

## 서비스

### 무선 인터넷 데이터 거래 플랫폼

U-Tong은 사용자가 자신의 데이터를 주식 방식으로 쉽게 거래하고, 실시간 시세 변동과 다양한 거래 기능을 제공하는 웹 플랫폼입니다.

## 개발 기간

2025. 06. 30 - 2025. 08. 07 (6주)

[서비스 링크](#)

[상세정보\(노션\)](#)

[디자인 링크\(피그마\)](#)

<https://github.com/Ureka-final-project-team-3>

JavaScript TailwindCSS React React Router

## 프론트엔드 기술 스택

HTML5 Vite Framer JWT Figma

## 구성원 (프론트엔드 3명, 백엔드 4명)

- FE ( 김현우 / 유동석 / 이채민 )
- BE ( 손민혁 / 신수현 / 장승범 / 이도연 )

## 역할 (팀장)

### Design

프로토타입 제작 (figma)

UI/UX 디자인 [반응형\_Mobile, PC] (figma)

### FrontEnd

쿠폰 페이지

충전 페이지

포인트 상점

서비스 가이드

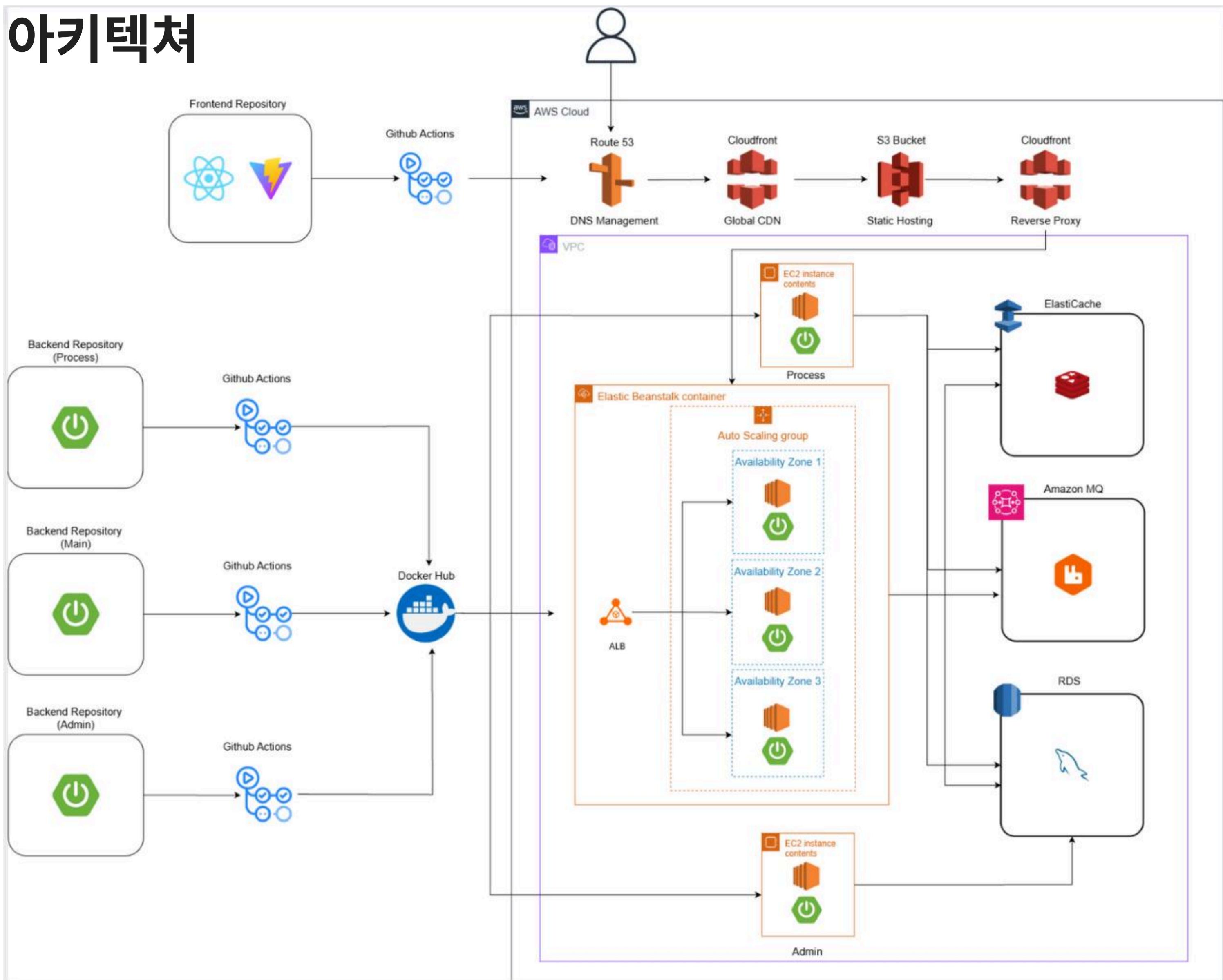
기프티콘 보관함

## 프로젝트 회고

- 팀원 간 원활한 소통을 위해 Swagger로 API 문서화를 하고, Github Project와 데일리 스크럼을 통해 진행 상황을 공유했습니다. 이를 통해 협업 효율성을 높이고 개발 속도를 유지할 수 있었습니다.
- 백엔드 분들과의 첫 소통이 걱정됐지만, 좋은 팀원들을 만나 성공적으로 프로젝트를 마칠 수 있었습니다.

- 기능 구현뿐 아니라 UI/UX까지 직접 고민하며 디자인 감각도 키울 수 있었던 값진 경험이었습니다.
- 데이터 거래를 주식거래처럼 풀어내어 다양한 문제 상황과 로직을 해결 한 과정을 경험 할 수 있어 좋았습니다.

