EPC Lookup Corruption Fix - Implementation Summary

Problem Statement

Issue: Real property addresses were being overwritten with “EPC Lookup” during the data processing pipeline, resulting in corrupted output where legitimate property addresses like “32 Summerfields Drive” were replaced with the text “EPC Lookup”.

Root Cause: The EPC lookup row (a special placeholder row created in `prepareOutput()`) was being inadvertently treated as a duplicate property during the duplicate detection and merging process. When properties with the same postcode were merged, the “EPC Lookup” address could overwrite real addresses.

Solution Overview

We implemented a multi-layered defense strategy to prevent this corruption:

1. **Special Marker**: Added `_isEPCLookupRow: true` to identify EPC lookup rows
2. **Skip Logic**: Modified duplicate detection to skip EPC lookup rows entirely
3. **Merge Protection**: Added validation in merge function to reject EPC lookup merges
4. **Comprehensive Logging**: Added debug logging throughout the pipeline to track address changes
5. **Output Validation**: Added final validation to detect corruption in output

---

Implementation Details

1. Modified Files

`src/utils/epcHandler.js`

- **Function**: `createEPCLookupRow()`
- **Changes**:
- Added `_isEPCLookupRow: true` marker to the created row
- Added debug logging to track when EPC lookup rows are created
- Added warning that this row should not be merged with real properties

```
const epcLookupRow = {
    'Address': 'EPC Lookup',
    'Postcode': targetPostcode,
    // ... other fields ...
    '_isEPCLookupRow': true  // ← Special marker
};
```

`src/utils/duplicateDetector.js`

- **Function**: `detectAndMergeDuplicates()`
- **Changes**:
- Added check to skip EPC lookup rows: `if (property.Address === 'EPC Lookup' || property._isEPCLookupRow)`
- Added comprehensive debug logging for each property being processed
- Enhanced logging to show address, postcode, and isTarget for each property
- **Function**: `mergeProperties()`
- **Changes**:
- Added CRITICAL VALIDATION to reject merge attempts involving EPC lookup rows
- Returns the non-EPC-Lookup property unchanged if merge is attempted
- Added extensive logging to track:
  - Input properties (existing and new)
  - Completeness scores
  - Base property selected for merge
  - Final merge result
  - Corruption detection if final address is “EPC Lookup”

```
// CRITICAL VALIDATION: Never merge with "EPC Lookup" row
const existingIsEPCLookup = existing.Address === 'EPC Lookup' || existing._isEPCLookupRow;
const newDataIsEPCLookup = newData.Address === 'EPC Lookup' || newData._isEPCLookupRow;

if (existingIsEPCLookup || newDataIsEPCLookup) {
    log.error('❌ CRITICAL: Attempted to merge with "EPC Lookup" row!');
    // Return non-EPC-Lookup property unchanged
    if (existingIsEPCLookup) {
        return newData;
    } else {
        return existing;
    }
}
```

`src/main.js`

- **Function**: `prepareOutput()`
- **Changes**:
- Added comprehensive logging at the start to show input properties
- Added validation to detect if any property has “EPC Lookup” as address before EPC lookup row is created (early corruption detection)
- Added detailed logging for each property category (Rightmove searches, individual listings, etc.)
- Added final output validation to:
  - Detect corrupted properties (Address = “EPC Lookup” but no marker)
  - Verify EPC lookup row exists and is correct
  - Verify target property is present and not corrupted

```
// CRITICAL VALIDATION: Check if any property already has "EPC Lookup" as address
const epcLookupProperties = properties.filter(p => p.Address === 'EPC Lookup');
if (epcLookupProperties.length > 0) {
    log.error('❌❌❌ CORRUPTION DETECTED IN INPUT:');
    // Log details...
}
```

2. Test Implementation

Created `test-epc-lookup-corruption-fix.js` with three comprehensive tests:

Test 1: EPC Lookup Row Creation
- Validates that `createEPCLookupRow()` creates row with correct marker
- Checks Address = “EPC Lookup” and `_isEPCLookupRow = true`
- **Result**: ✅ PASSED

Test 2: Duplicate Detection with EPC Lookup Row
- Tests scenario with 2 duplicate properties + 1 EPC lookup row
- Validates that:
- EPC lookup row is preserved (not merged)
- Duplicate properties are merged normally (2→1)
- No corrupted properties in result
- **Result**: ✅ PASSED

