

대학 졸업 작품 전시 후, 해당 작품을 판매하는 쇼핑몰

UniPiece 유니피스



| 6팀 멋쟁이 사람들

은정민 허성욱 강예인 주성원 엄민식

목차



01

프로젝트
개요

02

아키텍처

03

사용 기술

04

시연 영상

05

트러블 슈팅

06

결과

07

회고

유니피스 UniPiece

University + Piece

대학 졸업 전시를 마친 작품과
학생들의 개인 작품을 판매하고
새로운 예비 작가들을 발굴 및 후원하는 쇼핑몰

전시 이후 작품들은 어디로 갈까?



교내 보관



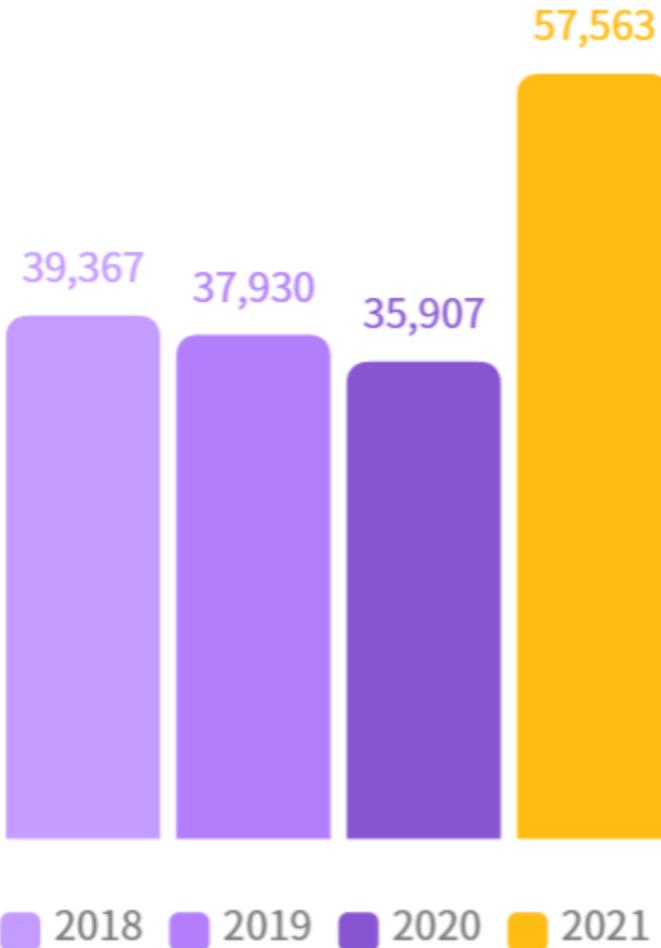
개인 보관



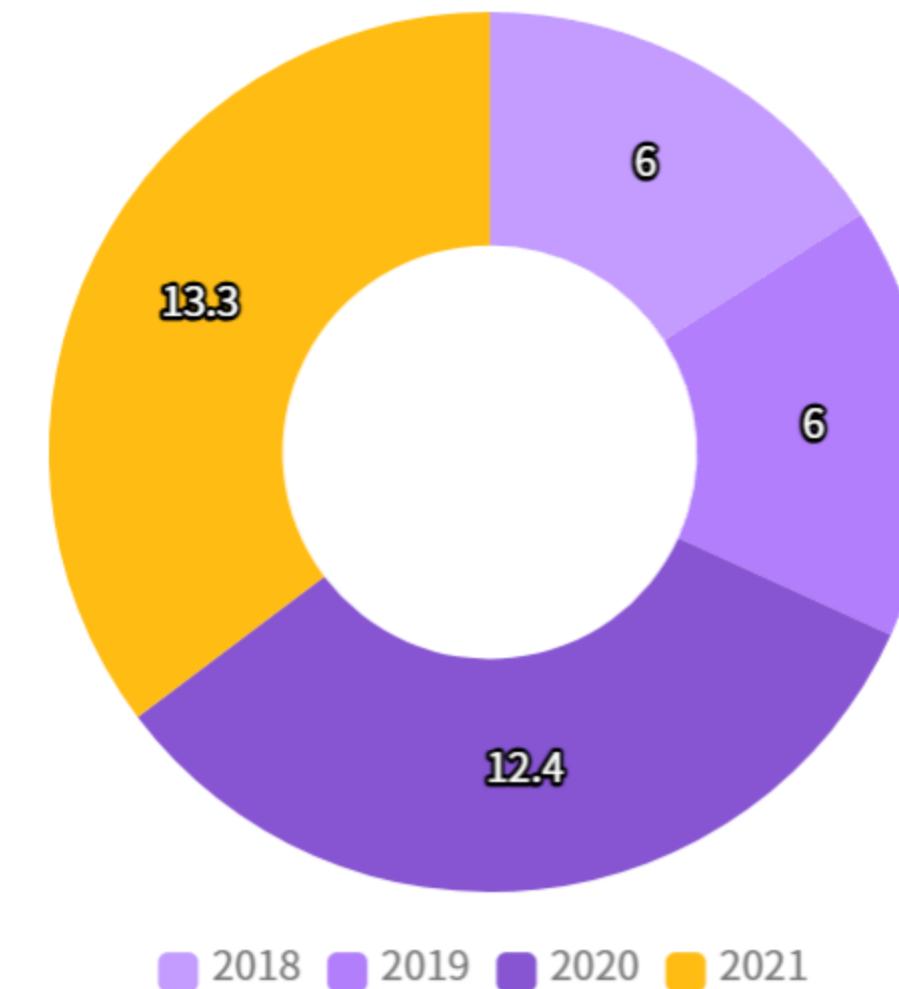
폐기 및 재활용

미술 시장의 확대

거래 작품 수
(단위 : 점)



온라인 미술 시장 규모 추이
(단위 : 십억달러)



출처 | 문화체육관광부 2022 미술시장실태조사

타서비스 분석



모아
(MOOA)

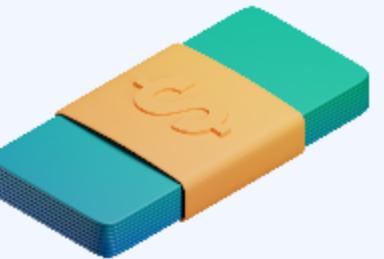
예비 작가인
학생의 작품에 대한
경쟁력이 아쉬움



플리옥션
(Flea Auction)

옵션을 통해서만
미술 작품 판매

기획 의도



작품 노출
기회 제공

- 자신의 작품을 소개하는 기회 제공
- 작가로서의 경력을 시작하는 데 도움
- 작품을 더 폭넓게 알릴 수 있는 발판

경제적 지원

- 작품을 판매하여 자신의 예술 활동을 지속할 수 있는 경제적 지원

작품
보존 및 기록

- 작품을 판매함으로써, 작품들을 보존하고 기록

문화적 가치
제고

- 작품을 판매하여 문화적인 가치를 높이는 데 기여

개발 목표

사용자에게 제시하는 목표



작가

- 작품을 판매함으로써 다양한 사람들에게 작품을 노출시킬 수 있다
- 이를 통해 창작물에 대한 인식과 인기를 확대할 수 있다



구매자

- 작품을 구매함으로써 새로운 예술가를 지원할 수 있다
- 작가가 창작활동을 지속할 수 있도록 도울 수 있다

개발 목표

앱 구현에 필요한 기술적 목표



소셜 로그인 API 적용

- 소셜 로그인으로 편리한 유저 관리



Firebase FireStore
Firebase Storage 사용

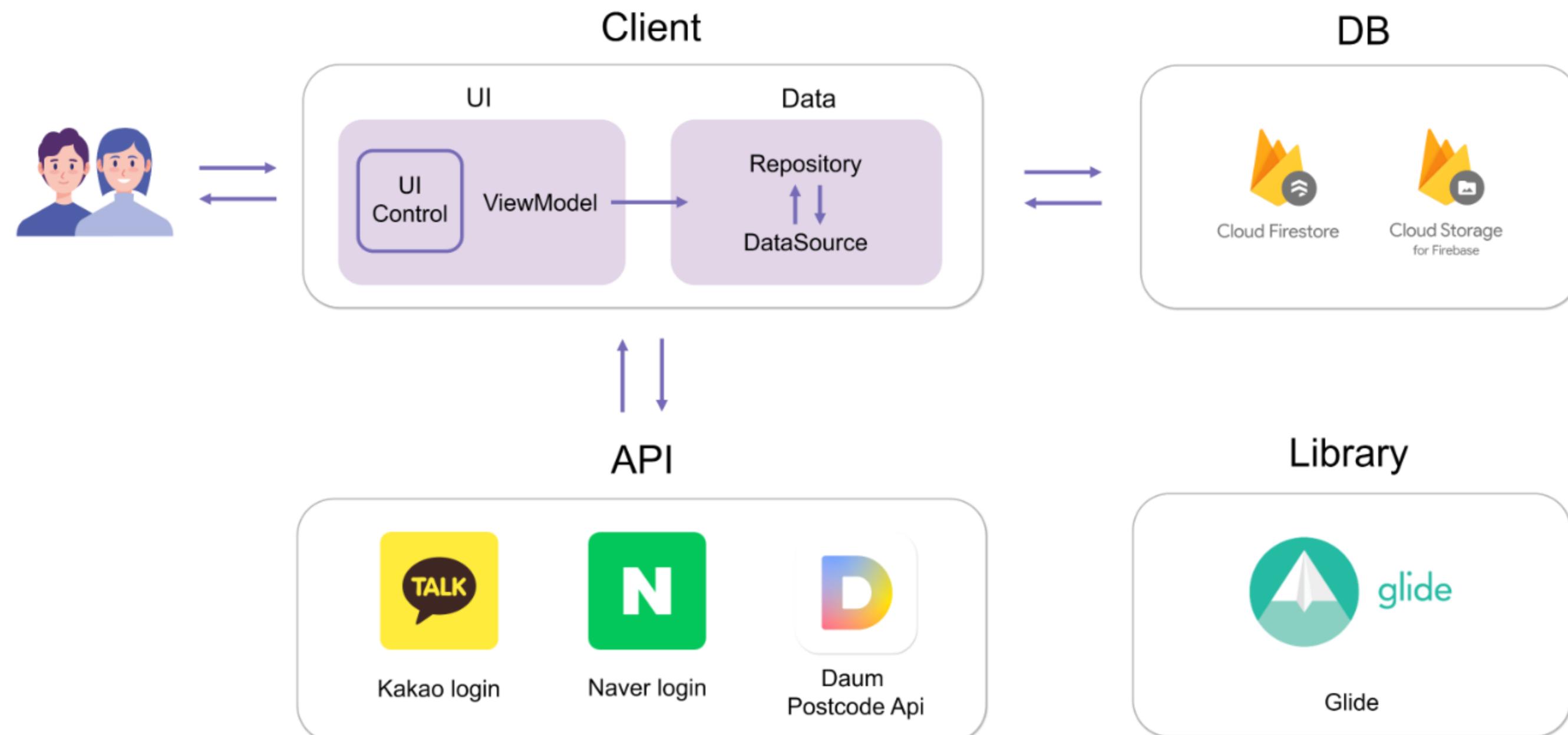
- 작품 리스트 또는 행사 정보와 같은 작품과 관련된 정보를 Firebase FireStore를 통해 보여주고, 검색 기능도 제공
- Firebase Storage를 통해서 관련 사진 제공