# 아키텍쳐



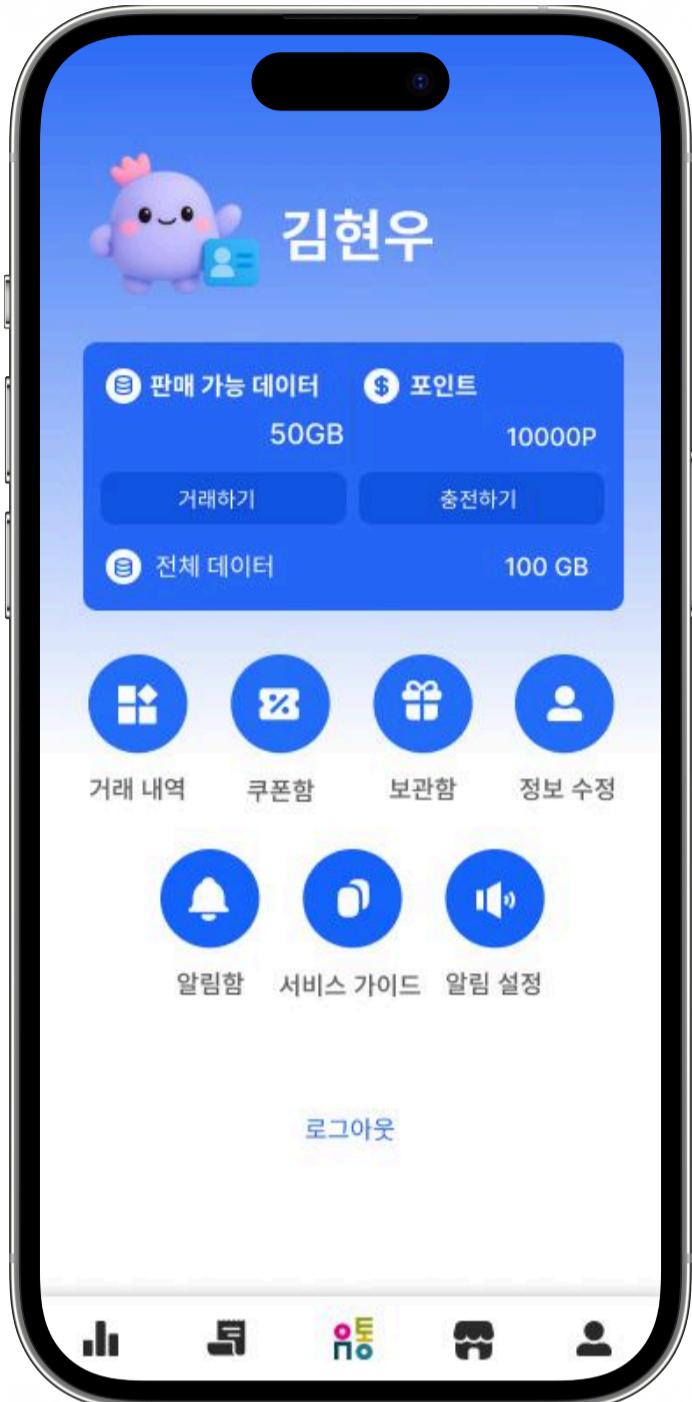
## 배경

서비스 안정성과 가용성을 위해 무중단 배포(Zero Downtime)가 필요했습니다.  
 단일 서버 직접 배포 시 발생할 수 있는 중단 문제를 해결하기 위해 롤링 배포(Rolling Deployment)를 도입했습니다.  
 CloudFront를 리버스 프록시로 두어 캐싱 및 트래픽 분산을 적용했습니다.  
 백엔드는 Docker 컨테이너 기반으로 운영하며, 이미지를 Docker Hub에 업로드해 배포 파이프라인에서 활용했습니다.  
 최종적으로 Elastic Beanstalk 위에서 롤링 업데이트 방식으로 안정적인 배포 환경을 구축했습니다.

## 작업 내용

- Docker 컨테이너화: 애플리케이션을 Dockerfile로 빌드 후 Docker Hub에 푸시
- Elastic Beanstalk 배포: Docker 기반 환경을 생성하고 롤링 방식으로 무중단 배포
- CloudFront 구성: 프론트 요청을 프록시하여 캐싱/HTTPS 처리 및 응답 최적화
- CI/CD 파이프라인: GitHub Actions 또는 CodePipeline으로 코드 푸시 시 자동 빌드·배포

## 마이페이지



### 주요 기능 및 구현 기술

#### 회원 정보 연동

- 로그인한 사용자의 정보를 서버와 연동하여 마이페이지 등에서 실시간으로 불러오고 표시할 수 있도록 구현.

#### 로그아웃 시 JWT 토큰 제거

- 로그아웃 시 클라이언트 측에 저장된 JWT 토큰을 삭제하고, 서버 세션과의 연계를 끊어 보안을 강화. 이를 통해 불필요한 인증 정보가 남지 않도록 하여 안전한 사용자 환경을 보장함.

### 기술적 도전 및 해결 방안

#### JWT 보안 취약성

- 문제: 클라이언트 저장소(LocalStorage/SessionStorage)에 토큰이 남아 있을 경우 재사용 공격 위험 존재
- 해결: 로그아웃 시 클라이언트 저장소에서 토큰을 완전히 삭제하고, 서버에서도 블랙리스트 또는 만료 처리 로직을 적용하여 이중 보안 강화

#### 리프레시 토큰 기반 자동 갱신

- 페이지가 갱신될 때마다 리프레시 토큰을 통해 액세스 토큰 만료 시간을 30분으로 재설정. 이를 통해 사용자가 장시간 서비스를 이용할 때 재로그인 없이 편리하게 유지할 수 있도록 하면서, 보안성과 사용자 편의성을 동시에 확보함.

### 주요 성과

- 안정적인 회원 정보 관리  
서버와 연동된 실시간 회원 정보 표시 기능을 통해 사용자가 언제든지 최신 정보를 확인할 수 있도록 구현,  
→ 사용자 신뢰도와 편의성 향상.
- 보안 강화  
로그아웃 시 액세스·리프레시 토큰을 모두 삭제하고 서버 세션도 종료하여, 불필요한 인증 정보가 남지 않도록 함  
→ 보안 취약점을 예방.
- 사용자 경험 개선  
페이지 갱신 시 리프레시 토큰을 활용해 액세스 토큰을 30분 단위로 자동 갱신,  
→ 사용자가 장시간 서비스를 이용하더라도 재로그인 없이 편리한 사용 경험 제공.

