**SheetCell Ex1**

**User Manual**

1. regarding the expression behavior: we decided to stay in the approach that was describe in the assignments word doc. An expression cannot be calculated if the arguments are not from the same type. Nan and Undefine will be shown only if the function gets the requested arguments types but cannot be calculated(dividing by 0 or a substring with end index that’s bigger then the string size).

2. we implemented the bonus for the load and save sheet.

3. all user command that request a file path will need to be an absolute path.

4. you cant concat to a number the start with spaces for example {CONCAT, HELLO, 123}.

5.you cant use ref to an empty cell.

**Architectural Choices**:

At first we worked on the core parts of the application: Engine, cell, sheet we started by declaring the main functionality of each part the main focus was how cell will be then we started thinking about how an expression will be interpreted in our program.

**Expression**

 **Initial Approach**: Initially, we represented expressions as strings, assuming that evaluating an expression would always result in a string. However, we quickly realized that the UI should receive data in its actual type to correctly display it.

 **Revised Approach**: We decided that cell evaluations could be of different types and represented them as objects. Later, we refined this further by introducing a Return Value class that could either be a string or a number.

**Functions Implementation**

**Function Representation**: Functions were represented as expressions with varying numbers of arguments (Unary, Binary, Trinary).

**Ref Function Challenge**: Initially, we deviated from the lecture's approach by deciding that the Ref function would not create a new class but would return the instance of an expression from the referenced cell. This approach made it challenging to recalculate cells when a referenced cell was updated. תמונה שמכילה טקסט, צילום מסך, תוכנה, תכונות מולטימדיה

התיאור נוצר באופן אוטומטי

**Final Solution**: To resolve these challenges, we used recursion to traverse Ref calls and update referenced expressions. We employed an Accept and Visit design pattern to navigate the expression tree, depending on the expression type (Unary, Binary, Trinary). Ultimately, we changed the implementation to make Ref an expression and used topological sorting of the reference graph for efficient cell updates. תמונה שמכילה טקסט, צילום מסך, תוכנה

התיאור נוצר באופן אוטומטי

תמונה שמכילה צילום מסך, טקסט

התיאור נוצר באופן אוטומטי

.