

1. 문제제기

사회적 문제 정의



든든한 육아 도우미

옹알옹알. 맡겨주세요!

| 문제 제기

사회적 문제 제기

사회적 지원 부족

정보의 부족

| 육아 부담 증가

맞벌이 부부의 시간적, 정신적 여유 부족.

육아 결정 부담.

| 부모 교육과 정보 접근성 부족

정확, 신뢰적, 최신적 정보 습득 어려움.

육아의 기회 부족으로 작용.

| 사회적 지지 부족

육아는 외롭고 고립된 경험이 될 수 있음.

첫 아이, 혹은 사회적 지지망이 약한 부모에게 치명적.

| 아기 발달 단계 이해 부족

아기 발달 특성에 대한 이해 부족.

적절한 대응 어려움.



든든한 육아 도우미

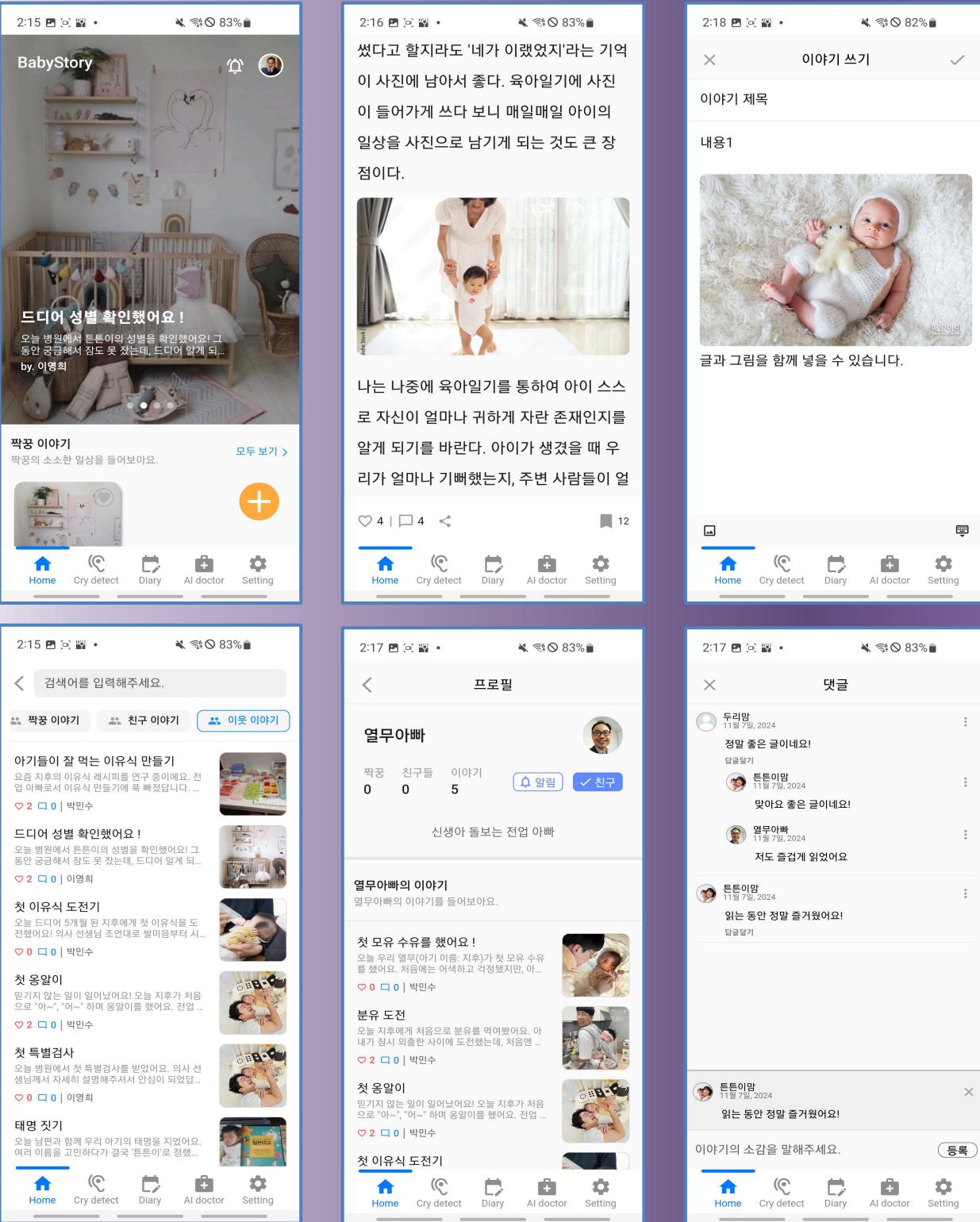
옹알옹알. 맡겨주세요!

