

# iOS Seminar 2

Wafflestudio 2024



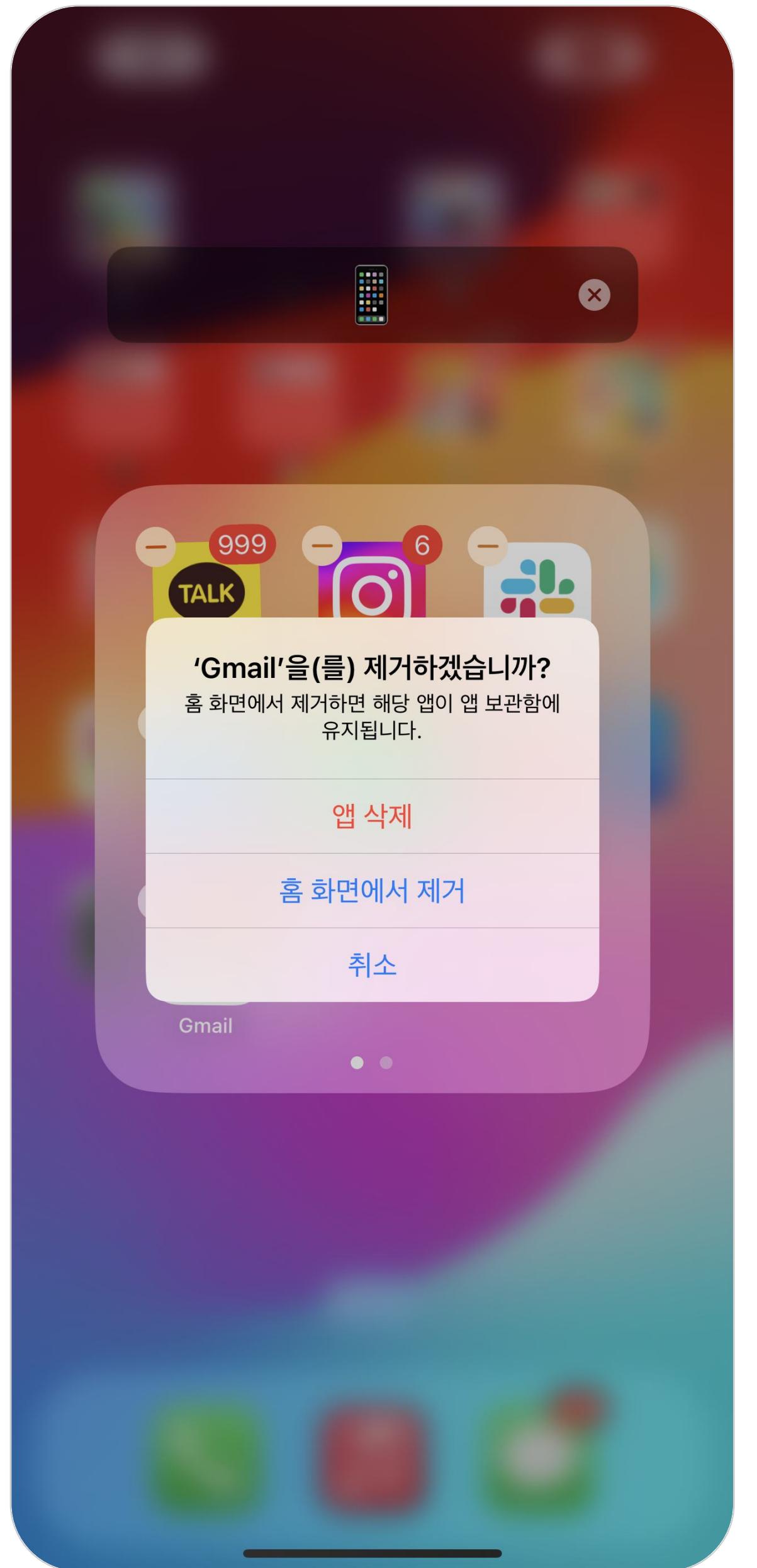


# Networking



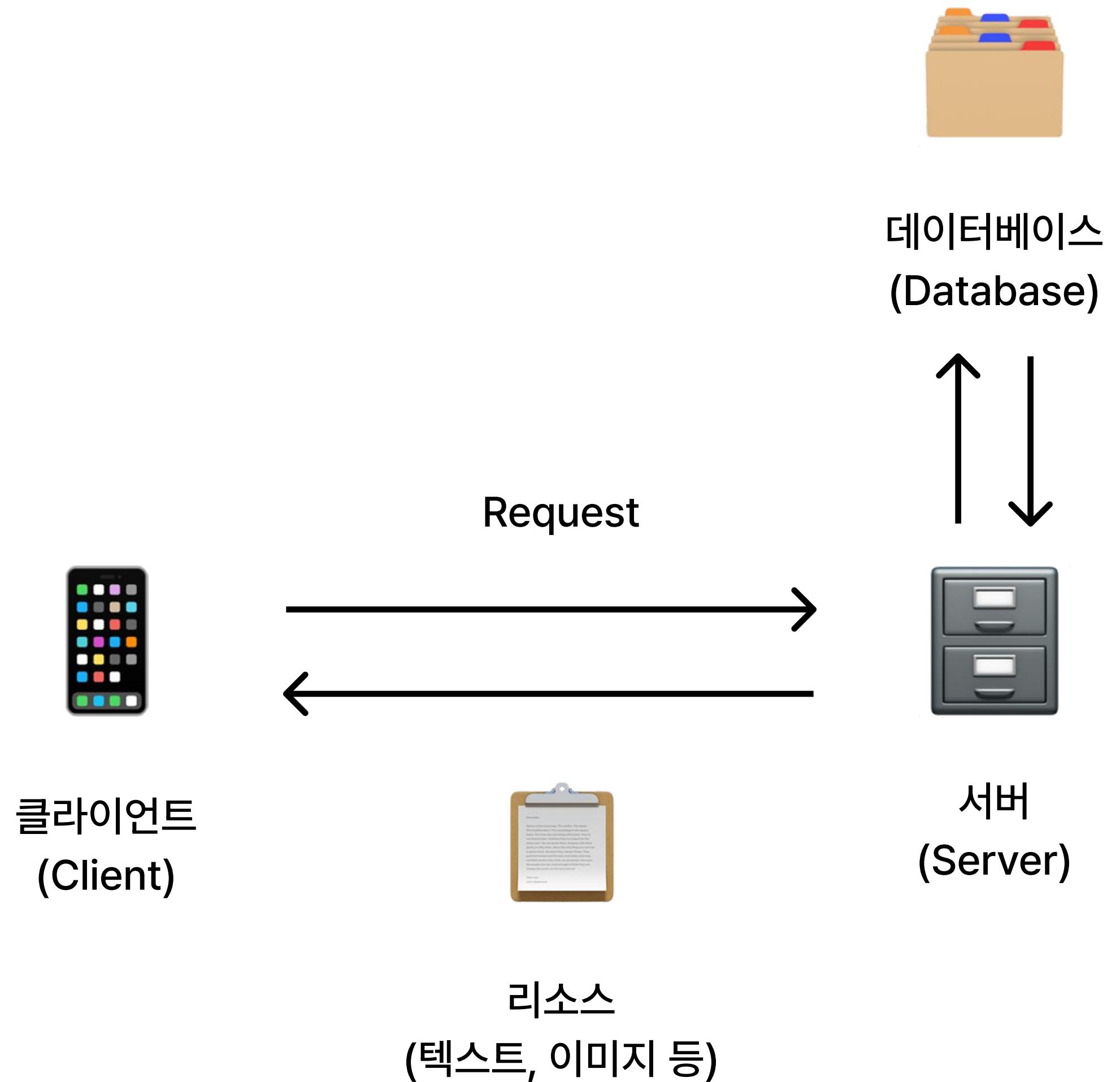
## UserDefaults (공식문서)

- 간단한 캐싱을 위한 클래스
  - key-value pair 이용
  - 앱을 지우기 전까지 유지되는 정보
- 정보를 저장하는 다른 방법들
  - FileManager
  - Core Data
- 과제0, 과제1에서 사용





## Networking





## REST API

- HTTP URI로 리소스 특정하고, HTTP Method로 리소스에 행할 동작을 정의
- 대표적인 메서드
  - **GET** : 리소스 조회
    - GET /user : 유저 정보 조회
  - **POST** : 리소스 생성
    - POST /user : 유저 생성
  - **PUT** : 기존 리소스 업데이트
    - PUT /user/password : 유저의 password 업데이트
  - **DELETE** : 리소스 삭제
    - DELETE /user : 유저 정보 삭제



## REST API

- Query Parameter
  - URL 뒤에 `?`와 함께 붙는 키-값(Key-Value) 쌍
  - GET <https://movie.com/user?id=wafflestudio&type=POPULAR>
- Path Variable
  - `/{{value}}`로 특정 리소스 지칭
  - GET <https://movie.com/top100list>
- Request Body
  - POST 요청에 많이 사용되며, 일반적으로 JSON 형식



