



# Java 21 · Spring Boot 기반 채널계 Monolith + 계정계· 대외계 연계 시뮬레이터 구축 명세서

## 범위와 현실성 가정

이 문서는 “채널계(고객 접점) Monolith + 계정계(Core Banking) + 대외계(FEP) 연계”를 국내 금융사에서 흔히 이야 기되는 **층위(채널/계정/정보/대외)**의 개념에 맞춰, 취업 준비생 포트폴리오 수준에서 직접 구현 가능한 형태로 구체화한 **프로젝트 명세다.** ①

국내 지급결제 관점에서, 고객이 인터넷·모바일뱅킹으로 수행하는 이체 등은 금융기관 내부의 “주전산(핵심) 시스템”을 통해 처리되고, 필요 시 외부 공동망(예: 결제원, 중앙은행 등)을 거쳐 타 기관과 상호 연동된다는 식의 큰 흐름이 교육 자료에서 설명된다. 또한 정보계·대외계 같은 용어도 “고객·계좌 관련 정보 활용(정보계)”과 “대외 공동망 연계(대외 계)”라는 취지로 정리된다. ②

다만 실제 금융망(타행이체 공동망 등)과 실기관 연계는 법·규정·계약·보안 요구사항이 매우 높고 개인 프로젝트로 직접 붙는 것은 현실적으로 어렵다. 따라서 본 프로젝트는 **대외계/FEP를 ‘시뮬레이터’로 구현하고**, “전문 변환·세션/재전송·에러코드 매핑” 같은 **구조/패턴을 학습**하는 데 목적을 둔다. ③

기술 제약과 선택 근거는 다음과 같이 두었다.

- Backend: Java 21 + Spring Boot 3.x (Spring Boot 3.2 계열 문서 기준, Java 17이 베이스라인이며 Java 21까지 호환으로 안내됨) ④
- Session/Cache: Redis (Spring Session으로 HttpSession을 Redis에 저장하는 방식이 공식 가이드에 존재) ⑤
- RDB: MySQL(InnoDB) (ACID 트랜잭션과 row-level locking, 격리수준 등을 공식 문서가 명시) ⑥
- “Kafka/대규모 MSA”는 피한다: 대신 (1) 단일 채널계 모듈리식, (2) 계정계·대외계는 연계 구조를 보여주기 위한 **보조 프로세스(시뮬레이터)** 정도로 최소화한다. (이 부분은 설계 선택이며, 현실 금융권의 다양한 형태를 단정하지 않는다.)

관련 기관·표준은 문맥상 다음을 기준으로 참조한다: 한국은행 ⑦, 금융결제원 ⑧, Internet Engineering Task Force ⑨, OWASP ⑩, W3C ⑪. ⑫

