 필터링
  const filtered = chatHistories.filter((chat) =>
    chat.title.toLowerCase().includes(searchValue) // 검색어가 포함된 항목 필터링
  );
  setFilteredChats(filtered);
};
```

기능

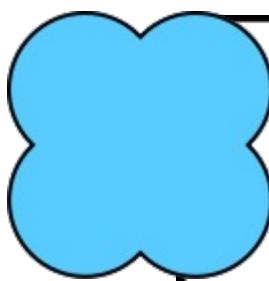
1. 사용자가 입력한 검색어를 소문자로 변환해, `chatHistories` 배열에서 검색어가 포함된 항목만 필터링
2. 필터링된 결과는 `filteredChats` 상태로 저장되어 실시간으로 UI에 반영

히스토리 서비스



Legal Brokerage Service

법률 예약 가능



```
//  
{step === 3 && (  
  <div className="popup-step-3">  
    <h3>상담 내용을 입력하세요</h3>  
    <textarea  
      className="contents-step-3"  
      value={consultationDetails}  
      onChange={(e) => setConsultationDetails(e.target.value)}  
      placeholder="상담 내용을 입력하세요"  
    ></textarea>  
    <div className='button-container-3'>  
      <button  
        className="step-3-before"  
        onClick={handlePreviousStep}>이전</button>  
      <button  
        className="step-3-next"  
        onClick={handleNextStep} disabled={!consultationDetails.trim()}>다음</button>  
    </div>  
  </div>  
)  
{step === 4 && (
```

순서

- 1st 상담 희망 일시 선택
- 2nd 상담할 전문가 선택
- 3rd 상담 내용 입력
- 4th 상담 내역 확인

React App

localhost:3000/LegalService

법률 중개 예약

법률 중개 기사작성

빠른 법률중개 연락

리뷰 남기기

+ Start a new chat

빠르고 정확한 법률 중개 서비스

Cross Check

혹시 전세사기를 당하셨나요?
저희 서비스를 통해 전세사기 피해를 최소화하세요!

원하는 변호사에게 상담을 요청하세요!

김아름 변호사
법무법인 김앤장

이권우 변호사
법무법인 세종

송한주 변호사
법무법인 광장

김진욱 변호사
법무법인 태평양

Chat Bot

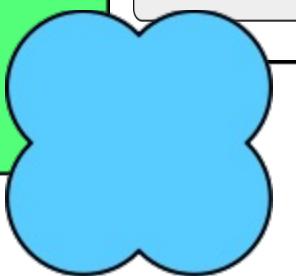
History

Legal Brokerage Service

Community

Community

게시판 소통



```
const Community = () => {
  const [activeBoard, setActiveBoard] = useState('free'); // 'free' or 'scam'
  const navigate = useNavigate(); // useNavigate 흑 사용

  return (
    <div className="community-container">
      <div className="community-content">
        <div className="board-tabs">
          <button
            className={`board-tab ${activeBoard === 'free' ? 'active' : ''}`}
            onClick={() => setActiveBoard('free')}
          >
            자유게시판
          </button>
          <button
            className={`board-tab ${activeBoard === 'scam' ? 'active' : ''}`}
            onClick={() => setActiveBoard('scam')}
          >
            전세사기 게시판
          </button>
        </div>
      </div>
    </div>
  );
}
```

분류

자유게시판

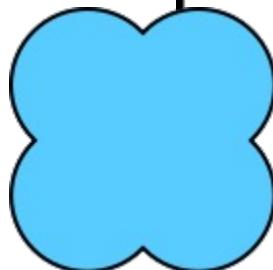
전세 사기 게시판

Community

상세 페이지 보기

```
const handleAddComment = (e) => {
  e.preventDefault();
  if (!commentInput.trim()) return;
  const newComment = {
    id: Date.now(),
    user: currentUser.name,
    comment: commentInput,
    replies: [],
  };
  setComments([...comments, newComment]);
  setCommentInput("");
};

const handleAddReply = (e, commentId) => {
  e.preventDefault();
  if (!replyInput.trim()) return;
  const updatedComments = comments.map((comment) => {
    if (comment.id === commentId) {
      return {
        ...comment,
        replies: [
          ...comment.replies,
          { id: Date.now(), user: currentUser.name, comment: replyInput },
        ],
      };
    }
    return comment;
  });
}
```



댓글 기능

1. 새로운 댓글을 추가하여 댓글 리스트를 업데이트하는 기능
2. 사용자가 댓글을 입력하고 "보내기" 버튼을 누르면 호출됨
3. 입력된 댓글이 유효한지 확인 후, comments 상태를 업데이트하여 새로운 댓글을 리스트에 추가

Community

글 작성 기능

