Wise Integration

Architecture Changes & Issue Resolution

Generated: October 22, 2025 at 17:55

Executive Summary

Removed full Wise API integration in favor of manual CSV upload. Encountered database connection pool exhaustion issues during CSV import implementation. Resolved through proper pool configuration.

1. Initial Architecture (Before Changes)

Wise API Integration Components

Backend (8 files removed):
• backend/src/services/wiseService.js - Wise API client wrapper
• backend/src/controllers/wiseWebhookController.js - Webhook event processing
• backend/src/routes/wiseRoutes.js - API sync endpoints
• backend/src/routes/wiseDebug.js - Debug/diagnostic routes
• backend/src/utils/wiseScaSigner.js - SCA (Strong Customer Authentication) signing
• backend/src/utils/wiseSignatureValidator.js - Webhook signature verification
• backend/src/webhookMonitor.js - In-memory webhook monitoring
• backend/src/routes/webhookMonitor.js - Monitor UI routes

Scripts (5 files removed):
• scripts/test-wise-sca.js
• scripts/create-wise-webhook.js
• scripts/monitor-wise-webhooks.js
• scripts/test-webhook-call.js
• scripts/test-webhook-no-sig.js

Frontend (1 component removed):
• frontend/src/components/WiseReviewQueue.jsx - Transaction review UI
• Removed 'Wise Sync' tab from AccountingApp.jsx

Features Removed

• Real-time transaction sync via Wise API
• Webhook-based balance updates
• SCA authentication flow
• Transaction review and classification queue
• Automated sync scheduling

2. New Architecture (Current State)

Components Retained

Backend (3 files kept):
• backend/src/routes/wiseImport.js - CSV upload endpoint (POST /api/wise/import)
• backend/src/services/wiseClassifier.js - Transaction category mapping
• backend/src/models/wiseTransactionModel.js - Database model

Frontend (1 new component):
• frontend/src/components/WiseImport.jsx - CSV upload modal (276 lines)
• 'Import CSV' button added to Dashboard

Current Workflow

1. User exports CSV from Wise.com (Account → Statements → Export CSV)
2. User uploads CSV via dashboard modal
3. Backend parses 21-column Wise CSV format
4. Automatic transaction classification (income/expense)
5. Duplicate prevention by Wise transaction ID
6. Bulk insert with database transaction
7. Balance auto-update after import

3. Issues Encountered & Resolution

Issue #1: CSV Import Returning 500 Error

Symptom:

```
POST /api/wise/import → 500 Internal Server Error
{error: 'Database connection failed', details: 'Unable to connect to database...'}
```

Investigation Steps:

■ Verified backend code correct
■ Verified frontend code correct
■ Verified database accessible (dashboard working)
■ CSV import specifically failing

Issue #2: Variable Naming Inconsistency

Problem:

```
// wiseImport.js (WRONG)
const db = require('../config/database');
client = await db.pool.getClient(); // db.pool is undefined!

// Other files use (CORRECT)
const pool = require('../config/database');
client = await pool.getClient();
```

Fix: Changed wiseImport.js to use 'pool' naming convention (commit: 8636c6d)

Issue #3: Connection Pool Exhaustion (ROOT CAUSE)

Problem:

• Database pool had NO configuration (using pg defaults)
• Default: max 10 connections, no timeouts
• Dashboard uses pool.query() → grab connection, use immediately, auto-release
• CSV import uses pool.getClient() → hold connection for entire transaction
• Pool exhausted → getClient() fails → 500 error

Evidence:

• Dashboard works fine (simple queries)
• CSV import fails (long-running transactions)
• Error: 'Database connection failed'

4. Solution Implemented

Root Cause:

```
// database.js (BEFORE - NO POOL CONFIG)
const pool = new Pool(poolConfig);
```

Solution:

```
// database.js (AFTER - WITH POOL CONFIG)
const poolConfig = {
  connectionString: process.env.DATABASE_URL,
  ssl: { rejectUnauthorized: false },
  max: 5,                     // Limit to 5 connections (Railway free tier)
  idleTimeoutMillis: 30000,   // Close idle connections after 30s
  connectionTimeoutMillis: 10000, // Fail fast after 10s
  allowExitOnIdle: false      // Keep pool alive
};
```

Fix commits:
• be8a265 - Added connection pool configuration
• 1a52b57 - Added diagnostic endpoint (GET /api/wise/test-connection)
• 35413cc - Improved error handling and CSV validation

5. Technical Improvements Made

Enhanced Error Handling

• Check 21-column format
• Validate required columns (ID, Status, Direction)
• Row-by-row field count verification
• Clear error messages with troubleshooting tips

Comprehensive Logging

```
console.log('=== CSV Import Started ===');
console.log('User:', req.user?.id, req.user?.username);
console.log('File:', req.file?.originalname, req.file?.size, 'bytes');
console.log('Total lines (including header):', lines.length);
console.log('CSV Headers found:', headerFields.length, 'columns');
// ... detailed logging throughout process
```

Diagnostic Endpoint

GET /api/wise/test-connection
• Tests pool.query() - simple queries
• Tests pool.getClient() - dedicated connection
• Returns pool statistics (totalCount, idleCount, waitingCount)

6. Deployment Details

Environment:
• **Backend:** Railway (https://business-accounting-system-production.up.railway.app)
• **Frontend:** Netlify (https://ds-accounting.netlify.app)
• **Database:** Railway PostgreSQL

Key Configuration:

```
# Backend .env (Railway)
DATABASE_URL=postgresql://...
NODE_ENV=production
PORT=3001

# Frontend .env (Netlify)
VITE_API_URL=https://business-accounting-system-production.up.railway.app/api
```

7. Lessons Learned

1. Connection Pool Configuration is Critical

• Never rely on defaults for production databases
• Always set max connections, timeouts, and idle timeouts
• Monitor pool statistics during development

2. Naming Conventions Matter

• Code imported database module as both 'db' and 'pool'
• Inconsistency caused db.pool.getClient() to fail
• Standardized on 'pool' convention

3. Different Query Patterns, Different Behavior

• pool.query() - auto-manages connections (works with defaults)
• pool.getClient() - manual connection management (needs proper pool config)

4. Testing in Production Reveals Issues

• CSV import was never tested end-to-end before deployment
• Error messages were too generic initially
• Added diagnostic endpoint for future troubleshooting

8. Files Modified (Final State)

Modified:
- backend/src/config/database.js - Added pool configuration
- backend/src/routes/wiseImport.js - Fixed imports, added validation, added diagnostic endpoint
- backend/src/server.js - Removed Wise API routes, kept CSV import route
- frontend/src/components/DashboardView.jsx - Added CSV import button
- frontend/src/components/AccountingApp.jsx - Removed Wise Sync tab

Created:
- frontend/src/components/WiseImport.jsx - New CSV upload modal

Deleted:
- 8 backend files (services, controllers, routes, utils)
- 5 scripts (testing, monitoring)
- 1 frontend component (WiseReviewQueue)

9. Current Status

■ **Working:**
• CSV manual upload functional
• Transaction classification automatic
• Duplicate prevention working
• Balance updates correct
• All Wise API code removed

■ **Tested:**
• CSV import with 21-column Wise format
• Database connection pool under load
• Error handling and validation

■ **Notes:**
• Pool configured for Railway free tier limits (max 5 connections)
• Diagnostic endpoint available for troubleshooting
• Comprehensive logging added for production debugging

Git Commits: 9472b00, 16088d3, 35413cc, 6a9f43b, 8636c6d, 1a52b57, be8a265