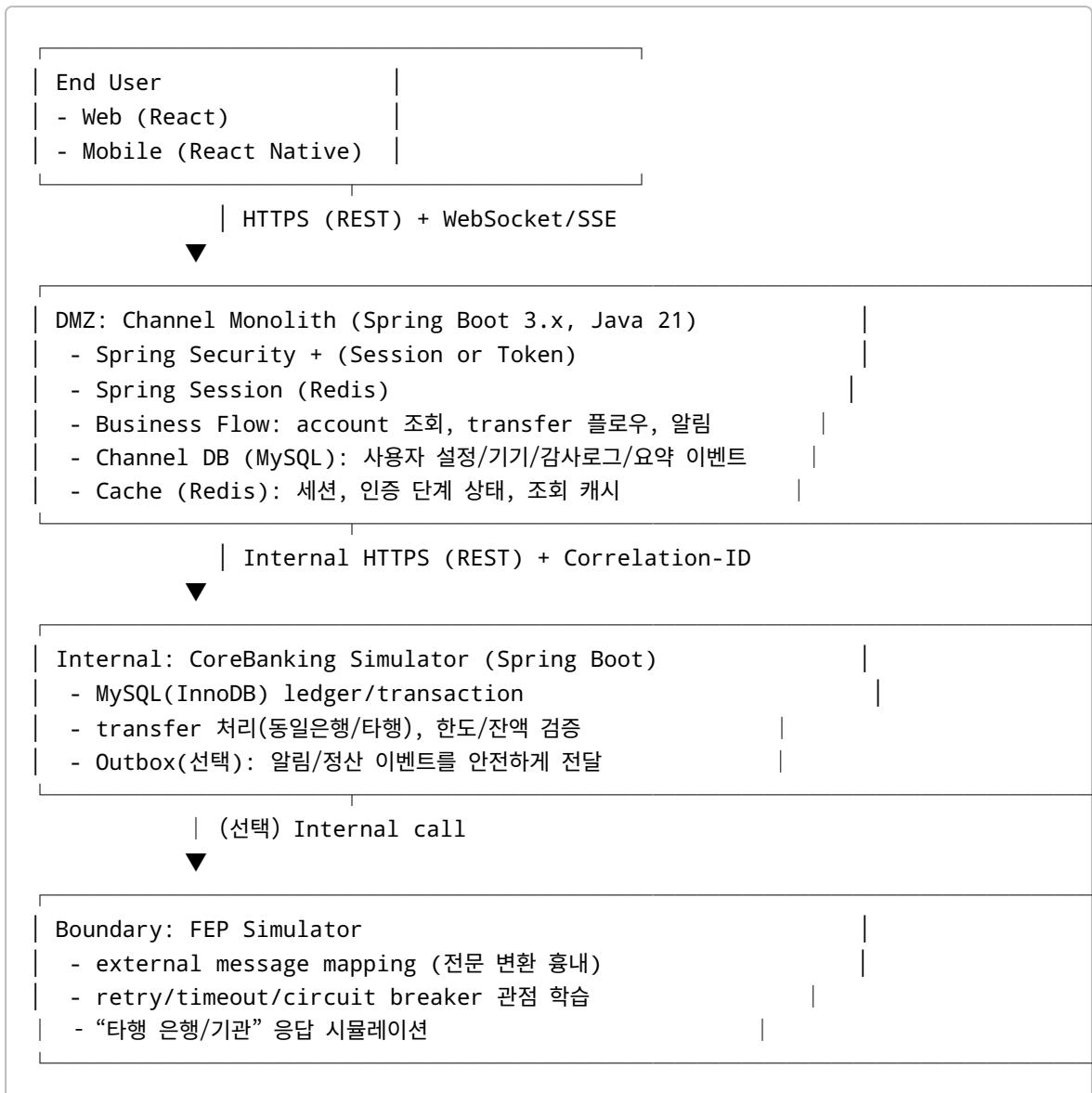
## 목표 아키텍처 설계

프로젝트는 “현실 금융사의 네트워크 분리/역할 분리”를 학습하기 위해 **논리적 3영역**을 둔다.

- 고객 접점: Web(React), Mobile(React Native)
- DMZ: 채널계 Monolith(인증/세션/화면·API/알림/캐시/감사로그)
- 내부망: 계정계(CoreBanking) 시뮬레이터 + 대외계(FEP) 시뮬레이터

“대외 공동망을 거쳐 타 기관과 연결되는 흐름”이라는 큰 그림은 금융 IT 교육 자료에서 언급되는 구조를 따르되, 실제 연계는 시뮬레이터로 대체한다. ⑬

아키텍처(학습용 구현 버전)는 아래처럼 제안한다.



핵심 의도는 다음과 같다.

- 채널계는 **고객 경험과 보안 세션**을 담당하고, 계정계는 **정합성/원장 반영**을 담당한다(역할 분리). ①
- 계정계의 데이터 무결성을 위해 MySQL InnoDB의 트랜잭션·락·격리수준 개념을 실제로 사용한다. ⑥
- “관측 가능성(Observability)”은 취업 포트폴리오에서 강점이 되므로, 최소 구성으로 **메트릭/트레이싱/보안** 로깅을 넣는다(Actuator+Micrometer, TraceContext). ⑯

## 채널계 Monolith 상세 명세

채널계는 “웹/앱/외부 API 등 고객이 직접 호출하는 경계”에서 세션·인증·화면/DTO 변환·알림을 책임진다. ⑮  
본 프로젝트에서는 “실제 금융권 채널계”를 그대로 복제하려 하기보다, **면점에서 설명 가능한 단위**로 기능을 쪼개 구현 가능한 범위를 명확히 한다.