```
const Posting = () => {
  const navigate = useNavigate();

  const handleSubmit = (e) => {
    e.preventDefault(); // 기본 동작(페이지 새로고침) 방지
    alert("등록이 되었습니다."); // 팝업 메시지
    navigate("/Community") // CommunityBoard.jsx로 이동
  };

  return (
    <div className="posting">
      <Sidebar />
      <div className="posting-container">
        <form className="posting-form" onSubmit={handleSubmit}>
          {/* 제목 */}
          <div className="form-group">
            <label htmlFor="title" className="label">
              제목
            </label>
            <input
              type="text"
              id="title"
              className="input"
              placeholder="제목을 입력해주세요"
              required
            />
          </div>
        </form>
      </div>
    </div>
  );
}
```



1. 정보 공유: 유용한 정보를 커뮤니티와 나눌 수 있음
2. 문제 해결 도움:
작성한 글을 통해 도움을 받을 수 있음

커뮤니티 서비스

React App

localhost:3000/Community

앱 코딩 github 취업 Drive 인프런 NAVER 영어사전 캠팡 지도 Map Claude 공지 한성 이클 YouTube Music 넷플릭스 모든 북마크

Cross Check

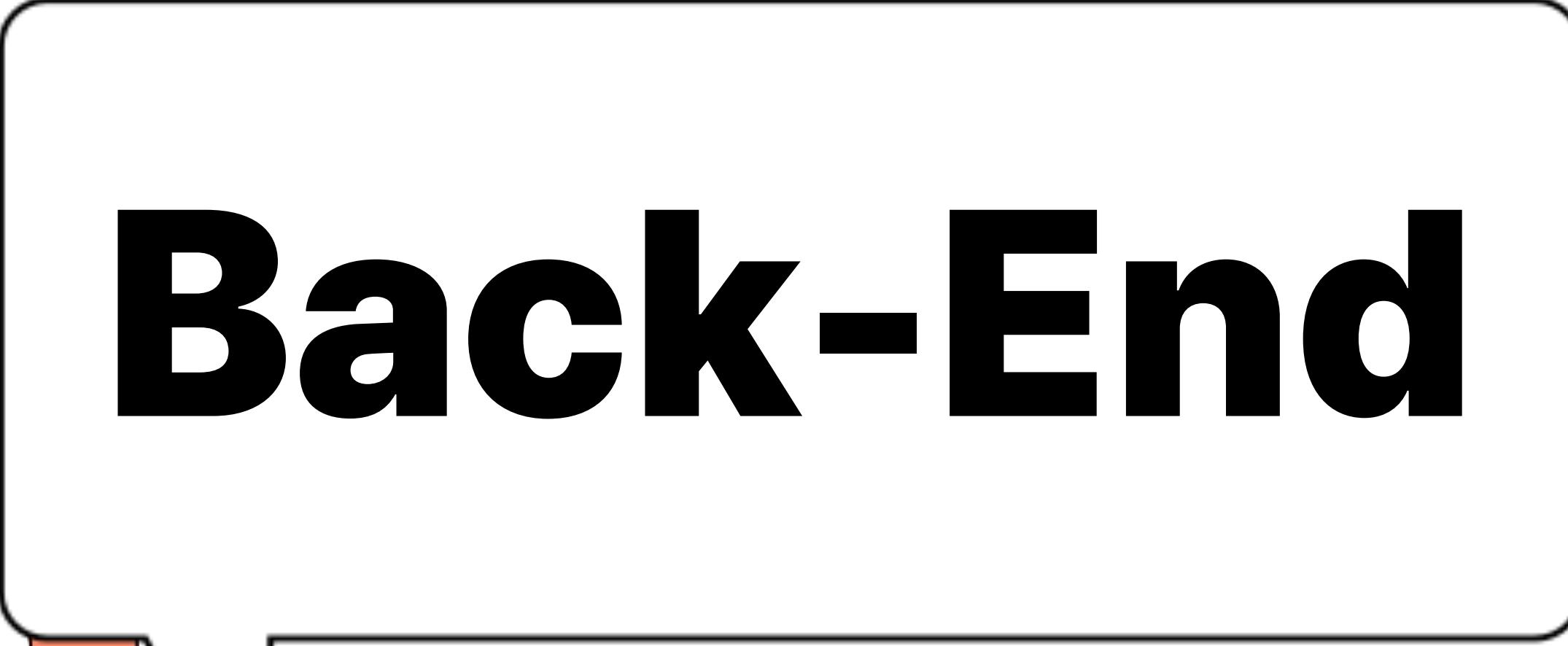
