Credit Transaction Fix Summary

Date: November 13, 2025

Problem

The API verify endpoint (`/api/v1/verify`) was working but failing when trying to deduct credits with the following error:

```
null value in column "user_id" of relation "credit_transactions" violates not-null
constraint
```

Root Cause

- The `credit_transactions` table had a NOT NULL constraint on both `customer_id` and `user_id`
- API transactions are authenticated via API keys, which are associated with customers (not users)
- When the `deductCredits()` function in `api-auth.ts` tried to create a transaction record, it only provided `customer_id` but the database required `user_id`
- This created a mismatch between the API authentication model (customer-based) and the transaction tracking model (requiring users)

Solution Implemented

1. Database Schema Change

File: `database/migrations/014_make_user_id_nullable_in_credit_transactions.sql`

Made the `user_id` column nullable in the `credit_transactions` table to support two types of transactions:
- **User transactions** (purchases via UI): Have `user_id` populated
- **API transactions** (usage via API): Have `user_id` as NULL (only `customer_id`)

```
ALTER TABLE credit_transactions
  ALTER COLUMN user_id DROP NOT NULL;
```

2. Code Changes

File: `src/app/lib/api-auth.ts`

Updated the `deductCredits()` function to:
- ✅ Lock the row using `FOR UPDATE` to prevent race conditions
- ✅ Retrieve `balance_before` from `customer_credits`
- ✅ Check for sufficient credits before attempting to deduct
- ✅ Calculate `balance_after` correctly
- ✅ Update both `balance` and `total_used` fields
- ✅ Insert transaction record with `user_id = NULL` for API transactions
- ✅ Include proper balance tracking (`balance_before` , `balance_after`)
- ✅ Use correct transaction type: `'USAGE'` (matching the credits library convention)

Changes Made:

Before:

```
await query(
  `INSERT INTO credit_transactions
   (customer_id, amount, transaction_type, description)
   VALUES ($1, $2, 'debit', $3)`,
  [customerId, -amount, description]
);
```

After:

```
await query(
  `INSERT INTO credit_transactions
   (customer_id, user_id, transaction_type, amount, balance_before, balance_after, de-
scription)
   VALUES ($1, NULL, 'USAGE', $2, $3, $4, $5)`,
  [customerId, -amount, balanceBefore, balanceAfter, description]
);
```

Files Created/Modified

New Files:

1. `database/migrations/014_make_user_id_nullable_in_credit_transactions.sql` - Migration SQL
2. `database/migrations/APPLY_014_MIGRATION.md` - Migration instructions and documentation
3. `CREDIT_TRANSACTION_FIX.md` - This summary document

Modified Files:

1. `src/app/lib/api-auth.ts` - Updated `deductCredits()` function

How to Apply the Fix

Step 1: Apply Database Migration

You need to apply the migration to your Supabase database. See detailed instructions in:
- `database/migrations/APPLY_014_MIGRATION.md`

Quick Steps:
1. Go to Supabase Dashboard → SQL Editor
2. Copy contents of `014_make_user_id_nullable_in_credit_transactions.sql`
3. Run the migration
4. Verify: `user_id` column should be nullable

Step 2: Deploy Code Changes

The code changes in `api-auth.ts` will be automatically deployed when you push to main branch.

Step 3: Test

After applying both changes:
1. Make an API call to `/api/v1/verify` with your API key
2. Verify credits are deducted successfully
3. Check that transaction records are created with `user_id = NULL`

```sql
SELECT
    id, customer_id, user_id, transaction_type,
    amount, balance_before, balance_after, description
FROM credit_transactions
WHERE user_id IS NULL
ORDER BY created_at DESC
LIMIT 5;
```

Benefits

✅ **Fixes API Credit Deduction** - API calls can now successfully deduct credits
✅ **Proper Balance Tracking** - Maintains `balance_before` and `balance_after` for audit trail
✅ **Data Integrity** - Uses row locking to prevent race conditions
✅ **Backward Compatible** - Existing user transactions continue to work with `user_id` populated
✅ **No Data Loss** - All existing transaction records remain intact

Transaction Types Supported

After this fix:

Transaction Type	user_id	customer_id	Use Case
PURCHASE	✅ Set	✅ Set	User purchases credits via UI
USAGE (User)	✅ Set	✅ Set	User performs searches via UI
USAGE (API)	❌ NULL	✅ Set	API calls via API keys
REFUND	✅ Set (optional)	✅ Set	Credit refunds
ADJUSTMENT	✅ Set (optional)	✅ Set	Manual credit adjustments

Verification Queries

Check if migration was applied:

```sql
SELECT column_name, is_nullable
FROM information_schema.columns
WHERE table_name = 'credit_transactions' AND column_name = 'user_id';
-- Should show: is_nullable = 'YES'
```

View API transactions:

```sql
SELECT * FROM credit_transactions
WHERE user_id IS NULL
ORDER BY created_at DESC;
```

View customer credit balance:

```
SELECT c.id, c.name, cc.balance, cc.total_used, cc.total_purchased
FROM customers c
JOIN customer_credits cc ON c.id = cc.customer_id;
```

Rollback Plan

If issues arise, you can rollback the migration:

⚠️ **Warning:** Rollback will delete all API transaction records!

```
BEGIN;
-- Delete API transactions (user_id IS NULL)
DELETE FROM credit_transactions WHERE user_id IS NULL;

-- Restore NOT NULL constraint
ALTER TABLE credit_transactions ALTER COLUMN user_id SET NOT NULL;
COMMIT;
```

Testing Checklist

- [ ] Migration applied successfully to database
- [ ] Code deployed to production
- [ ] API verify endpoint works without errors
- [ ] Credits are deducted correctly
- [ ] Transaction records created with NULL user_id
- [ ] Balance tracking is accurate (balance_before/after)
- [ ] Existing user transactions still work
- [ ] No performance degradation

Related Documentation

- API Documentation: `/api-docs` (Swagger UI)
- Credits System: `src/app/lib/credits/index.ts`
- API Authentication: `src/app/lib/api-auth.ts`
- Database Schema: Supabase Dashboard → Table Editor

Support

If you encounter any issues:
1. Check the migration was applied: See “Verification Queries” above
2. Check application logs for errors
3. Verify API key is valid and customer has credits
4. Test with Postman/Thunder Client before production use

Status: ✅ Ready for deployment
Priority: High - Fixes critical API functionality
Breaking Changes: None
Rollback Risk: Low (only if you need to revert user_id to NOT NULL)