## 기능 범위

채널계가 제공할 ‘대외’ 기능은 다음을 최소 기능(MVP)로 둔다.

- 회원/고객(간소화)
- 가입(샘플), 로그인, 로그아웃, 세션 만료, 강제 로그아웃(관리자)
- 계좌 조회
- 내 계좌 목록/잔액/최근 거래내역(요약)
- 이체
- 동일은행 이체(내부 이체) + 타행 이체(대외계 경유 시뮬레이터)
- 2단계 확인(입력→검증→재인증→실행→결과)
- 알림
- 이체 결과/입금/보안 이벤트를 WebSocket 또는 SSE로 전달(학습용)

이 구성은 “금융기관 서비스가 내부 처리 후 외부 공동망 연계로 이어질 수 있음”이라는 큰 흐름을 ‘이체 유즈케이스’ 하나로 집약해 보여주기 위함이다. <sup>13</sup>

## 코드/모듈 구조

Monolith이지만 MSA로 “쪼개지기 쉬운 모놀리식”을 목표로 한다. 권장 멀티모듈 구조 예시는 아래다(단일 레포, Gradle/Maven 멀티모듈).

- channel-app : Spring Boot entry, configuration, security, actuator
- channel-auth : 인증/세션/리스크 기반 재인증(거래인증) 유즈케이스
- channel-account : 계좌조회 유즈케이스 + DTO 조립
- channel-transfer : 이체 플로우(상태머신) + 검증
- channel-notification : WS/SSE, 디바이스 토큰, 알림 템플릿
- channel-common : 에러코드, 로깅 마스킹, 공통 응답 포맷

이렇게 모듈을 분리하면, 후일 혹시 분리하고 싶을 때도 “모듈 경계”가 설명 가능해진다(설계 선택).

## 인증·세션·거래 재인증

**세션 저장소:** Spring Security는 기본적으로 인증 정보를 HTTP 세션에 저장하며, 수평 확장 시 세션을 캐시/DB 등에 저장하도록 커스터마이징할 수 있음을 문서에서 언급한다. 따라서 본 프로젝트는 “세션 기반 채널계”를 택하고, Spring Session + Redis로 세션을 외부화한다. <sup>16</sup>

- Web(React): JSESSIONID (HttpOnly/Secure/SameSite) 기반 세션
- Mobile(React Native): 쿠키 기반 또는 X-SESSION-ID 헤더 기반(학습용으로 단순화)

**거래 재인증(2FA/OTP, step-up):** 금융 거래는 로그인만으로 끝나지 않고 “고위험 행위(이체, 설정 변경 등)”에 재인증을 요구하는 것이 일반적인 보안 설계 논리다(OWASP Authentication Cheat Sheet는 위험 이벤트 후 재인증을 강조). <sup>17</sup>

구현은 다음 중 하나를 선택한다.

- (권장) TOTP 기반 거래 OTP: TOTP는 HOTP를 시간 기반으로 확장한 OTP 알고리즘이며 RFC로 표준화되어 있다. <sup>18</sup>
- (대안) “간편 비밀번호(6자리)” + Redis에 시도 횟수/클다운 저장(학습용)

**CSRF:** 쿠키 기반 세션을 쓰는 Web UI는 CSRF 방어가 필요하며, OWASP CSRF Cheat Sheet는 ‘Signed Double-Submit Cookie’ 같은 패턴을 권장한다. (Spring Security의 CSRF 기능을 활용해도 된다.) <sup>19</sup>

## API 명세(예시)

응답 포맷(일관성):

```
{  
  "success": true,  
  "data": {},  
  "error": null,  
  "traceId": "..."  
}
```

`traceId`는 트레이싱/로그 상관관계를 위한 값이며, TraceContext(예: `traceparent`) 기반으로 확장 가능하다.

