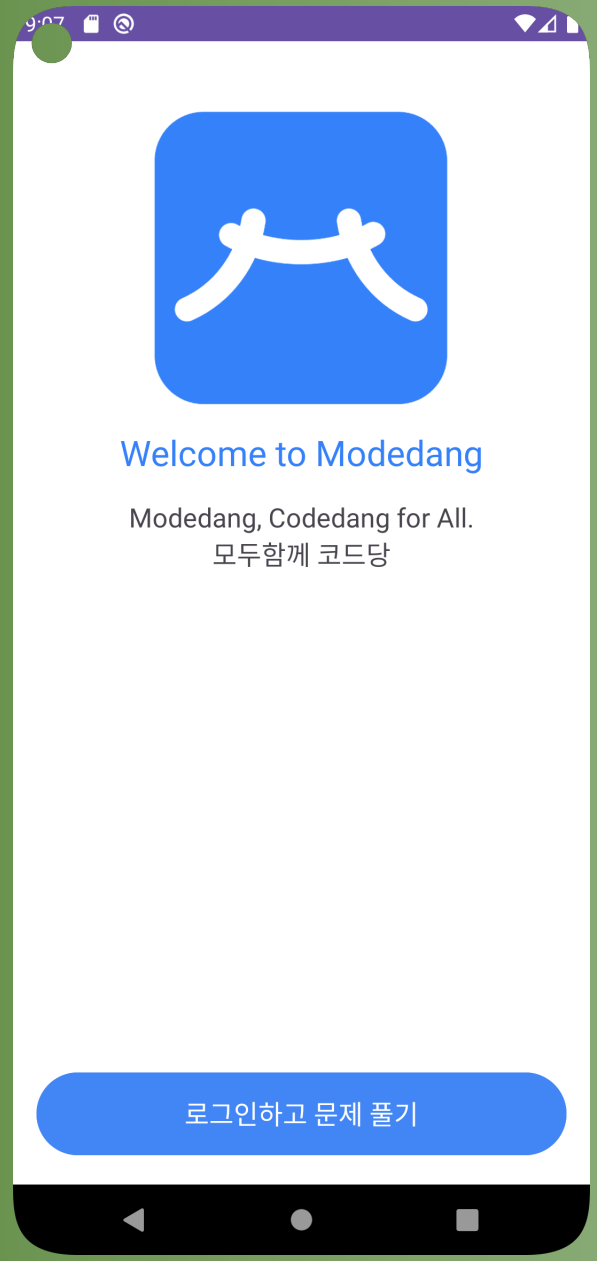
Modedang, Codedang for All

모두함께 코드당, 모두당

소프트웨어학과 하지민

모드당이란?

- 모바일 앱에서도 PS를 할 수 있으면 좋을 것 같은데 ?!
- 간단하고 예쁜 UI로 필요한 정보만!
- 대중교통을 이용하면서,
  노트북이 없는 상황에서도,
  핸드폰이나 태블릿PC를 사용하여
  편하게 코딩 가능

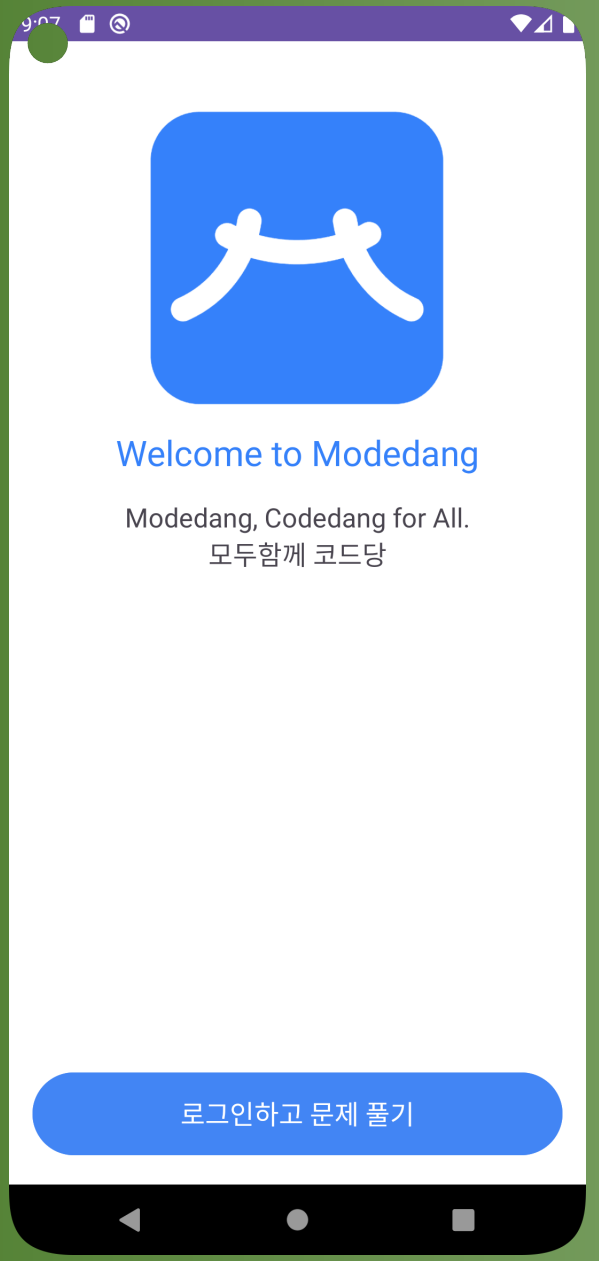


UI / UX

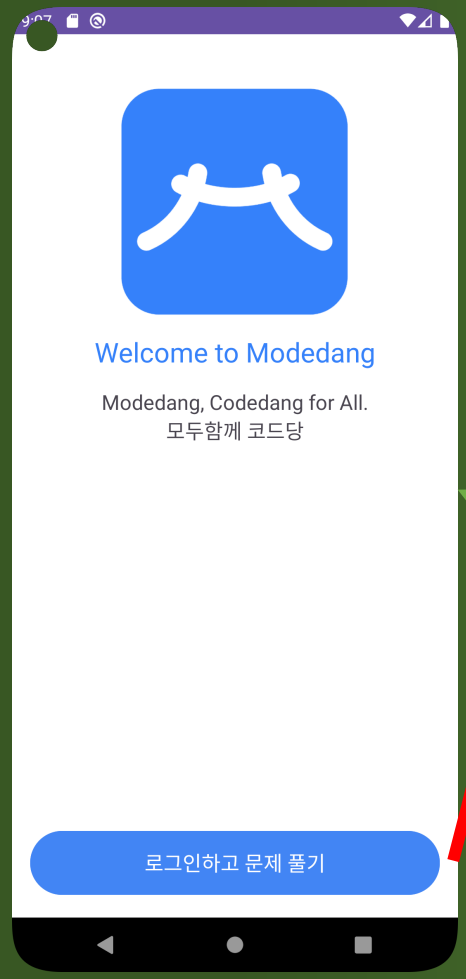
- 작은 화면을 사용하는 사용자에 대한 배려
- 사용자가 처한 상황에 필요없는 정보는 과감하게 배제
- 필요한 정보를 크게 보여주는 방식

