# AI Script Analyzer Fixes

This document outlines the fixes implemented to address issues with the AI script analyzer in the PSScript application.

## Issues Fixed

1. **Field Mapping Mismatch**: There was a mismatch between the field names used in the AI service (`optimization`) and the ones expected in the backend controller (`optimization_suggestions`).
2. **MS Learn Documentation References Format**: The format of Microsoft Learn documentation references was inconsistent, causing display issues in the frontend.
3. **Rating Scale Inconsistencies**: Rating scales weren't properly documented and displayed in the analysis results.
4. **Error Handling in Test Scripts**: Improved error handling in test scripts to better handle edge cases and missing fields.

## Implementation Details

### 1. Field Mapping Fix

The field mapping issue was fixed in `ScriptController.ts` by updating references from `optimization_suggestions` to `optimization` to match what the AI service returns:

```
// Before
optimizationSuggestions: analysis.optimization_suggestions || [],

// After
optimizationSuggestions: analysis.optimization || [], // Fixed fi
```

### 2. MS Learn Documentation References Format Fix

The MS Learn documentation references are now consistently formatted as objects with `command`, `url`, and `description` fields:

```
// Mock data with proper MS Learn reference format
ms_docs_references: [
  {
    command: 'PowerShell Scripts',
    url: 'https://learn.microsoft.com/en-us/powershell/scripting/
    description: 'Overview of PowerShell scripting'
  },
  {
    command: 'About Scripts',
    url: 'https://learn.microsoft.com/en-us/powershell/module/mic
    description: 'Information about PowerShell scripts and execut
  }
]
```

### 3. Rating Scale Documentation

The AI analyzer now includes clear rating scale guidelines in the prompt template:

- **Security Score**:
- 1-3: Minimal security risks
- 4-6: Moderate security risks that should be addressed
- 7-10: Severe security risks requiring immediate attention
- **Code Quality Score**:
- 1-4: Poor code quality requiring significant refactoring
- 5-7: Acceptable code with some improvements needed
- 8-10: Excellent code following best practices
- **Risk Score**:
- 1-3: Minimal execution risk
- 4-6: Moderate execution risk requiring caution
- 7-10: High execution risk requiring careful review and controlled environment
- **Reliability Score**:
- 1-4: Poor error handling requiring significant improvements
- 5-7: Adequate error handling with some improvements needed
- 8-10: Robust with excellent error handling

### 4. Error Handling in Test Scripts

The test script was enhanced to handle edge cases and missing fields:

```
// Improved handling of MS Learn documentation references
if (analysis.ms_docs_references && analysis.ms_docs_references.le
  analysis.ms_docs_references.forEach((ref, index) => {
    console.log(`${index + 1}. ${ref.command || 'Command'}: ${ref
    console.log(`   ${ref.description || 'No description provided
  });
}

// Improved handling of command details with support for both arra
if (analysis.command_details) {
  // Handle both array and object format
  if (Array.isArray(analysis.command_details)) {
    analysis.command_details.forEach((detail, index) => {
      console.log(`- Command ${index + 1}: ${JSON.stringify(detai
    });
  } else if (typeof analysis.command_details === 'object' && analy
    Object.entries(analysis.command_details).forEach(([command, de
      console.log(`- ${command}: ${typeof details === 'string' ? d
    });
  }
}
```

## Testing the Fixes

A new test script has been added to verify the fixes:

```
# Run the test script
./test-run-script-analysis.sh
```

This script will:

1. Test the AI script analyzer with a sample PowerShell script
2. Display the analysis results with properly formatted fields
3. Validate that all the field mappings are correct

## Future Improvements

1. Add a linting rule to ensure consistent field naming between backend and AI service
2. Create a shared type definition for analysis results to prevent field mapping issues
3. Enhance the rating scale display in the frontend with visual indicators (e.g., color-coding)
4. Implement fallback mechanism for all fields to ensure consistent data structure

Generated 2026-01-16 21:23 UTC