

정산 시스템 전체 점검 보고서

작성일: 2026년 2월 7일 | 프로젝트: 탑셀 주문관리 시스템

1. DB 스키마 현황

1-1. members 테이블 잔액 컬럼

컬럼명	데이터 타입	기본값	필수	설명
deposit	integer	0	YES (NOT NULL)	예치금 잔액
point	integer	0	YES (NOT NULL)	포인트 잔액

□ DB 확인 완료 - 두 컬럼 모두 정상 존재

1-2. settlement_history 테이블 (정산 이력)

컬럼명	데이터 타입	필수	기본값	설명
id	varchar	YES	gen_random_uuid()	PK
member_id	varchar	YES	-	회원 ID (FK: members.id)
order_id	varchar	YES	-	주문 ID
settlement_type	text	YES	'auto'	정산 유형 (auto/manual)
pointer_amount	integer	YES	0	포인트 차감액
deposit_amount	integer	YES	0	예치금 차감액
total_amount	integer	YES	0	정산 총액
pointer_balance	integer	YES	0	정산 후 포인트 잔액
deposit_balance	integer	YES	0	정산 후 예치금 잔액
description	text	NO	-	설명
created_at	timestamp	YES	now()	생성일시

인덱스: member_id, order_id, created_at

1-3. deposit_history 테이블 (예치금 변동 이력)

컬럼명	데이터 타입	필수	기본값	설명
id	varchar	YES	gen_random_uuid()	PK
member_id	varchar	YES	-	회원 ID (FK: members.id)
type	text	YES	-	유형 (charge/refund/deduct)
amount	integer	YES	-	변동 금액
balance_after	integer	YES	-	변동 후 잔액
description	text	NO	-	설명
admin_id	varchar	NO	-	처리 관리자 ID
related_order_id	varchar	NO	-	관련 주문 ID
created_at	timestamp	YES	now()	생성일시

인덱스: member_id, created_at, type

1-4. pointer_history 테이블 (포인터 변동 이력)

컬럼명	데이터 타입	필수	기본값	설명
id	varchar	YES	gen_random_uuid()	PK
member_id	varchar	YES	-	회원 ID (FK: members.id)
type	text	YES	-	유형 (grant/deduct)
amount	integer	YES	-	변동 금액
balance_after	integer	YES	-	변동 후 잔액
description	text	NO	-	설명
admin_id	varchar	NO	-	처리 관리자 ID
related_order_id	varchar	NO	-	관련 주문 ID
created_at	timestamp	YES	now()	생성일시

인덱스: member_id, created_at, type

2. API 목록 및 상세

2-1. 관리자 API

[API 1] 예치금 충전

```
URL: POST /api/admin/members/:memberId/deposit/charge
API 명: server/routes.ts (9363)
API: 예치금 충전
API 설명: { amount: number, description?: string }
API 반환: {}
SELECT FOR UPDATE: {} (tx.select().for('update'))
deposit_history: {} (type: "charge")
API 예외:
- 401: 유효하지 않은 토큰
- 403: 권한 없음
- amount <= 0 || 400
- 404: 회원 없음
```

[API 2] 예치금 환불

```
URL: POST /api/admin/members/:memberId/deposit/refund
API 명: server/routes.ts (9402)
API: 예치금 환불 (환급)
API 설명: { amount: number, description?: string }
API 반환: {}
SELECT FOR UPDATE: {}
deposit_history: {} (type: "refund")
API 예외:
- 400: ("금액이 0 이하입니다")
- 404: 회원 없음
```

[API 3] 포인트 지급

```
URL: POST /api/admin/members/:memberId/pointer/grant
API 명: server/routes.ts (9445)
API: 포인트 지급
API 설명: { amount: number, description?: string }
API 반환: {}
SELECT FOR UPDATE: {}
pointer_history: {} (type: "grant")
API 예외: 404, amount <= 0 || 400
```

[API 4] 정산 이력 조회

```
GET /api/admin/settlements
server/routes.ts (9484)
: (, )
memberId, startDate, endDate, type, page, limit
: { records, total, page, limit }
KST : (parseDateRangeKST )
```

[API 5] 예치금 이력 조회

```
GET /api/admin/deposit-history
server/routes.ts (9547)
: 
memberId, startDate, endDate, type, page, limit
: { records, total, page, limit }
```

[API 6] 포인터 이력 조회

```
GET /api/admin/pointer-history
server/routes.ts (9607)
: 
memberId, startDate, endDate, type, page, limit
: { records, total, page, limit }
```

[API 7] 회원 잔액 목록 조회

```
GET /api/admin/members-balance
server/routes.ts (9667)
: 
: [{ id, companyName, grade, deposit, point, username }]
```