### 주요 기능 및 구현 기술

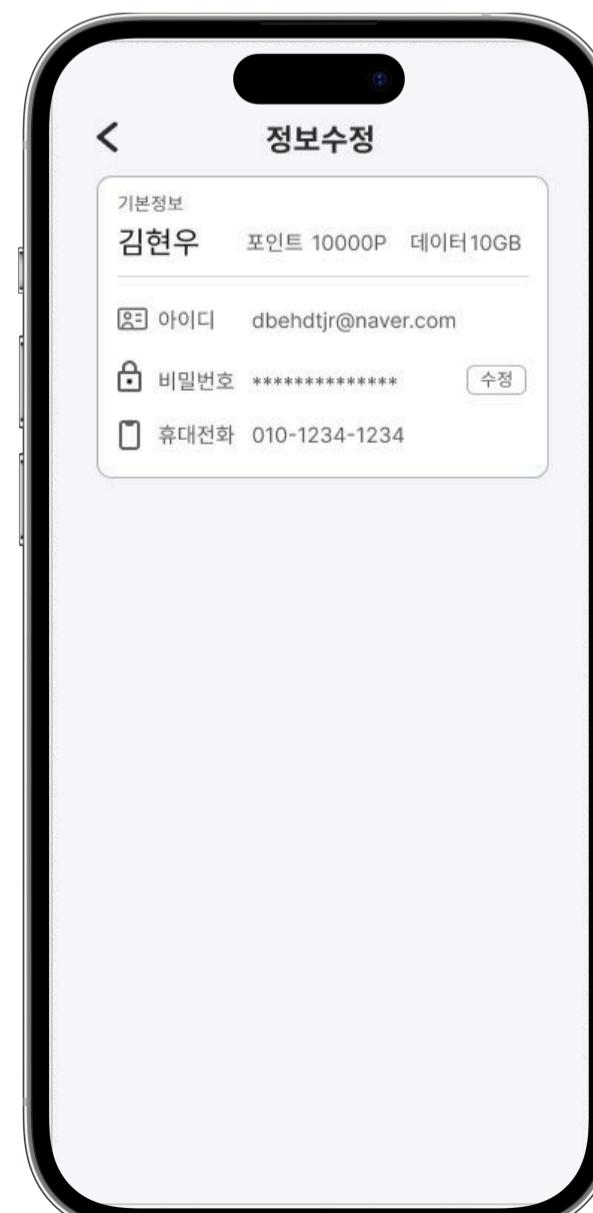
### 정보 수정 페이지

#### 회원 정보 조회 및 수정

사용자의 이름, 이메일, 전화번호 등 기본 회원 정보를 불러오고, 사용자가 직접 수정 할 수 있도록 구현. 마이페이지와 연동하여 실시간으로 변경 사항이 반영되도록 설계.

#### 이메일 변경 인증 절차

사용자가 이메일을 수정하면, 기존에 등록된 이메일로 인증 메일을 발송.  
링크가 포함되어 있으며, 사용자가 해당 링크를 클릭하면 이메일 변경 페이지로 이동. 인증 절차를 완료해야만 최종적으로 이메일 변경이 반영되도록 하여 보안성과 신뢰성을 확보.