+ Start a new chat

Chat Bot History Legal Brokerage Service Community

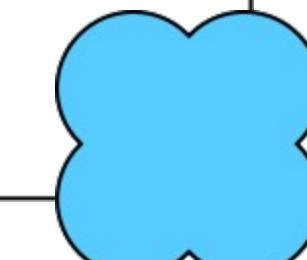
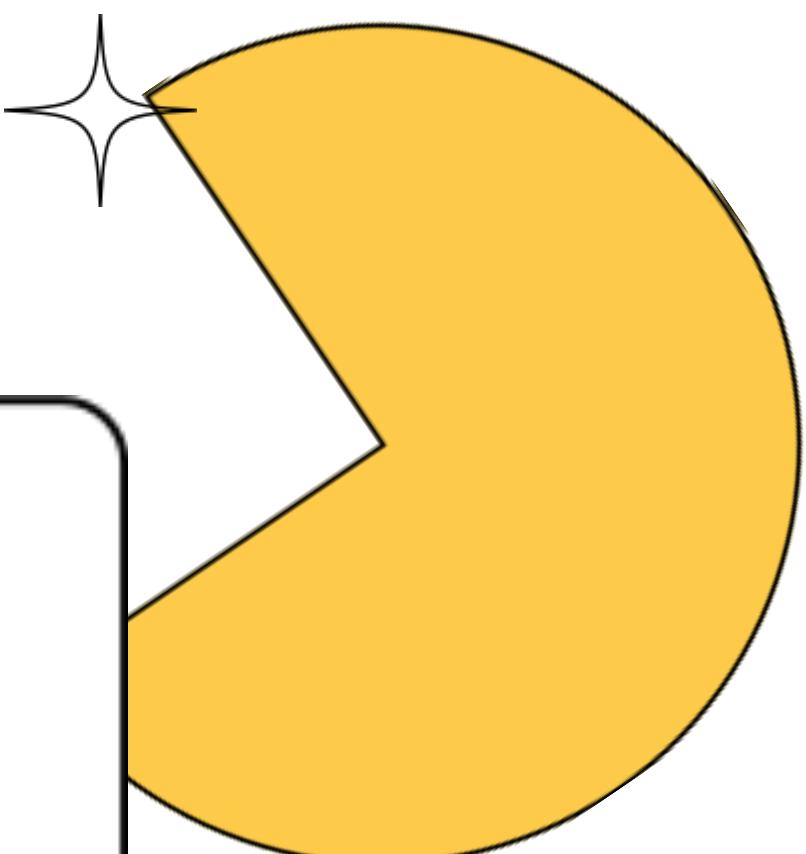
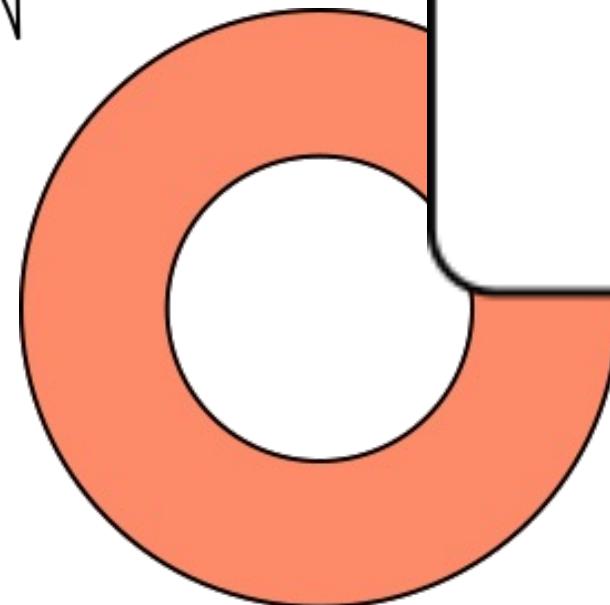
자유게시판 전세사기 게시판 + write

No.	Title	Writer	Date	Views	Likes	Comments
01	집 계약할 때 뭐 먼저 봐야하나요?	집들이	2024.11.02	112	30	5
02	자취생 저녁 메뉴 추천	자취생	2024.10.28	329	54	10
03	첫 전세 계약할 때 쟁겨야 할 서류	법사	2024.10.25	219	27	16
04	부동산 중개수수료 협의 가능한가요?	도미야	2024.10.23	498	38	16
05	전세자금대출 심사 얼마나 걸리나요?	하우스	2024.10.20	284	13	16