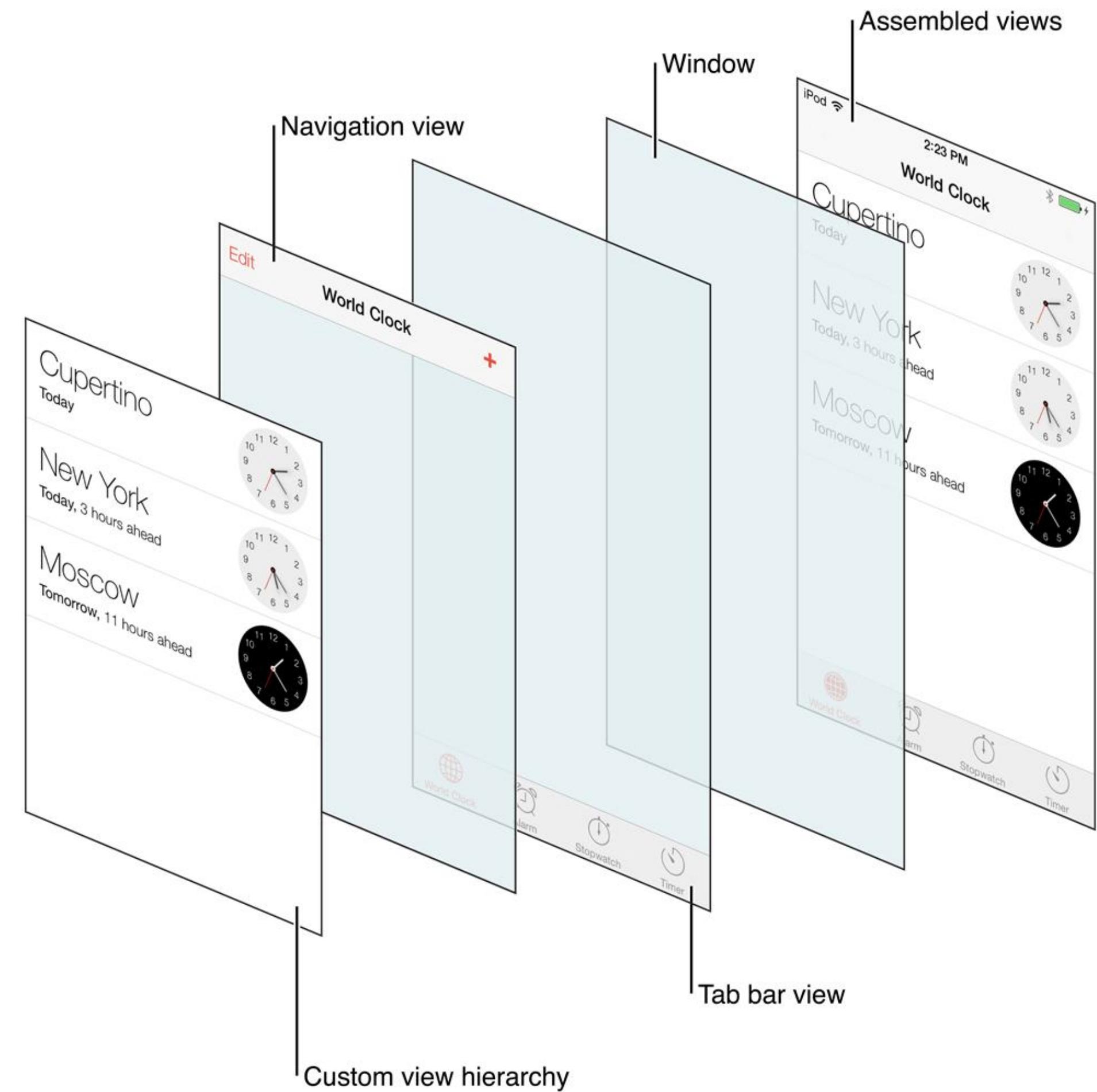
UITabBarController

# UITabBarController



## UITabBarController (공식문서)

- 여러 Child View Controller를 보여주기 위한 Container View Controller
- 각 탭은 별개의 ViewController를 root로 함
- 하단 Tab Bar 아이템을 커스텀화하려면 UITabBarItem 사용
- UITabBarControllerDelegate를 상속하여 Tab Bar 인터페이스와 상호작용이 일어났을 때의 동작을 정의할 수 있음





Concurrency

# Concurrency

## Asynchronous Code

- 🍔 햄버거 하나 만들어 줘
- 그릴에 고기가 구워지는 동안 다른 작업 진행

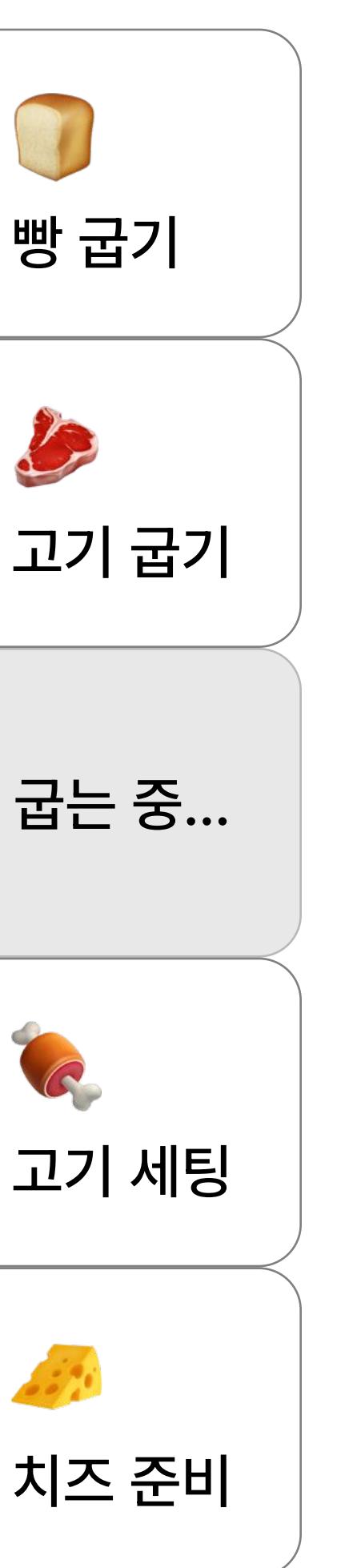


## Parallel Code

-  **햄버거 하나 만들어 줘**
- 두 명의 요리사가 햄버거 만들기

Before

A



Parallel Code

A

B





## async/await

- Concurrency (관련 문서)
  - combination of asynchronous and parallel code
  - 함수/메서드에 `async` 추가: 이 함수/메서드는 실행 중간에 suspension될 만한 지점이 있다!
    - 그러한 suspension point에 `await`를 붙임
      - throw가 발생할 만한 지점에 try 사용하는 것과 유사
    - 실질적으로는 다른 asynchronous 메서드를 호출하는 경우에만 실제로 실행이 유예됨



# Asynchronous Function + Parallel

- `await`를 이용하여 asynchronous 메서드를 호출하면 딱 그 한 줄이 실행된다
- 동시에 진행되어도 괜찮은 작업과 그렇지 않은 작업을 적절히 구분
  - `async-let`

```
let firstPhoto = await downloadPhoto(named: photoNames[0])
let secondPhoto = await downloadPhoto(named: photoNames[1])
let thirdPhoto = await downloadPhoto(named: photoNames[2])

let photos = [firstPhoto, secondPhoto, thirdPhoto]
```

```
downloadPhoto()
```

```
downloadPhoto()
```

```
downloadPhoto()
```

```
let photos = [firstPhoto, secondPhoto, thirdPhoto]
```



```
async let firstPhoto = downloadPhoto(named: photoNames[0])
async let secondPhoto = downloadPhoto(named: photoNames[1])
async let thirdPhoto = downloadPhoto(named: photoNames[2])

let photos = await [firstPhoto, secondPhoto, thirdPhoto]
```

```
downloadPhoto()
```

```
downloadPhoto()
```

```
downloadPhoto()
```

```
let photos = await [firstPhoto, secondPhoto, thirdPhoto]
```





Build Configuration File

# Build Configuration File



## Build Configuration File (공식문서)

- API key, access token 등의 정보
  - 그대로 github의 public repository에 올려버린다면?
- Dev 환경, Prod 환경에 따라 다르게 쓰여야 하는 정보
  - <https://snutt-api-dev.wafflestudio.com/~>
  - <https://snutt-api.wafflestudio.com/~>
  - 똑같은 변수로 다루고 싶은데, 환경에 따라 다른 값이 필요하다면?