20

핵심 앤드포인트(예시):

- `POST /api/v1/auth/login` : 로그인(세션 생성)
- `POST /api/v1/auth/logout` : 로그아웃(세션 무효화)
- `GET /api/v1/accounts` : 내 계좌 및 잔액 요약(캐시 가능)
- `GET /api/v1/accounts/{accountId}/transactions?from&to` : 거래내역
- `POST /api/v1/transfers/prepare` : 이체 입력 검증 + “거래 세션(`transferSessionId`)” 발급(Redis)
- `POST /api/v1/transfers/{transferSessionId}/confirm` : OTP/간편비번 검증
- `POST /api/v1/transfers/{transferSessionId}/execute` : 최종 실행(계정계 출금)
- `GET /api/v1/transfers/{transferId}` : 결과 조회(재조회/재시도 대비)

이 단계적 플로우는 “이체는 한 번의 POST로 끝나지 않고, 검증/인증/확정 같은 단계 상태가 존재한다”는 금융 도메인 특성을 학습하기 위한 의도다(설계 선택).

## 로깅·감사·민감정보 마스킹

보안 로깅은 “무엇을 기록할지”와 “무엇을 기록하면 안 되는지”가 동시에 중요하다. OWASP Logging Cheat Sheet는 로그 보호(변조/삭제/무단접근 방지)와 민감정보 포함 위험을 경고한다. 모바일 관점에서도 OWASP MAS(모바일) 쪽은 “로그에 민감 데이터 삽입” 자체를 취약점으로 분류한다. 21

채널계에서 남길 감사로그(권장 최소):

- 로그인 성공/실패(사유 분류), 세션 생성/만료/강제종료
- 계좌조회(민감 데이터는 마스킹)
- 이체: 준비/인증/실행/실패(에러코드), 타행 여부, 거래번호(내부 식별자)
- 권한 상승/관리자 행위

로그에 직접 남기지 말 것(원칙):

- 세션 토큰/쿠키 값, 비밀번호/OTP, 주민번호 등 PII, 전체 계좌번호(마스킹/토론판화 후 일부만) 22

## 계정계 CoreBanking 시뮬레이터 상세 명세

계정계는 “정합성·원장·거래 이력”이 중심이며, 채널계에서 들어온 요청을 실제로 반영하는 역할을 맡는다. 15  
현실 금융권은 더 보수적 스택/메인프레임도 많지만, 본 프로젝트는 **Spring Boot + MySQL(InnoDB)**로 원장 일관성의 핵심 개념(락/트랜잭션/격리)을 구현해 포트폴리오로 증명한다. 23

### 데이터 모델(권장)

원장/거래 모델은 개념적으로 아래 엔티티를 권장한다.

- `customer` : 고객(식별자, 이름 등 최소)
- `account` : 계좌(계좌상태, 상품유형, 통화, 한도)
- `balance` : 잔액(가용/지급가능/보류금) — 단순화 가능
- `ledger_entry` : 원장 분개(차변/대변, 금액, 기준시각, 참조거래)
- `transfer` : 이체 거래(상태 머신: REQUESTED → AUTHED → POSTED/FAILED)

“원장(ledger\_entry)을 별도로 둔다”는 건 학습용이지만, 트랜잭션 추적·감사에 유리하고, 면접에서 ‘정합성을 어떻게 보장했는지’ 설명하기 쉽다(설계 선택).

### 트랜잭션·락·동시성 제어

MySQL InnoDB는 트랜잭션이 ACID 모델을 따르며(row-level locking, consistent read 등), 동시성/정합성 제어 메커니즘을 제공한다. 24

또한 InnoDB는 SQL 표준의 4개 격리수준을 제공하고, 기본 격리수준이 REPEATABLE READ임을 공식 문서에서 명시한다. 25

이체 처리에서 최소로 지켜야 할 동시성 규칙(구현 규칙):

- 출금 계좌의 잔액 차감은 **하나의 DB 트랜잭션**에서 처리
- 같은 계좌에 대한 동시 출금 경합은 **locking read**(예: `SELECT ... FOR UPDATE`)로 직렬화
- MySQL 공식 문서는 카운터 증가 같은 케이스에서 `FOR UPDATE` locking read 후 업데이트를 예시로 든다.  
동일한 원리로 잔액 업데이트 경합을 통제할 수 있다. 26

트랜잭션 범위 설계에서 중요한 점은 “원장 커밋과 외부 호출을 한 트랜잭션으로 묶지 않는 것”이다. Spring 문서는 원격 호출을 가로지르는 트랜잭션 전파를 기본적으로 지원하지 않으며, 일반적으로 원격까지 트랜잭션을 늘리는 것을 신중히 보라고 언급한다. 따라서 본 프로젝트는 다음 전략을 권장한다. 27

- 내부 이체(동일은행): DB 트랜잭션으로 차감/가산을 한 번에 커밋
- 타행 이체:
  - 1) “외부 송금 요청 생성(REQUESTED)”을 먼저 기록
  - 2) FEP 시뮬레이터 호출(타임아웃/재시도/서킷브레이커 적용)
  - 3) 결과에 따라 “POSTED/FAILED”로 상태 전이 + 원장 반영(또는 보류금 모델로 2단계 반영)

이 구조는 “외부가 실패해도 내부 원장이 꼬이지 않게” 설계하는 학습 목적이다.