육아 보조 및 커뮤니케이션 앱

정보 공유의 가치

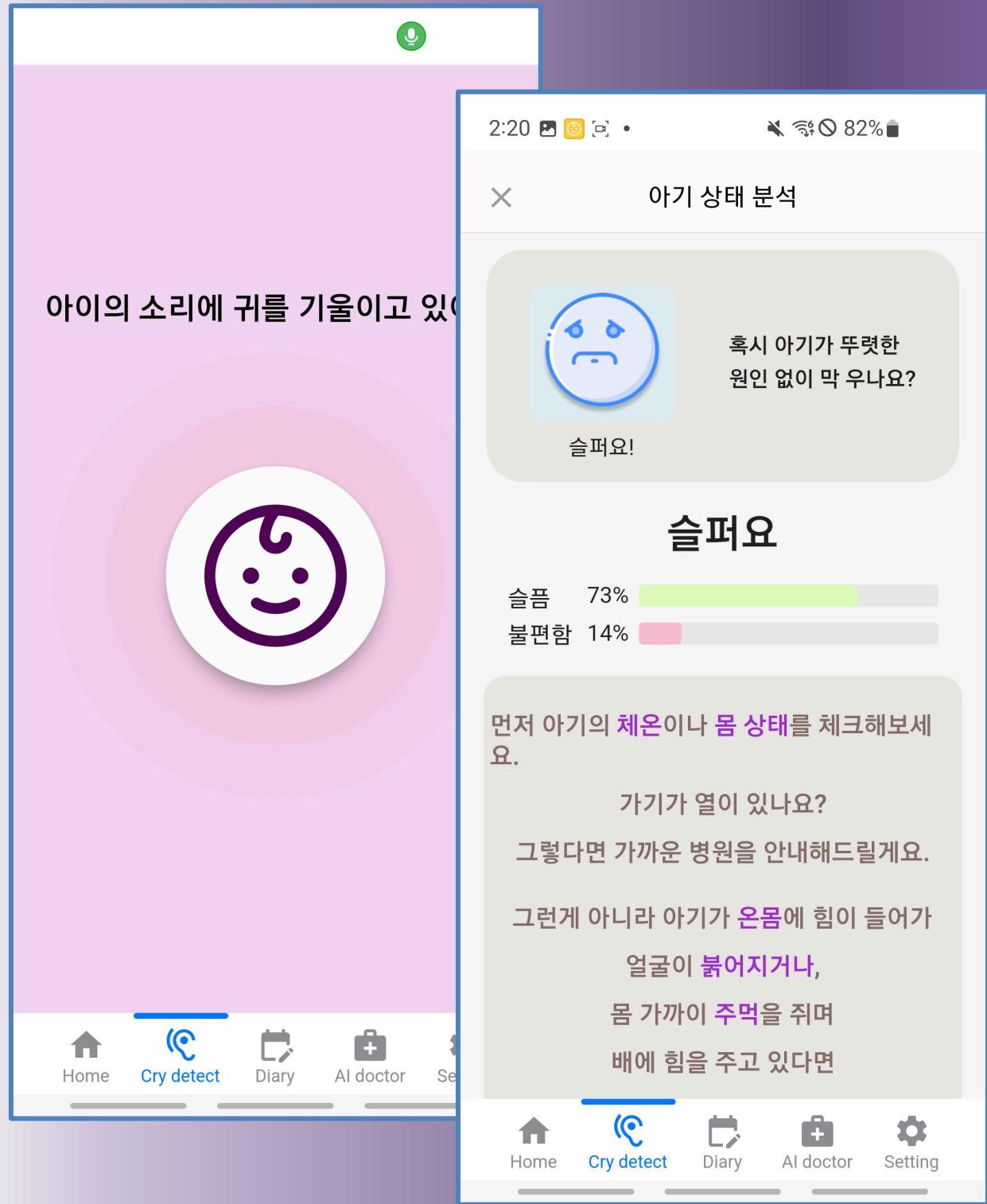
육아 정보 커뮤니티

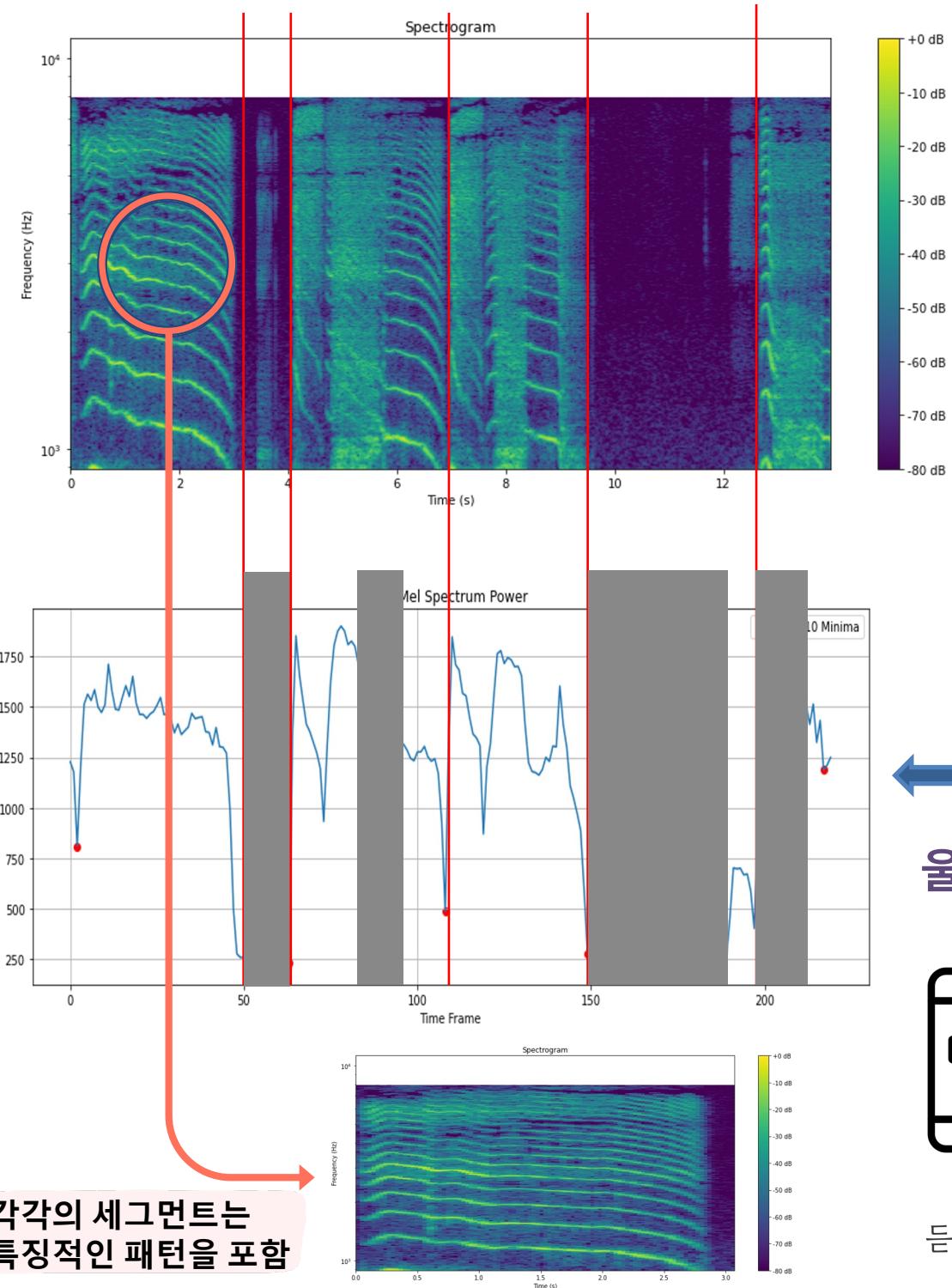
- 목적:** 사용자가의 글을 나눔으로써 가치를 창출한다.
- 기능 페이지 개수:** 12개
- Backend API Endpoint 개수:** 51개
- 주요 기능**
 - 메인 페이지에서 종합적인 글을 확인할 수 있다.
 - 이야기를 범위와 키워드로 검색할 수 있다.
 - 글과 이미지를 교차하여 글을 작성할 수 있다.
 - 글의 공개범위에 따라 접근을 제한할 수 있다.
 - 이야기에 좋아요, 댓글, 스크랩을 할 수 있다.



