포팅 메뉴얼

1.1 Frontend

▼ Android

- compileSDK = 35

```
implementation(platform(libs.androidx.compose.bom))

    implementation(libs.androidx.core.ktx)
    implementation(libs.androidx.lifecycle.runtime.ktx)
    implementation(libs.androidx.activity.compose)
    implementation(libs.androidx.ui)
    implementation(libs.androidx.ui.graphics)
    implementation(libs.androidx.ui.tooling.preview)
    implementation(libs.androidx.material3)
    implementation(libs.androidx.material.icons.extended)
    implementation(libs.androidx.navigation.compose)
    implementation(libs.material3)

    implementation(libs.accompanist.systemuicontroller)
    implementation(libs.androidx.core.splashscreen)

    // Coil
    implementation(libs.coil.compose)
    implementation(libs.coil.gif)

    // ML Kit Text Recognition
    implementation(libs.play.services.mlkit.text.recognition)
    implementation(libs.text.recognition.korean)

    // Orbit MVI
    implementation(libs.orbit.core)
    implementation(libs.orbit.viewmodel)
    implementation(libs.orbit.compose)
    implementation(libs.androidx.work.runtime.ktx)
```

```
    implementation(libs.androidx.compose.material)
    implementation(libs.androidx.lifecycle.process)
    implementation(libs.androidx.browser)
    testImplementation(libs.orbit.test)

    // Kotlin Serialization & Immutable Collections
    implementation(libs.kotlinx.serialization.json)
    implementation(libs.kotlinx.collections.immutable)

    // Hilt
    implementation(libs.hilt.android)
    ksp(libs.hilt.android.compiler)
    ksp(libs.hilt.ext.compiler)
    implementation(libs.hilt.navigation.compose)

    // Testing
    testImplementation(libs.junit)
    androidTestImplementation(libs.androidx.junit)
    androidTestImplementation(libs.androidx.espresso.core)
    androidTestImplementation(libs.androidx.ui.test.junit4)
    debugImplementation(libs.androidx.ui.tooling)
    debugImplementation(libs.androidx.ui.test.manifest)
    testImplementation(libs.kotlinx.coroutines.test)
    testImplementation(libs.turbine)
    testImplementation(kotlin("test"))

    // Retrofit + Okhttp + Gson
    implementation(libs.retrofit)
    implementation(libs.converter.gson)
    implementation(libs.okhttp)
    implementation(libs.logging.interceptor)
    implementation(libs.gson)

    // DataStore
    implementation(libs.androidx.datastore.preferences)

    // detekt
    detektPlugins(libs.detekt.formatting)
```

```
    // MockK
    testImplementation(libs.mockk)
    testImplementation(libs.mockk.android)

    // Lottie
    implementation(libs.lottie.compose)

    implementation(libs.androidx.navigation3.runtime)
    implementation(libs.androidx.navigation3.ui)
    implementation(libs.androidx.lifecycle.viewmodel.navigation3.androi
d)

    // Firebase
    implementation(platform(libs.firebase.bom))
    implementation(libs.firebase.messaging)
    implementation(libs.firebase.analytics)
```

1.2 Backend

▼ Spring

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-data-
jpa'
    implementation 'org.springframework.boot:spring-boot-starter-secur
ity'
    implementation 'org.springframework.boot:spring-boot-starter-oauth
2-resource-server'
    implementation 'org.springframework.boot:spring-boot-starter-must
ache'
    implementation 'org.springframework.boot:spring-boot-starter-mail'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.springframework.boot:spring-boot-starter-actua
tor'
    implementation 'io.micrometer:micrometer-registry-prometheus'
    implementation 'org.flywaydb:flyway-core:10.13.0'
    implementation 'org.flywaydb:flyway-database-postgresql:10.13.0'
```

```
// implementation 'com.vladmihalcea:hibernate-types-60:2.21.1'

    implementation 'org.springframework.cloud:spring-cloud-starter-openfeign'
    implementation platform('org.springframework.cloud:spring-cloud-dependencies:2025.0.0')
    implementation 'io.jsonwebtoken:jjwt-api:0.11.5'
    implementation 'net.logstash.logback:logstash-logback-encoder:7.4'
    implementation 'org.springframework.boot:spring-boot-starter-data-redis'
    implementation 'com.google.firebase:firebase-admin:9.2.0'
    implementation 'org.springframework.boot:spring-boot-starter-cache'
    implementation 'org.apache.commons:commons-pool2:2.11.1'

    compileOnly 'org.projectlombok:lombok'

    runtimeOnly 'org.postgresql:postgresql'
    runtimeOnly 'io.jsonwebtoken:jjwt-impl:0.11.5'
    runtimeOnly 'io.jsonwebtoken:jjwt-jackson:0.11.5'
    runtimeOnly 'io.micrometer:micrometer-registry-prometheus'

    annotationProcessor 'org.projectlombok:lombok'

    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    testImplementation 'io.rest-assured:rest-assured'
    testImplementation 'org.springframework.restdocs:spring-restdocs-restassured'
    testImplementation 'org.springframework.security:spring-security-test'
    testImplementation 'com.h2database:h2:2.2.220'

    testImplementation 'org.testcontainers:testcontainers:1.19.0'

    testRuntimeOnly 'org.junit.platform:junit-platform-launcher'

    implementation 'net.datafaker:datafaker:2.4.4'
```

```
    implementation 'org.springframework.boot:spring-boot-starter-webflux'
}
```

▼ Java

OpenJDK v21.0.7 HotSpot VM

1.3 Server

- Ubuntu v22.04
- Docker v27.5.1
- docker-compose v2.39.1

1.4 Database

- PostreSQL v15.3
- Redis v7.2.4

1.5 형상/코드관리

- SSAFY Gitlab
- Jira
- Notion

2. 환경변수

2.1 Frontend

▼ Android

