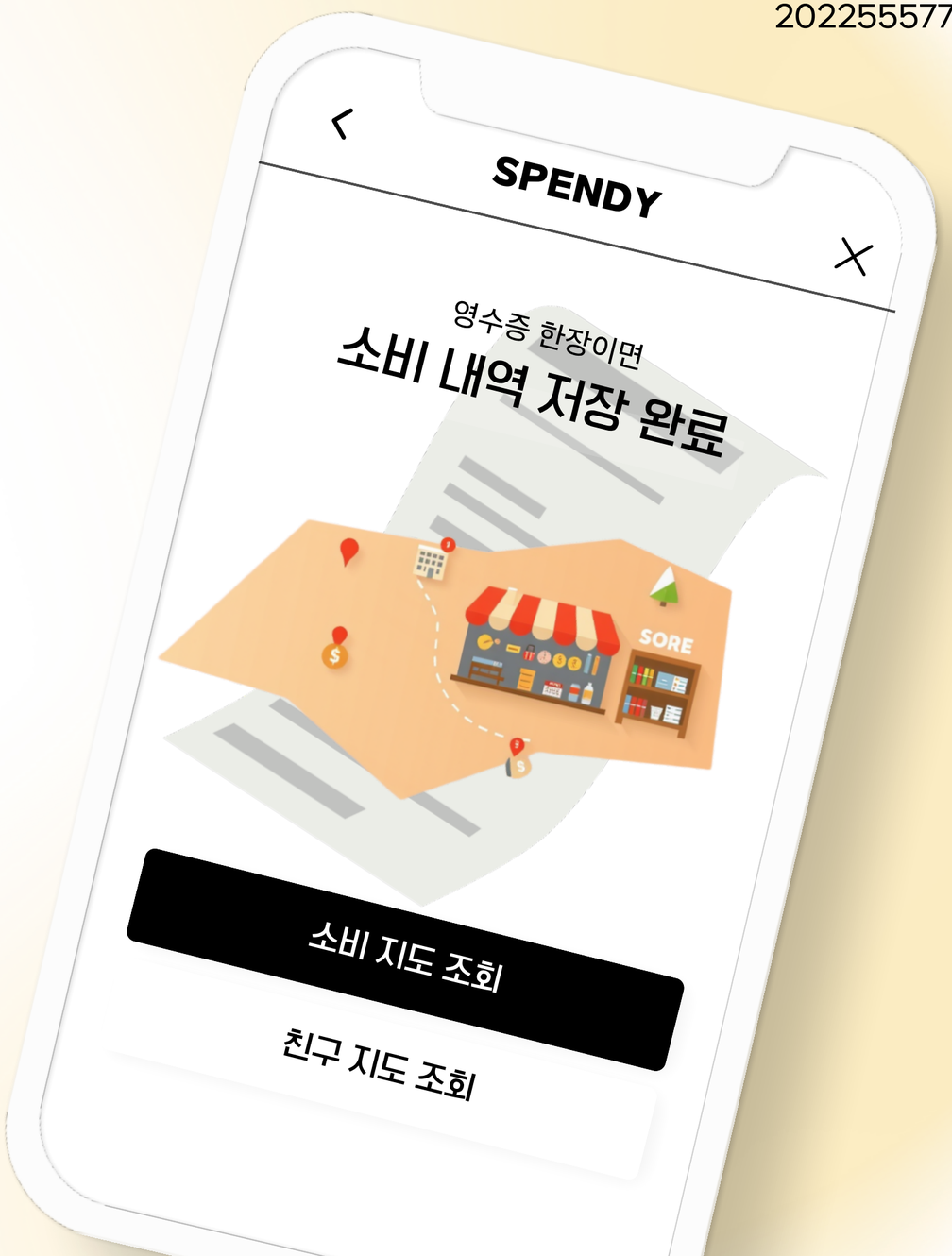
202155594 이학진
202255577 윤혜진

어디서, 얼마나, 누구와? - 소비 관리 가계부

SPENDY



01. 득점 기능

JETPACK

```
// 네비게이션 설정
val navView: BottomNavigationView = findViewById(R.id.bottom_navigation)
val navHostFragment = supportFragmentManager.findFragmentById(R.id.nav_host_fragment) as androidx.navigation.fragment.NavHostFragment
val navController = navHostFragment.navController
navView.setupWithNavController(navController)
```