### 요청 idempotency(중복 실행 방지)

네트워크 타임아웃이 나면 클라이언트가 재시도할 수 있다. HTTP 표준은 “멱등(idempotent)” 개념을 정의하며, 같은 요청을 여러 번 보내도 서버 상태 효과가 1회와 동일한 성질을 의미한다. 28

하지만 이체 실행은 보통 `POST`(비멱등 가능)로 표현되므로, 앱 레벨에서 **중복 실행 방지 키**가 필요하다(설계 선택).

권장 구현:

- 채널계가 `clientRequestId(UUID)` 를 생성해 계정계로 전달
- 계정계 `transfer` 테이블에 `(clientRequestId, fromAccountId)` 유니크 제약
- 동일 키 재호출 시 “이미 처리된 transferId/상태”를 반환

이 처리는 “재시도에 안전한 금융 API”라는 포인트를 만들기 좋다.

## 대외계 FEP 시뮬레이터 및 연계 규약

대외계(FEP)는 외부 기관/공동망과의 연계를 전담하는 영역으로 설명되며, 교육 자료에서는 이체가 필요할 때 “결제원/중앙은행 쪽을 거쳐 처리되는 흐름”을 언급한다. 13

또한 금융결제원 8 은 (영문 소개에서) 금융 결제 네트워크를 다루는 기관이라는 취지로 안내된다. 29

본 프로젝트의 FEP 시뮬레이터는 “외부 전문/프로토콜”을 그대로 구현하기보다, 다음 3가지를 학습 목표로 삼아 최소 구현한다.

- 메시지 변환(내부 JSON ↔ 외부 전문 포맷 흉내)
- 세션/타임아웃/재전송(재시도) 시나리오
- 장애/지연/부분 성공 케이스의 에러코드 매핑

### FEP 인터페이스(학습용 표준)

계정계 ↔ FEP 시뮬레이터 간 통신은 단순화를 위해 REST로 시작하고(최소 구현), 옵션으로 TCP 전문을 붙인다.

- 기본(권장): REST
- POST /fep/v1/interbank/transfers
- 응답: ACCEPTED, REJECTED, TIMEOUT\_SIMULATED 등 시뮬레이션 코드
- 옵션(심화): TCP 전문(고정 길이 또는 구분자 기반)
- “전문 매핑/프레이밍/재전송” 개념을 체험

### 장애·재시도·회로차단

외부 연계는 항상 성공하지 않으므로, “타임아웃·재시도·fallback” 정책을 코드로 박아두는 것이 학습 가치가 크다. Resilience4j 문서는 CircuitBreaker와 Retry를 조합해 호출을 데코레이션하고, 실패 시 fallback으로 복구하는 예시를 제공한다. 30

권장 정책(학습용 디폴트):

- timeout: 1s~2s
- retry: 최대 2~3회(지수 backoff)
- circuit breaker: 실패율/슬라이딩 윈도우 기준으로 OPEN 전환(기본 설정은 Resilience4j 설정 방식 참고)

31

### 트레이싱·상관관계 ID

연계 시스템을 만드는 포트폴리오에서 “요청 추적”은 강점이다. OpenTelemetry는 서비스 간 컨텍스트 전파를 설명하며, 기본 전파 방식이 W3C TraceContext 헤더를 사용한다고 안내한다. W3C TraceContext는 HTTP 헤더 기반 컨텍스트 전파 표준을 정의한다. 32

최소 구현안:

- 채널계가 `traceparent` 또는 `X-Correlation-Id`를 생성
- 계정계/대외계 호출에 그대로 전달
- 모든 로그/감사로그/에러 응답에 traceId 포함

## 개발·테스트·운영 계획 및 산출물

이 절은 “실제로 프로젝트를 끝까지 완성하고, 면접에서 ‘운영 가능한 시스템’으로 설명”하기 위한 구체 가이드다. “오픈 소스/공식 문서 기반”의 선택지를 최대한 활용한다.