### 기술적 도전 및 해결 방안

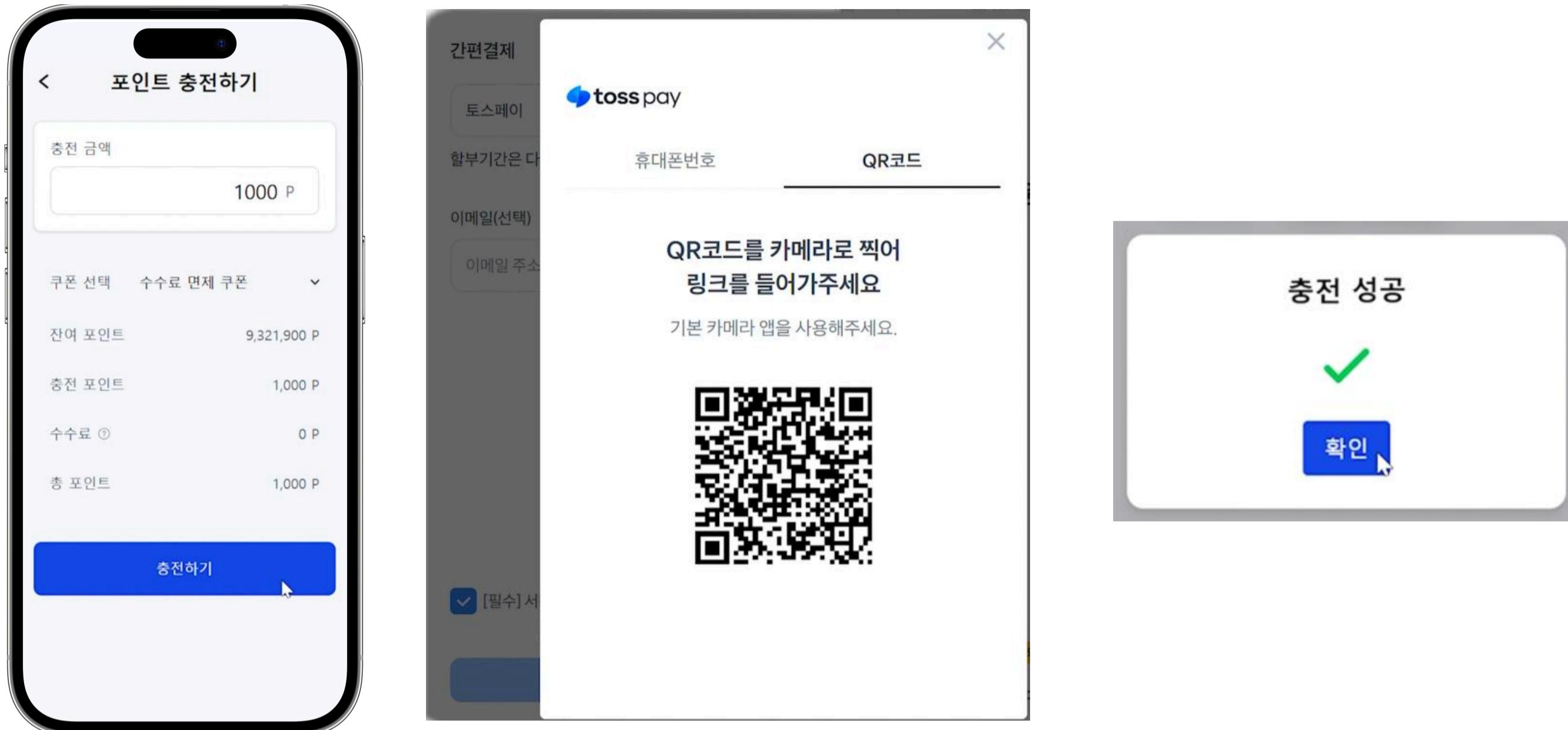
#### 비밀번호 변경 보안

- 문제: 비밀번호 변경 시 인증되지 않은 요청으로 계정 탈취 위험 존재
- 해결: 기존 비밀번호 확인 절차를 추가하고, 비밀번호는 해시 처리 후 DB에 저장.  
변경 성공 시 이메일 발송으로 이중 보안 확보

### 주요 성과

- 사용자가 직접 정보를 수정할 수 있는 기능을 제공하여 계정 관리 자율성 확보
- 비밀번호 변경 시 이메일 알림을 통해 보안 강화 및 사용자 신뢰성 향상

## 포인트 충전 페이지



## 주요 기능 및 구현 기술

## • 포인트 충전 기능 구현

사용자가 원하는 금액을 선택하여 포인트를 충전할 수 있도록 페이지를 구성. 충전 요청 시 TossPayments API와 연동하여 결제 프로세스를 처리하고, 결제 완료 후 서버를 통해 해당 사용자의 포인트가 갱신되도록 설계.

## • 외부 결제 API 연동

포인트 충전 과정에서 외부 결제 모듈(TossPayments)을 활용하여 안정적인 결제 환경 제공. 결제 성공/실패 여부는 서버 응답을 통해 처리되며, 성공 시 사용자 계정에 포인트가 적립됨.

## • 충전 성공 애니메이션

포인트 충전이 완료되면 Framer-motion을 활용한 애니메이션 모달을 표시. 단순 알림이 아닌, 시각적으로 직관적인 성공 애니메이션을 제공하여 사용자 경험을 강화함.

## • 쿠폰 적용 기능

충전 시 쿠폰을 입력·적용할 수 있도록 구현하여, 결제 금액 할인 혹은 추가 포인트 적립과 같은 혜택을 제공. 쿠폰은 서버에서 검증 후 적용되며, 충전 과정에 실시간으로 반영되어 최종 결제 금액과 충전 포인트가 결정됨.

## 기술적 도전 및 해결 방안

## • 결제 API 보안 처리

- 문제: 클라이언트에서 직접 결제 요청을 보내면 위·변조 위험이 존재
- 해결: 서버에서 결제 요청/검증을 처리하고, 클라이언트는 서버 응답만 받아 UI를 업데이트하도록 구조화

## • 포인트 데이터 동기화

- 문제: 결제 성공 후 포인트가 즉시 반영되지 않으면 사용자 혼란 발생
- 해결: 결제 완료 시 서버 DB 업데이트 → 클라이언트 상태를 실시간 갱신하여 일관성 유지