UI / UX

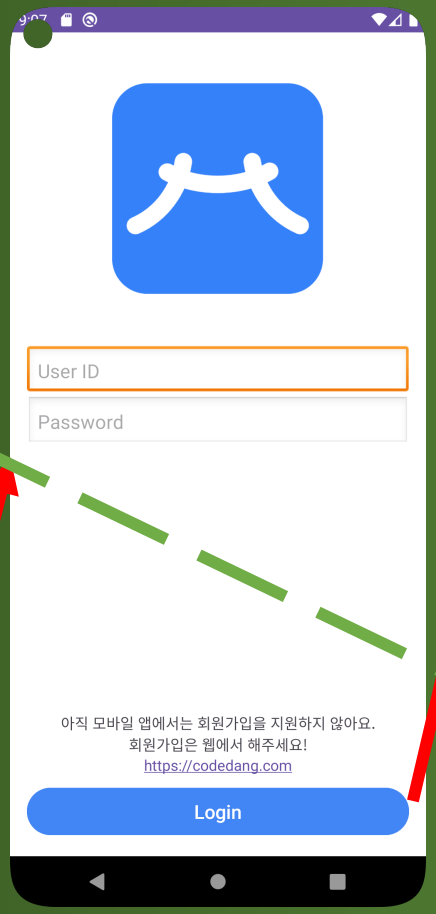
- 모바일 앱에서는 화면의 윗부분의 터치하기가 어려움
- 모든 버튼은 일관성있게 화면의 아래에 배치 (토스 방식 사용)
- 사용자 Flow를 단순하게 구성하여 손쉽게 사용할 수 있게 하였음



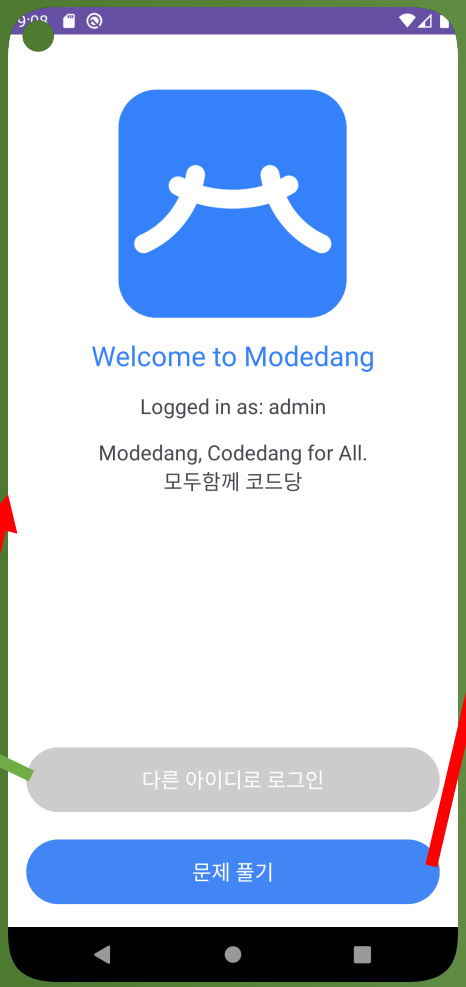
User Flow



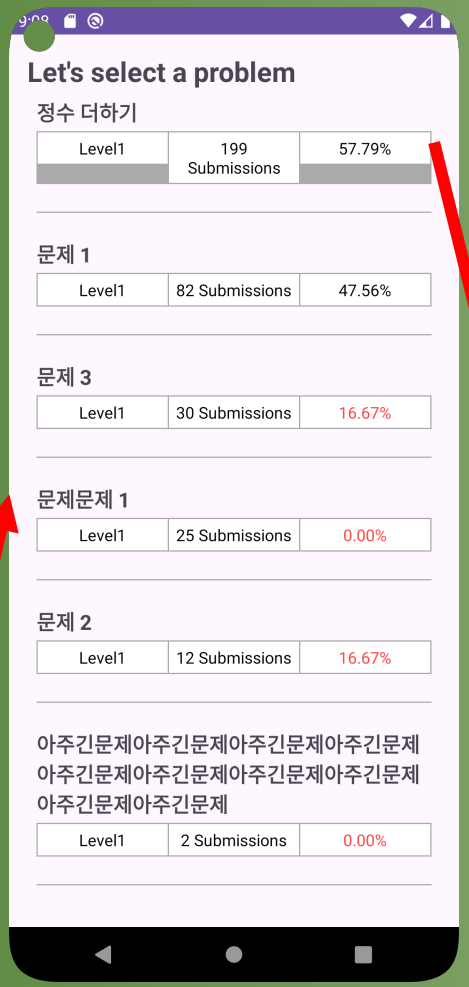
ActivityMain



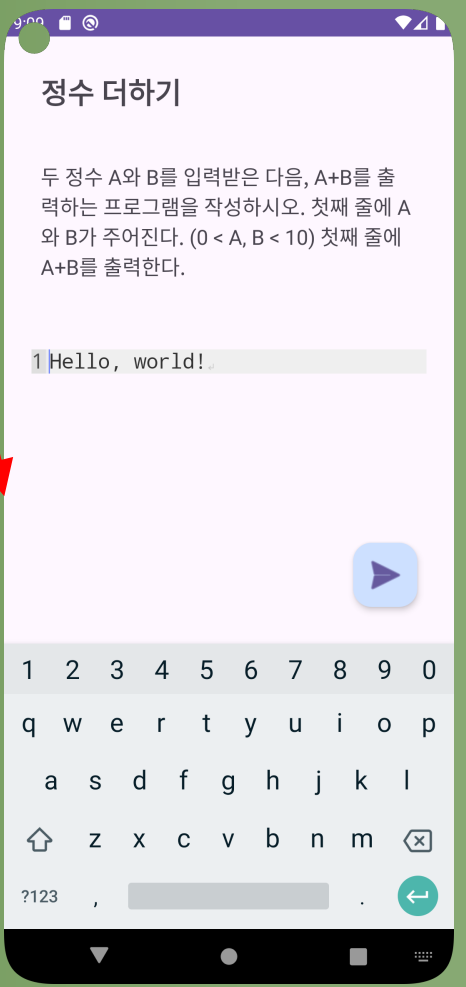
ActivityLogin
(이미 로그인되어 있으면 pass)



