📊 Advanced Column Mapping & Transformation Tool

## Complete Architecture & Flow Guide

**Document Version: 1.0**Date: June 4, 2025  
Author: Technical Documentation

# Table of Contents

1. Executive Summary  
2. File Structure & Responsibilities  
3. Application Flow Sequence  
4. Inter-Module Dependencies  
5. Key Processing Points  
6. User Journey Flow  
7. Technical Details  
8. Enhanced Features

# 1. Executive Summary

The Advanced Column Mapping & Transformation Tool is a Streamlit-based web application designed to help users map, transform, and consolidate data from multiple input files into a single output file. The tool provides an intuitive interface for managing column mappings, applying filters, handling static values, and formatting data with enhanced mapping file support for configuration reusability.

### Key Capabilities:

* Multi-file data consolidation
* Dynamic column mapping
* Advanced filtering capabilities
* Static value assignment
* Date formatting controls
* Enhanced mapping file format with backward compatibility
* Real-time error validation
* Multiple export formats (Excel, TXT, CSV)

# 2. File Structure & Responsibilities

## 2.1 app.py - Main Orchestrator (Entry Point)

Primary Role: Central controller that coordinates the entire application workflow

### Key Responsibilities:

* Application Configuration: Sets up Streamlit page configuration, layout, and styling
* File Upload Management: Handles file uploads for input files, output templates, and mapping configurations
* Sheet Selection Logic: Manages Excel sheet selection for multi-sheet workbooks
* Mapping File Validation: Validates and processes both basic and enhanced mapping file formats
* Workflow Coordination: Orchestrates the complete data processing pipeline
* Enhanced Features Support: Detects and enables advanced mapping capabilities

## 2.2 ui\_sections.py - User Interface Components

Primary Role: Provides reusable UI components for clean code separation

### Key Responsibilities:

* File Upload Interface: Creates the file upload section with proper styling and validation
* User Guide Display: Renders comprehensive help documentation with examples
* Footer Management: Displays application footer and branding
* UI Consistency: Ensures consistent styling and user experience across components

## 2.3 file\_utils.py - File Operations & Utilities

Primary Role: Handles all file reading and data validation operations

### Key Responsibilities:

* File Reading: Efficiently reads CSV and Excel files with optimized performance
* Data Validation: Handles file format validation and error reporting
* Column Management: Ensures required columns exist in DataFrames
* Error Handling: Provides comprehensive error reporting for file-related issues

## 2.4 mapping\_logic.py - Core Business Logic Engine

Primary Role: The heart of the application - contains all mapping and processing logic

### Key Responsibilities:

* Mapping Interface Creation: Builds dynamic tabbed interface for each input file/sheet
* Enhanced Mapping Support: Processes and applies enhanced mapping configurations
* Data Processing: Handles filtering, static value assignment, and date formatting
* Output Generation: Creates final consolidated output with multiple export options
* Configuration Export: Exports current mapping configurations for reuse
* Real-time Validation: Provides immediate feedback on mapping errors

# 3. Application Flow Sequence

## Phase 1: Initialization & Setup

1. Application Startup (app.py)  
 • Import all required dependencies (streamlit, pandas, custom modules)  
 • Configure Streamlit application (page title, layout, colors)  
 • Set maximum upload size limits  
 • Display application header and introduction

## Phase 2: File Upload & Validation

2. File Upload Process (ui\_sections.py → app.py)  
 • Display upload interface via show\_upload\_section()  
 • Accept input files (multiple CSV/Excel files)  
 • Accept output template (single CSV/Excel file)  
 • Accept optional mapping file (CSV/Excel with mapping config)  
 • Validate uploaded files and report errors  
   
 3. Sheet Selection Logic (app.py)  
 • Process Excel files to extract sheet information  
 • Display multiselect widget for each Excel file  
 • Allow users to select specific sheets for processing  
 • Handle CSV files (no sheet selection needed)  
 • Build input\_file\_sheets list with all combinations