MVVM 디자인 패턴 적용

- 뷰 로직과 비지니스 로직을 분리하여 생산성을 높임
- 중복되는 로직을 모듈화 해서 여러 뷰에 적용

아키텍처



회원가입/로그인

자동 로그인

SharedPreferences를 활용한 자동 로그인
앱 재실행에도 빠르게 로그인 가능

01

간편 로그인

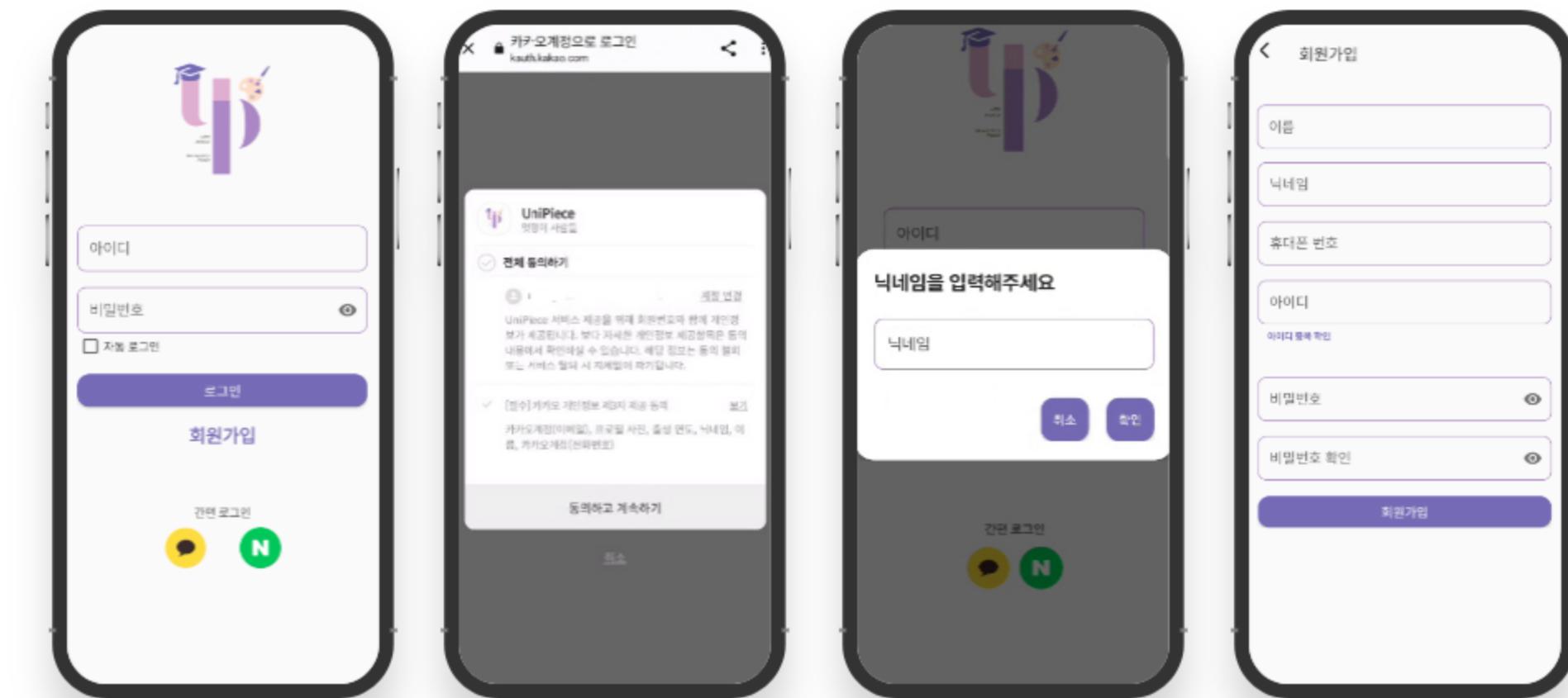
kakao, Naver API 활용으로
간편한 회원가입과 로그인 지원
간편 로그인으로 제공하지 않는 추가 정보는
다이얼로그를 통해 입력을 받고 회원가입

02

회원가입

간편 로그인을 사용하지 않는 경우
FireStore DB을 활용한 직접 회원가입

03



03. 사용 기술 - 앱 기능 소개

홈

4개의 섹션으로 구성

전시 홍보, 소식, 작가, 전시실 작품 섹션 각각
FireStore DB에서 받아온 데이터와
Storage에서 받아온 사진으로
ViewPager2로 콘텐츠 자동넘김 기능까지
사용자 중심의 콘텐츠 위주 홈화면 구성



작품 구매

작품 모아보기

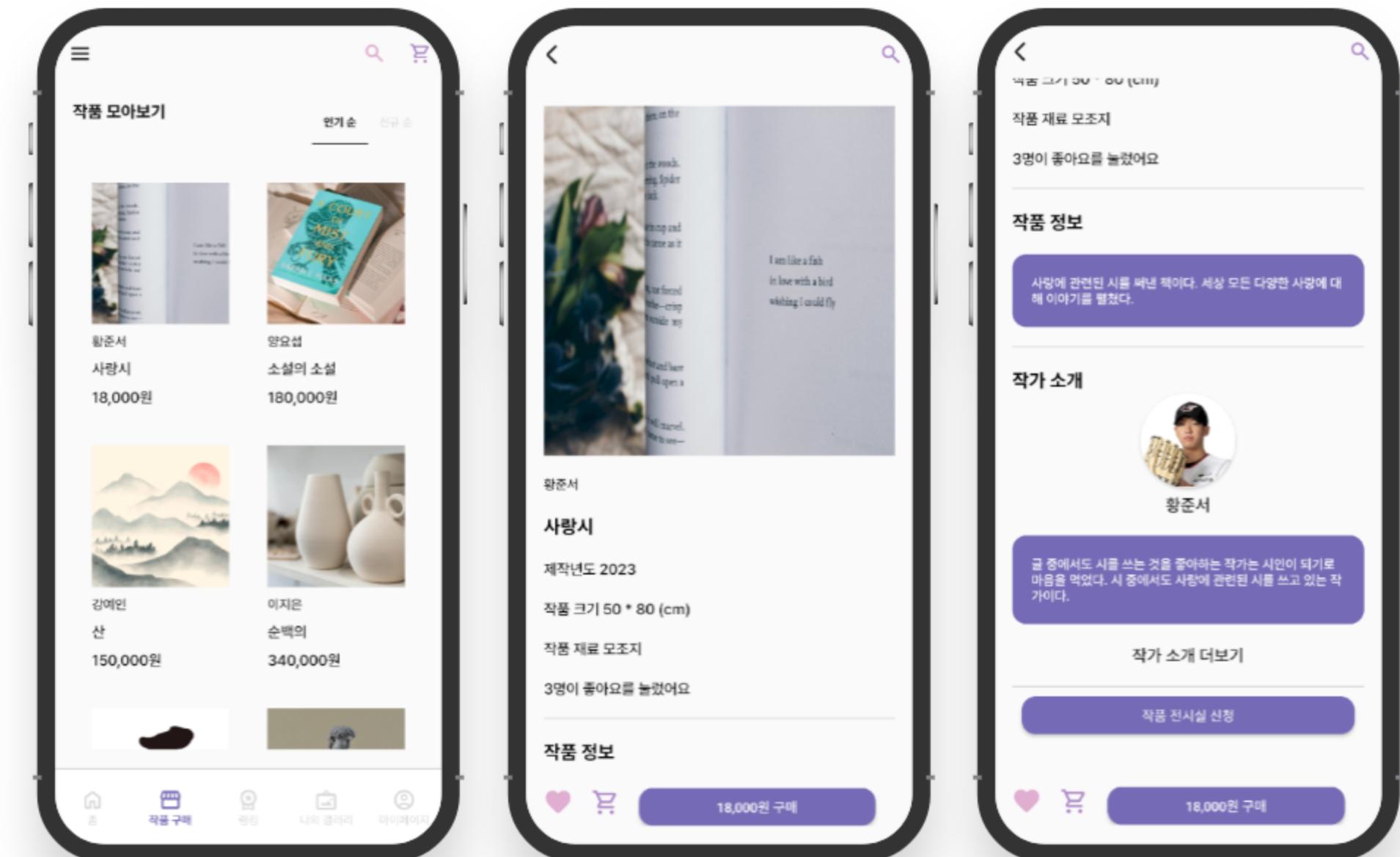
TabLayout과 ViewPager2로 구현
유니피스에 등록된 작가의 판매 작품 표시
FireStore DB의 orderby 쿼리 기능을 활용
인기 순, 신규 순으로 정렬 기능

01

작품 상세 정보

FireStore DB와 Storage 사용
작품 및 작가 정보 소개
작품을 직접 볼 수 있는 전시실 방문 신청
작품 장바구니에 넣기
해당 작품 구매
해당 작품에 대한 좋아요 표시