ActivityMain

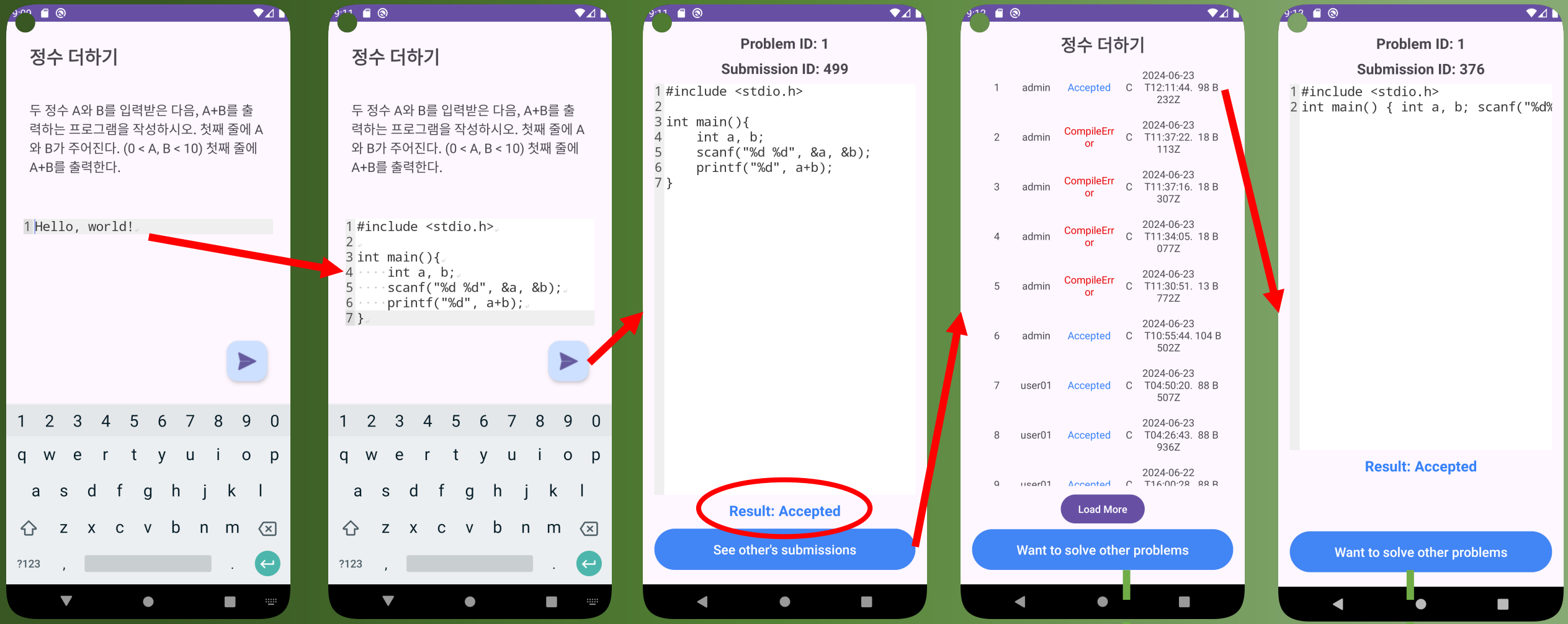


ActivityProblemList

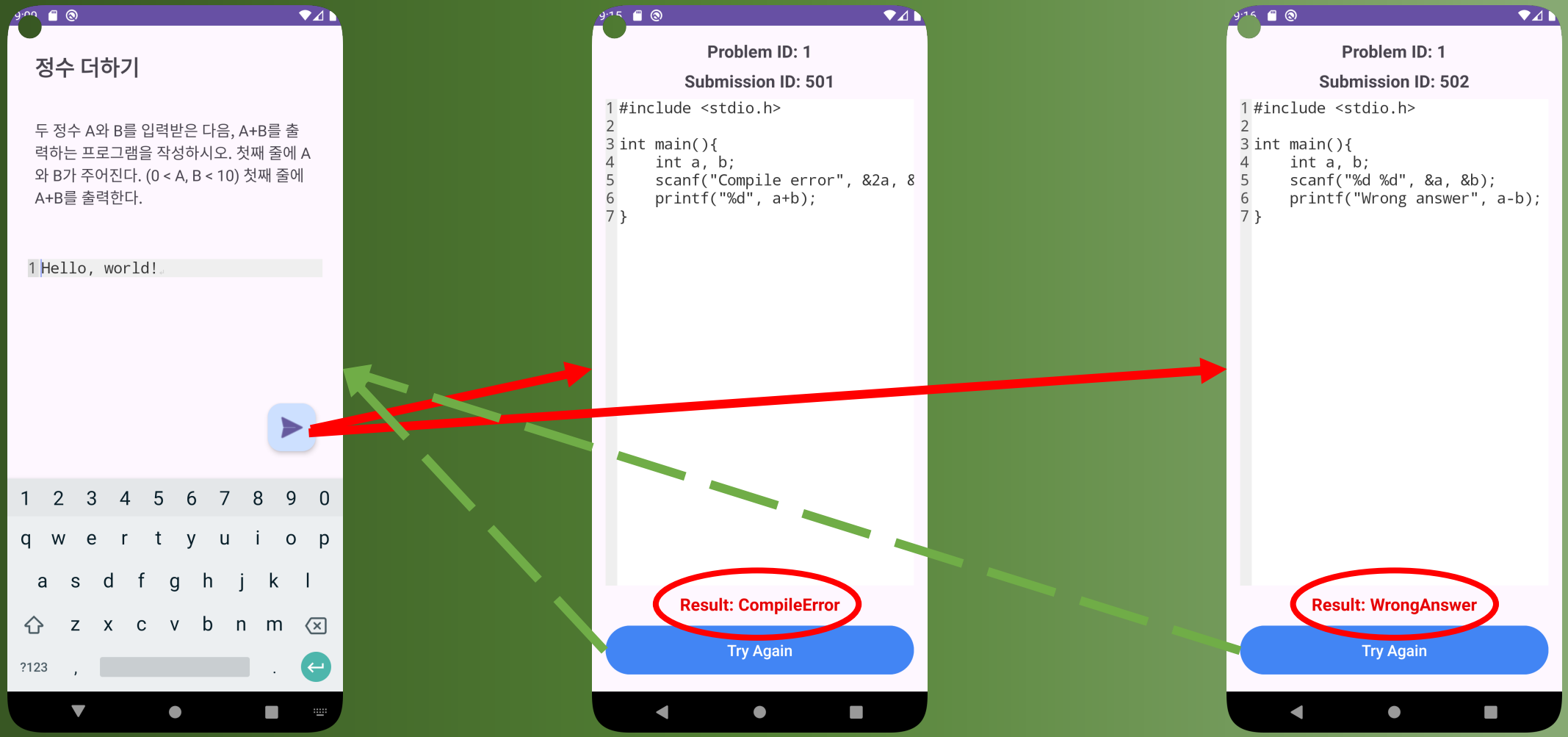


ActivitySolveProblem

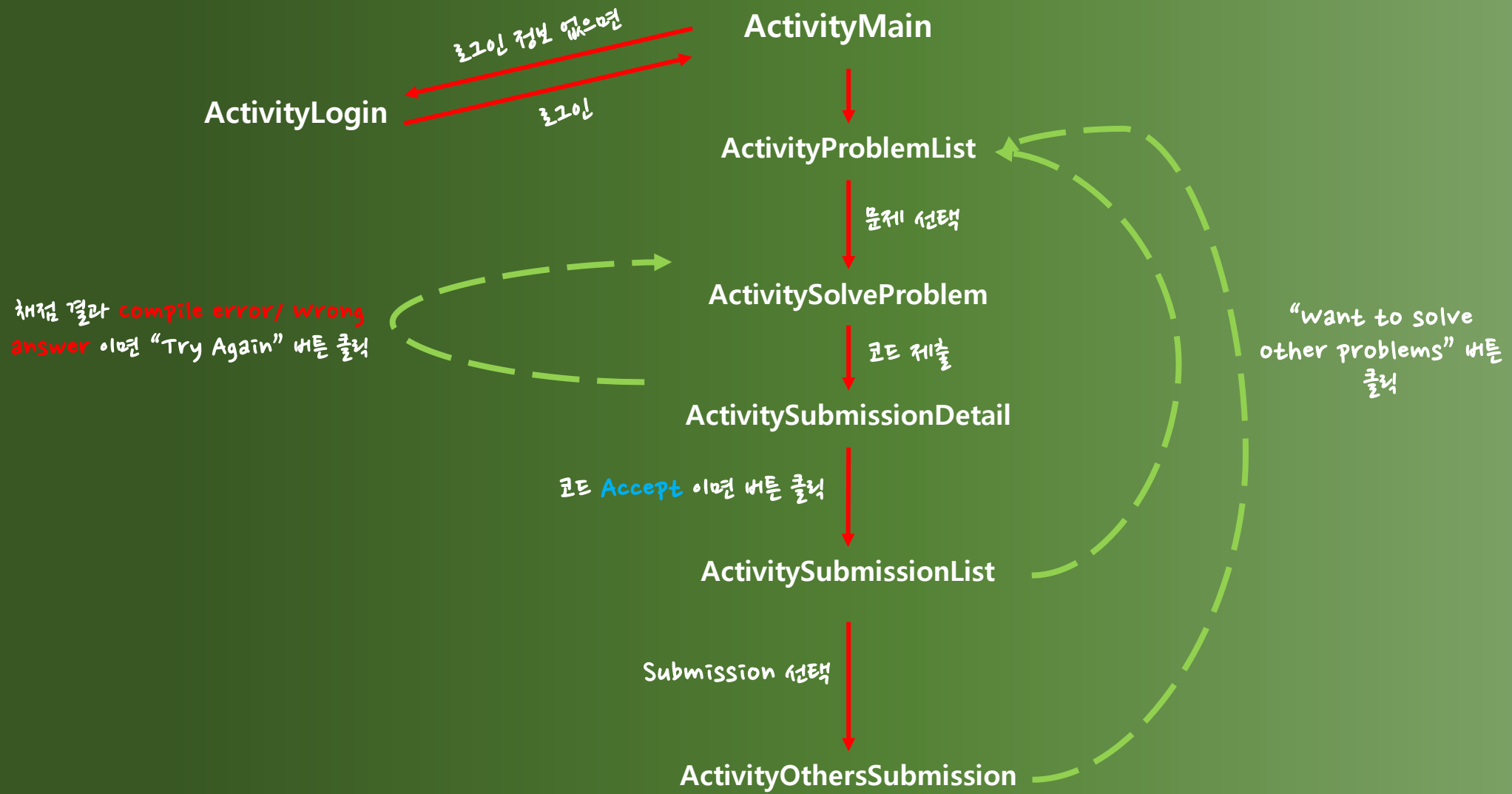
User Flow – 답이 맞는 경우



User Flow – 답이 틀린 경우

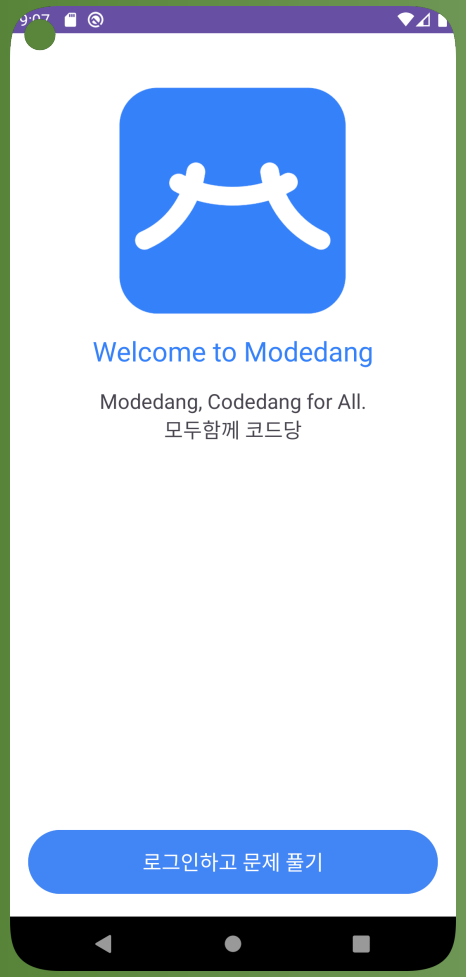