MainActivity.kt / NavHostFragment, NavController, setupWithNavController

```
private fun setupUI() {
    binding.rvRecentTransactions.layoutManager = LinearLayoutManager(context)
    receiptAdapter = ReceiptAdapter(emptyList()) { docId ->
```

HomeFragment.kt / RecyclerView, LinearLayoutManager

```
override fun onCreateView(
    inflater: LayoutInflater,
    container: ViewGroup?,
    savedInstanceState: Bundle?
): View {
    _binding = FragmentHomeBinding.inflate(inflater, container, false)
    return binding.root
}
```

HomeFragment.kt / Fragment

Navigation, Fragment, RecyclerView, ViewBinding, Layout 등 Jetpack 라이브러리를 실제 코드에서 적극적으로 사용

01. 득점 기능

API 연동

```
private fun runClovaOcrRest(bitmap: Bitmap, onResult: (String) -> Unit) {
    val imageBase64 = bitmapToBase64(bitmap)
    val json = org.json.JSONObject().apply {
        put("version", "V2")
        put("requestId", java.util.UUID.randomUUID().toString())
        put("timestamp", System.currentTimeMillis())
        put("images", org.json.JSONArray().apply {
            put(org.json.JSONObject().apply {
                put("name", "sample_image")
                put("format", "png")
                put("data", imageBase64)
            })
        })
    }
```

HomeFragment.kt / NAVER CLOVA OCR API

runClovaOcrRest 함수로
이미지 → Base64 → **네이버 OCR API** POST → JSON 응답 파싱

파싱한 주소 값을 **GEOCODING API**를 사용하여 정확한 위도, 경도로 변환

위도, 경도 값을 **NAVER MAP API**를 사용하여 지도에 마킹

```
import com.naver.maps.map.MapView
import com.naver.maps.map.NaverMap
import com.naver.maps.map.NaverMapSdk
import com.naver.maps.map.OnMapReadyCallback
import com.naver.maps.map.overlay.Marker
import com.naver.maps.geometry.LatLng
import com.naver.maps.map.CameraUpdate
```

MapFragment.kt / Naver Map API

```
private suspend fun getLatLngFromAddress(address: String): Pair<Double, Double> {
    return try {
        val apiKey = "KakaoAK a43aa2abb1d988b7e8c15c509e791f3a"
        val url = "https://dapi.kakao.com/v2/local/search/address.json?query=" +
            java.net.URLEncoder.encode(address, "UTF-8")
        val request = okhttp3.Request.Builder()
            .url(url)
            .addHeader("Authorization", apiKey)
            .build()
        val client = okhttp3.OkHttpClient()
        val response = client.newCall(request).execute()
        val body = response.body?.string() ?: return 0.0 to 0.0

        android.util.Log.d("KakaoGeocoding", "주소: $address\n요청 URL: $url\n응답: $body")

        val json = org.json.JSONObject(body)
        val documents = json.optJSONArray("documents")
        if (documents != null && documents.length() > 0) {
            val doc = documents.getJSONObject(0)
            val lat = doc.optString("y", "0.0").toDoubleOrNull() ?: 0.0
            val lng = doc.optString("x", "0.0").toDoubleOrNull() ?: 0.0
            android.util.Log.d("KakaoGeocoding", "파싱된 위도: $lat, 경도: $lng")
            return lat to lng
```

HomeFragment.kt / GEOCODING API

01. 득점 기능

APP 연동 & DB & 코루틴

```
private val galleryLauncher = registerForActivityResult(
    ActivityResultContracts.GetContent()
) { uri ->
    uri?.let { processImage(it) }
}
```

HomeFragment.kt / 갤러리 연동
사용자가 이미지를 선택하면, processImage(uri)로 이미지가 전달

```
private fun checkContactsPermissionAndLoad() {
    if (ContextCompat.checkSelfPermission(requireContext(), Manifest.permission.READ_CONTACTS) != PackageManager.PERMISSION_GRANTED) {
        requestPermissions(arrayOf(Manifest.permission.READ_CONTACTS), 1002)
    } else {
        loadContacts()
    }
}

private fun loadContacts() {
    contactList.clear()
    val cursor = requireContext().contentResolver.query(
        ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
        arrayOf(
            ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME,
            ContactsContract.CommonDataKinds.Phone.NUMBER
        ),
```