02



장바구니/주문하기/결제

FireStore DB와 Storage 사용 활용

장바구니

01

전체 선택 / 개별 선택
선택한 작품 제거
주문 단계로 넘어가기

주문하기

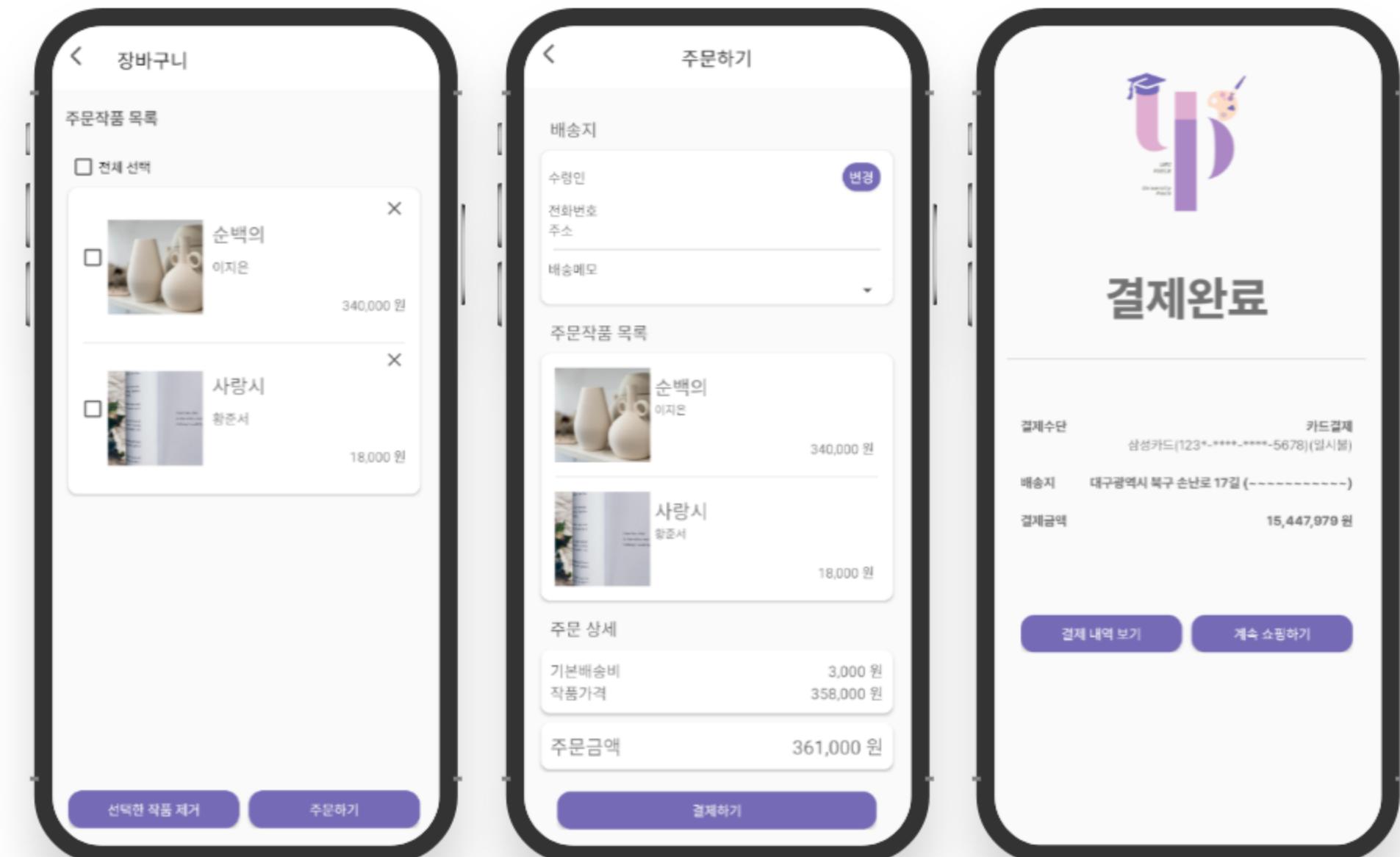
02

배송지 추가/수정/삭제
배송메모 선택 가능
주문 작품 목록 및 상세 내역 표시

결제

03

결제 완료 화면 표시



랭킹

FireStore DB의 orderby 쿼리 기능 활용

작품 랭킹

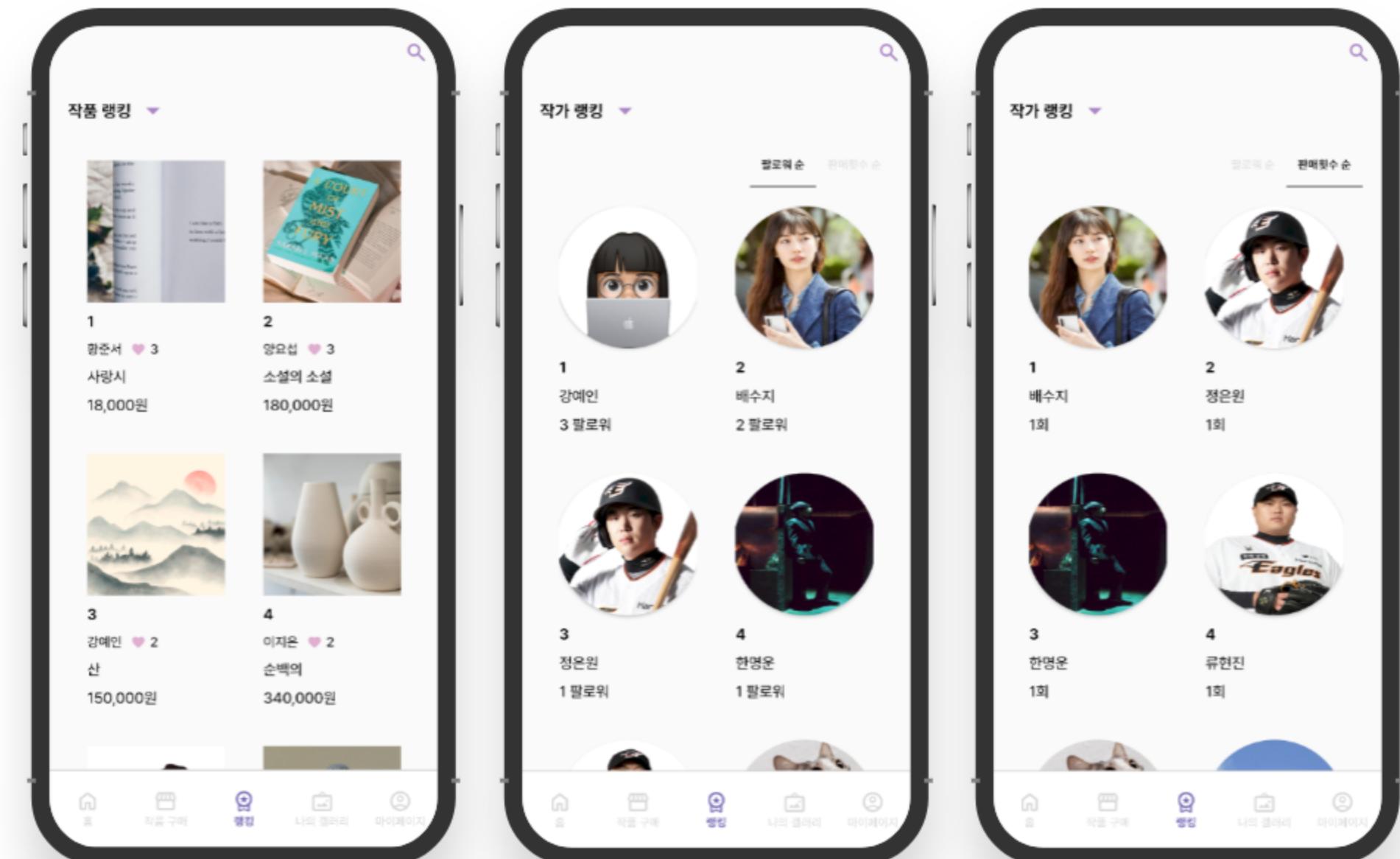
좋아요를 가장 많이 받은 작품순서로
작품 랭킹 구성

01

작가 랭킹

TabLayout과 ViewPager2로 구현
팔로워가 많은 순서
판매횟수가 많은 순서로
작가 랭킹 구성

02



작품 검색

FireStore DB와 Storage 사용 활용
쿼리 실행 후 자체 검색 필터링 기능 추가

검색 기능

