Assignment 1 : Web Application Mimicking Google Sheets

Objective:

Develop a web application that closely mimics the user interface and core functionalities of Google Sheets, with a focus on mathematical and data quality functions, data entry, and key UI interactions.

Features:

1. **Spreadsheet Interface:**
    - **Mimic Google Sheets UI:** Strive for a visual design and layout that closely resembles Google Sheets, including the toolbar, formula bar, and cell structure.
    - **Drag Functions:** Implement drag functionality for cell content, formulas, and selections to mirror Google Sheets' behavior.
    - **Cell Dependencies:** Ensure that formulas and functions accurately reflect cell dependencies and update accordingly when changes are made to related cells.
    - Support for basic cell formatting (bold, italics, font size, color).
    - Ability to add, delete, and resize rows and columns.

2. **Mathematical Functions:**
    - Implement the following mathematical functions:
        1. `SUM`: Calculates the sum of a range of cells.
        2. `AVERAGE`: Calculates the average of a range of cells.
        3. `MAX`: Returns the maximum value from a range of cells.
        4. `MIN`: Returns the minimum value from a range of cells.
        5. `COUNT`: Counts the number of cells containing numerical values in a range.
3. **Data Quality Functions:**
    - Implement the following data quality functions:
        1. `TRIM`: Removes leading and trailing whitespace from a cell.

2. `UPPER`: Converts the text in a cell to uppercase.
3. `LOWER`: Converts the text in a cell to lowercase.
4. `REMOVE_DUPLICATES`: Removes duplicate rows from a selected range.
5. `FIND_AND_REPLACE`: Allows users to find and replace specific text within a range of cells.

4. **Data Entry and Validation:**
    - Allow users to input various data types (numbers, text, dates).
    - Implement basic data validation checks (e.g., ensuring numeric cells only contain numbers).
5. **Testing:**
    - Provide a means for users to test the implemented functions with their own data.
    - Display the results of function execution clearly.

Bonus Features:

- Implement additional mathematical and data quality functions.
- Add support for more complex formulas and cell referencing (e.g., relative and absolute references).
- Allow users to save and load their spreadsheets.
- Incorporate data visualization capabilities (e.g., charts, graphs).

Evaluation Criteria:

- **Fidelity to Google Sheets UI:** How closely the application's look and feel matches Google Sheets, including drag functions and cell dependency handling.
- Functionality and completeness of the implemented features.
- Accuracy of the mathematical and data quality functions.

- Usability and intuitiveness of the user interface.
- Code quality and maintainability.
- Implementation of bonus features.
- Readme should give a clear explanation of Datastructures and tech stack used and why so.

Assignment 2: Building a Support Agent Chatbot for CDP "How-to" Questions

Objective:

Develop a chatbot that can answer "how-to" questions related to four Customer Data Platforms (CDPs): Segment, mParticle, Lytics, and Zeotap. The chatbot should be able to extract relevant information from the official documentation of these CDPs to guide users on how to perform tasks or achieve specific outcomes within each platform.

Data Sources:

- Segment Documentation: https://segment.com/docs/?ref=nav
- mParticle Documentation: https://docs.mparticle.com/
- Lytics Documentation: https://docs.lytics.com/
- Zeotap Documentation: https://docs.zeotap.com/home/en-us/

Core Functionalities:

1. **Answer "How-to" Questions:**
    - The chatbot should be able to understand and respond to user questions about how to perform specific tasks or use features within each CDP.
    - Example questions:
        - "How do I set up a new source in Segment?"
        - "How can I create a user profile in mParticle?"
        - "How do I build an audience segment in Lytics?"
        - "How can I integrate my data with Zeotap?"

2. **Extract Information from Documentation:**
    - The chatbot should be capable of retrieving relevant information from the provided documentation to answer user questions.
    - It should be able to navigate through the documentation, identify relevant sections, and extract the necessary instructions or steps.

3. **Handle Variations in Questions:**
    - Size variations. E.g extremely long question should not break it down.
    - Questions irrelevant to CDP. e.g Which Movie is getting released this week.

Bonus Features:

- **Cross-CDP Comparisons:**
    - The chatbot can answer questions about the differences in approaches or functionalities between the four CDPs.
    - Example question: "How does Segment's audience creation process compare to Lytics'?"
- **Advanced "How-to" Questions:**
    - The chatbot can handle more complex or platform-specific "how-to" questions.
    - It can provide guidance on advanced configurations, integrations, or use cases.

Evaluation Criteria:

- Accuracy and completeness of assignment.
- Code quality and build.
- Handling of variations in question phrasing and terminology.
- Implementation of bonus features (cross-CDP comparisons, advanced questions).
- Overall user experience and chatbot interaction.

Additional Notes:

- You can use any natural language processing (NLP) libraries or frameworks to build the chatbot. Instead of NLP you can also use simple document indexer.
- You can choose to implement the chatbot as a web application using any tool.
- The focus of this assignment is software engineering and not Model building.