[API 8] 특정 회원 잔액 상세 조회

```
GET /api/admin/members/:memberId/balance
server/routes.ts (9338)
: 
: { memberId, companyName, grade, deposit, point, totalBalance, pendingOrdersTotal, availableBalance }
```

2-2. 회원 API

[API 1] 내 잔액 조회

```
GET /api/member/my-balance
server/routes.ts (9696)
```

```
return { deposit, point, pendingOrdersTotal, availableBalance }  
calculateAvailableBalance return { }
```

[API 2] 내 정산 이력 조회

```
method: GET /api/member/my-settlements  
file: server/routes.ts (9715)  
params: startDate, endDate, page, limit  
return: { records, total, page, limit }  
return: return ID return
```

[API 3] 내 예치금 이력 조회

```
method: GET /api/member/my-deposit-history  
file: server/routes.ts (9755)  
return: { records, total, page, limit }
```

[API 4] 내 포인트 이력 조회

```
method: GET /api/member/my-pointer-history  
file: server/routes.ts (9795)  
return: { records, total, page, limit }
```

3. 잔액 검증 로직 상세

3-1. "사용 가능 잔액" 계산

```
파일명: server/routes.ts (5828줄) - calculateAvailableBalance 함수

// 함수 선언
function calculateAvailableBalance(member: Member, currentProducts: Product[]): Promise<BalanceInfo> {
  // 1. priceConfirmed=true인 supplyPrice 찾기 -> current_products에서 찾기
  // 2. supplyPrice가 0인 supplyPrice 찾기
  // 3. supplyPrice가 0인 current_products에서 찾기
  (getSupplyPriceByGrade(grade: START->startPrice, DRIVING->drivingPrice, TOP->topPrice)

  // 반환 값
  return {

```

3-2. 엑셀 업로드 시 잔액 검증

```
파일명: server/routes.ts (6524줄)

// 엑셀 업로드 시 (엑셀 파일):
// ① SHA-256 해시 생성
// ② 엑셀 파일 (current_products)
// ③ 엑셀 파일
// ④ 엑셀 파일
// ⑤ 엑셀 파일
// ⑥ 엑셀 파일
// ⑦ 엑셀 파일
// ⑧ 엑셀 파일
// ⑨ 엑셀 파일
// ⑩ 엑셀 파일
// ⑪ 엑셀 파일
// ⑫ 엑셀 파일
// ⑬ 엑셀 파일
// ⑭ 엑셀 파일
// ⑮ 엑셀 파일
// ⑯ 엑셀 파일
// ⑰ 엑셀 파일
// ⑱ 엑셀 파일
// ⑲ 엑셀 파일
// ⑳ 엑셀 파일
// ㉑ 엑셀 파일
// ㉒ 엑셀 파일
// ㉓ 엑셀 파일
// ㉔ 엑셀 파일
// ㉕ 엑셀 파일
// ㉖ 엑셀 파일
// ㉗ 엑셀 파일
// ㉘ 엑셀 파일
// ㉙ 엑셀 파일
// ㉚ 엑셀 파일
// ㉛ 엑셀 파일
// ㉜ 엑셀 파일
// ㉝ 엑셀 파일
// ㉞ 엑셀 파일
// ㉟ 엑셀 파일
// ㊱ 엑셀 파일
// ㊲ 엑셀 파일
// ㊳ 엑셀 파일
// ㊴ 엑셀 파일
// ㊵ 엑셀 파일
// ㊶ 엑셀 파일
// ㊷ 엑셀 파일
// ㊸ 엑셀 파일
// ㊹ 엑셀 파일
// ㊺ 엑셀 파일
// ㊻ 엑셀 파일
// ㊼ 엑셀 파일
// ㊽ 엑셀 파일
// ㊾ 엑셀 파일
// ㊿ 엑셀 파일

// 엑셀 파일
// pendingValidationRows (엑셀 파일)
// getSupplyPriceByGrade(currentProduct, member.grade)

// 엑셀 파일
// status: 'insufficient_balance'
// total: 엑셀 파일
// validCount: 엑셀 파일
// errorCount: 엑셀 파일
// totalOrderAmount: 엑셀 파일
// balanceInfo: { deposit, point, pendingOrdersTotal, availableBalance, shortage }
// errors: 엑셀 파일 (엑셀 파일)
```

订单金额校验失败: 金额 (订单金额)

validation_failed 校验失败原因:
totalOrderAmount, balanceInfo, balanceSufficient 金额不足

4. 배송중 전환 자동 정산 상세

파일 위치: server/routes.ts (7781줄)

API 주소: POST /api/admin/orders/to-shipping

작업 순서:

1. 모든 주문 조회 (모든 주문 조회, mode: all/filtered/selected)
2. 주문 상태가 배송중이 아닌 주문 조회
3. 주문 금액 조회
4. 주문 금액을 업데이트하고 (주문 금액을 업데이트):
 - a. SELECT FOR UPDATE로 락을 걸고
 - b. 주문 금액을 조회하고 (getSupplyPriceByGrade)
 - c. 주문 금액을 업데이트
 - d. 주문 금액을 업데이트 -> 주문 금액
 - e. 주문 금액을 "배송중"으로 변경, supplyPrice를 갱신, priceConfirmed=true
 - f. settlement_history를 기록
 - g. pointer_history를 기록 (주문 금액)
 - h. deposit_history를 기록 (주문 금액)
 - i. members 테이블에 주문 금액 기록
5. SSE 이벤트를 발생시킴 (pending-orders-updated, order-status-changed)

주문 금액을 업데이트하고 -> 주문 금액

주문 금액을 갱신: 0

주문 금액을 갱신: 0 (주문 금액을 갱신)

SELECT FOR UPDATE로 락을 걸고

settlement_history를 기록: 0

deposit_history를 기록: 0 (주문 금액)

pointer_history를 기록: 0 (주문 금액)

주문 금액을 갱신: 주문 금액을 갱신하고 주문 금액 (주문 금액을 갱신)

- 주문 금액을 갱신하고, 주문 금액/주문 금액을 갱신

주문 금액을 갱신: 0

- 주문 금액: pending_orders.supplyPrice, pending_orders.priceConfirmed
- 주문 금액: current_products 테이블에서 주문 금액 (getSupplyPriceByGrade)

5. 프론트엔드 현황

5-1. 관리자 정산관리 페이지

파일 위치: client/src/pages/admin/settlements.tsx (610줄)
URL 위치: /admin/settlements
컴포넌트: client/src/App.tsx (174줄)

주요 기능:

1. "members" - 회원 관리 (조회, 등록, 수정, 삭제)
2. "settlements" - 정산 관리 (조회, 등록, 수정, 삭제)
3. "deposit" - 입금 관리 (조회/등록/수정/삭제)
4. "pointer" - 지출 관리 (조회/등록/수정/삭제)

필터링 조건: 날짜 범위

- deposit-charge: 입금 금액
- deposit-refund: 입금 환불
- pointer-grant: 지출 금액

DateRangeFilter 위치: 날짜 범위 (년/월/일) 선택

페이지 번호: 1 (현재 페이지)

5-2. 회원 대시보드 잔액 표시

파일 위치: client/src/pages/dashboard.tsx

데이터 출처: DB 쿼리 (/api/member/my-balance)

데이터 출처: DB 쿼리 (/api/member/my-balance)

"잔액" 위치: activeTab="deposit" (입금)

컴포넌트 위치: client/src/pages/member/settlement-tab.tsx (369줄)

- MemberSettlementTab 컴포넌트
- 입금, 지출, 정산, 조회 버튼

5-3. 주문 등록 화면 잔액 배너

파일 위치: client/src/pages/dashboard.tsx (1247줄)

조건: canOrder=true && balanceData

3가지 경우:

- >= 1000: 잔액 배너
- > 0 & < 500: 잔액 배너 + "잔액 부족" 배너
- <= 0: 잔액 배너 + "잔액 부족" 배너 + "주문 등록" 배너

배너 위치: 주문 등록, 잔액 배너, 주문 등록

```
data-testid: card-balance-banner, text-balance-total, text-pending-total, text-available-balance
```

5-4. 잔액 부족 시 알림 UI (엑셀 업로드)

파일 위치: client/src/pages/dashboard.tsx (1749줄)

컴포넌트: 알림 대화상자 (Dialog)

- 잔액 부족 + 알림 OK: 잔액 부족 + 알림 버튼
- 알림 크기: 알림 버튼 크기 (max-w-500px)
- 잔액 + 알림: 알림 크기 (max-w-560px), "잔액 부족" 알림
- 잔액 부족 알림 OK: 잔액 + 알림 버튼 + "잔액 부족" 알림

알림 버튼:

- 알림 버튼, 알림 버튼, 알림 버튼
- 알림 버튼
- 알림, 알림, 알림 버튼, 알림 버튼
- 알림 버튼 (알림 버튼)

알림 버튼: 알림 버튼 크기 알림 "잔액 부족 알림" 알림

5-5. 회원 이력 조회 화면

파일 위치: client/src/pages/member/settlement-tab.tsx (369줄)

컴포넌트: 회원 이력 > "회원 이력" (회원 이력)

회원 이력: 회원 이력

회원 이력: 회원 이력

회원 이력: 회원 이력

회원 이력: 회원 이력 (DateRangeFilter 버튼)