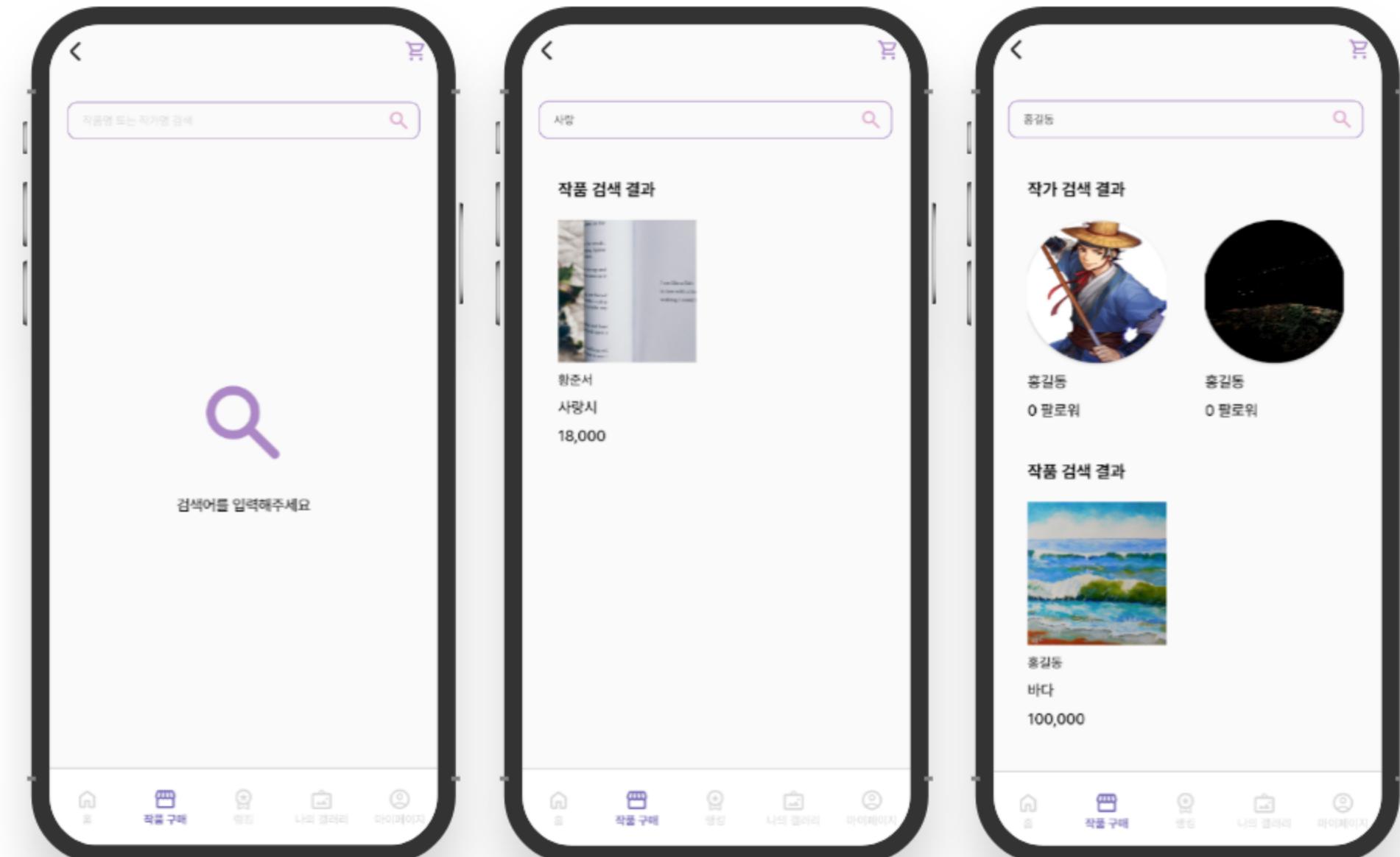
상단 우측의 검색 메뉴를 클릭하여 검색하면
해당 키워드에 맞는 작가, 작품 검색 결과 표시

01

작품, 작가 정보 이동

검색 결과를 클릭하면 해당 작가 또는
해당 작품의 상세 정보 페이지로 이동

02



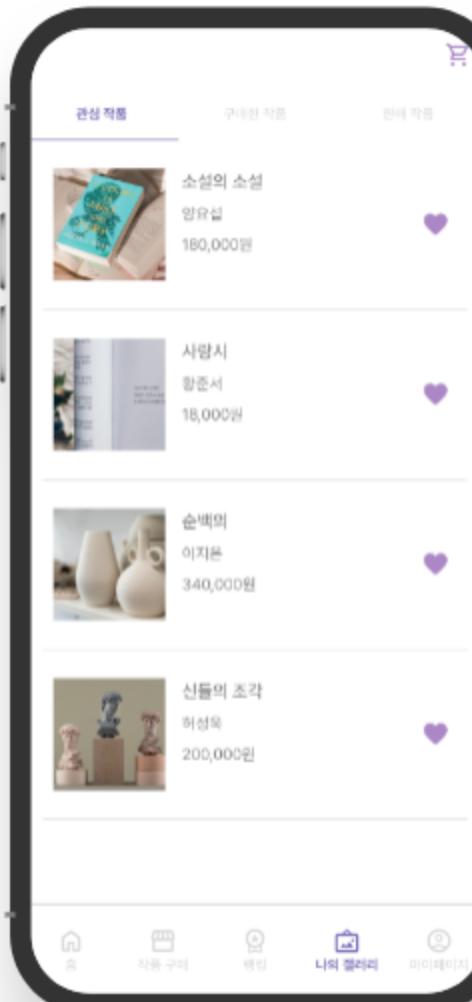
나의 갤러리

FireStore DB와 Storage 사용 활용

관심 작품

좋아요를 누른 작품 목록 표시
하트 버튼을 누르면 '좋아요' 취소 가능
해당 작품 클릭 시 작품 상세 정보로 이동

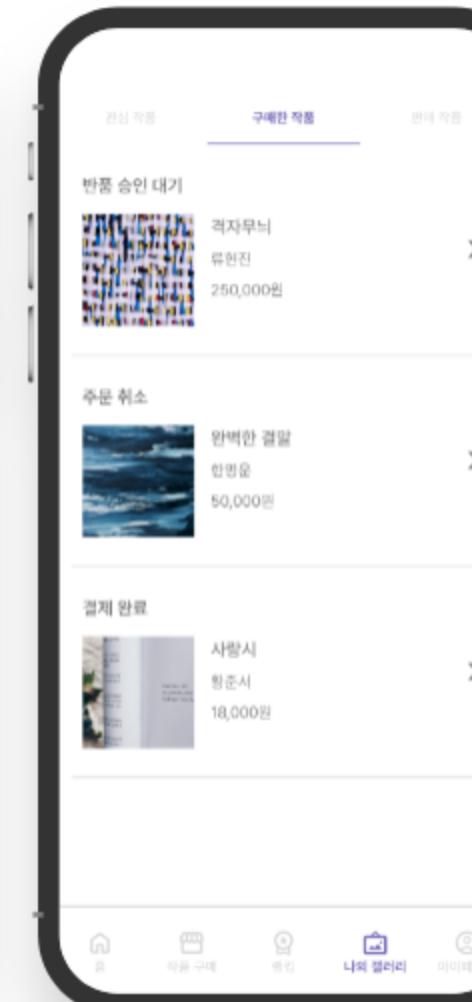
01



구매한 작품

구매한 작품 목록 표시
작품 클릭 시 상세 구매 내역으로 이동

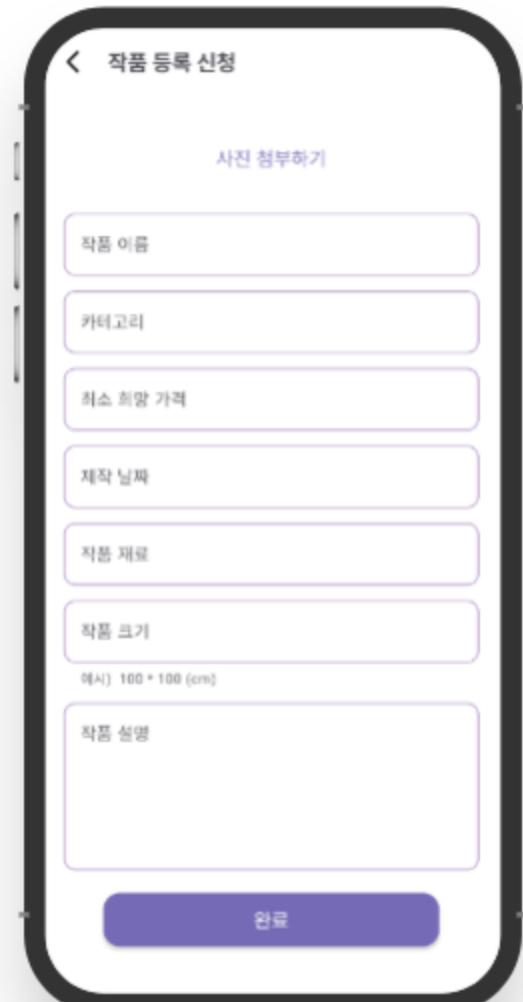
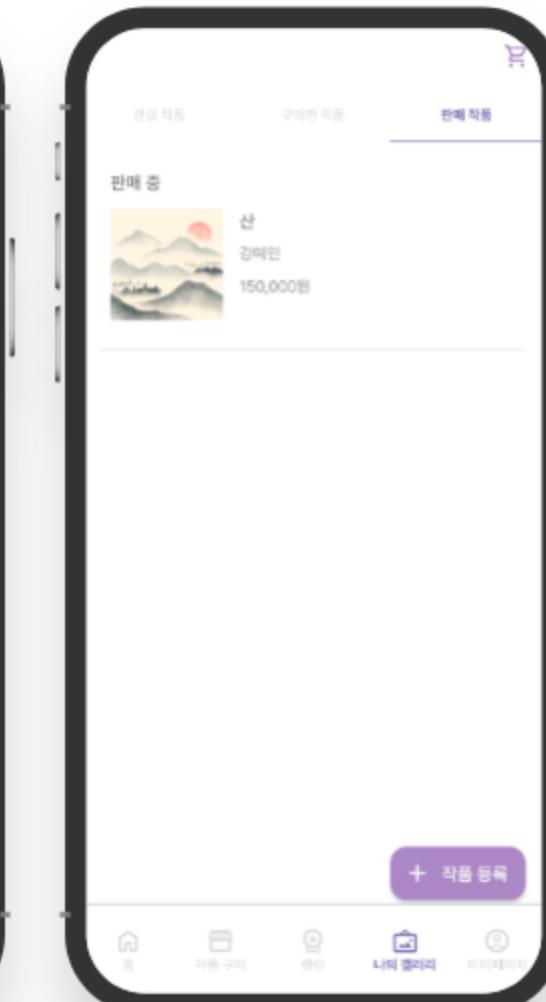
02