06	보증금 올려주면 월세 낮춰준다는데 괜찮을까요?	부동산 초보	2024.10.19	351	32	16
07	신축 아파트 하자보수 기간이 어떻게 되나요?	내 집	2024.10.16	257	14	16
08	집주인이 갑자기 계약 취소하자고 하네요ㅠㅠ	집순이	2024.10.02	184	40	16
09	집 계약할 때 뭐 먼저 봐야하나요?	집들이	2024.11.02	112	30	5
10	자취생 저녁 메뉴 추천	자취생	2024.10.28	329	54	10

< 1 >



Back-End



What is Serverless?

Serverless

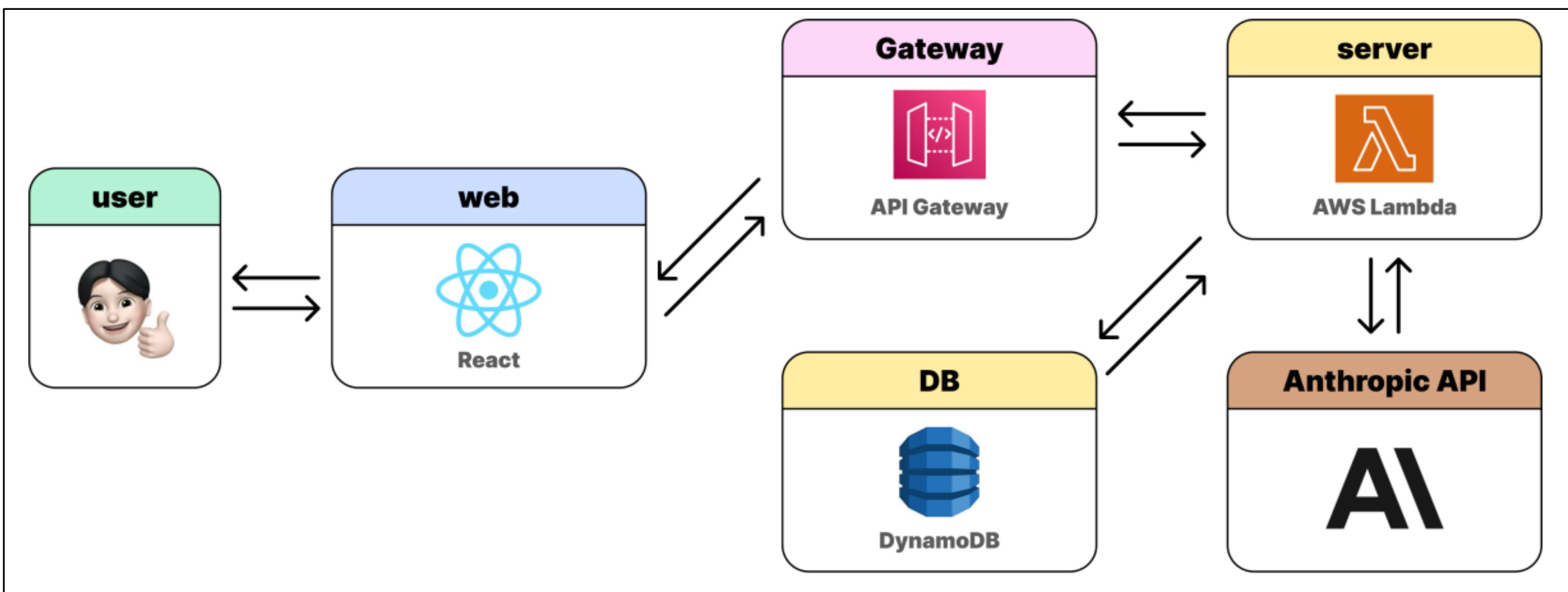
= FaaS (Function as a Service)

함수로 구현되어 있는 서버

넓은 의미에서 Serverless는 개발자가 서버 관리에 신경 쓰지 않고 코드 실행에만 집중할 수 있는 환경

서비스 아키텍처를 위한

기술 스택 정리



API Gateway

요청이 API Gateway로 도착하면 Lambda 함수를 호출함.