차별화된 AI 활용 아기 울음 감지 및 분석

- 목적:** 아기의 울음을 감지하고 원인을 분석한다.
- 기능 페이지 개수:** 11개
- Backend API Endpoint 개수:** 4개
- 주요 기능**
 - 백그라운드에서 실시간으로 울음을 감지한다.
 - 울음이 감지되면 울음을 분석한다.
 - 울음 원인을 해소방안과 함께 제시한다.
 - 울음이 감지되면 웨어러블 디바이스로 알린다.

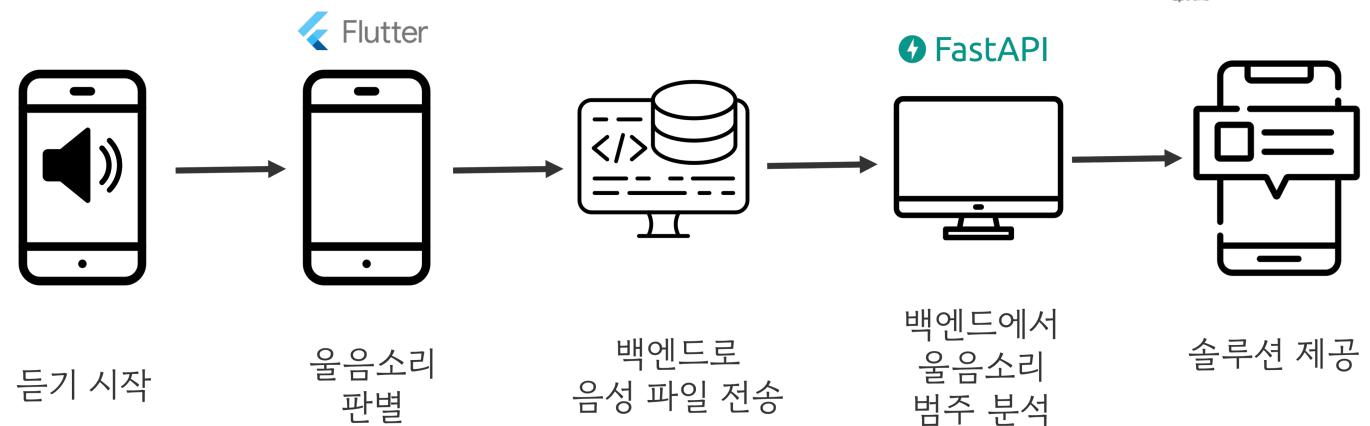




아기 울음 감지 및 울음분석 AI

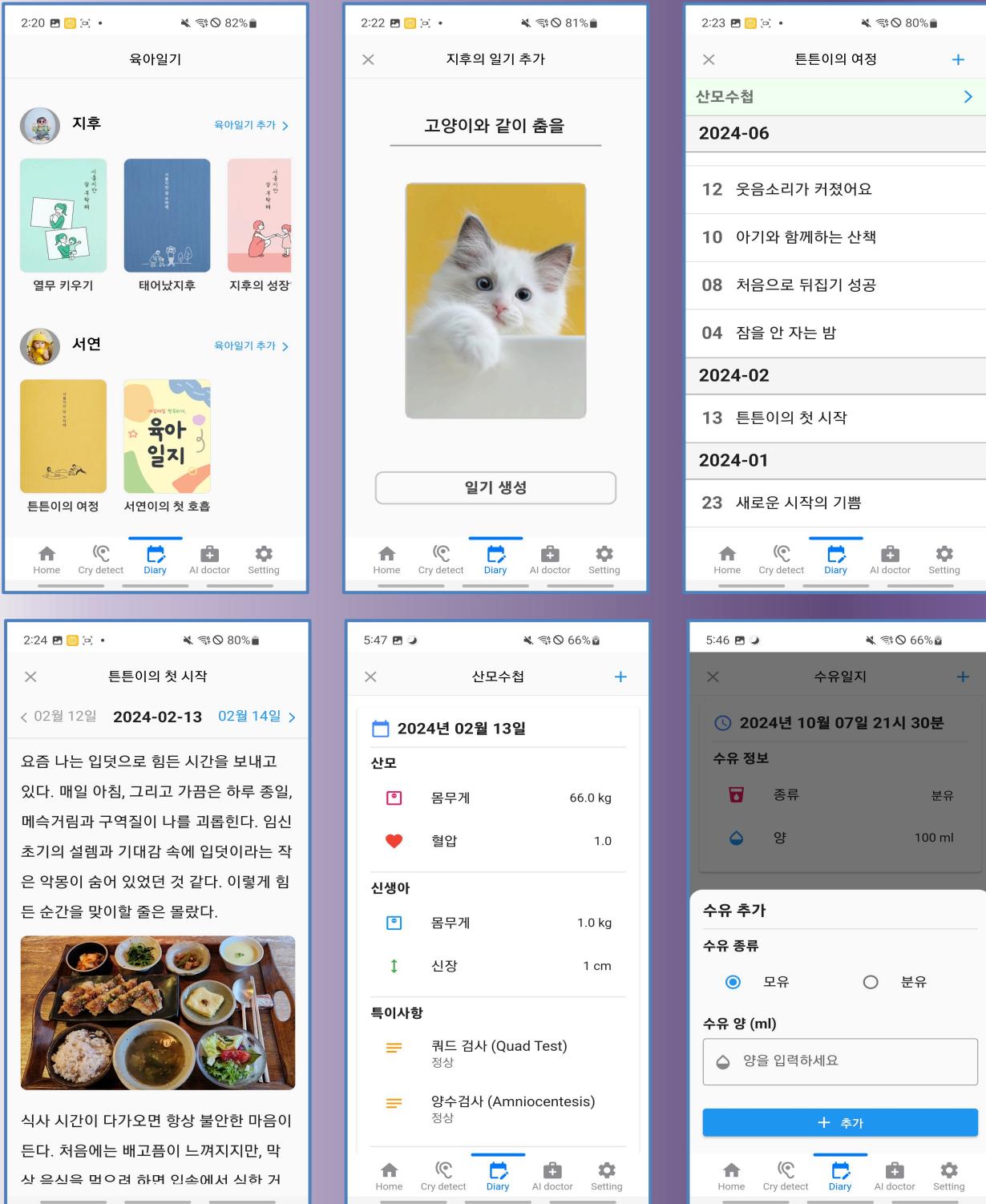
- **YamNet: 울음 감지 AI**
 - 521가지의 일상 음성을 분류하는 on-device AI.
 - MobileNet 아키텍쳐 기반의 가벼운 모델 사용.
- **ResNet fine-tuning: 울음 원인 분류**
 - 음성을 멜 스펙트로그램으로 변환하여 이미지로 취급한 뒤, 이미지 분류를 수행.
 - Sad, Hug, Diaper, Hungry, Sleepy, Awake, Uncomfortable 총 7가지 상태로 분류함.
 - 86%의 검증 성능을 달성함. → 파워 기반 특징 추출 전처리 수행.

울음 감지, 분석 프로세스



지속적인 앱 사용 유도 산모수첩 & 육아일기

- 목적:** 산모수첩으로 지속적인 앱 사용을 유도한다.
- 기능 페이지 개수:** 10개
- Backend API Endpoint 개수:** 29개
- 주요 기능**
 - 육아일기를 작성, 확인할 수 있다.
 - 산모수첩과 수유일지를 작성할 수 있다.
 - 병원 진단 기록과 수유 기록을 확인한다.
 - 다음 병원 검진 날짜를 확인한다.