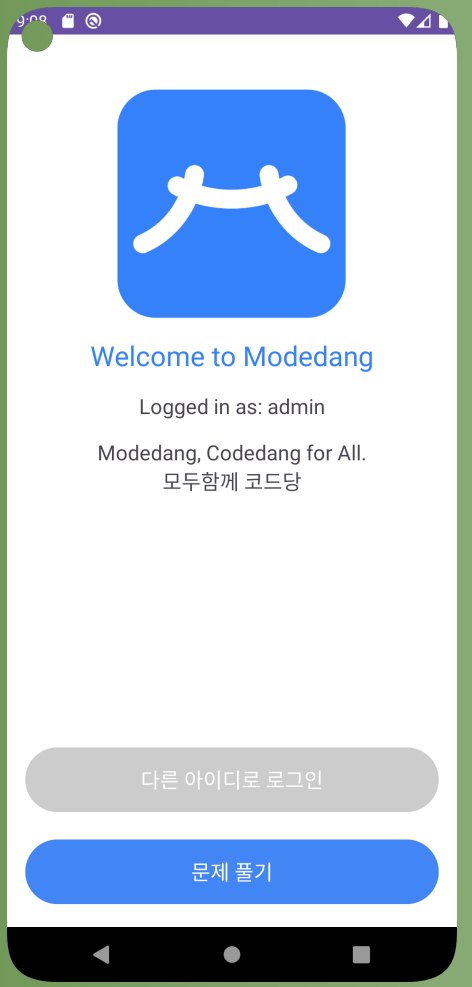
User Flow 전체도



UI / UX – ActivityMain

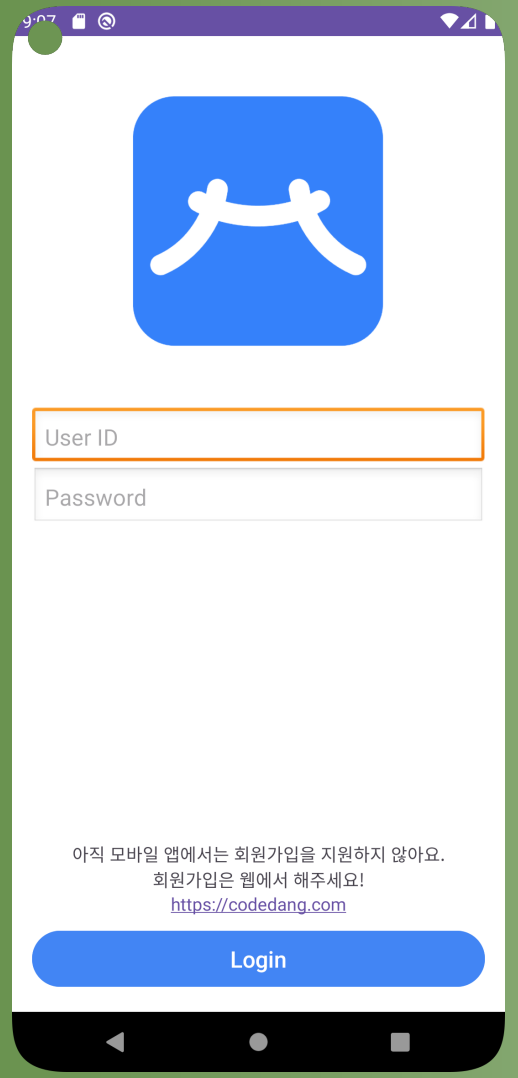
- 메인 화면
  - 모두당 로고, 로그인 정보 표시
  - 로그인 X 시 - 로그인 버튼을 눌러 로그인 가능
  - 로그인 O 시 - 다른 아이디로 로그인하거나 문제를 풀러갈 수 있음