6. 기존 기능 영향 확인

6-1. 주문 관련

확인 항목	상태	상세
엑셀 주문 등록 - 기존 검증 흐름	□ 정상	SHA-256 중복, 필수필드, 상품존재, 공급중지, 주소검증 순서 유지. 잔액 검증은 상품 검증 후, 주소 검증 전에 삽입
검증 오류 시 다이얼 로그	□ 정상	기존 validation_failed 응답에 balanceInfo, balanceSufficient 필드만 추가. 기존 필드 변경 없음
"정상건만 등록" 기능	□ 정상	잔액 충분 시 기존과 동일하게 작동. 잔액 부족 + 오류 동시 발생 시에만 버튼 비활성화
오류건 엑셀 자동 다운로드	□ 정상	errorExcelData 필드 기존 그대로 유지
주문 상태 전환 워크플로우	□ 정상	대기->상품준비중->배송준비중->배송중 흐름 유지. 배송중 전환에 정산 로직 추가
운송장 전달 기능	□ 정상	기존 로직 변경 없음
취소마감 기능	□ 정상	배송중 전환(mode=all) 시 취소마감 초기화 로직 유지
주문 취소 처리	□ 정상	취소된 주문은 잔액 계산 시 자동 제외 (InProgressStatuses에 미포함)

6-2. 회원 관련

확인 항목	상태	상세
PENDING/ASSOCIATE 등급 주문 차단	□ 정상	canOrder 로직 유지: START/DRIVING/TOP만 주문 가능
등급별 가격체계(3-tier)	□ 정상	getSupplyPriceByGrade 함수로 START/DRIVING/TOP 가격 매핑
회원 대시보드 기존 기능	□ 정상	기존 탭 유지 + "예치금충전" 탭 추가
사이드바 메뉴 구조	□ 정상	기존 메뉴에 "예치금충전" 항목만 추가
로그인/인증	□ 정상	세션 기반 인증 변경 없음

6-3. 관리자 관련

확인 항목	상태	상세
관리자 대시보드 기존 기능	□ 정상	매출/매입/주문관리/상품관리 기존 기능 변경 없음
관리자 사이드바 메뉴	□ 정상	기존 메뉴 유지 + "정산관리" 메뉴 추가 (admins.tsx에 settlement_management 권한 등록)
메뉴 접근 권한	□ 정상	SUPER_ADMIN이 ADMIN에게 정산관리 메뉴 권한 설정 가능

6-4. 시스템/인프라 관련

확인 항목	상태	상세
SSE 실시간 알림	□ 정상	배송중 전환 시 pending-orders-updated, order-status-changed 이벤트 발송 유지
엑셀 업로드/다운로드	□ 정상	기존 파싱/다운로드 로직 변경 없음
API 라우팅 충돌	□ 정상	신규 API 경로(/api/admin/settlements, /api/member/my-* 등) 기존 경로와 충돌 없음
DB 마이그레이션 안전성	□ 정상	신규 테이블 3개 추가 (settlement_history, deposit_history, pointer_history). members 테이블에 deposit/point 컬럼 추가 (기본값 0, 기존 데이터 보존)
기존 쿼리 정상 작동	□ 정상	신규 컬럼은 NOT NULL DEFAULT 0이므로 기존 쿼리에 영향 없음

6-5. UI/스타일 관련

확인 항목	상태	상세
기존 페이지 레이아웃	□ 정상	신규 컴포넌트는 기존 레이아웃 내에 탭/카드 형태로 추가. 기존 UI 변경 없음
반응형(모바일) 표시	□ 정상	잔액 배너, 정산 다이얼로그 모두 반응형 grid/flex 적용

기존 컴포넌트 스타일	□ 정상	DateRangeFilter, 테이블, 토스트 등 공통 컴포넌트 스타일 변경 없음
-------------	------	---

7. 종합 결론

정산 시스템 구현 상태: 완료

- DB 스키마: members 테이블 잔액 컬럼 + 3개 이력 테이블 모두 정상 생성
- 관리자 API: 8개 엔드포인트 구현 (충전/환급/지급, 이력 조회, 잔액 목록)
- 회원 API: 4개 엔드포인트 구현 (잔액 조회, 정산/예치금/포인트 이력)
- 잔액 검증: 엑셀 업로드 시 정상건 기준 검증, 부족 시 등록 차단
- 자동 정산: 배송중 전환 시 포인트 우선 차감, 트랜잭션/SELECT FOR UPDATE 적용
- 프론트엔드: 관리자 정산관리 페이지, 회원 잔액 배너/이력 조회 구현
- 기존 기능 영향: 모든 항목 정상 확인 (기존 기능 보존)