판매 작품

사용자가 작가로 등록된 경우
판매중인 작품 목록 표시 및 작품 등록
사용자가 일반회원인 경우
작가 등록 버튼 표시

03



마이페이지 - 회원 정보

FireStore DB와 Storage 사용 활용

회원 정보

회원 가입 시 등록한 회원정보 표시

01

회원 정보 수정 / 회원 탈퇴

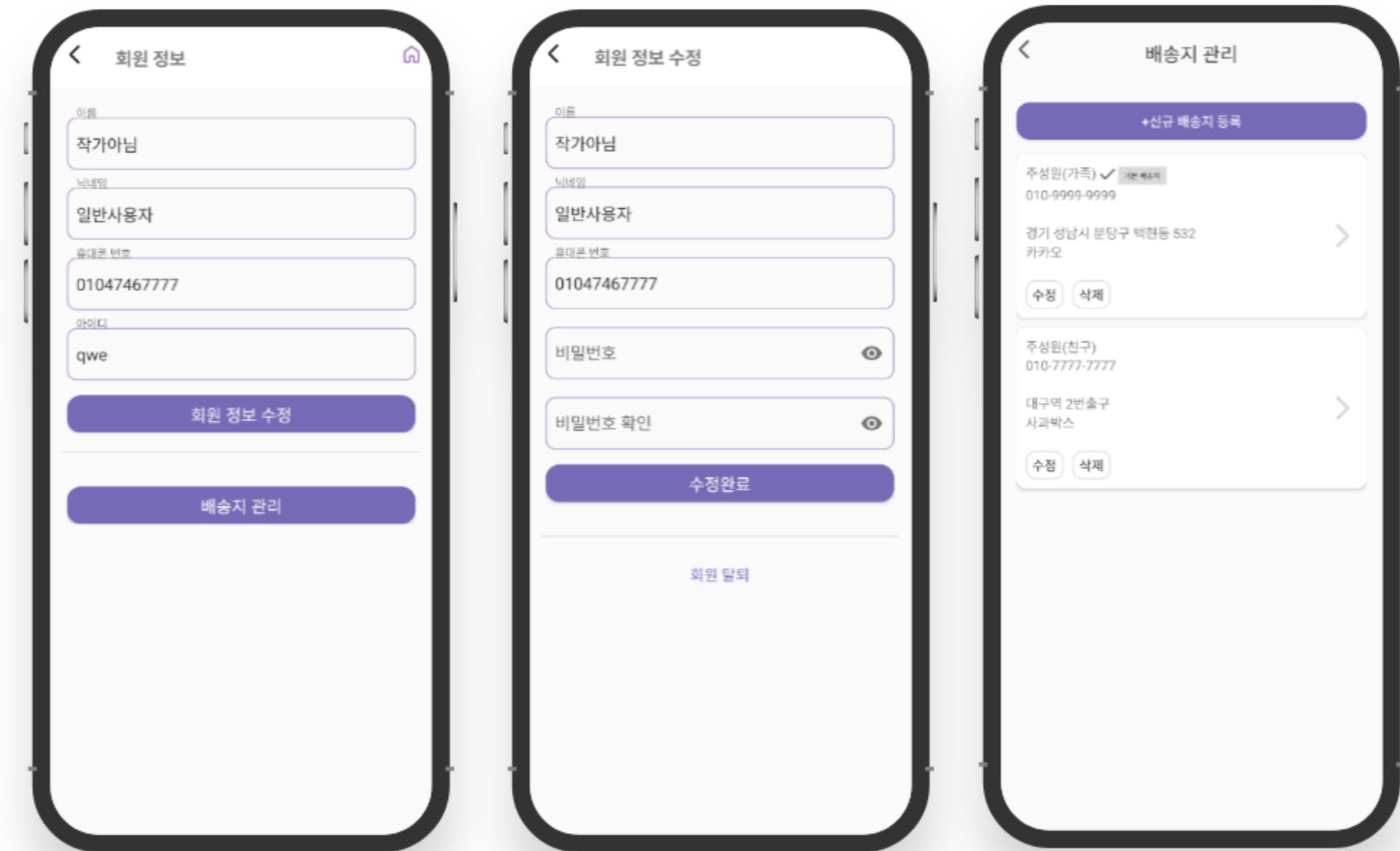
회원 정보 수정 및 회원 탈퇴 기능
회원 탈퇴 시 로그아웃 후
로그인 화면으로 이동

02

배송지 관리

작품 주문, 결제 시 등록할 배송지를
관리할 수 있는 화면으로 이동

03



마이페이지 - 전시실 방문 신청 목록

FireStore DB와 Storage 사용 활용

신청 내역 표시

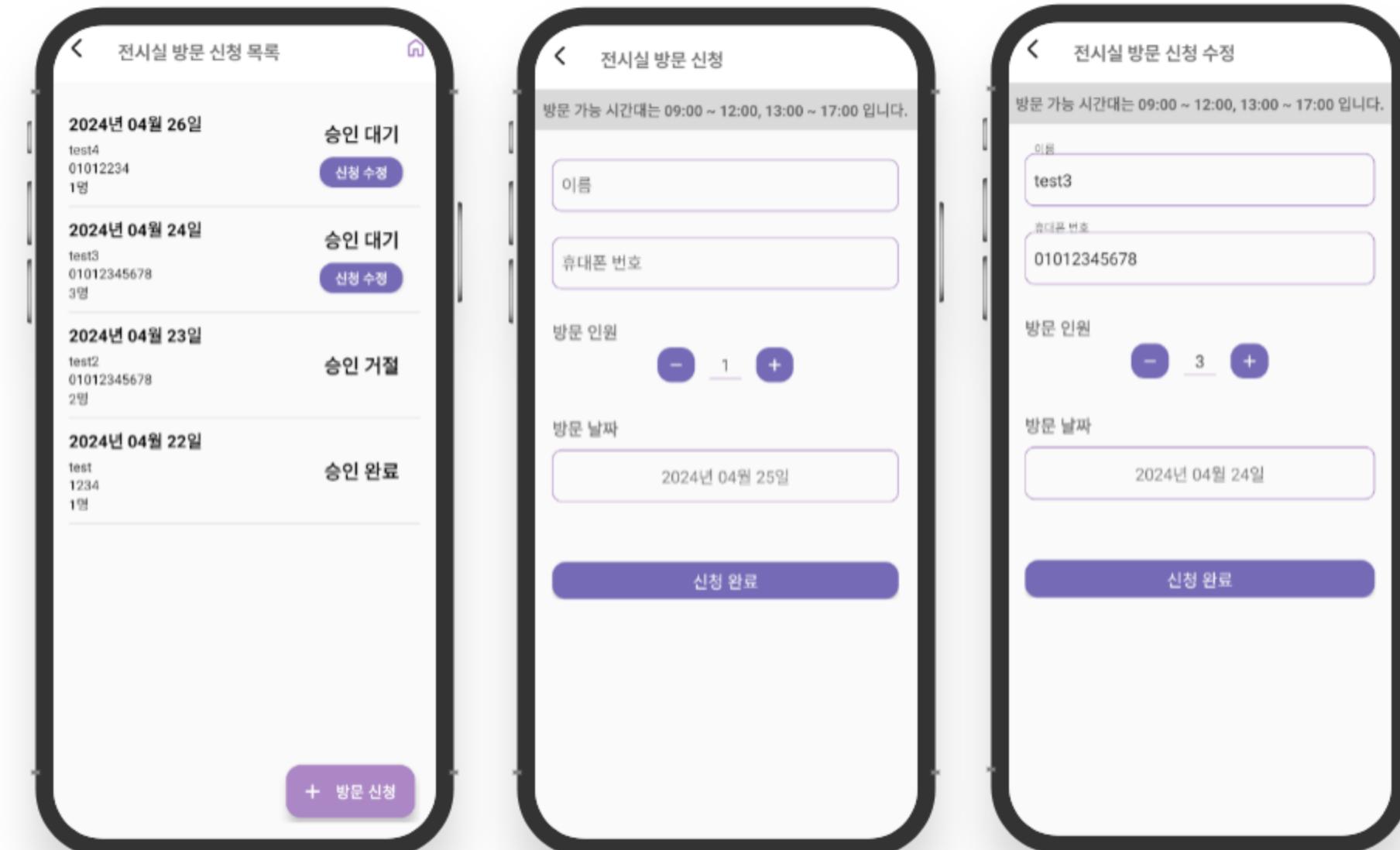
전시실 방문 신청한 내역을 표시
현재 신청 상태 확인 가능
신청 결과가 안나온 경우 수정 가능

01

방문 신청 / 수정

전시실 방문 가능 시간대를 상단에 표시
전시실 방문에 필요한 내용 작성

02



마이페이지 - 팔로우한 작가 목록

FireStore DB와 Storage 사용 활용

팔로우한 작가 목록 표시

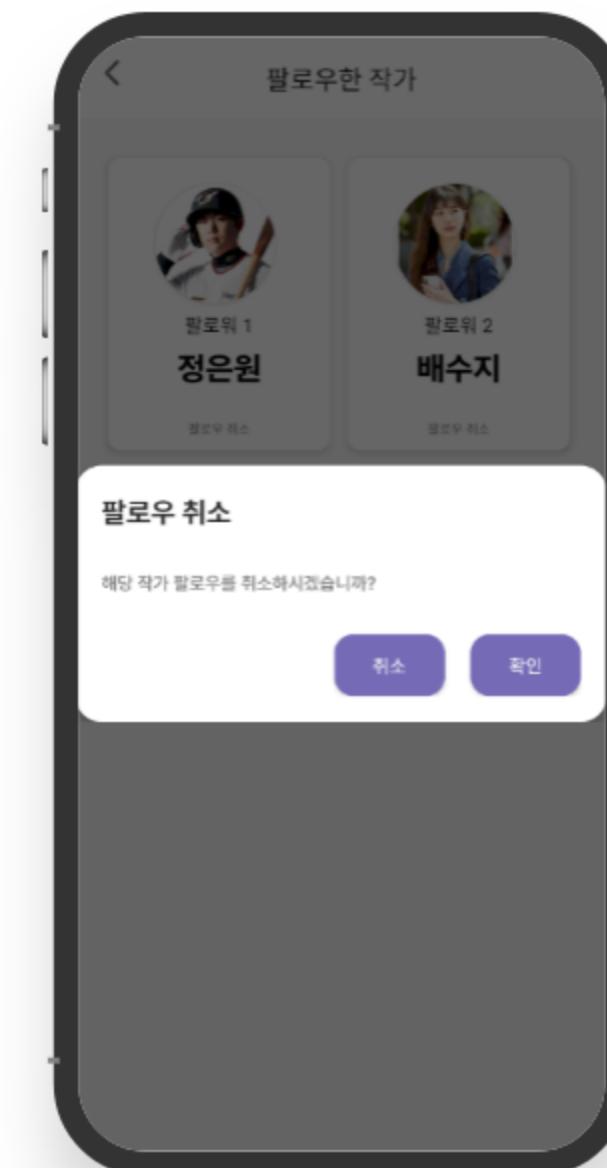
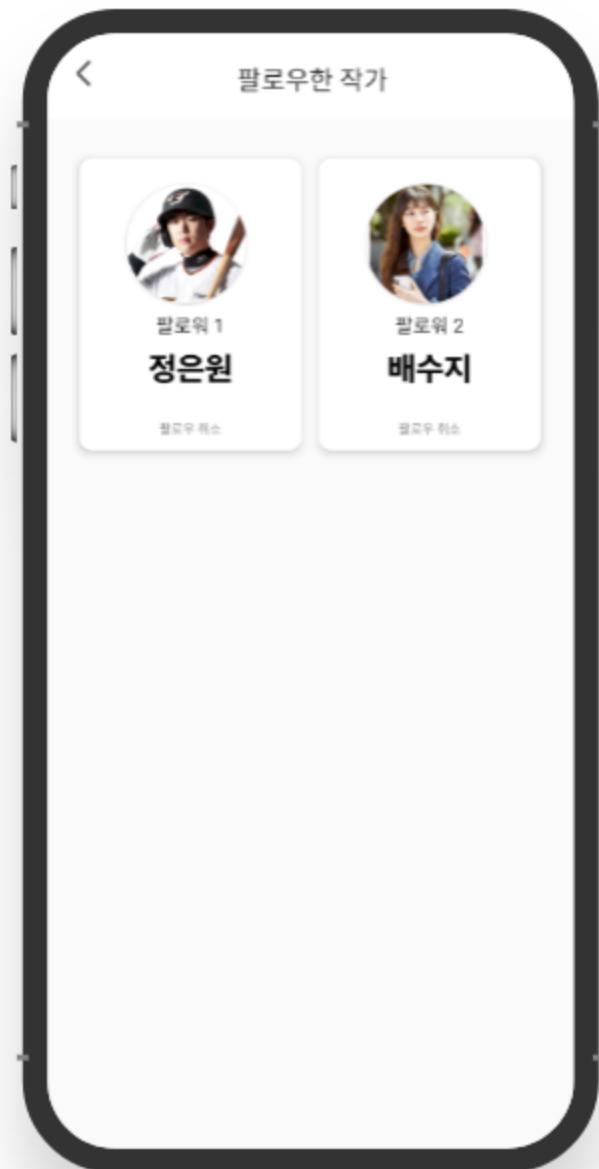
사용자가 팔로우한 작가들을 표시
해당 작가를 클릭하면 작가 상세 정보로 이동

