신한은행 해커톤 프론트 기술 스택 정리

1. 기술 스택 정리

React

- **간단한 설명:** 컴포넌트 기반의 프론트엔드 UI 라이브러리.
- **특징:** 가상 DOM(Virtual DOM)을 사용해 빠른 렌더링, 재사용 가능한 컴포넌트 구조.
- **선택 이유:** 금융 서비스에서 반복되는 UI(랭킹보드, 적금 챌린지 등)를 모듈화하기에 최적이며, 대규모 생태계로 안정성이 검증됨.

Vite

- **간단한 설명:** 초고속 번들러 및 개발 서버.
- **특징:** ESBuild 기반으로 CRA보다 **10배 이상 빠른 빌드**, HMR(Hot Module Replacement) 속도가 뛰어남.
- **선택 이유:** 최신 React 프로젝트에서 사실상 표준으로 자리잡았으며, 초기 로딩 속도와 빌드 최적화가 뛰어남.

TypeScript

- **간단한 설명:** JavaScript에 타입 안정성을 추가한 언어.
- **특징:** 정적 타입 체크로 런타임 에러 감소, 코드 자동 완성/리팩토링에 강점.
- **선택 이유:** 은행 서비스에서 필수적인 **안정성과 신뢰성**을 높이고, 유지보수를 쉽게 만듦.

Vercel

- **간단한 설명:** 정적 사이트 및 서버리스 기능을 제공하는 호스팅 플랫폼.
- **특징:** GitHub와 완벽하게 연동되어 **코드 푸시 시 자동 빌드 및 배포**, 글로벌 CDN으로 빠른 로딩 속도 제공.
- **선택 이유:** 배포 자동화와 Preview URL 기능 덕분에 협업 및 QA 과정이 매우 효율적.

GitHub

- **간단한 설명:** 코드 버전 관리 및 협업 플랫폼.
- **특징:** Git 기반, Pull Request, Issue 관리, GitHub Actions(CI/CD) 지원.
- **선택 이유:** Vercel과 자동 연동되어 브랜치별 Preview 배포가 가능하고, 협업 워크플로우 관리가 용이.

Sentry

- **간단한 설명:** 실시간 오류 추적 및 모니터링 플랫폼.
- **특징:** 에러 스택 트레이스, 브라우저 환경, 사용자 세션 등 **상세한 오류 데이터** 제공.
- **선택 이유:** 금융 앱 서비스에서 발생하는 클라이언트 오류를 빠르게 감지해 안정성을 확보하기 위해 선택.

---

2. 배포 파이프라인

전체 흐름

1. **로컬 개발**
   - React + Vite + TypeScript로 UI 및 기능 개발.
   - Sentry 연동(`@sentry/react`)으로 에러 추적 활성화.
2. **코드 관리 (GitHub)**
   - `feature` 브랜치에서 개발 → Pull Request(PR) → `main` 브랜치로 머지.
   - PR 시 GitHub Actions가 ESLint/TypeScript 빌드 검사 실행.
3. **자동 빌드 & Preview (Vercel 연동)**
   - `feature` 브랜치에 Push → Vercel이 자동으로 빌드 후 Preview URL 생성.
   - QA/테스트 후 main 브랜치로 Merge.
4. **프로덕션 배포 (Vercel)**
   - main 브랜치 Merge → Vercel이 **Production URL**로 자동 배포.
5. **모니터링 (Sentry)**
   - 배포 후 발생하는 오류/경고를 실시간으로 Sentry 대시보드에서 확인.

- Slack이나 이메일 알림 연동으로 문제를 즉시 감지.