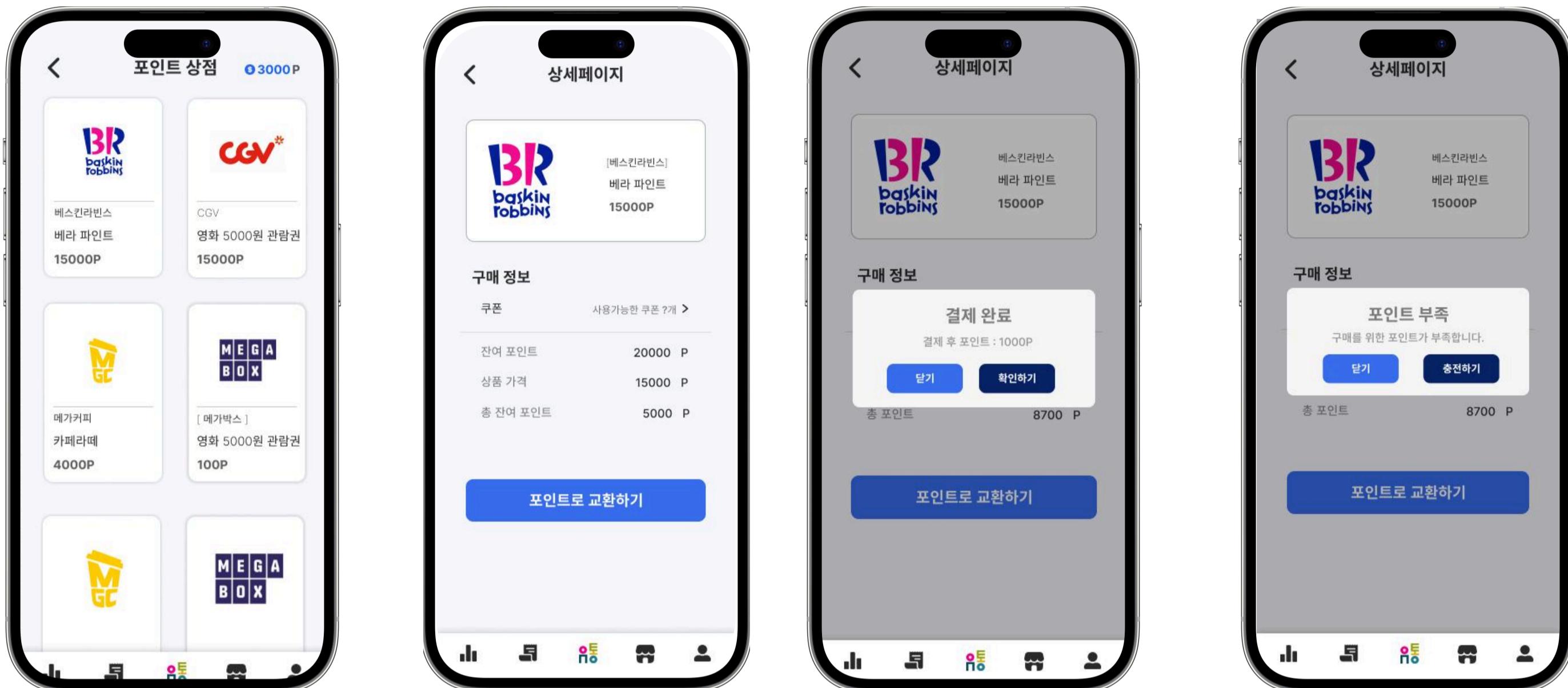
## • 충전 성공 UX 개선

- 문제: 단순 알림창은 사용자 만족도를 떨어뜨림
- 해결: Framer-motion 기반 애니메이션 모달을 도입하여 시각적 완성도를 높이고 긍정적인 경험 제공

## 주요 성과

- 안전하고 안정적인 포인트 충전 서비스 제공: 외부 결제 API(TossPayments) 연동으로 신뢰성 확보
- 실시간 포인트 반영: 결제 후 서버-클라이언트 데이터 동기화로 사용자 혼란 방지
- 향상된 사용자 경험: 충전 성공 모달 애니메이션을 통해 몰입감 있고 긍정적인 피드백 경험 제공

## 포인트 상점



## 주요 기능 및 구현 기술

- 카테고리 및 필터 기능**  
상품을 카테고리별로 나누고, 가격대·인기순·최신순 등의 필터를 제공하여 원하는 상품을 쉽게 찾을 수 있도록 설계.
- 상품 상세 보기 & 구매**  
특정 상품을 클릭하면 상세 정보를 확인 가능.  
구매 버튼 클릭 시, 사용자의 포인트를 차감하고 구매 내역에 저장되며, 기프티콘의 경우 즉시 발급.

## 기술적 도전 및 해결 방안

- 상품 데이터 관리**
  - 문제: 기프티콘, 쿠폰, 디지털 아이템 등 상품 유형별 속성이 달라 관리 복잡
  - 해결: 상품 스키마를 공통 필드(상품명, 가격, 유효 기간) + 옵션 필드(쿠폰코드, 기프티콘 바코드 등)로 분리 설계
- 사용자 경험(UX)**
  - 문제: 상품 수가 많아질수록 탐색이 불편하고, 구매 과정이 단조로움
  - 해결: 카테고리·필터 UI 제공, 구매 시 Framer-motion 모달 애니메이션 추가 → 사용자 몰입감 강화

## 주요 성과

- 포인트 활용성 확대:** 데이터를 판매하여 얻은 포인트로, 실제 상품 교환으로 이어지는 포인트 생태계 구축
- 사용자 경험 강화:** 직관적인 탐색·구매 흐름과 애니메이션 피드백으로 즐거운 상점 이용 경험 제공

## 기프티콘 보관함



### 주요 기능 및 구현 기술

#### 구매 상품 보관함

사용자가 포인트로 교환한 상품(기프티콘, 쿠폰 등)을 한눈에 확인할 수 있는 보관함 페이지 구현.  
교환 즉시 보관함에 자동 저장되며, 사용자는 언제든지 상세 정보를 확인 가능.

#### 상세 정보 조회

각 상품을 클릭하면 이미지, 사용 방법, 유효 기간, 바코드/쿠폰 코드 등 상세 정보를 확인할 수 있도록 구성.  
모바일 환경에서도 스캔 및 사용이 용이하도록 UI 최적화.

#### 사용 상태 관리

사용 완료된 상품은 “사용 완료” 상태로 표시하거나 자동으로 별도 영역으로 이동.  
유효 기간이 임박한 상품은 시각적 알림(색상 강조, 배지 표시)으로 사용자에게 알려줌.

### 기술적 도전 및 해결 방안

#### 유효기간 일관성 유지

- 문제: 클라이언트 시계 차이나 네트워크 지연으로 만료 표시가 어긋날 수 있음.
- 해결: 만료 판단은 서버 시간 기준으로 수행하고, 클라이언트에는 서버에서 계산된 남은 기간/상태를 보내도록 설계.