01

팔로우 취소

팔로우 한 작가들 중
팔로우 취소를 하고 싶은 작가가 있을 경우
취소 버튼을 통해 팔로우 취소

02



마이페이지 - 작가 상세 정보

작가 상세 정보

01

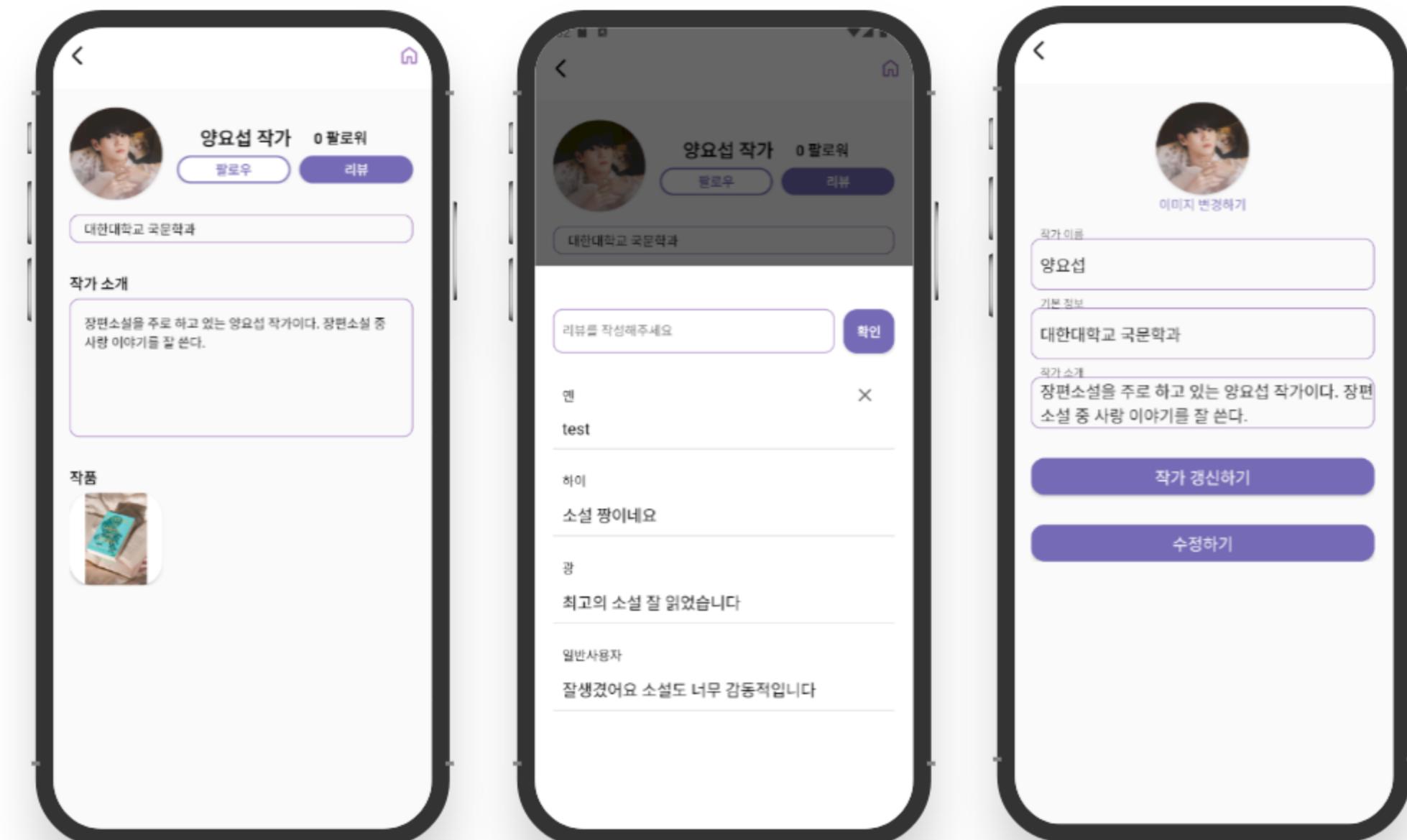
팔로우 버튼을 통해 팔로우 및 팔로우 취소
리뷰 버튼을 클릭하면 해당 작가에 대한
사용자들의 리뷰 확인 및 리뷰 작성
작품을 클릭하면 해당 작품 상세 정보로 이동

작가 정보 수정 / 갱신

02

본인이 작가인 경우 상단에 수정 아이콘 표시
프로필 이미지 및 작가 정보 수정
작가 등록 1년이 지난 경우 갱신 신청 가능

FireStore DB와 Storage 사용 활용



기술적인 도전

소셜 로그인

카카오, 네이버 API를 사용하여
소셜 로그인 구현

01

주소 검색 API

Firebase 호스팅을 사용하여
다음 주소 검색 API 적용

02



03

Coroutine

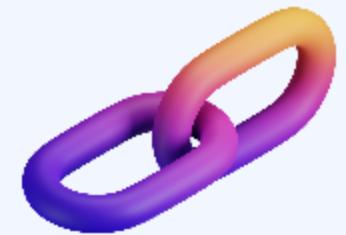
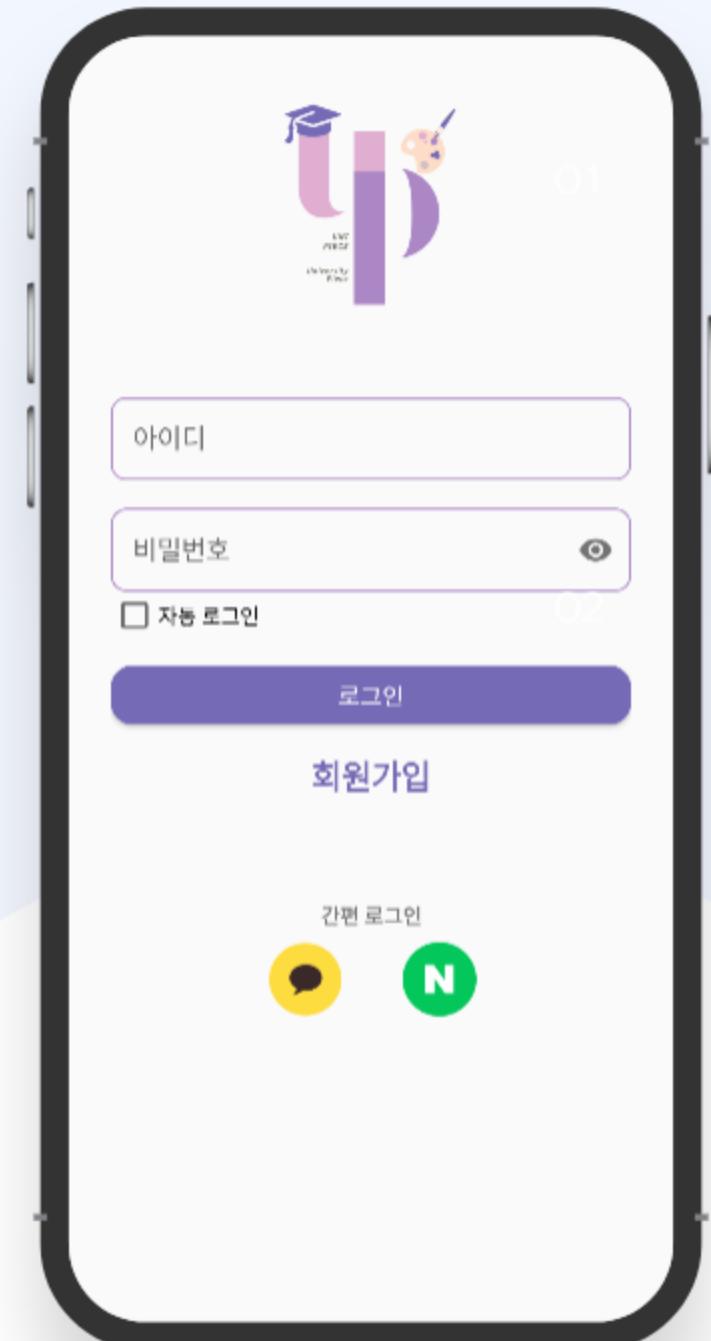
Firebase FireStore 및 Storage
쿼리 시 Coroutine을 사용해 동기적 처리

