Building Data Structuring Tool for Excel

Background

Organizations often deal with large volumes of Excel data that require cleaning, transformation, and formatting before analysis. Manual processing is time-consuming and error-prone. Your task is to build a tool that simplifies this process for end users.

Objective

Design and implement a user-friendly application that allows users to upload Excel files, specify data transformation logic, and receive a processed Excel file as output.

Functional Requirements

Step 1: File Upload

- The user should be able to upload one or more Excel files (.xlsx).
- The tool should validate the file format and structure.

Step 2: Logic Input

- Prompt the user to specify the operations they want to perform. Examples:
    - Remove duplicates
    - Filter rows based on conditions
    - Replace values
    - Merge columns
    - Convert date formats
    - Normalize text (e.g., lowercase, trim spaces)
- Mathematical Operations like, but not limited to
    - Add, subtract, multiply, divide columns
    - Apply formulas (e.g., percentage change, weighted average)
    - Aggregate functions (sum, mean, median, min, max)
    - Conditional calculations (e.g., if column A > 100, then...)

  Students must design a flexible input method (form, dropdown, or text parser) to capture user logic

Step 3: Execution

- The tool should parse the logic and apply the transformations.
- Handle errors gracefully (e.g., invalid column names, unsupported operations).

Step 4: Output

- Generate a new Excel file with the transformed data.
- Provide a download link or email the file to the user.

Technical Expectations

- Use Python (e.g., Pandas, OpenPyXL) or any other suitable language.
- Build a simple UI
- Ensure modular code for easy extension. **Mandatory:** Every function and class must include clear, concise documentation using docstrings and inline comments.
- Add a preview feature before final output.
- Support for multiple sheets and large files

Good to have:

- Allow saving and reusing transformation templates.
- Include logging for audit and debugging.

Deliverables

- Source code with documentation.
- A short demo video or walkthrough.
- Sample input and output files.

Milestone

#	Milestone	Description	Timeline
1	Requirements & Design	Define UI flow, backend logic, and user interaction model	Week 1
2	File Upload Module	Implement file validation and preview	Week 2
3	Logic Input Interface	Build UI for selecting operations and entering formulas	Week 3
4	Transformation Engine	Code backend logic to apply operations	Week 4
5	Output Generation	Create downloadable Excel output	Week 5
6	Documentation & Testing	Add docstrings, comments, and test cases	Week 6
7	Final Demo & Submission	Present working tool with sample files	Week 7

Qualifying Criteria

#	Criteria	Description
1	Functionality	All four steps (upload, logic input, execution, output) must work end-to-end
2	Mathematical Operations	At least 3 types of math operations must be supported
3	User Experience	Interface should be intuitive and responsive
4	Error Handling	Tool must handle invalid inputs gracefully
5	Code Quality	Code must be modular, readable, and well-documented
6	Documentation	Every function/class must include docstrings and inline comments
7	Demo	A short video or live walkthrough must be provided
8	Sample Files	Include at least one input and one output Excel file