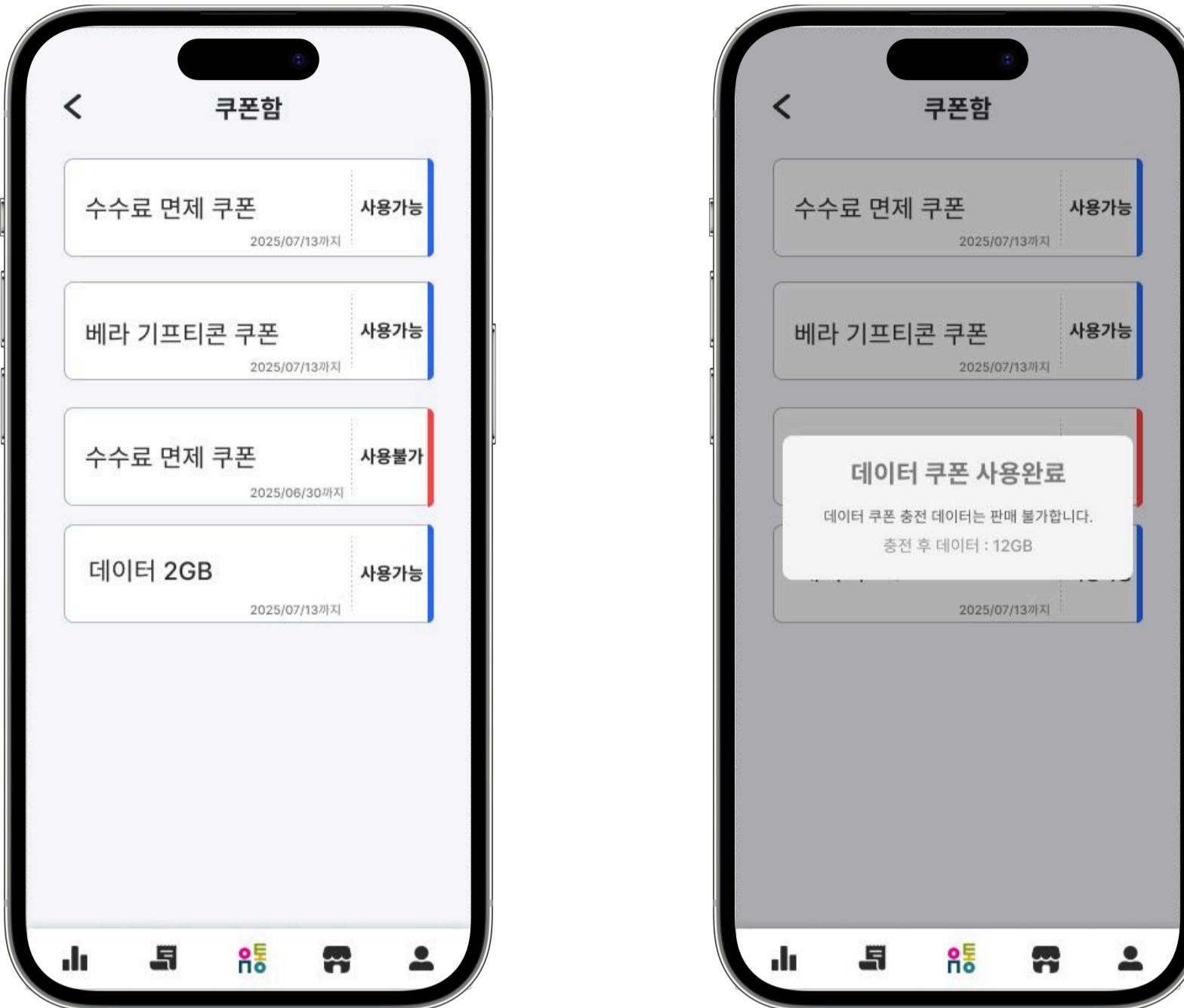
#### 즉시 반영성(구매 → 보관함)

- 문제: 구매 직후 보관함에 반영되지 않으면 사용자 혼란 발생.
- 해결: 구매 성공 응답에서 보관함용 아이템 객체를 즉시 반환하고, 클라이언트 전역 상태(예: Zustand/Redux)를 원자적으로 업데이트.

### 주요 성과

- 신뢰성 확보: 서버 기준 유효기간·상태 관리로 만료/사용 상태의 일관성 보장
- 사용자 안내 개선: 만료 임박 알림과 상태 표시로 사용자 혼란 최소화

## 쿠폰 보관함



### 주요 기능 및 구현 기술

#### • 이벤트 연동 쿠폰 획득

사용자는 이벤트 룰렛 참여를 통해 다양한 쿠폰(데이터 충전 쿠폰, 포인트 수수료 면제 쿠폰 등)을 랜덤으로 획득. 획득한 쿠폰은 자동으로 쿠폰 보관함에 저장되어 추후 사용 가능.

#### • 쿠폰 보관함 관리

쿠폰 보관함에서 사용자는 자신이 보유한 쿠폰 목록을 확인할 수 있음.

쿠폰별로 남은 유효기간과 사용 가능 여부가 표시되며, 만료된 쿠폰은 자동으로 사용 불가 상태로 전환.

#### • 쿠폰 사용 로직

- 데이터 쿠폰: 쿠폰 보관함에서 직접 사용 가능 → 사용 시 서버를 통해 1GB 데이터 충전 처리.
- 포인트 수수료 면제 쿠폰: 포인트 충전 페이지에서 결제 시 선택 적용 → 충전 시 발생하는 수수료가 자동 면제 처리.

### 기술적 도전 및 해결 방안

- 문제: 만료 쿠폰 사용 시도, 혹은 동일 쿠폰 중복 적용 가능성  
해결: 서버에서 사용 전 유효성 검증(만료 체크, 사용 여부 체크) 후 처리 → 클라이언트에서는 UI에서 만료 · 사용완료 상태를 즉시 표시