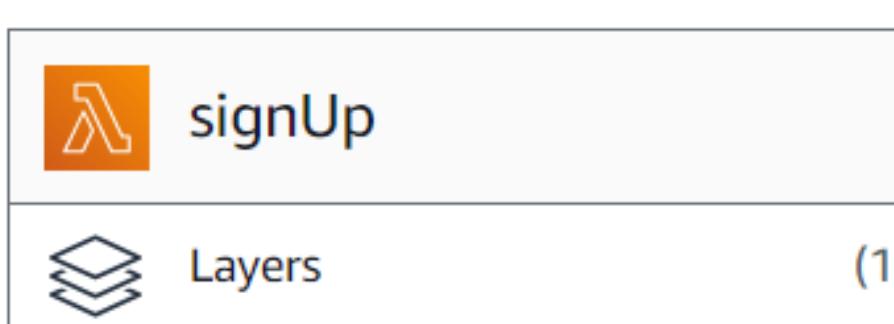
만들어둔 함수에 Gateway를 Trigger로 연결하면,

함수를 호출할 수 있게 End-point를 생성해줌.

▼ 함수 개요 정보

다이어그램

템플릿

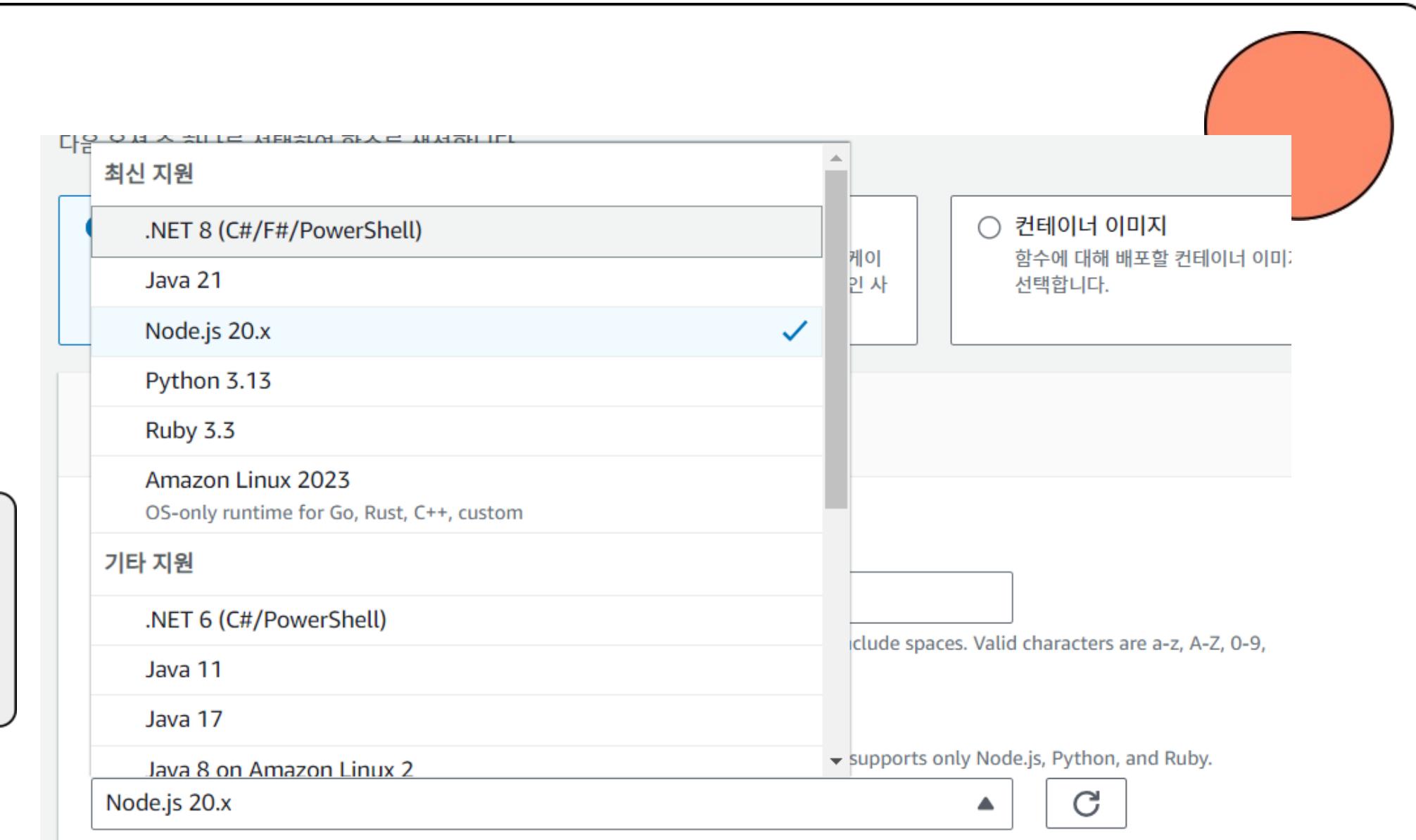


API 게이트웨이

+ 대상 추가

+ 트리거 추가

Lambda Function



람다 함수는 다양한 함수 런타임을 제공한다.
그리고 서버의 기능을 함수 단위로 수행하기 때문에
특정 함수는 Python 다른 함수는 Java와 같이 필요에 따라 작성할 수도 있다.

Lambda

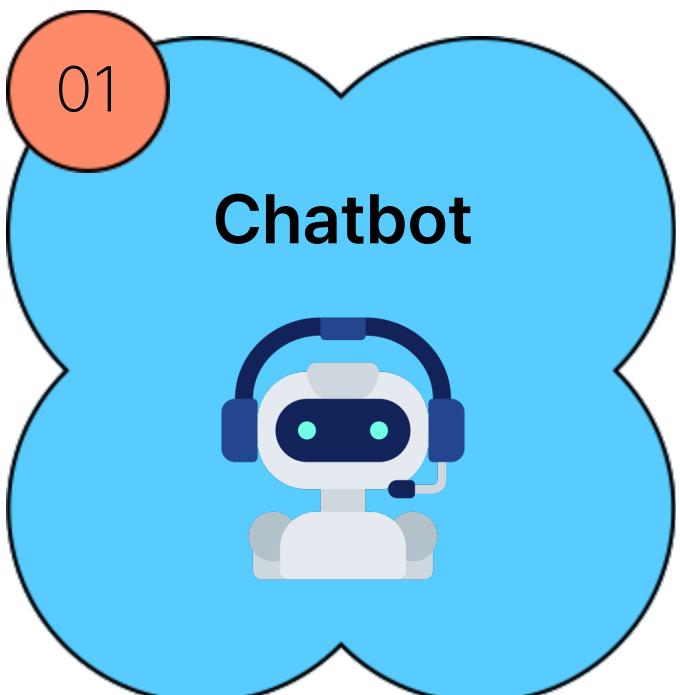
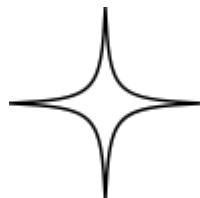
Function

함수가 트리거에 의해 실행되었다면 가장 먼저
실행되는 함수는 lambda_handler 이다.
Python 에서의 __main__과 같이 진입점의 역할을 수행한다.