Test 3: Direct Merge Attempt
- Tests merging a real property with EPC lookup row
- Validates that merge is rejected and real property is preserved
- **Result**: ✅ PASSED

Testing Results

```
================================================================================
🧪 TEST: EPC Lookup Corruption Fix
================================================================================

✅ TEST 1 PASSED: EPC Lookup row created correctly with marker
✅ TEST 2 PASSED: Duplicate detection correctly handles EPC Lookup row
✅ TEST 3 PASSED: Merge function correctly rejected EPC Lookup merge

================================================================================
📊 TEST SUMMARY
================================================================================

The fixes include:
  1. ✅ Added _isEPCLookupRow marker to identify special rows
  2. ✅ Skip EPC Lookup rows in duplicate detection
  3. ✅ Reject merge attempts involving EPC Lookup rows
  4. ✅ Added comprehensive logging throughout the pipeline
  5. ✅ Added final output validation to detect corruption
```

Debug Logging Features

The enhanced logging will help track down corruption issues in production:

1. Duplicate Detection Logging

```
🔍 Processing property 1/3:
   Address: "32 Summerfields Drive"
   Postcode: "DN9 3BG"
   isTarget: 0
```

2. Merge Operation Logging

```
🔄 MERGE PROPERTIES DEBUG
──────────────────────────────────────────────────
📝 EXISTING property:
   Address: "45 Main Street"
   Postcode: "DN15 7LQ"
📝 NEW property:
   Address: "32 Main Street"
   Postcode: "DN15 7LQ"
📊 Completeness scores: existing=10, new=12
🎯 Base for merge: NEW
   Base Address: "32 Main Street"
✅ MERGE RESULT:
   Final Address: "32 Main Street"
```

3. Output Validation Logging

```
📋 PREPARING OUTPUT
══════════════════════════════════════════════════
Total properties to process: 25
Target property: "45 Main Street", DN15 7LQ

🔍 VALIDATING FINAL OUTPUT...
✅ EPC Lookup row: OK (Address="EPC Lookup", Postcode=DN15 7LQ)
✅ Target property: OK (Address="45 Main Street", Postcode=DN15 7LQ)
```

Branch and Commit Information

- **Branch**: `fix/epc-lookup-corruption`
- **Modified Files**:
- `src/utils/epcHandler.js`
- `src/utils/duplicateDetector.js`
- `src/main.js`
- **New Files**:
- `test-epc-lookup-corruption-fix.js`
- `EPC_LOOKUP_CORRUPTION_FIX_SUMMARY.md` (this file)

How to Verify the Fix

1. **Run the test script**:
   ```
   bash
   node test-epc-lookup-corruption-fix.js
   ```
   All tests should pass with ✅ indicators.

2. **Check logs in production**:
   Look for these log messages:
   - ⚠️ `Skipping duplicate detection for "EPC Lookup" row` - Good, means protection is working
   - ❌ `CRITICAL: Attempted to merge with "EPC Lookup" row!` - If you see this, investigate what caused it
   - ❌❌❌ `CORRUPTION DETECTED` - If you see this, there's a new bug to fix

3. **Validate output CSV**:
   - Check that only ONE row has Address = "EPC Lookup"
   - Check that all other addresses are real property addresses
   - Verify target property address is correct (not "EPC Lookup")

Prevention Strategy

The multi-layered approach ensures:

1. **Prevention**: Skip EPC lookup rows in duplicate detection
2. **Protection**: Reject merge attempts at the merge function level
3. **Detection**: Validate final output and log errors
4. **Debugging**: Comprehensive logging to identify root cause if new issues arise

This defense-in-depth strategy ensures that even if one layer fails, the others will catch and prevent corruption.

Next Steps

1. ✅ Test the fix with sample data
2. ✅ Commit changes to branch
3. 🔄 Push to GitHub
4. 🔄 Create pull request
5. ⏳ Review and merge
6. ⏳ Test in production with real data

Notes

- The `_isEPCLookupRow` marker is intentionally kept as an internal field (starts with underscore)
- The marker is not exported to the final CSV output
- All validation checks use both the marker AND the Address field for maximum safety

- Logging is comprehensive but performance impact is minimal (only logs during processing, not scraping)

Author

DeepAgent - Abacus.AI
Date: December 8, 2025
Issue: EPC Lookup address corruption bug