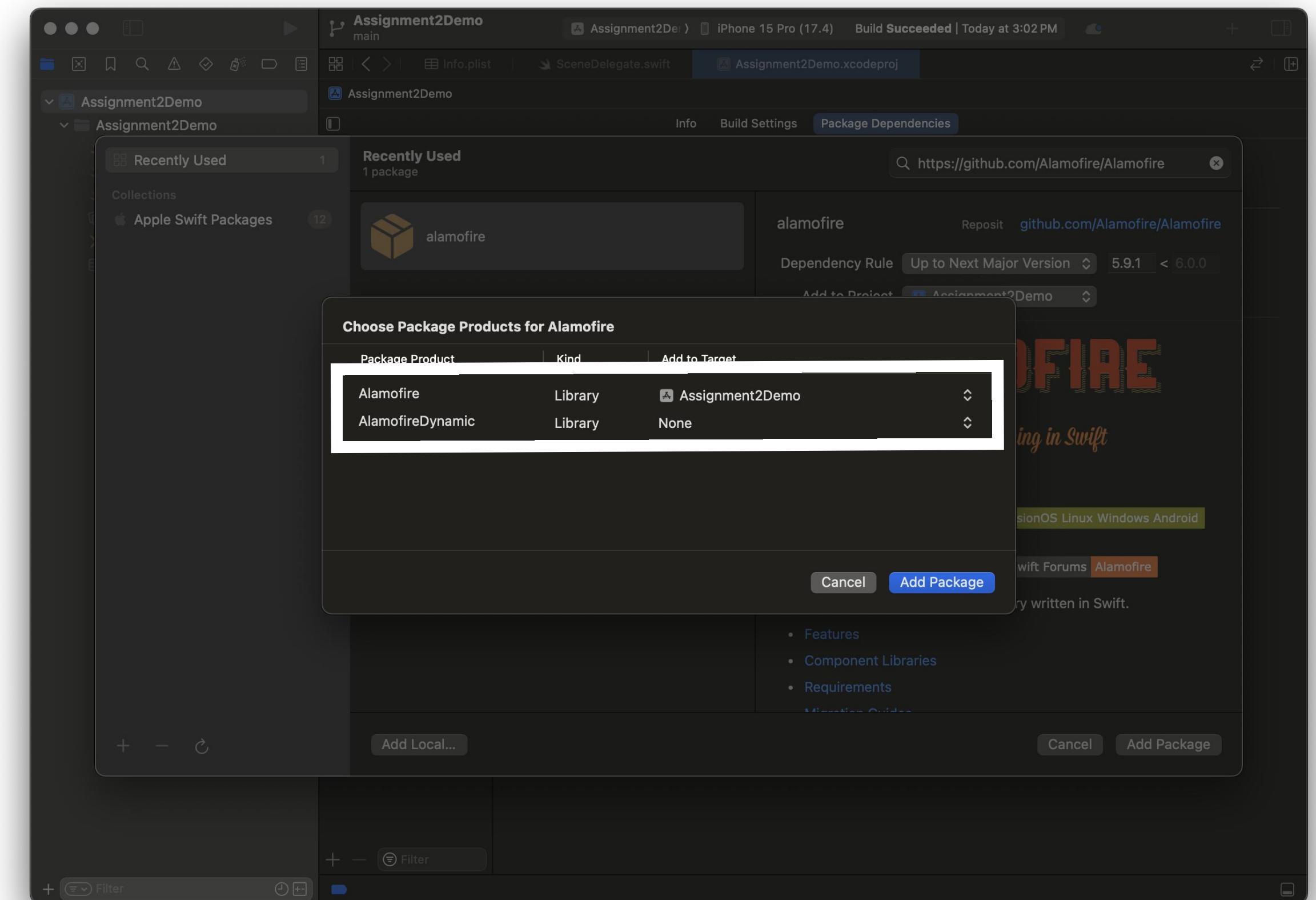
# Swift Package Manager



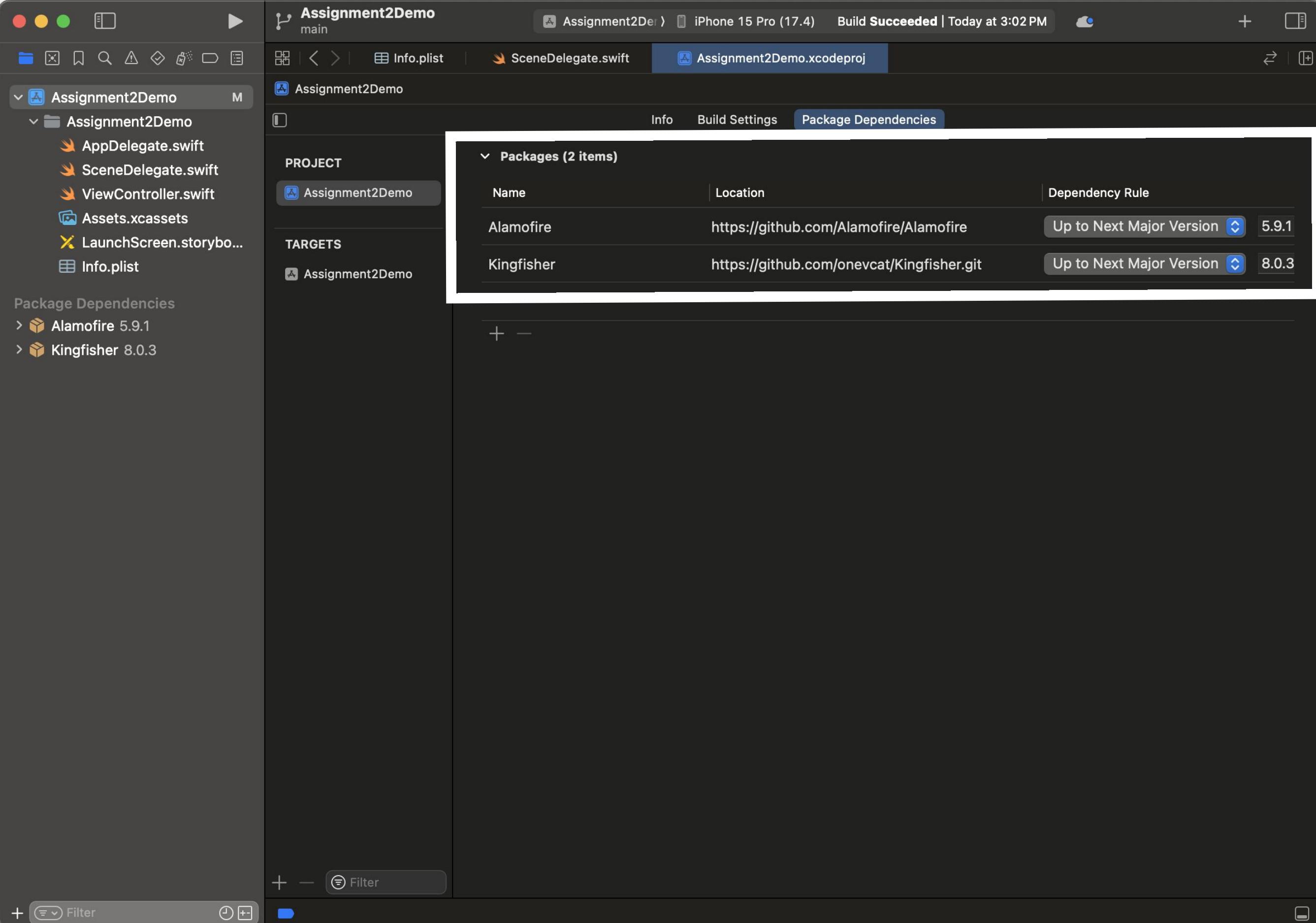


## 과제2에서 사용할 Library

- Alamofire
  - <https://github.com/Alamofire/Alamofire>
  - HTTP 네트워킹 라이브러리
  - AF 대신 URLSession을 사용해도 괜찮습니다
- Kingfisher
  - <https://github.com/onevcat/Kingfisher.git>
  - 웹에서 비동기로 이미지를 다운받고, 캐싱하는 작업을 도와줌