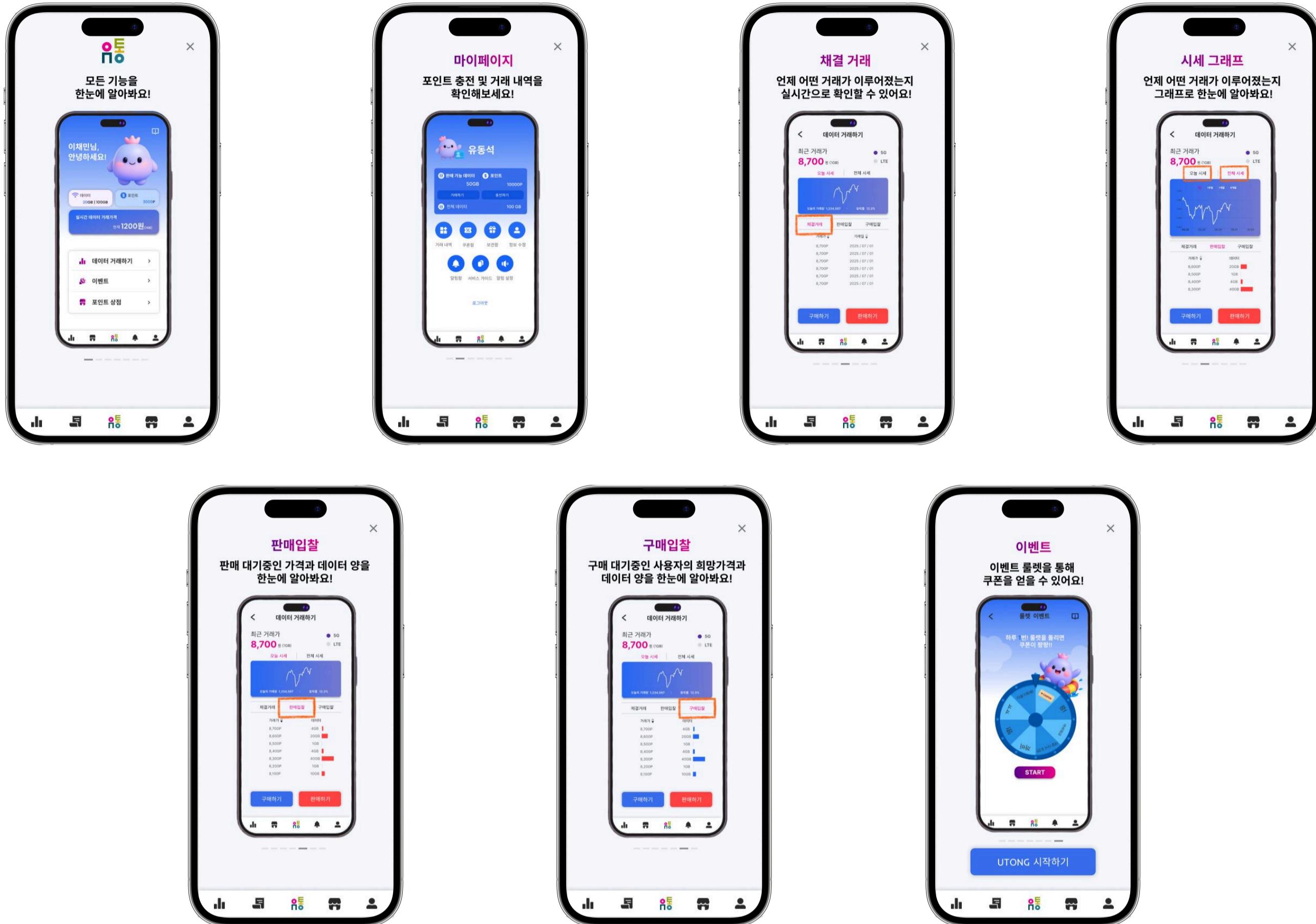
#### • 이벤트 연동 데이터 흐름

- 문제: 이벤트 룰렛에서 쿠폰을 획득해도 보관함에 반영이 지연될 경우 사용자 혼란 발생  
해결: 이벤트 성공 응답 시 쿠폰 객체를 함께 반환하고, 클라이언트 전역 상태에 즉시 반영 → 보관함에서 바로 확인 가능

### 주요 성과

- 이벤트와 상점 기능 연계: 단순 이벤트 참여를 넘어서 실제 서비스 기능(충전/수수료 면제)과 연결, 서비스 몰입도 강화
- 사용자 편의성 향상: 데이터 쿠폰은 즉시 사용, 포인트 쿠폰은 충전 시 선택적으로 적용 → 상황에 맞는 유연한 활용 가능
- 보안 및 무결성 확보: 서버 기반 유효성 검증과 상태 동기화로 중복 사용, 만료 쿠폰 사용을 차단하여 신뢰성 높은 쿠폰 관리 제공

## 서비스 가이드



## 주요 기능 및 구현 기술

## • 슬라이드 기반 안내

- 사용자가 좌우로 넘기며 확인할 수 있는 슬라이딩(캐러셀) UI로 구현.  
각 슬라이드에는 서비스의 핵심 기능이 간단히 요약되어 있어, 짧은 시간 안에 전체 흐름을 이해할 수 있음.

## 기술적 도전 및 해결 방안

## • 슬라이딩 애니메이션 최적화

- 문제: 다양한 해상도와 기기 환경에서 슬라이드 전환 시 애니메이션이 끊기거나 어색하게 보일 수 있음
- 해결: Framer-motion을 활용해 전환 효과를 최적화하고, 반응형 UI를 적용하여 모든 화면에서 부드러운 슬라이딩 경험 제공