04

Android Jetpack ViewModel

MVVM의 ViewModel에서의 패턴
(데이터를 관리하기 위한 패턴)을
관리하기 편하도록 JetPack ViewModel 사용

시연 영상



시연 영상 링크

1. ViewPager2 랜더링 이슈

요구 사항	각 화면은 서로 다른 콘텐츠를 보여주어야 하며, 사용자가 모든 탭을 한눈에 볼 수 있도록 해야 함
문제 확인	<p>ViewPager를 사용해서 Tablayout을 구현했을 때</p> <ol style="list-style-type: none">height가 한 화면의 height에 맞춰 다른 화면에도 적용됨화면에 아이템이 적을 경우 아래에 공백이 생김
원인 추론	<ul style="list-style-type: none">- 레이아웃 계층 구조에서 부모 뷰의 크기를 기반으로 설정되어 있을 때 발생할 수 있음- ViewPager 내부의 Fragment 또는 뷰의 height가 고정되어 있거나, height를 동적으로 조정하지 않아서 발생할 수 있음
해결 방법 선택지 및 의견조율	<ul style="list-style-type: none">- ViewPager의 height를 다르게 설정- Viewpager의 height를 동적으로 조정

1. ViewPager2 랜더링 이슈

의견 결정 ViewPager의 height를 동적으로 계산하여 조정

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)

    // ViewPager의 높이를 동적으로 계산하여 설정
    binding.viewpagerMyGallery.viewTreeObserver.addOnGlobalLayoutListener(object : ViewTreeObserver.OnGlobalLayoutListener {
        override fun onGlobalLayout() {
            val viewPager = binding.viewpagerMyGallery
            val screenHeight = resources.displayMetrics.heightPixels

            // ViewPager의 높이 계산
            val viewPagerLayoutParams = viewPager.layoutParams
            val density = resources.displayMetrics.density
            val additionalHeightInPixel = (80 * density).toInt()
            // 나머지 화면 높이 계산
            viewPagerLayoutParams.height = screenHeight - viewPager.top - additionalHeightInPixel
            viewPager.setLayoutParams(viewPagerLayoutParams)

            // OnGlobalLayoutListener 제거
            viewPager.viewTreeObserver.removeOnGlobalLayoutListener( victim: this)
        }
    })
}
```

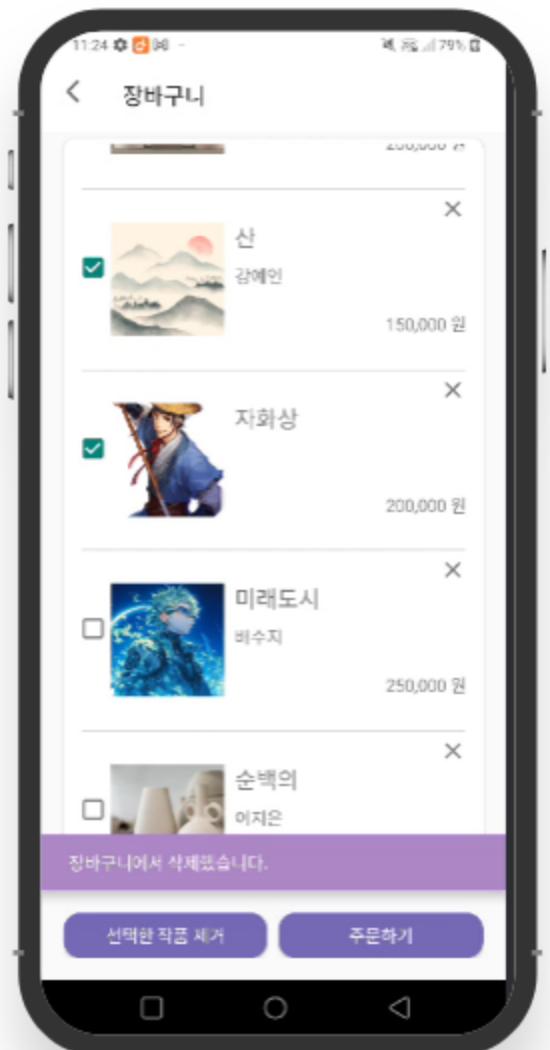
2. FireBase 속도 이슈

요구 사항	<ul style="list-style-type: none"> - 각 화면은 받아온 데이터를 사용자에게 보여줘야 함 - 사용자가 수정, 삭제한 데이터를 적용 후에, 데이터를 받아와서 사용자에게 보여줘야 함
문제 확인	<ul style="list-style-type: none"> - 데이터를 받아올 때까지 빈 화면으로 보여짐 - 데이터를 수정, 삭제하고 변경사항을 적용해서 데이터를 받아오지 않고, 변경사항 이전 데이터를 받아와서 보여짐
원인 추론	<ul style="list-style-type: none"> - 데이터를 받아오는 속도가 느려서, 빈화면이 출력이 된 후 기다린 후에 데이터를 받아옴 - 데이터를 수정, 삭제를 처리하는 시간이 걸려서, 기다리지 않고 바로 데이터를 받아와 이전 데이터가 보여짐
해결 방법 선택지 및 의견조율	<ul style="list-style-type: none"> - progressbar나 snackbar를 사용해서 사용자에게 진행 사항을 시각적으로 보여줌 - callback과 함께 코루틴을 사용하여 데이터 처리 성공 후에 데이터를 받아오도록 함

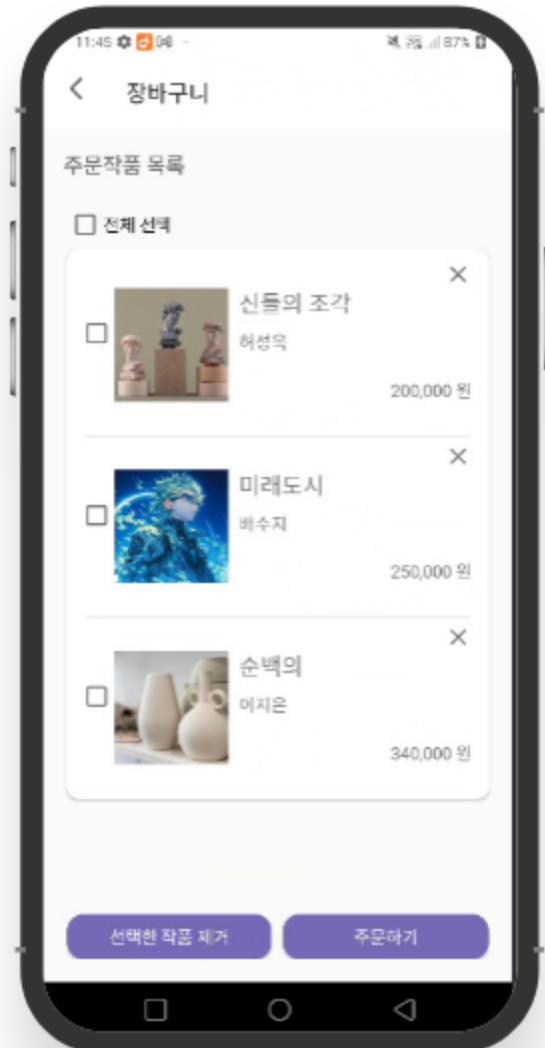
2. 데이터 처리 속도 이슈

의견 결정

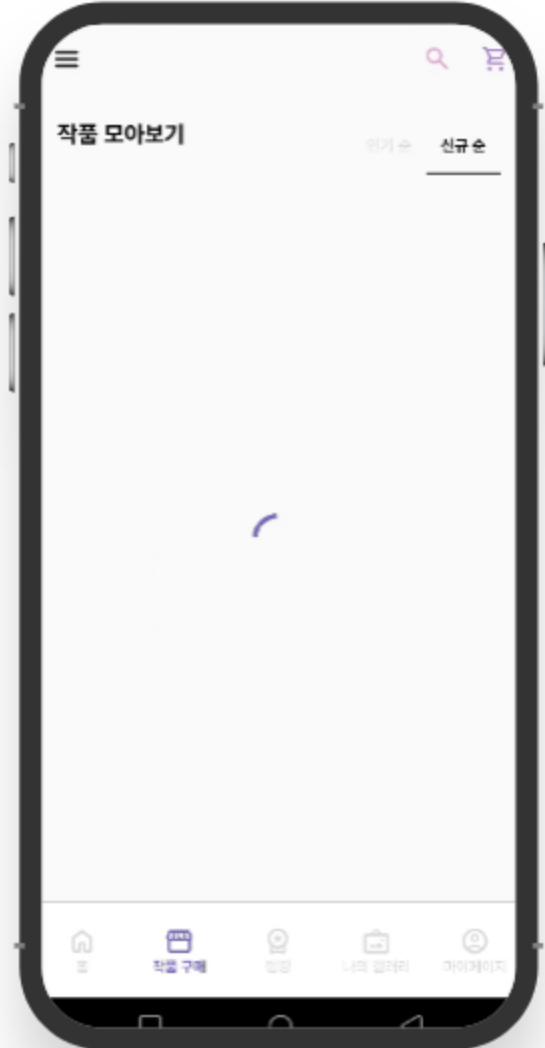
loading을 알 수 있는 LiveData
필드를 구현해서 상태에 맞
게 progressbar나 snackbar를
사용해서 사용자에게 진행 사항
을 시각적으로 보여줌



장바구니
삭제 전 스낵바 띄움



장바구니
삭제 후 스낵바 사라짐



데이터 받아오기 전
progressbar 보여줌

2. 데이터 처리 속도 이슈

의견 결정

loading을 알 수 있는 LiveData 필드를 구현해서 상태에 맞게 progressbar나 snackbar를 사용해서 사용자에게 진행 사항을 시각적으로 보여줌

```
private val _loading = MutableLiveData<Boolean>(
    value: true
)
val loading: LiveData<Boolean> = _loading

fun setLoading(){
    viewModel.loading.observe(owner: this@CartActivity) { loading ->
        with(binding){ this: ActivityCartBinding
            // 로딩 중일때
            if(loading){
                progressBar.visibility = View.VISIBLE
                cardView.visibility = View.GONE
            }
            // 로딩 중이 아닐 때
            else {
                progressBar.visibility = View.GONE
                cardView.visibility = View.VISIBLE
            }
        }
    }
}
```

2. 데이터 처리 속도 이슈

의견 결정 callback과 함께 코루틴을 사용하여 데이터 처리 성공 후에 데이터를 받아오도록 함