```
BASE_URL=https://i13d202.p.ssafy.io
BITLY_TOKEN=36f08e0fcee88e331a4a4b3d57c1691cb2cef6d0
```

2.2 Backend

▼ Backend

```
JWT_SECRET=3mUQmfXezNKncDhYTYi2vK85iaHp589H;
MAIL_PASSWORD=ndqcchlcpyfwynox;
SPRING_PROFILES_ACTIVE=local
```

2.3 외부API

- 알라딘 API

  도서 정보를 가져오는 API

  ```
  aladin:
    ttbkey: ttbhongsbien1441001
  ```

- GMS API

  DALL-E 모델로 이미지 생성을 하는 API

  ```
  openai:
    key: S13P11D202-5351d94c-0223-4f14-9467-2420945e6284
    url: https://gms.ssafy.io/gmsapi/api.openai.com/v1
  ```

3. 덤프 파일

```
```

용량 이슈로 Gitlab에 첨부

4. 시나리오

▼ 시연 시나리오

1. 홈 화면 (20초)

"Glim 앱의 홈 화면입니다.

홈 화면에서는 인기 글귀와 도서 그리고 관리자 페이지에서 설정한 추가 큐레이션을 확인할 수 있습니다."

- 인기 글귀 카드 클릭 → 글림 상세 화면으로 이동
- **"각 글귀를 클릭하면 글귀 숏폼 화면으로 이동합니다."**
- 뒤로가기로 홈 화면 복귀
- 도서 섹션의 책 표지 클릭 → 도서 상세 화면으로 이동
- **"도서를 클릭하면 해당 도서의 상세 정보를 볼 수 있습니다."**
- 뒤로가기로 홈 화면 복귀

2. 글림 탭 (30초)

"글림 탭에서는 다양한 글귀들을 피드 형태로 감상할 수 있습니다."

- 스크롤 3-4번 하며 여러 글귀 보기
- 좋아요 버튼 클릭
- **"마음에 드는 글귀에는 좋아요를 누를 수 있고,**

  상단에 다운로드 버튼으로 갤러리에 저장할 수 있습니다."
- 공유
  - 링크 공유 버튼 클릭
  - **"글귀 앱링크를 공유하거나 이미지를 인스타그램 스토리에 바로 올릴 수 있습니다."**
  - 인스타 스토리 공유 버튼 클릭
- 도서 정보 클릭 → 도서 상세 화면으로 이동
- **"글귀의 출처인 도서 정보를 클릭하면 해당 도서의 상세 페이지로 이동합니다."**
- 뒤로가기로 글림 탭 복귀

3. 생성 화면 (60초)

"생성 탭에서는 직접 글귀 카드를 만들어 업로드할 수 있습니다."

- **"먼저 업로드할 글귀의 도서를 추가해보겠습니다."**
- 도서 검색 후 선택
- **“다음으로 글귀를 추가할 때 텍스트를 직접 입력할 수도 있고”**
- 글귀 추가 버튼 클릭
- **"이미지에서 텍스트를 인식하는 기능도 있습니다."**

- 텍스트 인식 시연
- **"다음으로 배경 이미지를 설정해보겠습니다."**
- 배경 이미지 선택 화면 진입
- **"갤러리에서 이미지를 선택하거나, AI 이미지 생성 기능을 사용할 수 있습니다."**
- AI 이미지 생성 버튼 클릭
- **"AI 이미지 생성은 약 10-15초 정도 소요됩니다."**
- (이미지 생성 대기)
- **"AI가 글귀에 어울리는 이미지를 생성해주었네요."**
- **"마지막으로 스타일을 조정하며 글귀를 완성하면."**
- 텍스트 스타일, 배치, 이미지 투명도, 스케일 편집 시연
- **"이제 완료 버튼을 눌러서 업로드할 수 있습니다."**

4. 검색 화면 (30초)

"검색 탭에서는 원하는 키워드로 도서나 글귀를 찾을 수 있습니다."

- **"윤동주'를 검색해보겠습니다."**
- '윤동주' 검색어 입력
- 검색 결과에서 도서 확인
- **"윤동주 시인과 관련된 도서들이 검색되었네요."**
- **"이번에는 글귀 탭에서 특정 키워드가 포함된 글귀를 찾아보겠습니다."**
- 글귀 탭으로 전환
- 키워드 검색 (예: '사랑', '희망' 등)
- **"해당 키워드가 포함된 다양한 글귀들을 확인할 수 있습니다."**
- 검색된 글귀 클릭 → 글림 상세 화면으로 이동
- **"마음에 드는 글귀는 클릭하여 글림화면으로 이동할 수 있습니다."**

5. 마이페이지 (35초)

"마이페이지에서는 개인 활동 내역을 확인할 수 있습니다."

- **"앞서 생성한 글귀를 '업로드한 글림' 섹션에서 확인해보겠습니다."**

- 업로드한 글귀 확인
- **"좋아요를 누른 글귀들도 '좋아요한 글림' 섹션에서 모아볼 수 있습니다."**
- 좋아요한 글귀 탭 확인
- **"이외에도 프로필 설정, 알림 설정 등 다양한 개인화 기능을 이용할 수 있습니다.**
- 개인정보 변경 탭, 설정 탭 확인

6. 잠금화면 (20초)

“마지막으로 잠금화면 기능입니다."

- **“잠금화면에서도 글귀 피드를 바로 볼 수 있으며,”**
- **“글림 화면과 마찬가지로 좋아요와 이미지 저장 모두 가능합니다.”**
- **“우측 하단 버튼을 끌어 당기면 글귀의 상세 페이지로 바로 이동할 수도 있습니다.”**