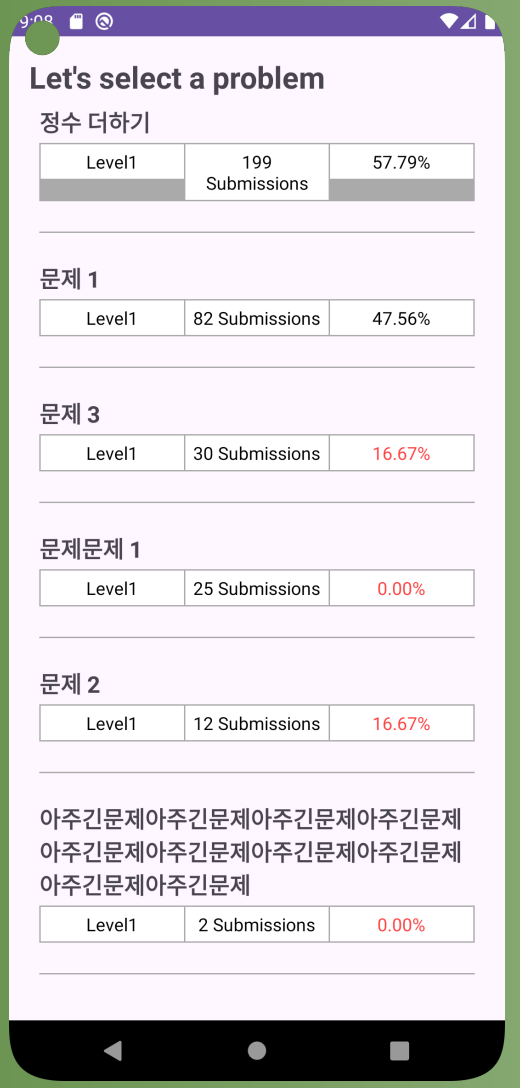
UI / UX - ActivityLogin

- 로그인
  - 따로 화면을 넘어가지 않고도 키보드에서 User ID (enter) - Password (enter)만 해도 로그인이 되도록 구현!
  - 한번 로그인을 하면 JWT Token을 앱 내부 DB(Shared Preference)에 저장하여 다시 로그인할 필요 없음



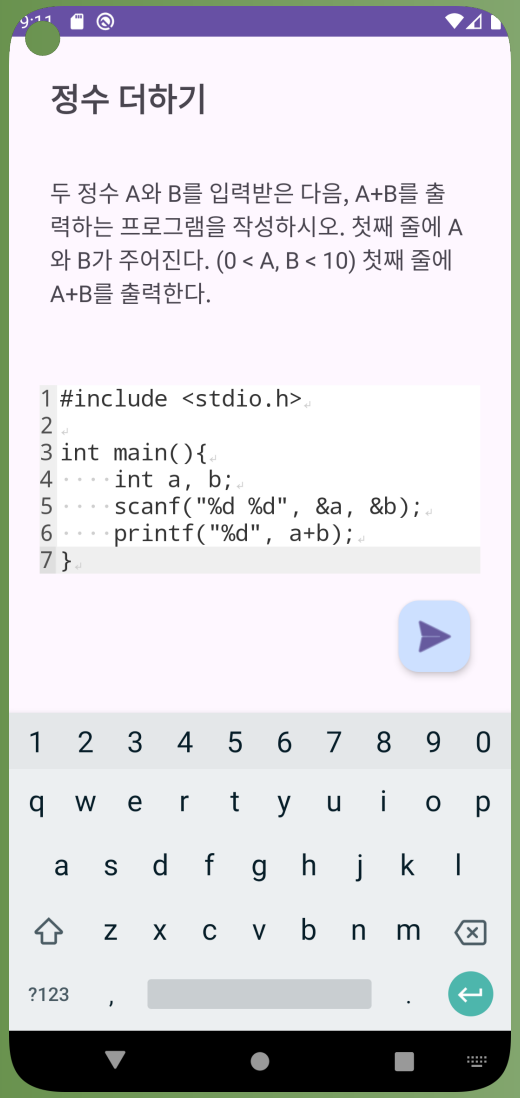
UI / UX - ActivityProblemList

- 문제 선택
  - 문제 이름, 난이도, 제출 횟수, 정답률까지만 간추려서 표시
  - 가시성 있게 표시하여 사용자가 쉽게 무슨 문제가 있는지 파악할 수 있음!



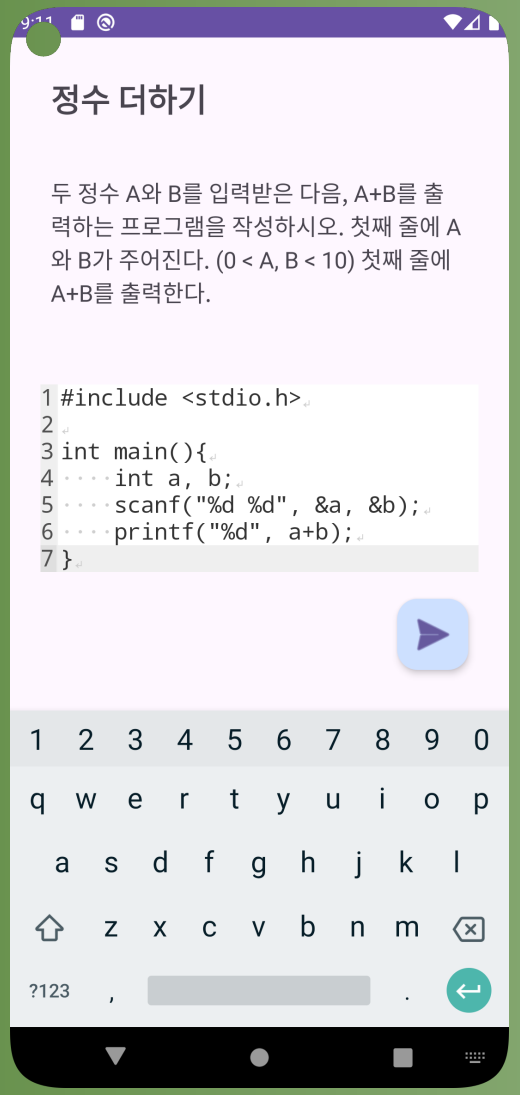
UI / UX - ActivitySolveProblem

- 문제 정보, Code Editor
- 화면 위쪽은 문제 정보, 화면 아래쪽은 코드 에디터
- 각자 Scroll 가능하게 구현
- 문제 정보: 문제 제목, Description, Input/Output 정보 표시