```
### lambda_handler 간단한 예시
def lambda_handler(event, context):
    # 요청 데이터 파싱
    body = json.loads(event['body'])
    name = body.get('name')
    loginId = body.get('loginId')
    password = body.get('password')
    birth = body.get('birth')
    return {
        "statusCode": 200,
        "body": json.dumps({
            "status": 200,
            "success": True,
            "message": "회원가입에 성공했습니다."
        })
    }
```

함수가 결과값을 반환하면
다시 API 게이트웨이를 거쳐 클라이언트 측으로 데이터가 전달되며
한 번의 요청을 처리하는 과정이 마무리된다.

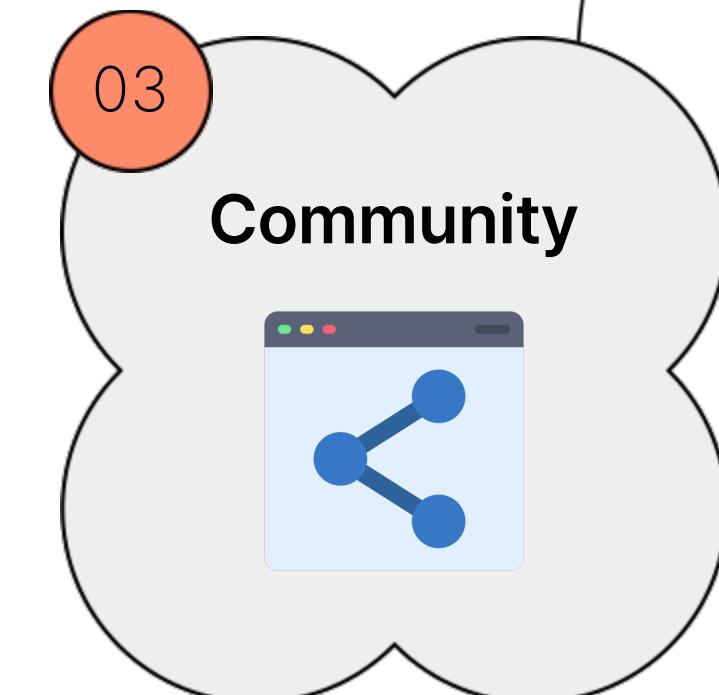
핵심 기능 소개



- `create_chat_room`
- `send_message`



- `get_history`
- `get_detailed_history`
- `del_history`



- `write_post, del_post`
- `get_community`
- `get_detailed_post`
- `write_comment`
- `del_comment`

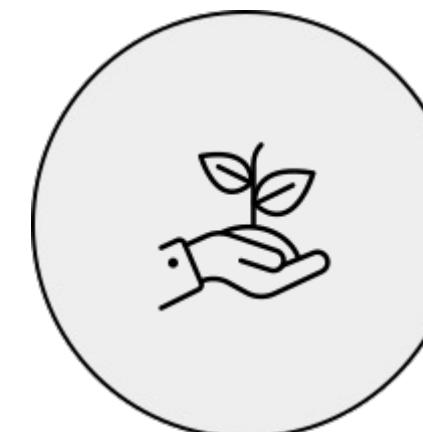
핵심 함수 소개

send_message



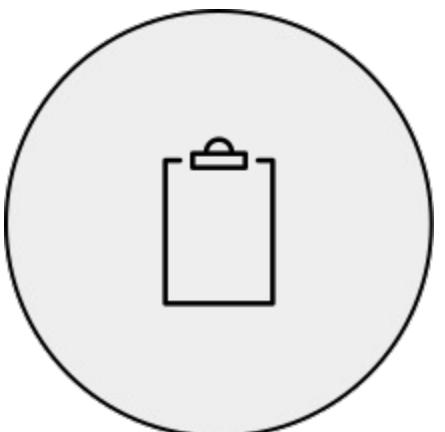
사용자가 입력한 메시지를 LLM에게 전달하고
API 호출을 통해 생성된 응답을 반환하는 함수

get_history



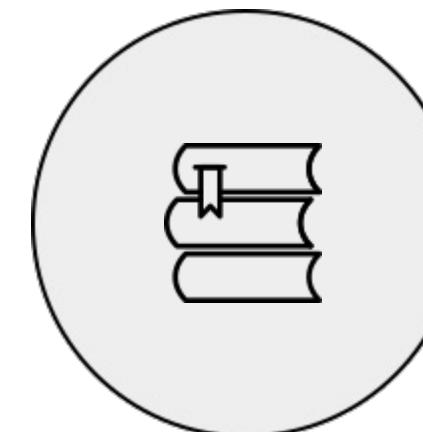
채팅 기록 전체 목록을 불러오는 함수

get_detailed_history



특정 채팅방에 대한 대화 내역을 모두 불러오는 함수

write_post



커뮤니티에 게시물을 작성하는 함수

Chat Bot

Send-Message