## Phase 3: Mapping File Processing

4. Mapping File Analysis (app.py)  
 • Read and validate mapping file structure  
 • Check for required basic columns (FileName, SheetName, OutputColumn, InputColumn)  
 • Detect enhanced features (StaticValue, FilterValues, DateFormatFlag, IncludeFlag)  
 • Add missing columns with defaults for backward compatibility  
 • Notify user when enhanced features are detected  
 • Set mapping\_file\_valid flag for downstream processing

## Phase 4: Core Mapping Interface

5. Dynamic Mapping Interface Creation (mapping\_logic.py)  
 • Create dedicated tab for each input file/sheet combination  
 • Read and process input data with performance optimization  
 • Handle column header detection and row offset  
 • Strip whitespace and deduplicate column names  
 • Parse enhanced mapping configuration  
 • Build comprehensive mapping interface with:  
 - Include/Exclude checkboxes  
 - Output column display  
 - Input column mapping dropdowns  
 - Static value text inputs  
 - Dynamic filter multiselects  
 - Date formatting checkboxes

## Phase 5: Data Processing & Output

6. Final Output Processing (mapping\_logic.py)  
 • Apply user-selected filters to input data  
 • Process column mappings with comprehensive logic  
 • Handle static values, column mappings, and blank mappings  
 • Apply date formatting when enabled  
 • Validate all mappings for completeness  
 • Combine all processed DataFrames into final result  
 • Generate multiple export formats (Excel, TXT, Enhanced Mapping CSV)

# 4. Key Processing Points

## 4.1 Enhanced Mapping File Support

The application provides full backward compatibility while enabling powerful new features through enhanced mapping files. Basic mapping files continue to work seamlessly, while enhanced files unlock advanced capabilities.

* Static Values: Assign fixed values to output columns
* Filter Values: Apply comma-separated filter criteria to input data
* Date Format Flags: Control date formatting on a per-column basis
* Include Flags: Control column inclusion/exclusion in output

## 4.2 Dynamic User Interface Generation

The interface adapts dynamically to the uploaded data and configuration, providing contextual controls and real-time validation.

# 5. User Journey Flow

1. 1. User Access: Navigate to Streamlit application with styled interface
2. 2. File Upload: Upload input files, output template, and optional mapping file
3. 3. Configuration: Select Excel sheets and configure mappings for each file/sheet
4. 4. Mapping Setup: Toggle column inclusion, map columns, assign static values, apply filters
5. 5. Output Generation: Click 'Generate Final Output' and receive validation feedback
6. 6. Export: Download consolidated Excel file, TXT file, and enhanced mapping configuration

# 6. Enhanced Features

## 6.1 Advanced Mapping Capabilities

The application supports sophisticated mapping configurations that go beyond simple column-to-column mapping, enabling complex data transformation workflows.

## 6.2 Configuration Management

Enhanced mapping files provide comprehensive configuration export including all user settings, enabling team collaboration and process documentation.

### Example Enhanced Mapping Configuration:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **FileName** | **SheetName** | **OutputColumn** | **InputColumn** | **StaticValue** | **FilterValues** | **DateFormatFlag** | **IncludeFlag** |
| data.xlsx | Sheet1 | CustomerName | Full\_Name |  |  | False | True |
| data.xlsx | Sheet1 | Status |  | Active |  | False | True |
| data.xlsx | Sheet1 | ProcessDate | Order\_Date |  |  | True | True |
| data.xlsx | Sheet1 | Category | Product\_Type |  | Premium,Standard | False | True |

# 7. Conclusion

This architecture provides a robust, scalable, and user-friendly solution for complex data mapping and transformation tasks while maintaining simplicity for basic use cases. The modular design enables easy maintenance and extension, while the enhanced mapping file format provides powerful configuration management capabilities for professional data processing workflows.

================================================================================

**Document End***This document provides a comprehensive technical overview of the Advanced Column Mapping & Transformation Tool architecture. For additional technical details or specific implementation questions, refer to the source code documentation within each module.*