### 로컬 실행 환경

로컬 개발 시 컨테이너로 의존성을 고정하면 재현성이 높다. Docker <sup>33</sup> 문서는 Docker Compose가 단일 YAML(`compose.yml`)로 멀티 컨테이너 앱을 정의/실행한다고 설명한다. <sup>34</sup>

권장 compose 구성:

- MySQL(스키마 2개: `channel_db`, `core_db`)
- Redis
- (선택) Keycloak(SSO 실습용)
- channel-app, corebank-app, fep-sim app

Keycloak은 OpenID Connect/OAuth 2.0/SAML 같은 표준 프로토콜을 지원한다고 공식 사이트에 명시되어 있어, “금융권 SSO/IDP 연계” 느낌을 포트폴리오로 담기 좋다. <sup>35</sup>

### DB 마이그레이션

스키마 변경 이력은 버전 관리돼야 한다. Flyway 문서는 `flyway_schema_history` 테이블이 적용된 마이그레이션과 checksum을 추적하며, 이미 적용된 버전드 마이그레이션은 수정하지 않는 것이 베스트 프랙티스라고 안내한다.

<sup>36</sup>

프로젝트 산출물로는 다음이 명확해야 한다.

- `V1__init.sql`, `V2__add_ledger.sql` … 같은 마이그레이션 파일
- 로컬 실행 시 자동 반영(부팅 시 마이그레이션)

### 테스트 전략

통합 테스트에서 실제 MySQL/Redis를 붙여보는 것이 강력하다. Spring Boot 문서는 Testcontainers가 Docker 컨테이너 기반으로 MySQL 같은 실제 백엔드 서비스를 테스트에 띄우는 데 유용하다고 설명한다. Testcontainers 문서는 JUnit 5(Jupiter) 통합과 컨테이너 lifecycle 관리 방법을 제공한다. <sup>37</sup>

권장 테스트 루틴(최소):

- 계정계: “동시 출금 경합” 테스트(멀티스레드/멀티요청) → 최종 잔액/원장 일관성 검증
- 채널계: 이체 플로우 상태머신(prepare→confirm→execute) 시나리오 테스트
- 대외계: timeout/retry/circuit breaker 동작 테스트(시뮬레이션 지연/에러 주입)

## 관측 가능성과 운영

Spring Boot Actuator는 Micrometer 자동 구성과 다양한 모니터링 시스템(예: Prometheus 등) 지원을 설명한다. Micrometer는 Spring 생태계에서 메트릭 계측 라이브러리로 쓰인다. <sup>38</sup> 또한 Actuator는 tracing에 대해 Micrometer Tracing 자동 설정을 제공한다고 안내한다. <sup>39</sup>

권장 운영 지표(학습용 SLO/대시보드):

- 채널계: p95 응답시간, 로그인 실패율, 이체 실패율, 외부 호출 timeout 비율
- 계정계: DB deadlock 감지 수, 잔액 업데이트 평균시간
- 대외계: circuit breaker OPEN 횟수, retry 횟수

캐시/세션 TTL은 Redis의 만료 기능으로 통제한다. Redis 문서는 `EXPIRE` 가 키 만료를 설정해 시간이 지나면 자동 삭제된다고 설명한다. <sup>40</sup>

## 브랜치 전략과 CI

Gitflow는 `main/develop/feature` 등 여러 브랜치를 이용해 릴리즈 중심으로 개발을 조직하는 워크플로로 소개된다. <sup>41</sup>

CI는 GitHub <sup>42</sup> Actions를 권장한다. GitHub 문서는 Actions가 저장소 내에서 CI 워크플로를 구성할 수 있다고 설명한다. <sup>43</sup>

권장 워크플로(최소):

- PR 생성 시: 빌드 + 단위테스트 + Testcontainers 통합테스트
- main 머지 시: Docker 이미지 빌드 + (선택) Compose 기반 e2e 스모크

## 프론트엔드 선택지

React는 UI를 컴포넌트로 구성해 웹 UI를 만들 수 있는 라이브러리라고 공식 문서에서 소개된다. React Native는 React 지식을 활용해 네이티브 앱을 만들 수 있다고 안내한다. <sup>44</sup>

포트폴리오 관점의 최소 화면 범위(권장):

- 로그인/로그아웃, 계좌목록, 계좌상세(거래내역), 이체(단계형), 알림(실시간)
- “이체 진행 상태(prepare/confirm/execute)”를 시각적으로 보여주면 백엔드 상태머신 설계가 전달된다(설계 선택).