앱 사용 각인 효과

AI 의사와 질의 응답

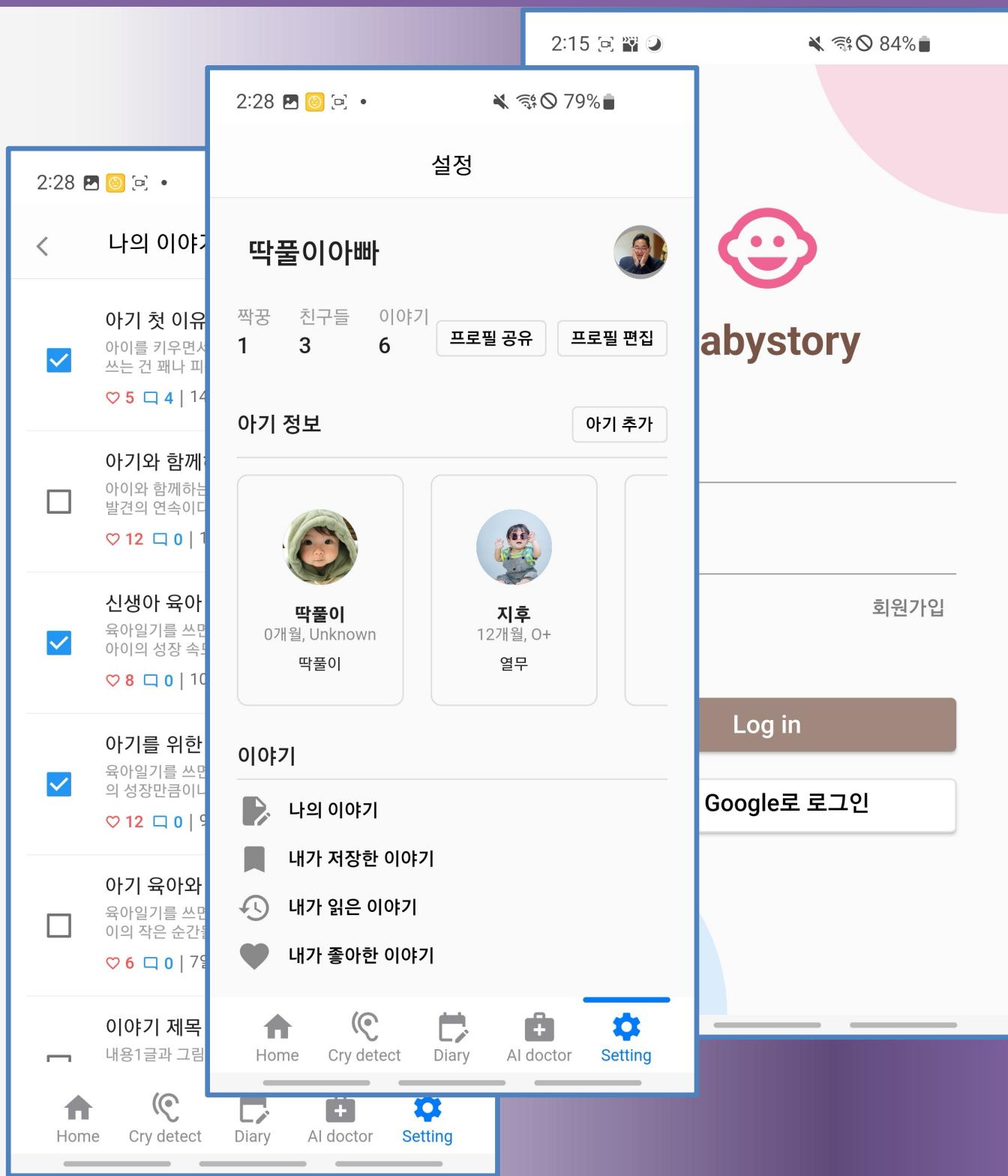
- 목적:** AI의사를 통해 긴급한 상황에 앱을 각인시킨다.
- 기능 페이지 개수:** 3개
- Backend API Endpoint 개수:** 4개
- 주요 기능**
 - AI의사에게 아기의 증상을 상담받는다.
 - Kakao Map API를 통해 가까운 병원으로 안내한다.
 - 상담 기록을 확인할 수 있다.
- RAG를 통한 사실 검증**
 - 아이사랑 기관의 전문가 질의 응답 데이터셋 사용.
 - 사용자의 질문과 유사한 질문을 답변을 참조.



