Mini Excel

Session 2023-2027

Supervised By:

Sir Nazeef ul Haq

Submitted By:

Muhammad Tayyab Amir (2023-CS-101)

Muhammad Harris Amin (2023-CS-160)

University of Engineering and Technology, Lahore

Table of Contents

1. Project Overview

1.1 Description

We all have used excel for some data work and it's a very versatile and useful application. The Mini Excel project is a web-based spreadsheet application that simulates the core functionalities of popular spreadsheet software like Microsoft Excel or Google Sheets. This project aims to provide users with an interactive platform for data manipulation, computation, and organization. The development process mainly focuses on using different data structures and algorithms to enhance the functionalities of the project. By implementing these concepts, the project performs efficient computations and ensures dynamic updates in the interconnected cells.

1.2 Business Case

In today's data-driven world, spreadsheets have become an essential tool for organizing, analyzing, and presenting information across various domains such as finance, education, research, and business management. While commercial spreadsheet applications like Microsoft Excel and Google Sheets dominate the market, they often come with limitations such as high costs, excessive complexity. The mini excel is a good solution for these limitations.

1.3 Motivation

The Mini Excel project was assigned by our instructor as an opportunity to apply theoretical knowledge of data structures and algorithms in a practical setting. The goal is to understand how real-world applications depends on computational efficiency and logical structuring to handle data.

1.4 Features

1. Interactive user interface
2. Basic arithmetic operations
3. Dynamic updates
4. Error handling
5. Data structures implementation
6. Cell formatting (bold, italic, underline, alignment)
7. Functional dependencies
8. Prevents circular dependencies
9. Efficient data searching in grid
10. Undo/Redo

1.5 Technology Stack

Table 1. Technology Stack

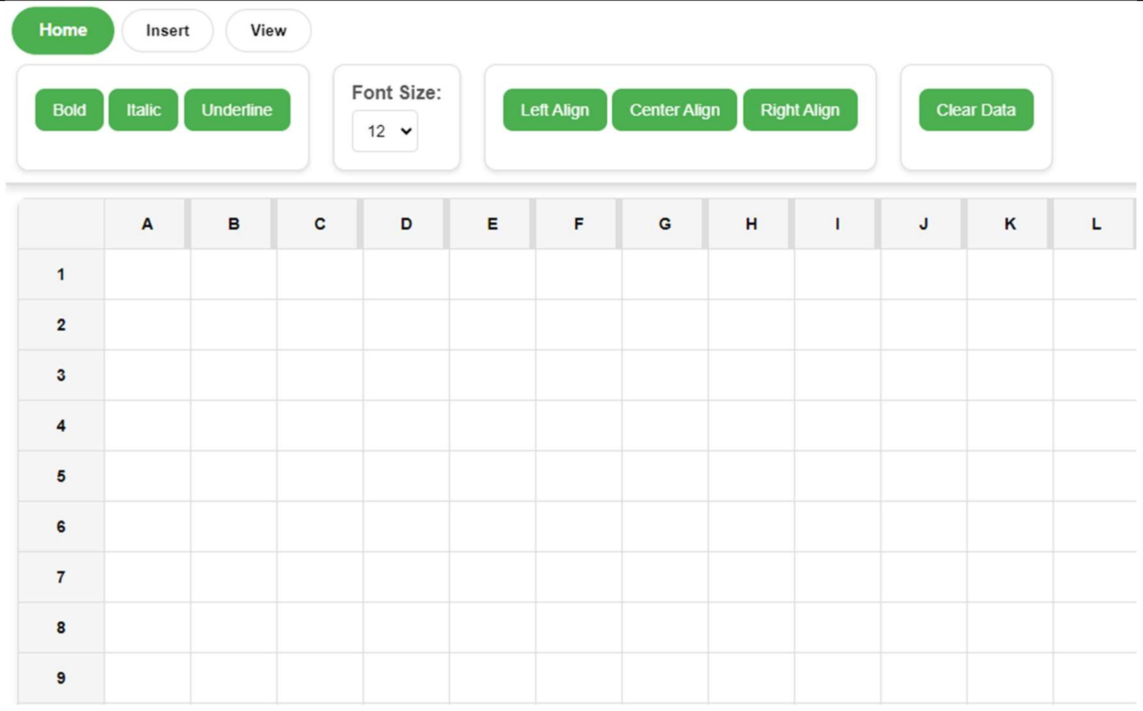
Front End	HTML, CSS, Javascript
Version control	Git
Development Environment	Visual Studio Code
Testing Tools	Console.log() and Chrome
Project Link	https://github.com/HARRISAMIN432/FinalProject OR https://github.com/TayabAmir/DSA_Final_Project