## Package Manager





## Assignment 2

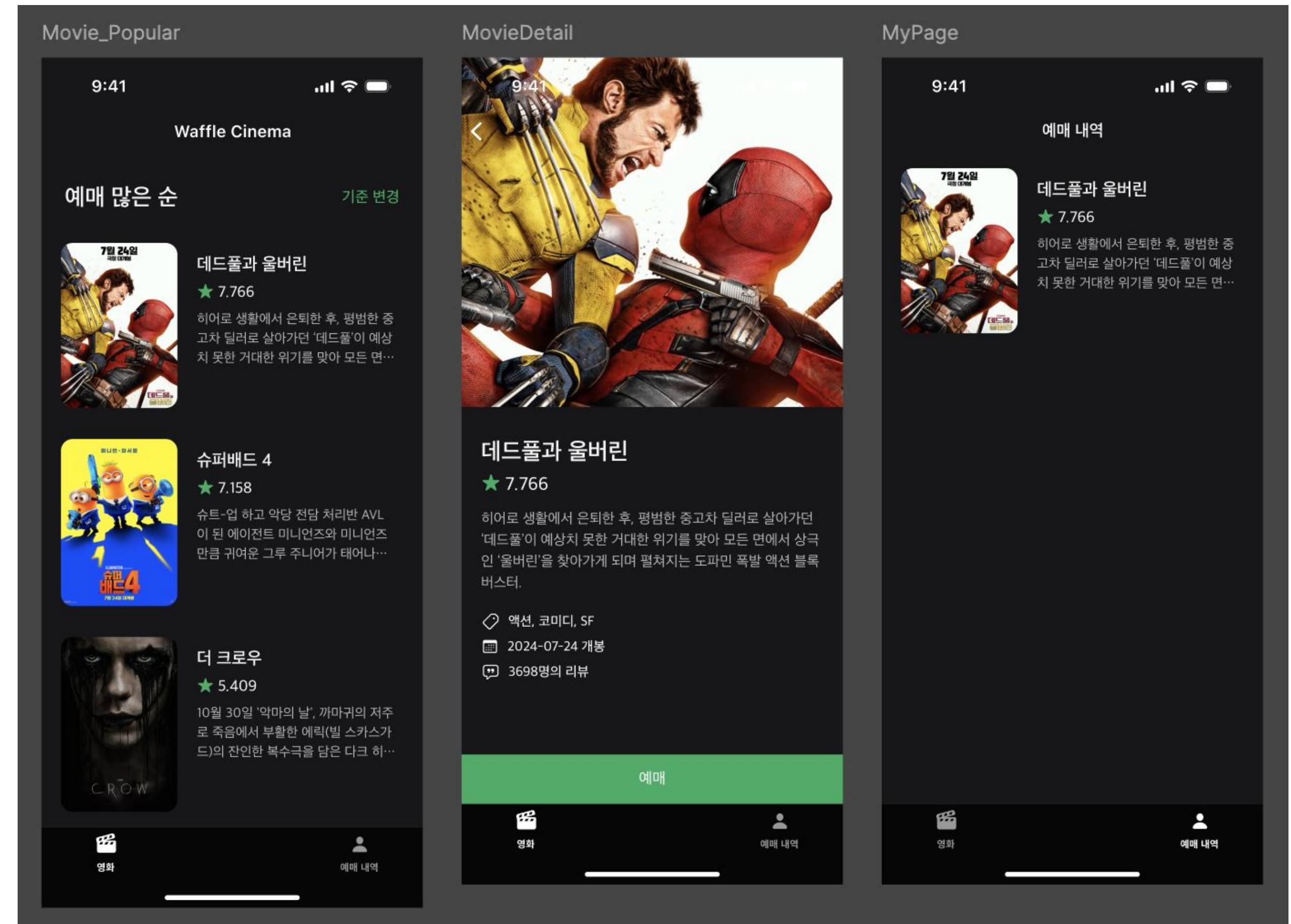
# 과제 2 안내



## Assignment 2

# 간단한 영화 예매 앱 만들기

- 과제 2 디자인 링크(Figma)
- 과제 2 스켈레톤 코드 링크(Github)
- 관련 키워드
  - Package Manager
    - Alamofire
    - Kingfisher
  - UITabBarController
  - UITableView + UITableViewCell
  - UINavigationController
  - UserDefaults
  - UIAlertController





## 과제 상세 스펙 및 주의사항 (1)

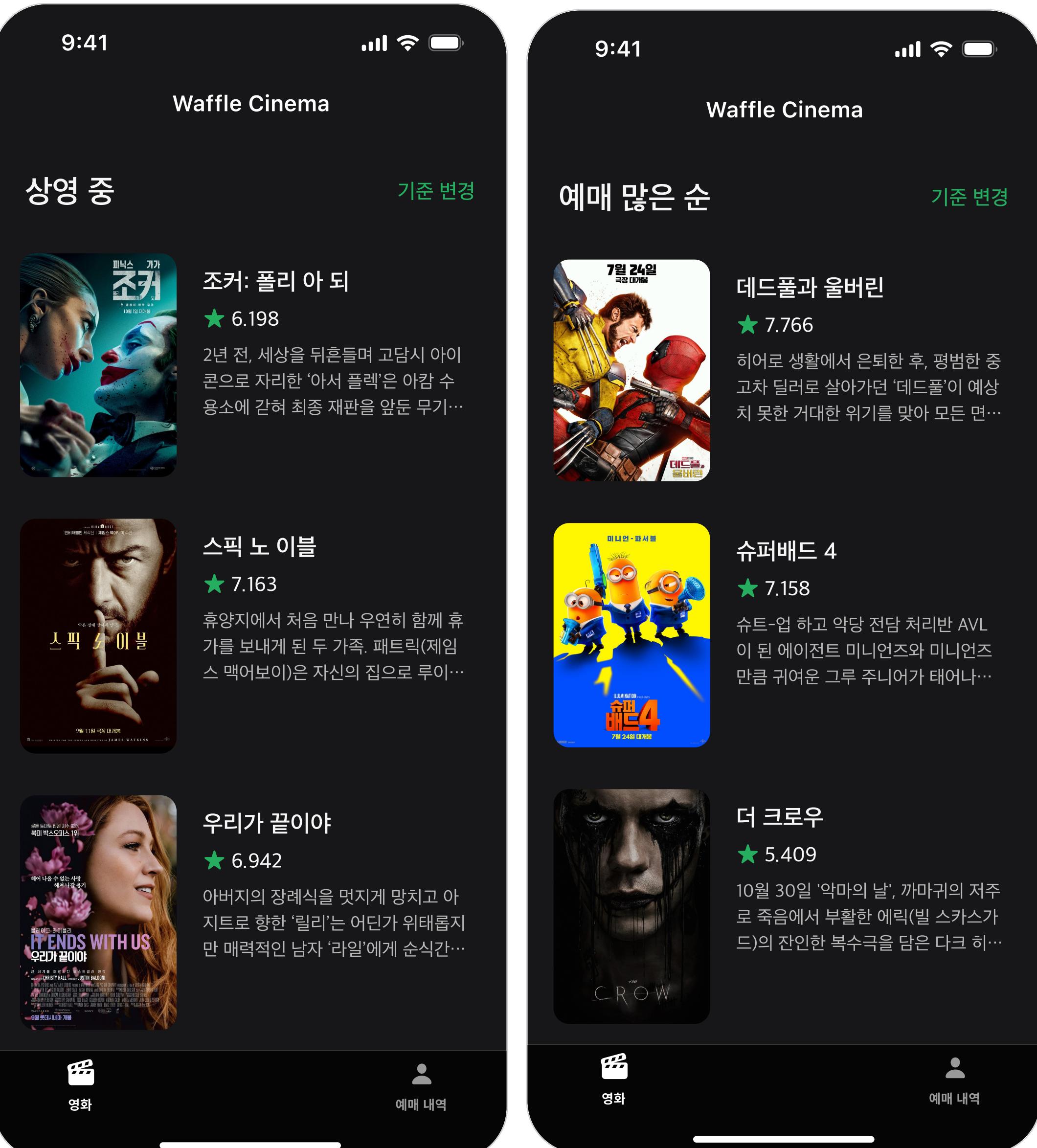
- 전체
  - Storyboard를 사용하지 않습니다 (LaunchScreen.storyboard 제외)
  - 모든 Text, Color, Constraint, Corner Radius 등은 피그마에 주어진 값을 사용합니다
  - Figma에 스펙이 누락된 부분이 있다면, 슬랙에서 질문 부탁드립니다
- 스켈레톤 코드
  - 스켈레톤 코드만으로는 과제를 해결하실 수 없습니다. 적절한 변수와 함수를 추가해 주세요
  - 스켈레톤 코드는 자유롭게 수정 가능하며, 아예 사용하지 않으셔도 괜찮습니다
    - 사용 여부는 과제 채점에 아무런 영향을 미치지 않습니다
- Font
  - SF Pro / Apple SD Gothic Neo는 Apple의 시스템 폰트이므로 과제 진행 시 size, weight, color만 고려합니다
- Appearance
  - Dark Mode로 강제합니다