존중받는 개인정보

로그인 & 설정페이지

- 목적:** 유저의 기본적인 CRUD를 수행한다.
- 기능 페이지 개수:** 11개
- Backend API Endpoint 개수:** 25개
- 주요 기능**
 - 이메일, 소셜 로그인/회원가입을 할 수 있다.
 - 사용자와 아기의 개인정보를 수정할 수 있다.
 - 자신과 상호작용한 이야기를 관리할 수 있다.
 - 자신과의 친구, 짹꿍을 관리할 수 있다.
 - 로그 아웃 할 수 있다.



체계적인 개발 진행

TDD 개발 방법론과 MVC 패턴 활용

Request

```
curl --location --request POST  
--header 'Content-Type: application/json'  
--data-raw '{  
    "parent_id": "string",  
    "password": "string",  
    "email": "string",  
    "nickname": "string",  
    "signInMethod": "string",  
    "emailVerified": 1,  
}'
```

Response

```
{  
    "parent": {  
        "parent_id": "string",  
        "password": "string",  
        "email": "string",  
        "nickname": "string",  
        "sign_in_method": "string",  
        "access_token": "string",  
        "refresh_token": "string",  
        "x_jwt": "string"  
    }  
}
```

```
def test_create_parent(client,  
                      response = client.post(  
                        "/parent",  
                        json=test_create_data  
                      )  
  
    assert response.status_code == 201  
    response_json = response.json()  
    assert "parent" in response_json  
  
    assert response_json["parent"]["parent_id"] == test_create_data["parent_id"]  
    assert bcrypt.checkpw(test_create_data["password"], response_json["parent"]["password"])  
    assert response_json["parent"]["email"] == test_create_data["email"]  
    assert response_json["parent"]["nickname"] == test_create_data["nickname"]  
    assert response_json["parent"]["sign_in_method"] == test_create_data["sign_in_method"]  
  
    assert "x-jwt" in response_json  
    assert "access_token" in response_json
```

```
class Parent(BaseModel):  
    parent_id: str  
    password: str  
    email: str  
    name: str  
    nickname: str  
    gender: str  
    sign_in_method: str  
    email_verified: bool  
    photo_id: str  
    description: str  
    main_address: str  
    sub_address: str  
    hash_list: str  
    class Config:  
        orm_mode = True
```

```
@router.post("/")  
def create_parent(createParentInput:  
    ...  
    부모 생성  
    --input  
    - createParentInput.parent_id: 부모 아이디  
    - createParentInput.email: 이메일  
    - createParentInput.password: 비밀번호  
    - createParentInput.nickname: 닉네임  
    - createParentInput.signInMethod: 로그인 방식  
    - createParentInput.emailVerified: 이메일 인증 여부  
    --output  
    - parent: 부모 정보  
    - x-jwt: JWT 토큰  
    ...  
    try:  
        parent = Parent.create(**createParentInput.dict())  
        if parent:  
            raise CustomException("parent already exists")  
    except CustomException:  
        db.rollback()
```

```
class ParentService:  
    # 부모 생성  
    def createParent(self, createParentInput: CreateParentInput):  
        ...  
        부모 생성  
        --input  
        - createParentInput.parent_id: 부모 아이디  
        - createParentInput.email: 이메일  
        - createParentInput.password: 비밀번호  
        - createParentInput.nickname: 닉네임  
        - createParentInput.signInMethod: 로그인 방식  
        - createParentInput.emailVerified: 이메일 인증 여부  
        --output  
        - parent: 부모 정보  
        ...  
        db = get_db_session()  
  
        # 부모 아이디 중복 확인  
        error = db.query(ParentTable).filter(  
            ParentTable.parent_id == createParentInput.parent_id).first()  
  
        if error:  
            raise CustomException("parent_id already exists")  
  
        # 패스워드 암호화  
        if createParentInput.signInMethod == 'email':  
            createParentInput.password = bcrypt.hashpw(  
                createParentInput.password.encode('utf-8'),  
                bcrypt.gensalt())  
  
        # 이메일 중복 확인
```

API 명세서 작성

→ 테스트 코드 작성

→ MVC 패턴에 따라 코딩