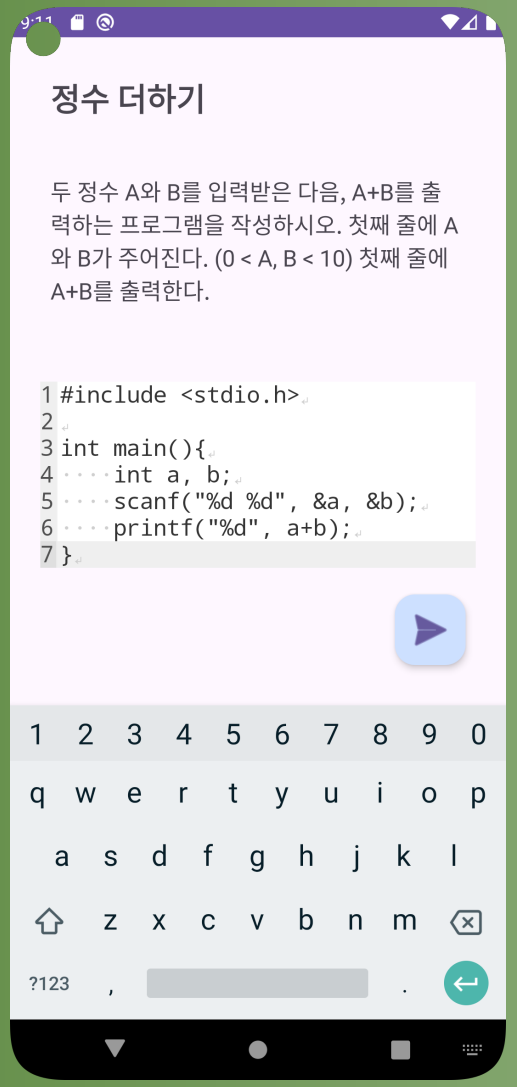
UI / UX - ActivitySolveProblem

- Code Editor의 경우 오픈소스인 sora-editor를 사용.
- 사용 이유: 안드로이드 코드 에디터 오픈소스가 많이 없음에도 Star가 900+ / 모바일임에도 많은 기능 제공



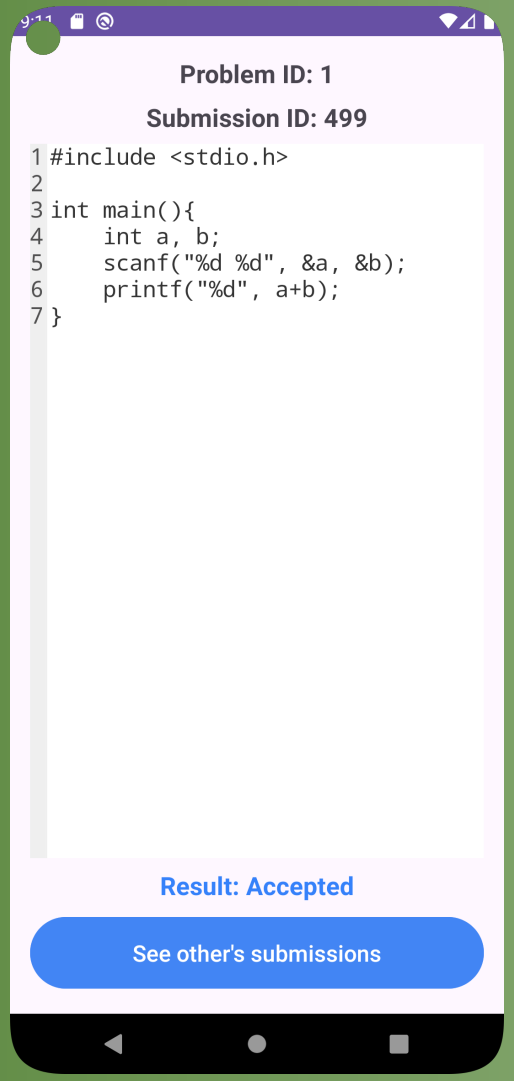
UI / UX - ActivitySolveProblem

- 최대한 코드 에디터를 크게 보여줄 수 있도록 하는 데에 초점.
- 제출 버튼도 다른 액티비티와는 다르게 작게 설정하였고, 코드 에디터 위에 뜨도록 구현



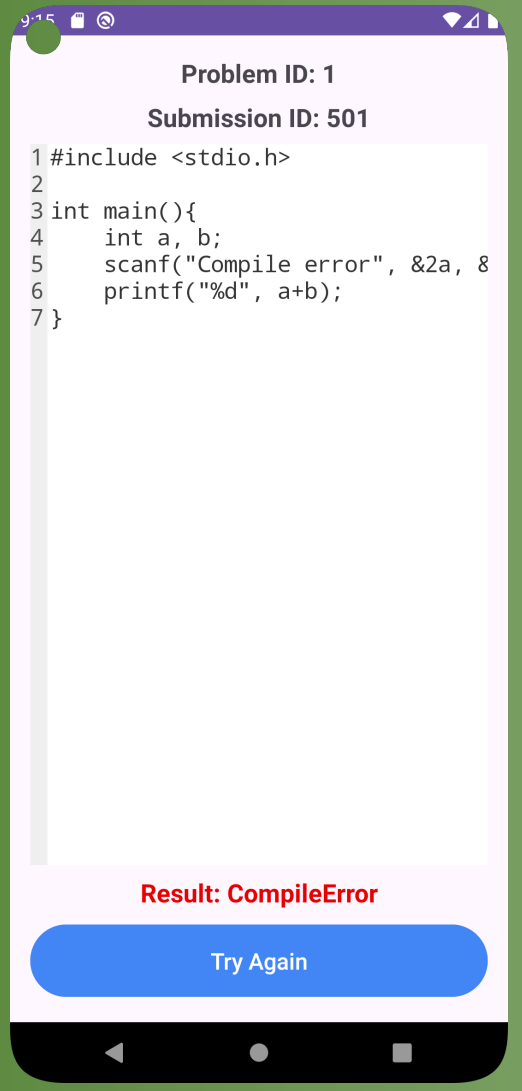
UI / UX – ActivitySubmissionDetail

- 문제 ID, Submission ID, 제출한 코드, 채점 결과 확인 가능
- 채점 결과가 맞았을 경우, 해당 문제의 다른 Submission을 보러 갈 수 있음



UI / UX – ActivitySubmissionDetail

- 채점 결과가 틀렸을 경우, Try Again 버튼을 통해서 다시 문제를 풀러 갈 수 있음.