## 최종 산출물 체크리스트

마지막에 “프로젝트가 금융권 채널/계정/대외 구조를 이해하고 구현했다”로 보이려면, 코드 외에도 문서 산출물이 중요하다.

- 시스템 컨텍스트(아키텍처 디어그램 + 데이터 흐름)
- API 스펙(요청/응답·에러코드·idempotency 키 규칙)
- DB ERD/マイ그레이션 히스토리(Flyway)
- 보안 설계(세션/CSRF/재인증/로깅 마스킹) — OWASP 근거 포함 <sup>45</sup>
- 장애 시나리오 리포트(대외계 timeout, 재시도, 서킷브레이커) — Resilience4j 근거 포함 <sup>46</sup>

이 체크리스트가 갖춰지면, 면접에서 “채널계는 고객 접점과 세션/보안, 계정계는 원장 정합성, 대외계는 외부 연동의 실패/재전송/전문 변환”이라는 구조를 근거(공식 문서·표준·공개 교육자료)와 함께 설명할 수 있다. <sup>47</sup>

- 
- 1 2 3 12 13 42 47 <https://www.bok.or.kr/portal/bbs/B0000217/view.do?menuNo=200144&nttId=221237&pageIndex=1>  
https://www.bok.or.kr/portal/bbs/B0000217/view.do?menuNo=200144&nttId=221237&pageIndex=1
- 4 7 10 <https://docs.spring.io/spring-boot/docs/3.2.0-SNAPSHOT/reference/htmlsingle/index.html>  
https://docs.spring.io/spring-boot/docs/3.2.0-SNAPSHOT/reference/htmlsingle/index.html
- 5 9 <https://docs.spring.io/spring-session/reference/guides/boot-redis.html>  
https://docs.spring.io/spring-session/reference/guides/boot-redis.html
- 6 23 24 **17.1 Introduction to InnoDB**  
[https://docs.oracle.com/cd/E17952\\_01/mysql-8.0-en/innodb-introduction.html?utm\\_source=chatgpt.com](https://docs.oracle.com/cd/E17952_01/mysql-8.0-en/innodb-introduction.html?utm_source=chatgpt.com)
- 8 15 33 <https://nayoungs.tistory.com/entry/%EA%B8%88%EC%9C%B5-IT-%EA%B8%88%EC%9C%B5-%EC%8B%9C%EC%8A%A4%ED%85%9C%EC%9D%98-%EA%B5%AC%EC%A1%B0-%EC%B1%84%EB%84%90%EA%B3%84-%EA%B3%84%EC%A0%95%EA%B3%84-%EC%A0%95%EB%B3%B4%EA%B3%84-%ED%86%BA%EC%95%84%EB%B3%B4%EA%B8%BA>  
<https://nayoungs.tistory.com/entry/%EA%B8%88%EC%9C%B5-IT-%EA%B8%88%EC%9C%B5-%EC%8B%9C%EC%8A%A4%ED%85%9C%EC%9D%98-%EA%B5%AC%EC%A1%B0-%EC%B1%84%EB%84%90%EA%B3%84-%EA%B3%84%EC%A0%95%EA%B3%84-%EC%A0%95%EB%B3%B4%EA%B3%84-%ED%86%BA%EC%95%84%EB%B3%B4%EA%B8%BA>
- 11 27 <https://docs.spring.io/spring-framework/docs/4.1.5.RELEASE/spring-framework-reference/html/transaction.html>  
<https://docs.spring.io/spring-framework/docs/4.1.5.RELEASE/spring-framework-reference/html/transaction.html>
- 14 38 **Metrics :: Spring Boot**  
[https://docs.spring.io/spring-boot/reference/actuator/metrics.html?utm\\_source=chatgpt.com](https://docs.spring.io/spring-boot/reference/actuator/metrics.html?utm_source=chatgpt.com)
- 16 **Authentication Persistence and Session Management - Spring**  
[https://docs.spring.io/spring-security/reference/servlet/authentication/session-management.html?utm\\_source=chatgpt.com](https://docs.spring.io/spring-security/reference/servlet/authentication/session-management.html?utm_source=chatgpt.com)
- 17 **Authentication - OWASP Cheat Sheet Series**  
[https://cheatsheetseries.owasp.org/cheatsheets/Authentication\\_Cheat\\_Sheet.html?utm\\_source=chatgpt.com](https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html?utm_source=chatgpt.com)
- 18 **RFC 6238 - TOTP: Time-Based One-Time Password ...**  
[https://datatracker.ietf.org/doc/html/rfc6238?utm\\_source=chatgpt.com](https://datatracker.ietf.org/doc/html/rfc6238?utm_source=chatgpt.com)
- 19 [https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site\\_Request\\_Forgery\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html)  
[https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site\\_Request\\_Forgery\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html)
- 20 <https://www.w3.org/TR/trace-context/>  
<https://www.w3.org/TR/trace-context/>
- 21 [https://cheatsheetseries.owasp.org/cheatsheets/Logging\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html)  
[https://cheatsheetseries.owasp.org/cheatsheets/Logging\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html)
- 22 45 **Logging - OWASP Cheat Sheet Series**  
[https://cheatsheetseries.owasp.org/cheatsheets/Logging\\_Cheat\\_Sheet.html?utm\\_source=chatgpt.com](https://cheatsheetseries.owasp.org/cheatsheets/Logging_Cheat_Sheet.html?utm_source=chatgpt.com)
- 25 <https://dev.mysql.com/doc/refman/en/innodb-transaction-isolation-levels.html>  
<https://dev.mysql.com/doc/refman/en/innodb-transaction-isolation-levels.html>
- 26 <https://dev.mysql.com/doc/en/innodb-locking-reads.html>  
<https://dev.mysql.com/doc/en/innodb-locking-reads.html>

