Transaction History and Pricing Display Fix

Date: October 31, 2024
Issues Fixed:
1. Exact searches with "No Results" not showing in transaction history
2. Pricing display missing thousand separators

Issue 1: Missing Transaction History for Free Searches

Problem

Exact searches that returned "No Results" were supposed to be logged as free (0 credits) transactions, but they were not appearing in the transaction history.

Root Cause

The database had a constraint that prevented zero-amount transactions:

```
CONSTRAINT credit_transactions_amount_non_zero CHECK (amount != 0)
```

This constraint was blocking the `recordFreeSearch()` function from inserting transactions with `amount = 0` for:
- Exact searches with no results
- Duplicate/cached searches (already fixed earlier)

Solution

Part 1: Database Migration

Created migration `012_allow_zero_credit_transactions.sql`:

```
-- Drop the existing non-zero constraint
ALTER TABLE credit_transactions
DROP CONSTRAINT IF EXISTS credit_transactions_amount_non_zero;

-- Create index for better performance
CREATE INDEX IF NOT EXISTS idx_credit_transactions_zero_amount
ON credit_transactions(customer_id, amount)
WHERE amount = 0;
```

Part 2: Admin API Endpoint

Created `/api/admin/migrate` endpoint that:
- Requires SUPER_ADMIN authentication
- Runs the migration safely
- Returns success/error status

To run the migration:

```
# As SUPER_ADMIN user, make a POST request:
curl -X POST https://your-domain.com/api/admin/migrate \
  -H "Cookie: next-auth.session-token=YOUR_SESSION_TOKEN"
```

Or access via browser console while logged in as SUPER_ADMIN:

```
fetch('/api/admin/migrate', { method: 'POST' })
  .then(r => r.json())
  .then(console.log);
```

Issue 2: Pricing Display Missing Commas

Problem

Credit package prices were displayed without thousand separators:
- ❌ $1000.00
- ❌ $5000.00
- ❌ $10000.00
- ❌ $20000.00

Solution

Updated the `formatPrice()` function in `/src/app/dashboard/page.tsx`:

Before:

```
const formatPrice = (priceCents: number) => {
  return (priceCents / 100).toFixed(2);
};
```

After:

```
const formatPrice = (priceCents: number) => {
  const dollars = priceCents / 100;
  return dollars.toLocaleString('en-US', {
    minimumFractionDigits: 2,
    maximumFractionDigits: 2,
  });
};
```

Result:
- ✅ $1,000.00
- ✅ $5,000.00
- ✅ $10,000.00
- ✅ $20,000.00

Additional Fix: Transaction Type Display

Problem

Transaction types in the database are stored in uppercase (`PURCHASE`, `USAGE`), but the dashboard was comparing them in lowercase, causing incorrect styling.

Solution

Updated transaction type comparisons to be case-insensitive:

Before:

```
transaction.transaction_type === 'purchase'
```

After:

```
transaction.transaction_type.toUpperCase() === 'PURCHASE'
```

This ensures correct color coding:
- 🟢 PURCHASE → Green badge
- 🔵 USAGE → Blue badge
- ⚪ Others → Gray badge

Files Changed

1. **Database Migration**
   - `database/migrations/012_allow_zero_credit_transactions.sql` (NEW)
   - `scripts/run-migration-012.js` (NEW)

2. **API Endpoint**
   - `src/app/api/admin/migrate/route.ts` (NEW)

3. **Dashboard**
   - `src/app/dashboard/page.tsx`
     - Updated `formatPrice()` function (line 212-218)
     - Fixed transaction type comparisons (lines 303, 405, 407)

Testing Checklist

After Running Migration:

1. **Test Free Exact Searches (No Results)**
   - Log in as regular user
   - Perform exact search that returns no results
   - Check transaction history → Should show entry with 0 credits
   - Check credit balance → Should not decrease

2. **Test Duplicate Cached Searches**
   - Perform same search twice after cooldown
   - Check transaction history → Should show 0-credit entry for duplicate
   - Check credit balance → Should not decrease on second search

3. **Test Pricing Display**
   - Go to “Buy More Credits” section
   - Verify all prices show with commas:

     - $1,000.00 ✓
     - $5,000.00 ✓
     - $10,000.00 ✓
     - $20,000.00 ✓

4. **Test Transaction Type Display**
   - View transaction history
   - Verify PURCHASE transactions have green badge
   - Verify USAGE transactions have blue badge
   - Verify 0-credit transactions display correctly

Migration Instructions

For Production Deployment:

1. **Run the migration** (SUPER_ADMIN only):
   `javascript`
   ```
     // In browser console while logged in as SUPER_ADMIN:
     fetch('/api/admin/migrate', { method: 'POST' })
       .then(r => r.json())
       .then(data => console.log(data));
   ```

2. **Verify migration success:**
   - Response should be: `{ success: true, message: "Migration completed..." }`

3. **Test free searches:**
   - Perform exact search with no results
   - Check if it appears in transaction history

Alternative: Direct Database Access

If you have direct database access:

```sql
-- Run this SQL directly:
ALTER TABLE credit_transactions
DROP CONSTRAINT IF EXISTS credit_transactions_amount_non_zero;

CREATE INDEX IF NOT EXISTS idx_credit_transactions_zero_amount
ON credit_transactions(customer_id, amount)
WHERE amount = 0;
```

Impact Summary

Before:

- ❌ Exact searches with no results: Not visible in transaction history
- ❌ Duplicate cached searches: Not visible in transaction history
- ❌ Pricing: $10000.00 (hard to read)
- ⚠️ Transaction badges: Potential styling issues

After:

- ✅ All searches logged in transaction history (including free ones)
- ✅ Complete audit trail of all search activity
- ✅ Pricing: $10,000.00 (easy to read)
- ✅ Correct color coding for all transaction types
- ✅ Better UX and transparency for customers

Build Status

✅ **Build Successful** - Ready for deployment

Next Steps

1. Deploy the updated code
2. Run the database migration as SUPER_ADMIN
3. Test all scenarios listed in the testing checklist
4. Monitor transaction history for free searches

Status: ✅ READY FOR MIGRATION AND DEPLOYMENT