## Assignment 2

### 과제 상세 스펙 및 주의사항 (2)

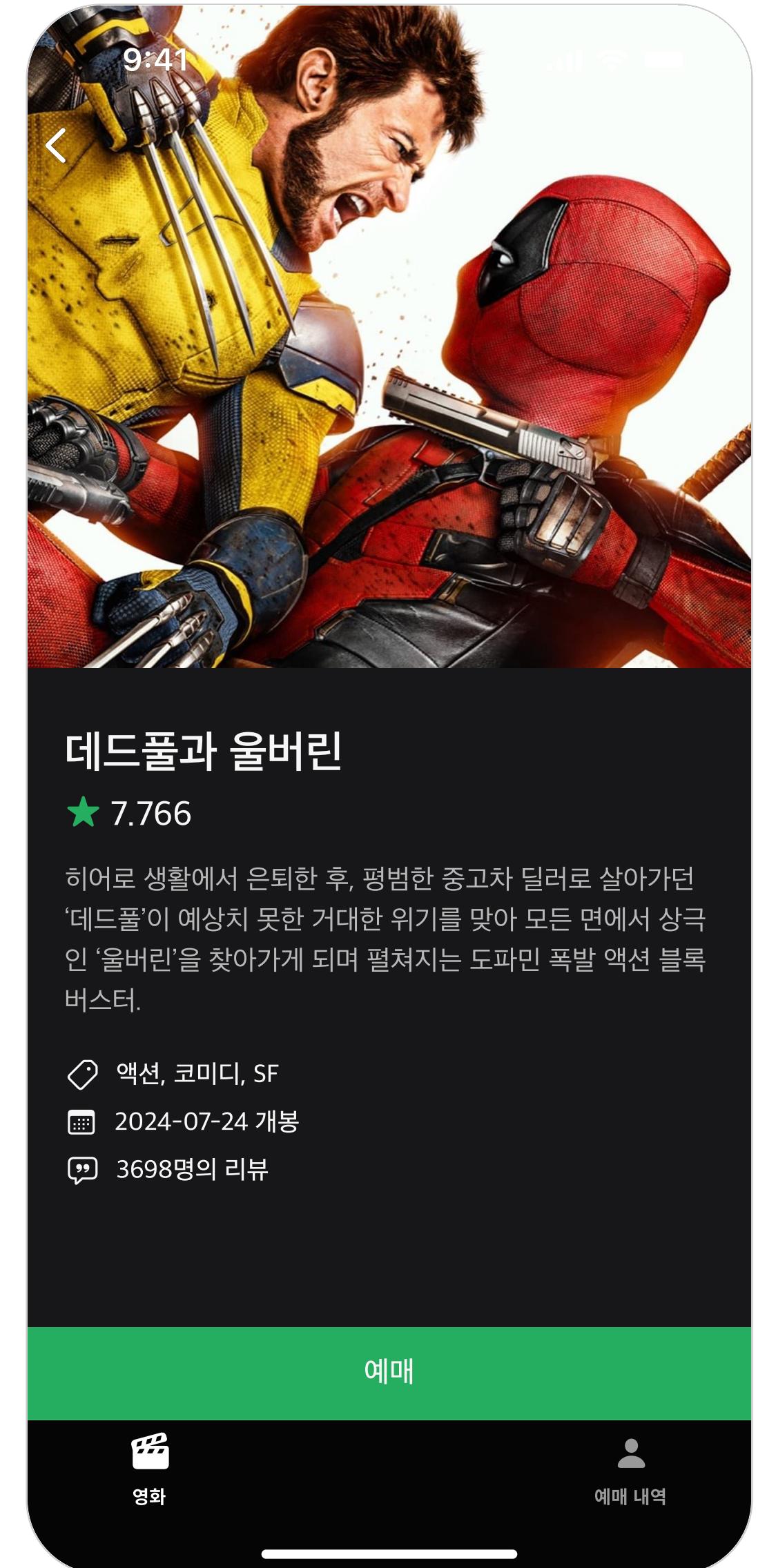
- 앱을 처음 시작하는 경우에는 항상 “상영 중”을 보여줍니다
  - 우측의 “기준 변경” 버튼을 눌러 “상영 중” <> “예매 많은 순” 전환
- 영화 목록은 반드시 UITableView를 이용하여 보여줍니다
  - 포스터 이미지는 비율을 유지하되 가로를 120으로 고정
    - 영화에 따라 UITableViewCell의 높이가 다를 수 있습니다
  - 영화의 개요는 최대 3줄까지만 보여줍니다
  - 영화의 평점은 별도의 조작 없이 API response값 그대로 보여줍니다
- Figma에는 편의상 영화 3개만을 보여주고 있으나, 실제로는 무한 scroll이 가능하도록 합니다
  - Pagination 이용





## 과제 상세 스펙 및 주의사항 (3)

- 앱을 처음 시작하는 경우에는 항상 “상영 중”을 보여줍니다
  - 우측의 “기준 변경” 버튼을 눌러 “상영 중” <> “예매 많은 순” 전환
- 영화 목록은 반드시 UITableView를 이용하여 보여줍니다
  - 포스터 이미지는 비율을 유지하되 가로를 120으로 고정
    - 영화에 따라 UITableViewCell의 높이가 다를 수 있습니다
  - 영화의 개요는 최대 3줄까지만 보여줍니다
  - 영화의 평점은 별도의 조작 없이 API response값 그대로 보여줍니다
- Figma에는 편의상 영화 3개만을 보여주고 있으나, 실제로는 무한 scroll이 가능하도록 합니다
  - Pagination 이용
- UIAlertController
  - 별도의 커스텀 디자인이 필요하지 않습니다. 스펙에 맞는 문구만 지정해 주세요