FriendsFragment.kt / 연락처 연동

```
private fun saveReceiptToFirestore(receipt: ReceiptInfo) {
    val db = FirebaseFirestore.getInstance()
    db.collection("receipts")
        .add(receipt)
        .addOnSuccessListener {
            // 저장 성공 시 최근 내역 새로고침
            loadRecentTransactions()
        }
        .addOnFailureListener { e ->
            Toast.makeText(context, "Firestore 저장 실패: ${e.localizedMessage}", Toast.LENGTH_SHORT).show()
        }
}
```

HomeFragment.kt / Firebase에 db 저장

```
// 주소 → 좌표 변환 (코루틴 launch)
if (parsedAddress.isNotBlank()) {
    GlobalScope.launch(Dispatchers.Main) {
        try {
            // 실제 Geocoding API 사용 (정확한 위치)
            val (lat, lng) = kotlinx.coroutines.withContext(Dispatchers.IO) {
                getLatLngFromAddress(parsedAddress)
            }
```

HomeFragment.kt / 코루틴 적용

이외에도 많은 기능에서 코루틴을 적극 사용

01. 득점 기능

머신러닝

1. 문제 정의
   목표: OCR로 추출한 소비 품목명을 입력으로 받아,
   적절한 소비 카테고리로 자동 분류

2. 데이터 구축
   - 카테고리 수: 9개 (식품, 음료, 생활용품, ...)
   - 총 데이터 수: 20,000개
   - 각 품목명에 대해 15개 템플릿 문장 자동 생성
   - 다양성 확보: 품목명, 문장 스타일, 표현 방식 다양화

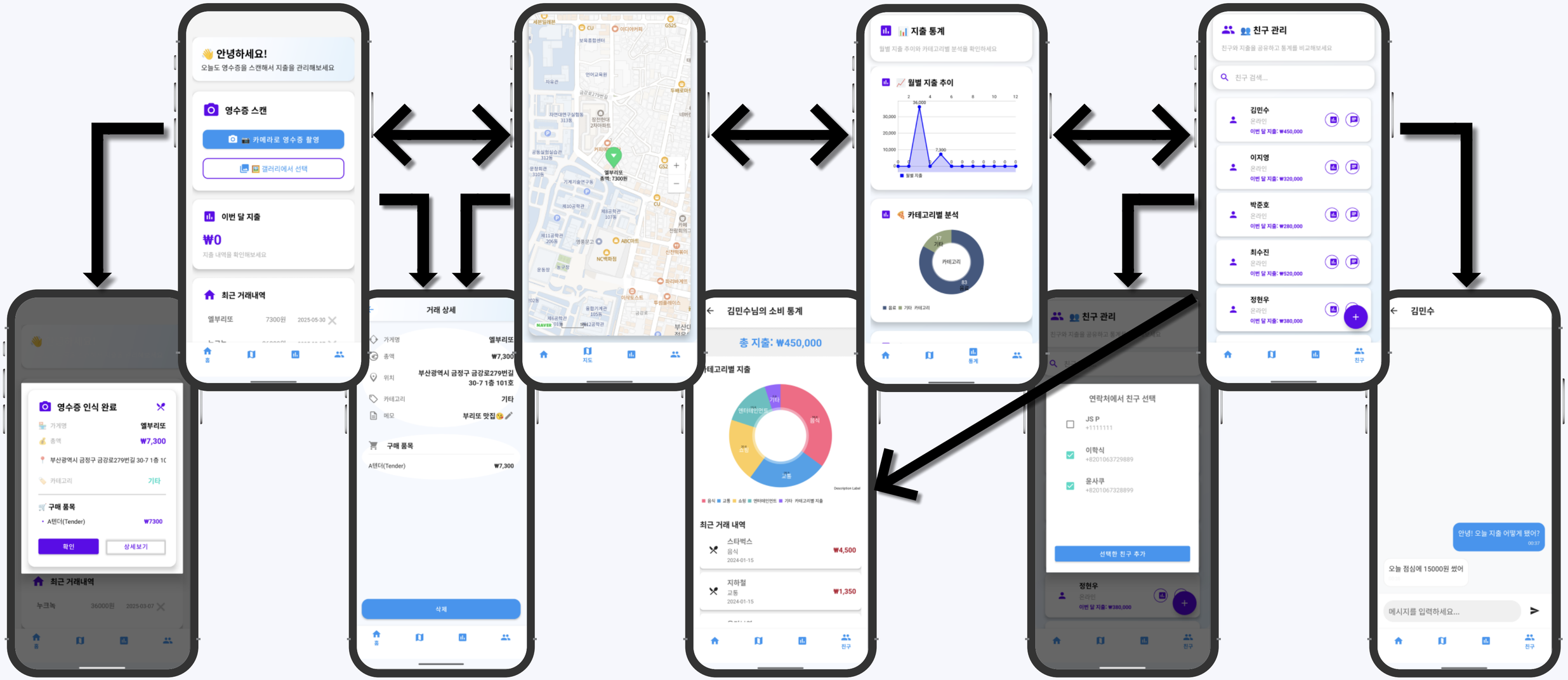
3. 모델 학습
   - 모델: beomi/KcELECTRA-base (한국어 BERT 계열 사전학습 모델)
   - Accuracy: 약 65–75% 수준
   - TFLite 변환으로 모바일 탑재 가능
   - 모델 경량화
   - float16, int8 버전으로 모델 크기 최소화 (~20MB까지)

