File Hash Deduplication Implementation Report

Overview

This report documents the implementation and testing of the file hash deduplication feature in the PSScript platform. The feature prevents duplicate scripts from being uploaded to the database by calculating a unique hash for each script file and checking for existing matches.

Implementation Details

Components

The file hash deduplication feature is implemented across several components:

1. **Database Schema**:
2. The `scripts` table includes a `file_hash` column (VARCHAR(255)) to store the MD5 hash of each script.

3. An index (`idx_scripts_file_hash`) was created on this column to speed up duplicate detection.

4. **Script Model** (`src/backend/src/models/Script.ts`):

5. The Script model includes a `fileHash` field defined as: `typescript fileHash: { type: DataTypes.STRING(32), allowNull: true, field: 'file_hash' }`

6. **File Integrity Utilities** (`src/backend/src/utils/fileIntegrity.ts`):

7. `calculateBufferMD5`: Calculates the MD5 hash of a file buffer.
8. `calculateStringMD5`: Calculates the MD5 hash of a string.
9. `checkFileExists`: Checks if a file with the same hash already exists in the database.
10. `verifyFileIntegrity`: Verifies file integrity by comparing hashes.
11. `updateFileHash`: Updates file hash in the database.

12. `batchUpdateFileHashes`: Batch updates file hashes for scripts without hashes.

13. **Script Controller** (`src/backend/src/controllers/ScriptController.ts`):

14. When a script is uploaded, the controller:
    1. Calculates the MD5 hash of the file content.
    2. Checks if a script with the same hash already exists.
    3. If a match is found, rejects the upload with a 409 Conflict response.
    4. If no match is found, saves the script with its hash.

Workflow

The file hash deduplication workflow is as follows:

1. User uploads a PowerShell script through the API.
2. The system calculates an MD5 hash of the file content.
3. The system checks if a script with the same hash exists in the database.
4. If a match is found:
5. The upload is rejected with a 409 Conflict response.
6. The user is informed that a script with identical content already exists.
7. The existing script ID is provided in the response.
8. If no match is found:
9. The script is saved to the database with its hash.
10. The script file is stored on the server.
11. A success response is returned to the user.

Testing Results

We conducted comprehensive testing of the file hash deduplication feature using various test scripts and scenarios:

Test 1: Upload Original Script

- **Script**: `test-script-unique.ps1`
- **Hash**: `7f2a8ffa3577ad81591e7f24d53bd642`
- **Result**: Successfully uploaded with ID 20
- **Conclusion**: The system correctly accepts a new script and stores its hash.

Test 2: Upload Duplicate Script

- **Script**: `test-script-unique.ps1` (same content)
- **Hash**: `7f2a8ffa3577ad81591e7f24d53bd642`
- **Result**: Rejected with 409 Conflict, referencing existing script ID 20
- **Response**: `json { "error": "duplicate_file", "message": "A script with identical content already exists", "existingScriptId": 20 }`
- **Conclusion**: The system correctly identifies duplicate content based on file hash.

Test 3: Upload Modified Script

- **Script**: `test-script-unique-modified.ps1` (modified version with parameter added)
- **Hash**: `43294651b4d5318371cc5d803cfb5746`
- **Result**: Successfully uploaded with ID 21
- **Conclusion**: The system correctly accepts a script with different content, even if it's similar to an existing script.

Test 4: Upload Duplicate of Modified Script

- **Script**: `test-script-unique-modified.ps1` (same content as Test 3)
- **Hash**: `43294651b4d5318371cc5d803cfb5746`

- **Result**: Rejected with 409 Conflict, referencing existing script ID 21
- **Conclusion**: The system correctly identifies duplicate content of the modified script.

Benefits

The file hash deduplication feature provides several benefits:

1. **Storage Efficiency**: Prevents duplicate scripts from consuming storage space.
2. **Data Integrity**: Ensures that each script in the database is unique.
3. **User Experience**: Informs users when they attempt to upload a script that already exists.
4. **Search Optimization**: Eliminates duplicates from search results.
5. **Performance**: Uses an indexed hash field for fast duplicate detection.

Recommendations for Future Enhancements

Based on our implementation and testing, we recommend the following enhancements:

1. **Similarity Detection**: Implement fuzzy matching to detect scripts that are similar but not identical.
2. **Version Control Integration**: Allow users to create new versions of existing scripts instead of requiring them to upload as new scripts.
3. **Duplicate Management**: Provide tools for administrators to manage and merge duplicate scripts.
4. **Alternative Hash Algorithms**: Support additional hash algorithms (SHA-256, etc.) for improved security.
5. **Content-Based Deduplication**: Implement more sophisticated content analysis to detect functionally equivalent scripts with minor formatting differences.

Conclusion

The file hash deduplication feature is working correctly and effectively prevents duplicate scripts from being uploaded to the database. The implementation is robust and provides a good foundation for future enhancements.

Generated 2026-01-13 06:26 UTC