```
// 선택한 작품 제거 버튼
with(buttonCartDelete) { this: Button
    setOnClickListener { it: View!
        // 삭제 스낵바
        showLikeSnackbar("장바구니에서 삭제했습니다.")
        val list = cartAdapter.getSelectedData()

        // 코루틴으로 삭제 완료 된 후 데이터 다시 불러오기
        lifecycleScope.launch { this: CoroutineScope
            val deletionTasks = list.map { item ->
                async { this: CoroutineScope
                    suspendCoroutine<Boolean> { cont ->
                        viewModel.deleteCartDataByUserIdx(userIdx, item.pieceInfo.pieceIdx) { success ->
                            cont.resume(success)
                        }
                    }
                }
            }
            val results = deletionTasks.awaitAll()

            if (results.all { it }) {
                viewModel.getCartDataByUserIdx(userIdx)
            }
        }
    }
}
```



```
// 특정 사용자의 특정 작품을 장바구니에서 삭제하는 함수
± tjddn +1*
fun deleteCartDataByUserIdx(userIdx: Int, pieceIdx: Int, callback: (Boolean) -> Unit) {
    viewModelScope.launch { this: CoroutineScope
        val success = try {
            cartRepository.deleteCartDataByUserIdx(userIdx, pieceIdx)
            _loading.value = false
            true
        } catch (e: Exception) {
            Log.e( tag: "Firebase Error", msg: "Error vmDeleteCartDataByUserIdx : ${e.message}")
            false
        }
        // 콜백으로 성공 여부 반환
        callback(success)
    }
}
```

3. 안드로이드 앱 아이콘이 나타나지 않는 오류

요구 사항	<ul style="list-style-type: none">- 안드로이드 앱은 실행 후 앱 아이콘이 메뉴에 등록이 되어야 함
문제 확인	<ul style="list-style-type: none">- 안드로이드 앱은 정상적으로 실행이 되지만, 앱 아이콘이 메뉴에서 나타나지 않고 등록되지 않음
원인 추론	<ul style="list-style-type: none">- 카카오톡 API 사용시 AndroidManifest에서 android.intent.action.MAIN action과 android.intent.category.LAUNCHER category가 있는 <intent-filter>에 <data> 태그를 추가하는 경우 앱 아이콘이 사라짐
해결 방법 선택지 및 의견조율	<ul style="list-style-type: none">- <data>가 필요한 경우 <intent-filter>를 분리해서 2개로 나눠야함

3. 안드로이드 앱 아이콘이 나타나지 않는 오류

의견 결정 <data>가 필요한 경우 <intent-filter>를 분리해서 2개로 나눠야함

```
<activity
    android:name=".ui.login.SplashScreenActivity"
    android:exported="true"
    android:theme="@style/Base.Theme.SplashScreen.Unipiece">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
    <intent-filter>
        <data
            android:host="oauth"
            android:scheme="${NATIVE_APP_KEY}" />
    </intent-filter>
</activity>
```

기대 효과



- 01 쇼핑몰을 통해 작품을 소개함으로써 작품 노출 기회를 제공해 학생들이 작가로서의 경력을 시작하는데 중요한 발판이 될 수 있게 한다
- 02 작품을 판매하여 학생들이 경제적인 지원을 받아 예술적 활동을 계속할 수 있는 가능성이 열린다
- 03 작가들이 지속적으로 활동할 수 있게 하여 문화적 가치를 높이고 사회적 다양성을 증진시키는 데 기여할 수 있다
- 04 졸업 전시나 과제 제출 후 무분별하게 버려지는 작품들을 가치 있게 활용할 수 있다

향후 프로젝트 계획

부가 기능 구현

- 앱 내 알림
 - 팔로우
 - 작가 신청 및 갱신 승인
 - 전시실 방문 신청 승인
 - 판매 신청 승인
 - 주문 취소 승인
 - 반품 취소 승인
- 옵션 기능 (작품 경매)
- 랭킹 기능
 - 공동 순위 반영으로 수정 예정

데이터 추가 관리자 앱

- 관리자 승인 기능
 - 작가 신청 및 갱신 승인
 - 전시실 방문 신청 승인
 - 주문 취소 승인
 - 반품 취소 승인
 - 판매 신청 승인
- 승인한 작가 정보나 작품 정보를 쇼핑몰에 사용되는 DB에 데이터 추가하는 기능

리팩토링 진행

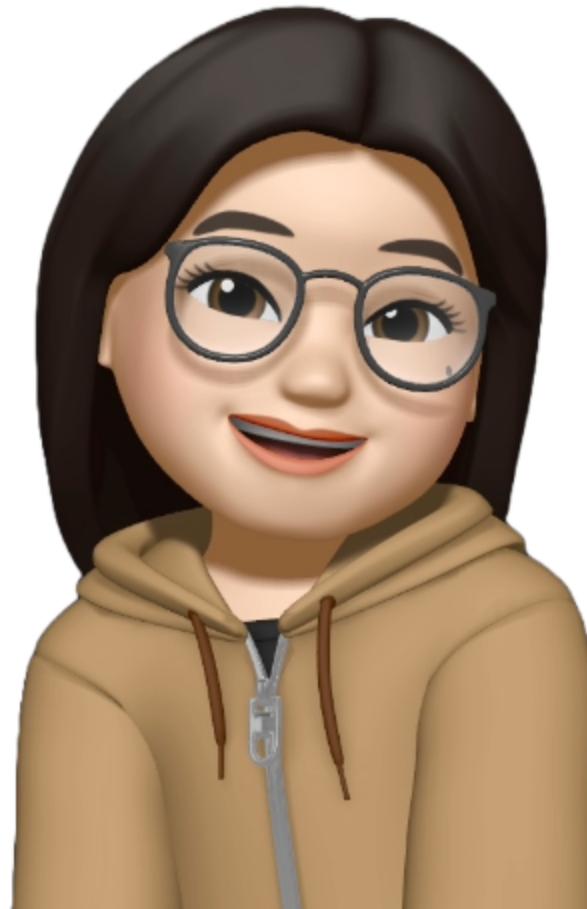
- 코드 정리
 - Repository
 - DataSource
 - ViewModel
- Directory 구조 정리

은정민

디자인 패턴과 코루틴을 적용해서 기능을 구현하는 경험은 새롭고 유익했으며, 많은 것을 배우고 성장할 수 있었습니다. 프로젝트의 리더는 처음이었기 때문에 부족한 점이 많았지만, 함께 으쌰으쌰 해준 팀원들 덕분에 무사히 잘 마무리할 수 있었습니다. 팀원 여러분, 정말 고생 많았고 다음에 또 함께 프로젝트 해볼 기회가 있기를! 고마웠습니다.

리더

- UI / DB 설계
- 관심 작품 목록
- 구매한 작품 목록
- 판매 작품 목록
- 작품 등록 신청 및 수정
- 주문 취소
- 반품 접수



허성욱

살면서 한 달이라는 시간이 이렇게 짧다고 느낀 적은 처음인 거 같습니다., 앱 스쿨 2기를 통해 처음으로 개발을 접하게 되었고 다른 팀원분들에 비해 실력이 많이 부족했지만 다들 잘 알려주시고 도와주셔서 마무리까지 잘할 수 있었던 거 같습니다! 다들 고생하셨고 다들 시간이 괜찮으시다면 저희 UniPiece 마무리까지 할 수 있었으면 좋겠습니다!

부리더

- UI / DB 설계
- 회원가입
- 로그인
- 홈
- 전시 소개
- 소식
- 전시실 작품 소개
- 작가 등록 신청 및 수정



주성원

프로젝트를 시작하기 전 이 프로젝트가 즐겁게 마무리됐으면 좋겠다는 목표가 있었는데 좋은 팀원들 덕에 이를 수 있어서 좋았고, 정해진 디자인패턴이나 깃허브로 개발 및 소통하는 방식 등 협업에 필요한 스킬들을 빠삭하게 알아가는 유익한 시간이었습니다. 협업프로젝트를 하면 훨씬 더 많이 성장할 수 있다는 말이 크게 공감하고, 앞으로 하게 될 많은 프로젝트들에서 넘어지지 않도록 기초를 잘 닦을 수 있었던 시간이었다고 생각합니다. 앞으로도 기억에 많이 남을 프로젝트 일 것 같습니다! 수고하셨습니다!

팀원

- UI / DB 설계
- 팔로잉한 작가
- 장바구니
- 배송지 추가 및 수정
 - 주소 검색 API
- 주문하기
- 결제



엄민식

기획부터 개발까지 다양한 의견과 생각을 접하고 생각의 폭이 넓어진 것 같아 정말 좋은 경험이었습니다. 혼자 공부 한 것보다 팀원으로 프로젝트를 진행하며 공부할 때 더 많이 발전한 것 같습니다.

팀원

- UI / DB 설계
- 마이페이지
- 회원 정보 수정
- 작가 정보
- 작가 정보 수정
- 전시실 신청 목록
- 전시실 신청 및 수정



강예인

디자인 패턴 MVVM과 코루틴을 처음 적용해서 프로젝트를 진행해 봤는데 부족한 점이 많았지만 이렇게 무사히 완성할 수 있어서 뿌듯했습니다! 이제 조금 알 것 같은 느낌이 들어서 많이 성장하고 배운 것 같습니다. 제가 이것저것 제안을 많이 했는데 팀원분들이 열심히 함께 따라주셔서 이렇게 무사히 마칠 수 있었던 것 같습니다 ! 다음에 또 같이 팀하고 싶습니다~ 다들 고마웠어요 :) 수고하셨습니다.

팀원

- UI / DB 설계
- 프로젝트 초기 설정
 - 깃허브
 - Bottom Navigation 구현
- 작품 구매
- 작품 상세보기
- 작품 랭킹
- 작품 검색
- 장바구니 삭제 기능



UniPiece 유니피스



유니피스는 당신의 작품을, 당신을 응원합니다!

감사합니다.



| 6팀 멋쟁이 사람들

은정민 허성욱 강예인 주성원 엄민식