Migration 014: Make user_id Nullable in credit_transactions

Purpose

This migration fixes the credit transaction issue in the API verify endpoint by making the `user_id` column nullable. This allows API transactions (which only have `customer_id` from API keys) to be recorded without requiring a `user_id`.

Problem Being Fixed

- API transactions are associated with customers via API keys, not users
- The `credit_transactions` table had `user_id` as NOT NULL, causing errors when API calls tried to deduct credits
- Error: “null value in column "user_id" of relation "credit_transactions" violates not-null constraint”

Solution

- Make `user_id` nullable to allow both types of transactions:
- **User transactions** (purchases via UI): Have `user_id` populated
- **API transactions** (usage via API): Have `user_id` as NULL

How to Apply

Option 1: Using Supabase SQL Editor (Recommended)

1. Go to your Supabase project dashboard at https://supabase.com/dashboard
2. Navigate to **SQL Editor**
3. Copy the contents of `014_make_user_id_nullable_in_credit_transactions.sql`
4. Paste into the SQL Editor
5. Click **Run** to execute the migration

Option 2: Using psql Command Line

```
psql "YOUR_SUPABASE_DATABASE_URL" < database/migrations/
014_make_user_id_nullable_in_credit_transactions.sql
```

Replace `YOUR_SUPABASE_DATABASE_URL` with your actual database connection string from Supabase settings.

Option 3: Direct SQL

Connect to your database and run:

```sql
BEGIN;

-- Make user_id nullable
ALTER TABLE credit_transactions
  ALTER COLUMN user_id DROP NOT NULL;

-- Add comment to explain the nullable user_id
COMMENT ON COLUMN credit_transactions.user_id IS
  'User ID for user-initiated transactions (purchases). NULL for API transactions (us-
age).';

COMMIT;
```

Verification

After running the migration, verify it worked by running this query:

```sql
SELECT
    column_name,
    data_type,
    is_nullable
FROM information_schema.columns
WHERE table_name = 'credit_transactions'
    AND column_name = 'user_id';
```

Expected result: `is_nullable = 'YES'` for the `user_id` column.

Testing

After the migration:

1. Make an API call to `/api/v1/verify` with your API key
2. Check that credits are deducted successfully
3. Verify the transaction was recorded:

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

You should see API transactions with `user_id = NULL`.

Rollback (if needed)

If you need to rollback this migration, you would need to:

1. Delete all transactions where `user_id` IS NULL
2. Then restore the NOT NULL constraint

```sql
BEGIN;

-- WARNING: This will delete all API transactions!
-- 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;
```

⚠️ **Warning:** Rollback will delete all API transaction records. Only do this if absolutely necessary.

Related Files Changed

- `database/migrations/014_make_user_id_nullable_in_credit_transactions.sql` - The migration SQL
- `src/app/lib/api-auth.ts` - Updated `deductCredits()` function to properly handle NULL user_id
- `database/schema.sql` - Updated to reflect the nullable user_id

Impact

- ✅ Fixes API verify endpoint credit deduction
- ✅ Allows proper tracking of both user and API transactions
- ✅ Maintains backward compatibility with existing user transactions
- ✅ No data loss or breaking changes