<sup>28</sup> <https://www.rfc-editor.org/rfc/rfc9110.html>

<https://www.rfc-editor.org/rfc/rfc9110.html>

<sup>29</sup> Korea Financial Telecommunications & Clearings Institute

[https://eng.kftc.or.kr/intro/prVideo?utm\\_source=chatgpt.com](https://eng.kftc.or.kr/intro/prVideo?utm_source=chatgpt.com)

<sup>30</sup> <sup>46</sup> <https://resilience4j.readme.io/docs/getting-started>

<https://resilience4j.readme.io/docs/getting-started>

<sup>31</sup> <https://resilience4j.readme.io/docs/getting-started-3>

<https://resilience4j.readme.io/docs/getting-started-3>

<sup>32</sup> <https://opentelemetry.io/docs/concepts/context-propagation/>

<https://opentelemetry.io/docs/concepts/context-propagation/>

<sup>34</sup> <https://docs.docker.com/compose/>

<https://docs.docker.com/compose/>

<sup>35</sup> <https://www.keycloak.org/>

<https://www.keycloak.org/>

<sup>36</sup> <https://documentation.red-gate.com/fd/versioned-migrations-273973333.html>

<https://documentation.red-gate.com/fd/versioned-migrations-273973333.html>

<sup>37</sup> <https://docs.spring.io/spring-boot/reference/testing/testcontainers.html>

<https://docs.spring.io/spring-boot/reference/testing/testcontainers.html>

<sup>39</sup> Tracing :: Spring Boot

[https://docs.spring.io/spring-boot/reference/actuator/tracing.html?utm\\_source=chatgpt.com](https://docs.spring.io/spring-boot/reference/actuator/tracing.html?utm_source=chatgpt.com)

<sup>40</sup> <https://redis.io/docs/latest/commands/expire/>

<https://redis.io/docs/latest/commands/expire/>

<sup>41</sup> <https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>

<https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>

<sup>43</sup> <https://docs.github.com/actions>

<https://docs.github.com/actions>

<sup>44</sup> <https://react.dev/>

<https://react.dev/>