category	count
기타	603
외식	585
식품	573
명품	558
음료	553
전자기기	547
생활용품	532
의류	525
뷰티	524

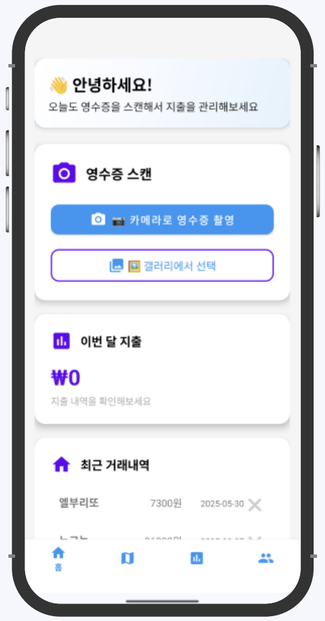
```
// 백그라운드에서 안전하게 추론
return runBlocking(Dispatchers.IO) {
    try {
        android.util.Log.d("TFLite", "추론 시작 - 입력 텍스트: $text")

        // KCELECTRA 토크나이저 사용
        val tokenIds = tokenizeForKCELECTRA(text)
        android.util.Log.d("TFLite", "토크나이징 완료 - 토큰 수: ${tokenIds.size}")

        val inputArray = Array(1) { tokenIds }
        val output = Array(1) { FloatArray(LABELS.size) }

        android.util.Log.d("TFLite", "입력 배열 형태: ${inputArray.size}x${inputArray[0].size}")
        android.util.Log.d("TFLite", "출력 배열 형태: ${output.size}x${output[0].size}")

        android.util.Log.d("TFLite", "추론 실행 시작")
        // 추론 실행
        tfliteInterpreter?.run(inputArray, output)
        android.util.Log.d("TFLite", "추론 실행 완료")

        // 출력 크기 확인
        android.util.Log.d("TFLite", "실제 출력 크기: ${output[0].size}")

        val maxIdx = output[0].indices.maxByOrNull { output[0][it] } ?: 0
        val category = LABELS[maxIdx]

        android.util.Log.d("TFLite", "KCELECTRA 분류 결과: $category (인덱스: $maxIdx, 신뢰도: ${output[0][maxIdx]})")
        category
```