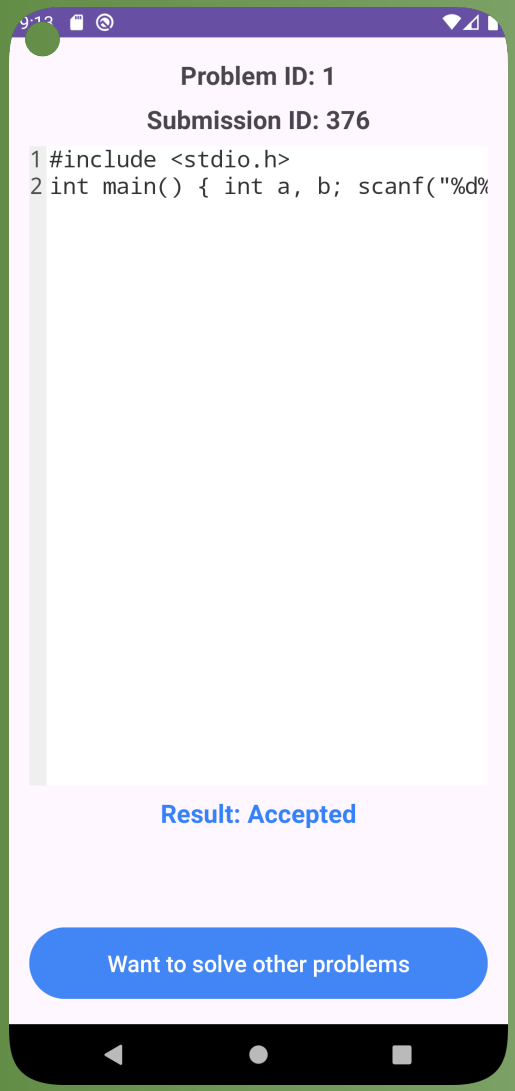


UI / UX – ActivitySubmissionList

- 다른 모든 사람의 Submission들을 표 형식으로 볼 수 있음.
- Submission 클릭 시 ActivityOthersSubmission으로 넘어감
- Submission을 100개씩 불러오고, 필요한 경우 Load More 버튼을 눌러 100개씩 더 가져올 수 있음.
- 화면 하단의 버튼을 누르면 다른 문제를 풀러갈 수 있음

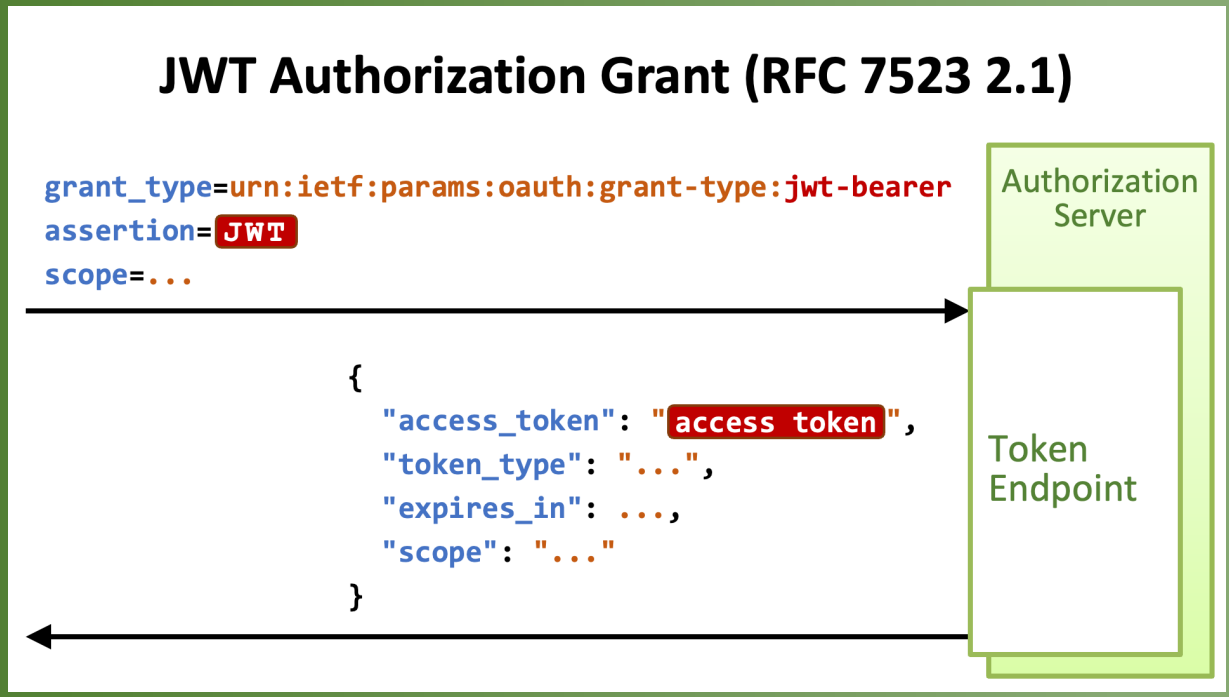
UI / UX – ActivityOthersSubmission

- ActivitySubmissionList에서 특정 Submission을 클릭하면 해당 Submission의 자세한 정보를 볼 수 있음.



implementation Challenge

- 1. 로그인 / 세션 구현
  - 어떻게하면 JWT Token을 여러 액티비티에서 쉽게 사용할 수 있을까 고민하였음.
  - 코드 제출, 다른 사람 Submission 보기 등의 기능은 아무나 할 수 없어야 하기에 모든 액티비티에서 JWT 사용을 해야했음.



출처: Authlete
https://www.authlete.com/developers/jwt_authorization_grant/

implementation Challenge

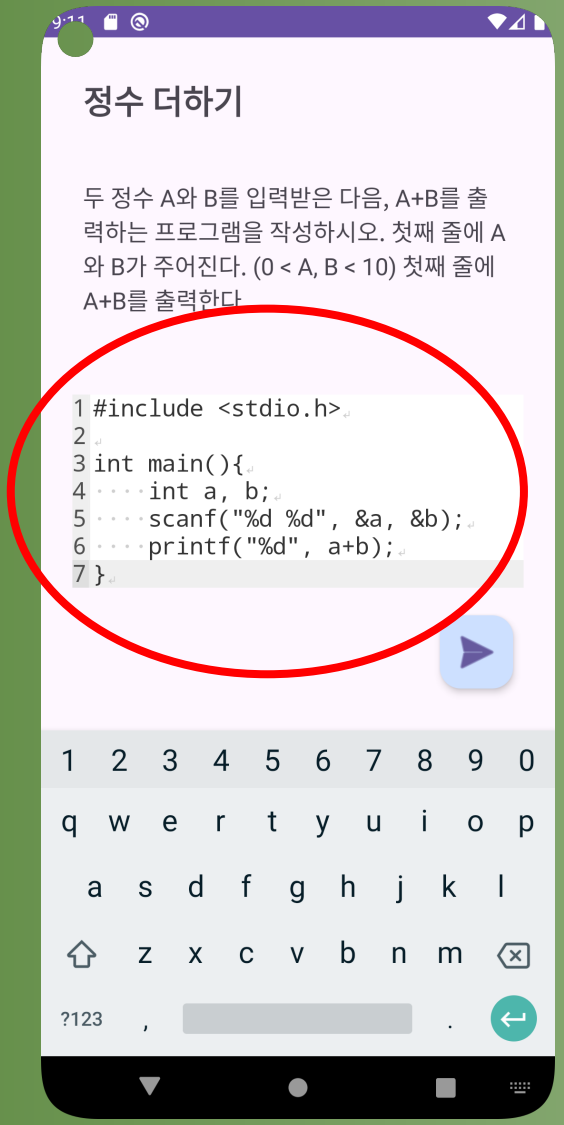
- 1. 로그인 / 세션 구현
  - SharedPreference를 사용하여 앱 로컬 DB에 JWT Token과 cookie를 저장.
  - 로그인 시에 JWT Token과 cookie를 받아서 SharedPreference에 저장함. 앱을 실행할 때마다 백엔드 서버에 session을 reissue하여 유효한 토큰값인지 확인받음.

```
if (response.isSuccessful) {
    val authToken = response.header( name: "authorization")
    val setCookie = response.header( name: "set-cookie")

    // Store tokens and username globally (SharedPreferences used here as an example)
    val sharedPreferences = getSharedPreferences( name: "AppPrefs", Context.MODE_PRIVATE)
    with(sharedPreferences.edit()) { this: SharedPreferences.Editor!
        putString("AUTH_TOKEN", authToken)
        putString("COOKIE", setCookie)
        putString("USERNAME", email) // 저장한 username
        apply()
    }
```