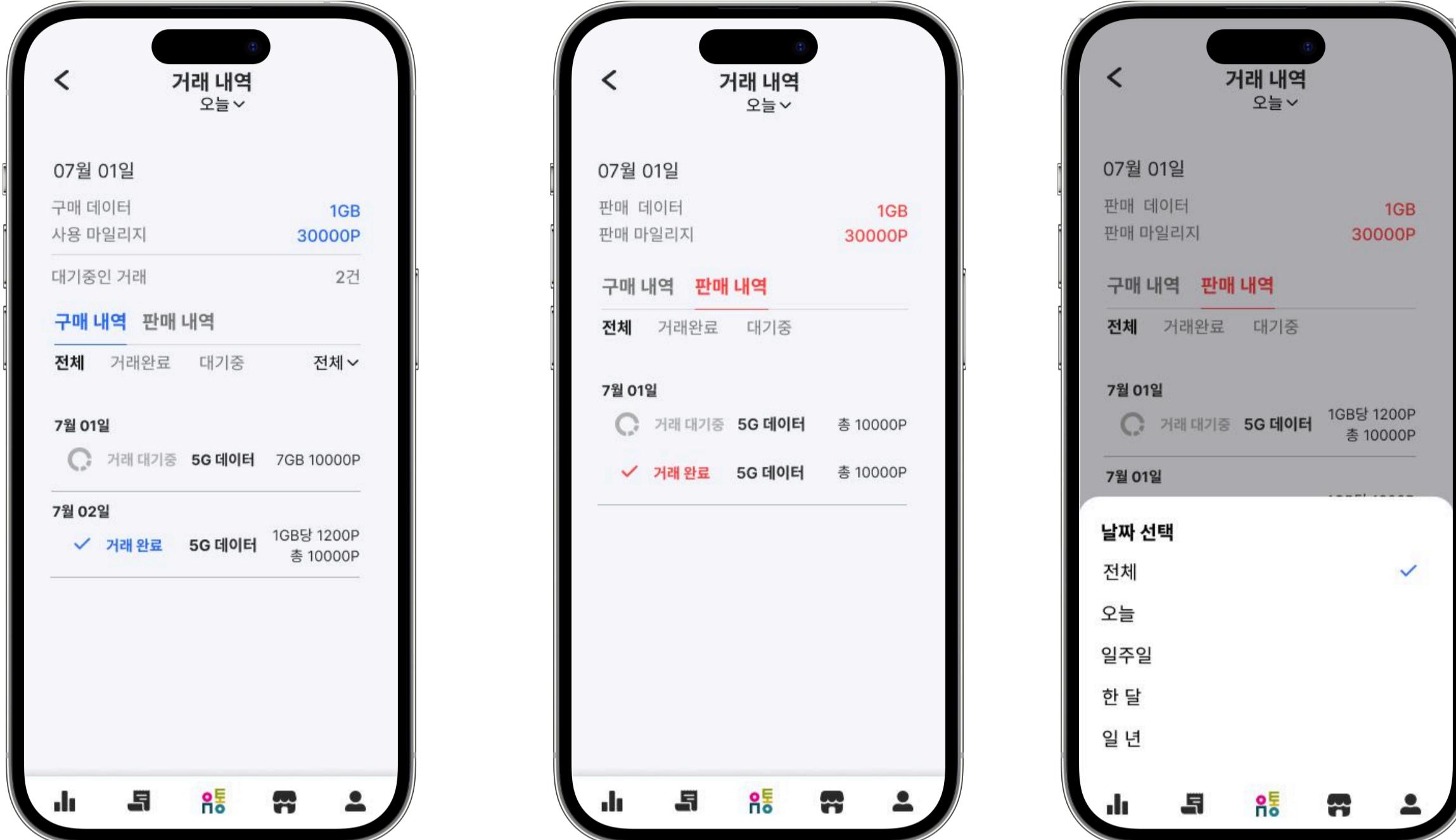
## • 사용자 경험(UX) 고려

- 문제: 온보딩 단계에서 지나치게 많은 정보를 보여주면 사용자가 지루함을 느낄 수 있음
- 해결: 각 슬라이드에 핵심 기능만 간결하게 배치하고, 건너뛰기/시작하기 버튼을 제공해 빠른 진입 가능

## UX 강화 요소

- Framer-motion 애니메이션을 적용하여 슬라이드 전환 시 부드러운 움직임 구현
- 직관적인 아이콘·일러스트를 활용해 기능 이해도를 높임

## 거래내역 페이지



## 주요 기능 및 구현 기술

## • 거래내역 페이지

사용자가 참여한 거래(구매/판매)를 한눈에 확인할 수 있도록 구현. 거래 상태(진행 중, 완료, 취소 등)와 금액, 포인트 변동 내역을 직관적으로 표시.

## • 날짜별 상세 거래내역

특정 거래를 클릭하면 상세 페이지로 이동하여 거래일시, 거래 금액, 수수료, 상대방 정보 등 세부 내역 확인 가능. 날짜별 필터링 기능을 제공하여 원하는 기간의 거래만 추려서 확인 가능.

## 기술적 도전 및 해결 방안

## 대량 데이터 조회 성능

- 문제: 거래내역이 많아질 경우 페이지 로딩 지연 및 불필요한 전체 데이터 불러오기 문제 발생
- 해결: 서버에서 날짜별 쿼리 필터를 지원하고, 클라이언트에서는 무한 스크롤/탭 전환 방식으로 렌더링 최적화

## 상세 거래 데이터 무결성

- 문제: 상세 페이지 진입 시 거래 요약 데이터와 상세 데이터가 불일치할 가능성 존재
- 해결: 상세 조회 시 서버에서 원본 거래 데이터를 다시 검증 후 제공 → 클라이언트에서는 UI만 바인딩

## 주요 성과

- 투명한 거래 경험 제공: 요약 + 상세 내역 확인 기능을 통해 사용자가 거래 흐름을 명확히 파악 가능
- 사용자 편의성 강화: 날짜별 필터링과 페이지네이션으로 원하는 거래만 빠르게 조회 가능
- 데이터 신뢰성 확보: 서버 원본 데이터 기반 상세 조회 및 실시간 동기화로 정확하고 일관된 거래 정보 제공

### MIXED CONTEXT 문제



1

### MIXED CONTENT 문제

HTTPS와 HTTP 혼용으로 mixed-content 발생 (쿠키 발급 실패)

프론트엔드 → 보안을 위한 HTTPS 배포

백엔드 → CloudFront의 Reverse-Proxy 이용

API 요청 → HTTP to HTTPS Redirection

#### 배경

- 사용자 인증에서 리프레시 토큰(Refresh Token)을 이용해 세션 활동이 있을 때마다 액세스 토큰(JWT)을 자동 연장하는 방식을 적용하고자 했음. 이를 위해 쿠키에 토큰을 안전하게 저장·관리하려 했으나, 프론트엔드와 백엔드의 프로토콜이 달라 Mixed Content 문제가 발생하며 쿠키 발급 자체가 정상적으로 이뤄지지 않는 상황에 직면함.

#### 문제상황 및 원인 분석

- 프론트엔드는 HTTPS, 백엔드는 HTTP로 배포되어 프로토콜 혼용 문제 발생
- 브라우저의 보안 정책상 HTTP 리소스가 차단되면서 쿠키 발급 실패
- 그 결과, 리프레시 토큰을 통한 세션 연장 로직이 동작하지 않음

#### 해결 과정

- 프론트엔드: HTTPS로 배포하여 보안 통신을 일관성 있게 유지
- 백엔드: CloudFront Reverse-Proxy를 적용해 모든 외부 요청이 HTTPS로 들어오도록 구성
- API 요청: HTTP 요청은 자동으로 HTTPS로 리다이렉션하여 혼용 문제를 제거

#### 결과 및 배운 점

- 프로토콜 일관성 확보를 통해 쿠키 발급 문제 해결, 리프레시 토큰 기반 세션 연장 로직이 정상적으로 동작
- HTTPS/HTTP 혼용 시 발생하는 보안 이슈와 브라우저 정책에 대해 실무적으로 이해
- 인증·보안 로직은 프론트엔드, 백엔드, 인프라가 긴밀히 맞물려야 한다는 점을 경험적으로 습득