02. 상세 기능

와이어프레임



홈



영수증 스캔

카메라로 직접 촬영하거나 갤러리에서 선택

이번 달 지출

해당 월의 총 지출을 한 눈에 확인

최근 거래내역

최근 소비를 가게명, 금액, 날짜만 간단하게 표시

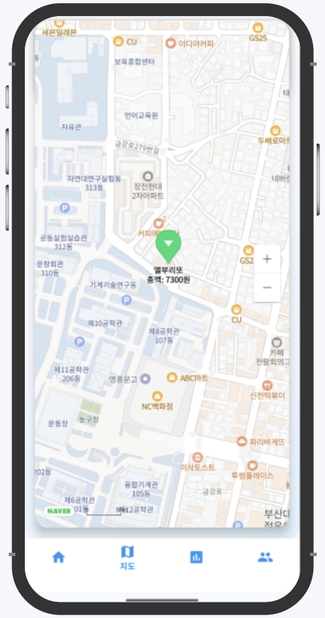
거래내역을 클릭하면 상세 거래 내용으로 이동

지도

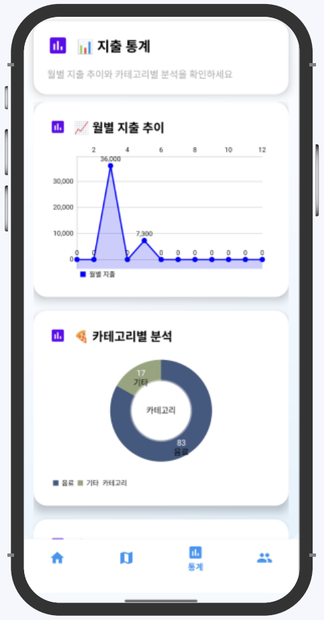
지도에 마커 표시

소비 장소를 가게명, 금액과 함께 간단히 지도에 표시

마커를 클릭하면 상세 거래 내용으로 이동



지출 통계



월별 지출 추이

그래프를 통해 한 눈에 월별 지출 추이 파악

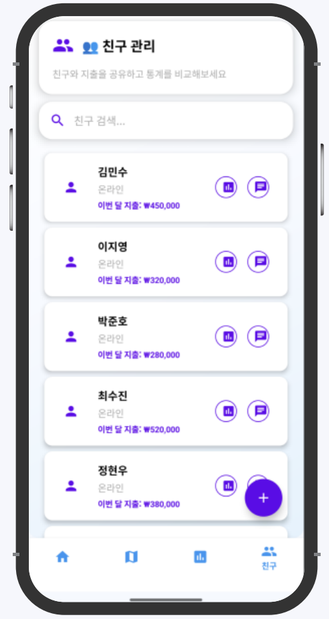
카테고리별 분석

카테고리별 비율 제공

카테고리별 상세

카테고리별로 지출한 금액 표시

친구 관리



친구 목록

친구 목록을 통해 소비 내용 공유

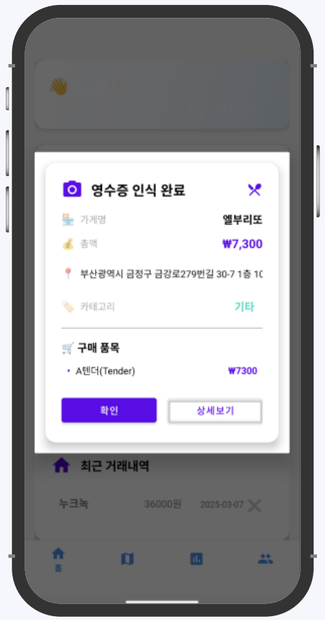
친구 추가

주소록과 연동하여 친구 추가 가능

다양한 소셜 기능

친구와의 소비 통계 공유 및 채팅 기능

영수증 인식



영수증 정보 제공

영수증 인식 성공시 가게명, 총액, 주소, 카테고리, 구매 품목 정보 제공

거래 내용 상세 페이지

상세한 거래 내용 제공

가게명, 총액, 주소, 카테고리, 구매 품목 정보 제공

구매 품목 별 가격

구매한 품목 별로 상세한 가격 제공

메모 기능

소비별로 간단한 메모 작성 기능

친구의 소비 통계

카테고리별 지출

친구의 카테고리별 지출 비율 열람 가능

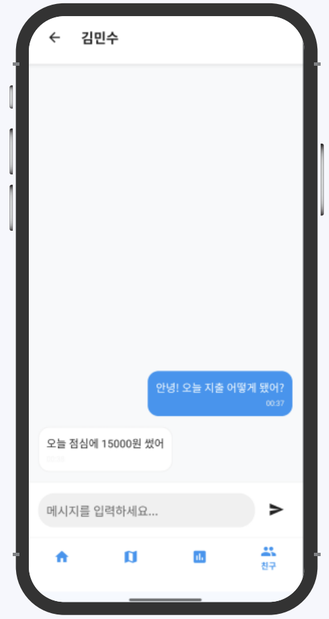
최근 거래 내역

친구의 최근 거래 내역을 간단히 열람 가능

친구와의 채팅

채팅 기능

친구와의 채탕기능



친구 추가

친구 추가 기능

주소록 앱과 연동하여 친구추가 가능

친구는 한 번에 원하는 수 만큼 선택하여 추가하도록 함

03.

역할분담

이학진

지도 API 및 차트 구현
Firebase Auth 및 친구 기능 구현
OCR 연동 및 텍스트 추출 설계

윤혜진

머신러닝 모델 학습 및 적용
Firebase 구조 설계 및 파싱 로직
UI/UX 디자인

THANK YOU