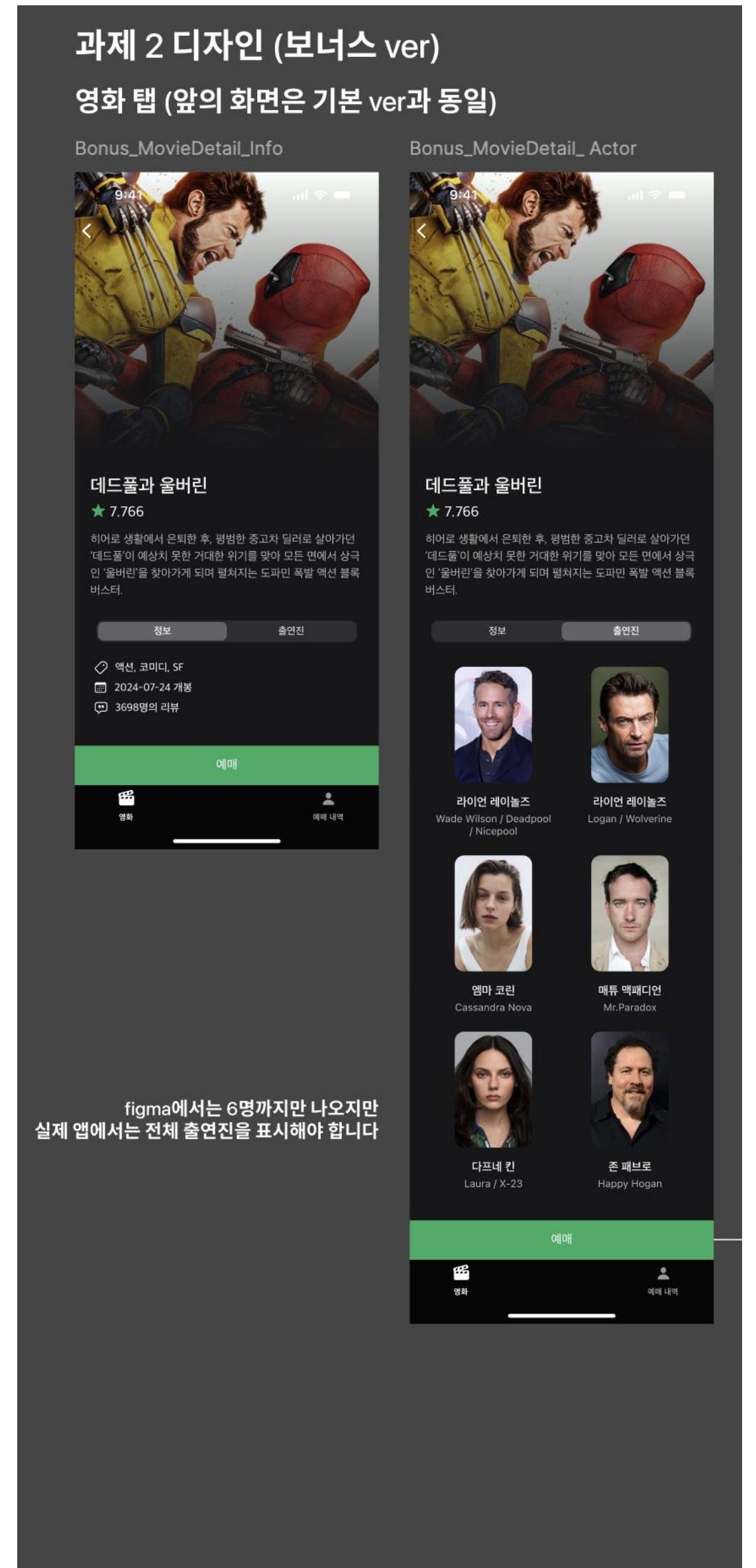




## Assignment 2

### 과제 Bonus 스펙 (1)

- 영화 상세 화면의 포스터 이미지를 피그마와 같이 보여줍니다
  - hint:
- 영화 설명 아래에 영화의 정보와 출연진을 나누어서 볼 수 있도록 합니다
  - figma에는 편의상 일부 출연진만을 보여주고 있지만, 실제로는 모든 출연진을 보여줍니다
  - 출연진은 반드시 UICollectionView를 이용하여 보여줍니다
  - hint: UISegmentedControl
    - UISegmentedControl에서 별도의 커스텀 디자인을 하지 않습니다
    - 각 메뉴의 이름을 올바르게 지정하고, 올바른 동작을 수행하도록 구현합니다





## Assignment 2

# 과제 Bonus 스펙 (2)

- 초기 잔액 200,000원 내에서 영화 예매
  - 예매 버튼을 누르면, sheet 형태로 예매 화면을 띄웁니다
    - 예매가 완료된 영화에 대해서는 이전과 마찬가지로 아무런 변화가 없도록 합니다
  - 성인은 명당 10,000원, 청소년은 명당 8,000원
  - 총 인원 수는 0명이 될 수 없으며 각 인원 수는 0명 이상 (max값은 알아서 설정해주시되, 총 금액이 잔액보다 많은 경우에는 결제를 진행할 수 없도록 합니다)
  - 예매 내역에 결제한 금액 Label을 함께 보여줍니다
    - 예매 내역이 추가/삭제되는 경우, 잔액 Label도 올바르게 업데이트합니다
- hint: UIStepper