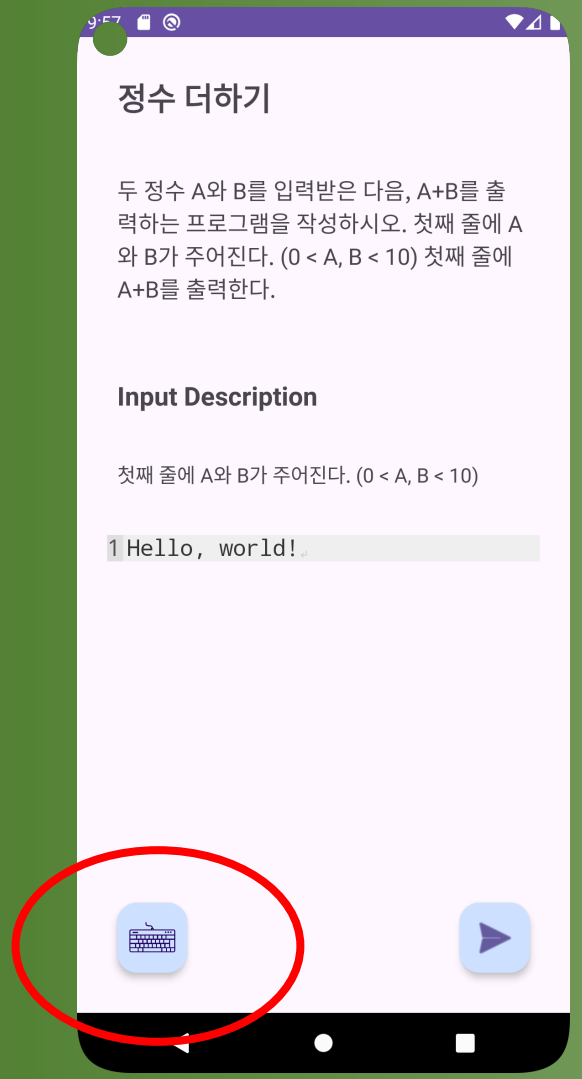
implementation Challenge

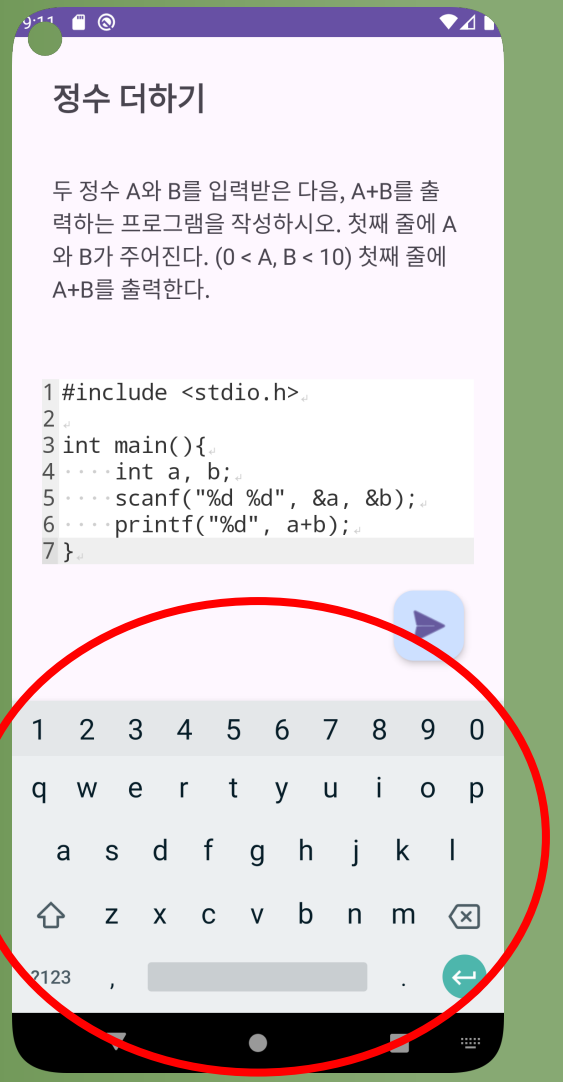
- 2. 코드 에디터 구현
  - 코드 에디터를 직접 구현하는 것은 굉장히 어려운 일이기에, 오픈소스를 사용하였음.
  - 생각보다 코드 에디터 오픈소스가 많이 없어서 그나마 제일 Star 수가 많은 Sora-editor 사용.
  - Sora editor마저도 잘 maintain되고 있지 않고 문서화도 부실하여 사용하기 어려웠음.



implementation Challenge

- 2. 코드 에디터 구현
  - 특히, 코드 작성 중 소프트 키보드를 넣어 버릴 경우, 다시 코드 에디터를 터치하여도 소프트 키보드가 나오지 않는 현상이 발생 하였음
  - 키보드가 들어가 있을 때를 감지하여 키보드를 꺼내는 버튼을 화면 왼쪽 아래에 넣어서 키보드를 강제로 보이도록 구현하였음.
  - 키보드가 보이는 경우에는 이 버튼은 display되지 않음.





implementation Challenge

- 2. 코드 에디터 구현
  - 코드 에디터의 기본적인 기능 중 하나인 Syntax Highlighting 기능을 사용하려고 하였음.
  - Documentation도 따라해보고, 공식 예제도 따라해보았으나 계속해서 문제 발생.
  - 결론적으로는, API 29에서는 현재 불가능함
  - 사용하고 있는 sora-editor가 코드 하이라이팅을 위해서 textmate라는 라이브러리를 사용하는데, textmate가 API 33 이상에서 동작함.
  - 이에 따라 API 29에서 textmate를 사용하려면 desugaring이라는 작업을 해주어야 하는데, sora-editor가 이 desugaring을 하면 textmate와 상호작용할 때 버그가 생김
  - 현재 hotfix로 고쳐진 상태이지만, release를 하고 있지 않아서 사용할 수가 없어서 포기하였음.

implementation Challenge

- 3. 사용자 Flow 단일화, UI 단순화
  - 모바일 앱에서는 유저 로직을 최대한 단일화시키는 것이 사용성에 있어 좋을 것이라고 판단
  - 최소한의 버튼을 사용하여 미리 사용자가 할 수 있는 행동의 범위를 정해주었음
  - 각 액티비티가 보여주는 정보를 최소화하여 사용자가 쉽게 사용 가능