```
def send_message_to_claude(messages):
    try:
        # Claude API 설정
        headers = {
            "Content-Type": "application/json",
            "anthropic-version": "2023-06-01",
            "x-api-key": CLAUDE_API_KEY
        }

        # 메시지를 객체 형식으로 변환
        message_objects = [{"role": "user", "content": msg} for msg in messages]

        payload = {
            "model": "claude-3-opus-20240229",
            "max_tokens": 1000,
            "system": """
                당신은 전세사기 위험을 전문적으로 분석하는 AI 전문가입니다.
                법률, 부동산, 금융 분야의 깊은 지식을 가진 전문가로서 다음 가이드라인을 엄격히 준수하세요.
            """,
            "messages": message_objects
        }
    
```

프롬프트 엔지니어링을 통해 전세사기 전문가로서의 역할을 하도록 LLM모델 튜닝 후 입력된 메시지와 함께 전달

```
        response = requests.post(
            CLAUDE_API_URL,
            headers=headers,
            json=payload,
            timeout=30 # 30초 타임아웃 설정
        )
    
```

Chat Bot

Send-Message

메시지 저장

```
timestamp = datetime.utcnow().isoformat()
message_item = {
    'messageId': message_id,
    'chatRoomId': chatRoomId,
    'timestamp': timestamp,
    'type': messageType,
    'content': content.strip() if content else '',
    'claudeResponse': claude_response if claude_response else '',
    'file': file_url if file_url else None,
    'pdfExtractedText': extracted_text if file_url else None
}

message_table.put_item(Item=message_item)
```

대화의 맥락 이해,
히스토리 조회를 위한 메시지 저장

이전 메시지 가져오기

```
response = message_table.scan(FilterExpression=Key('chatRoomId').eq(chatRoomId))
previous_messages = [item['content'] for item in response.get('Items', []) if 'content' in item]
```

History

get-history

```
def get_sorted_chat_rooms_by_user(userId):
    try:
        # userId에 해당하는 모든 채팅방 내역 조회
        response = table.scan(
            FilterExpression=Attr('userId').eq(userId)
        )

        # 결과 처리: endedAt(마지막 채팅 시간) 기준으로 내림차순 정렬
        chat_rooms = response.get('Items', [])

        # endedAt가 None인 경우를 처리하여 정렬
        sorted_chat_rooms = sorted(
            chat_rooms,
            key=lambda x: x.get('endedAt') if x.get('endedAt') is not None else '',
            reverse=True
        )
    
```

userId를 사용하여
특정 유저의 모든 채팅방 조회

```
return {
    'statusCode': 200,
    # Decimal 변환 함수 사용
    'body': json.dumps(sorted_chat_rooms, default=decimal_default_proc)
}
except Exception as e:
    return {
        'statusCode': 500,
        'body': json.dumps({'error': str(e)})
}
```

History

get-detailed-history

```
def get_chat_messages(chat_room_id):
    try:
        # chatRoomId를 기준으로 Scan 수행
        response = table.scan(
            FilterExpression=Attr('chatRoomId').eq(chat_room_id)
        )

        # 결과 처리
        messages = response.get('Items', [])
        return {
            'statusCode': 200,
            'body': json.dumps(messages, default=decimal_to_float)
        }
    except Exception as e:
        return {
            'statusCode': 500,
            'body': json.dumps({'error': str(e)})
        }
```

chat_room_id를 이용해 해당 채팅방의 모든 메시지 반환

Community

write_post

```
def lambda_handler(event, context):
    body = json.loads(event['body'])

    userId = body['userId']
    c_name = body['c_name']
    title = body['title']
    image = body['image']
    content = body['content']

    community_table = dynamodb.Table('communityDB')
    post_table_name = f'{c_name}_post_DB'
    post_table = dynamodb.Table(post_table_name)

# 게시글 개수 조회 및 ID 생성
    response = community_table.get_item(Key={'c_name': c_name})
    current_post_count = response['Item']['post_count']
    post_id_counter = response['Item']['postIdCounter']
    new_post_id = f'{userId}_{c_name}_{post_id_counter + 1}'
```

```
# 게시글 저장
    post_table.put_item(Item={
        'userId': userId,
        f'{c_name}_post_id': new_post_id,
        'title': title,
        'image': image,
        'createdAt': datetime.now().isoformat(),
        'content': content,
        'like': 0,
        'watched': 0,
        'comment_count': 0,
        'commentIdCounter': 0
    })
```

게시물 정보를 전달받아 게시물 ID 생성 후 DB에 저장

발표 끝~

감사합니다