2. Use Cases

2.1 Main Screen

Table 2. Main Screen

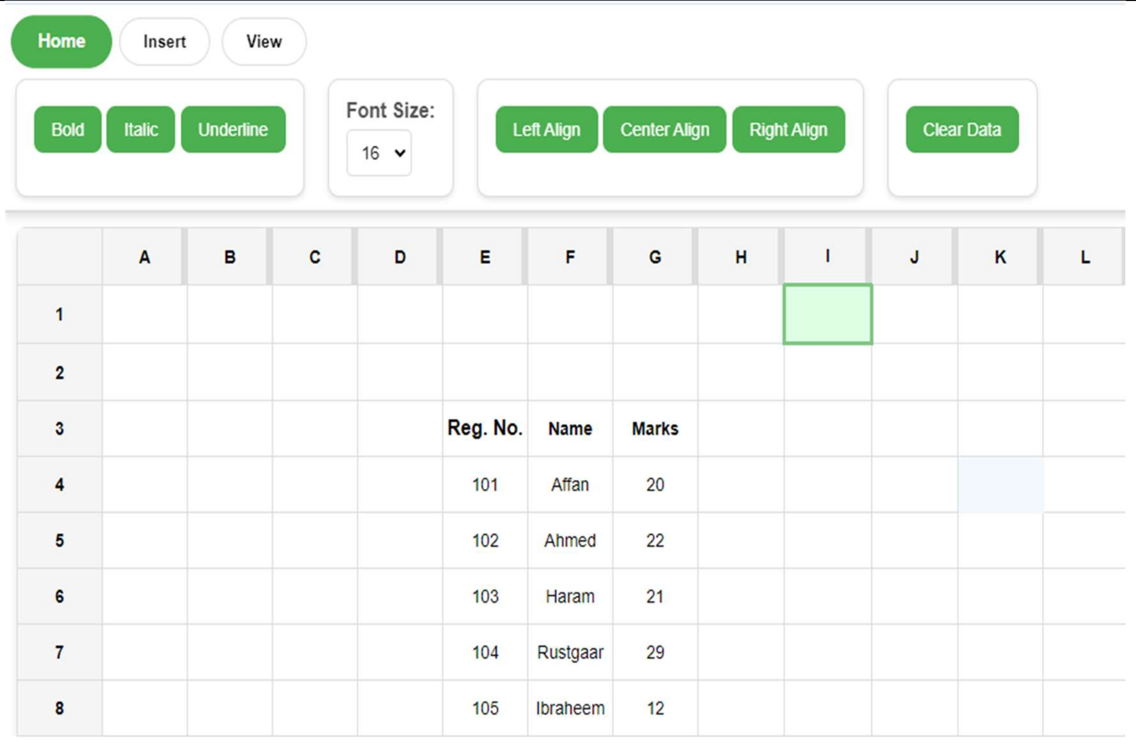
Use Case Id	U01
Name	Main Screen
Actor	User
Description	It displays the spreadsheet on the screen

UI	 Home Insert View Bold Italic Underline Font Size: 12 Left Align Center Align Right Align Clear Data A B C D E F G H I J K L 1 2 3 4 5 6 7 8 9 Fig1. Main Screen

2.2 Writing Data

Table 3. Writing data

Use Case Id	U02
Name	Writing data
Action	User
Description	It provides access to write data in cells

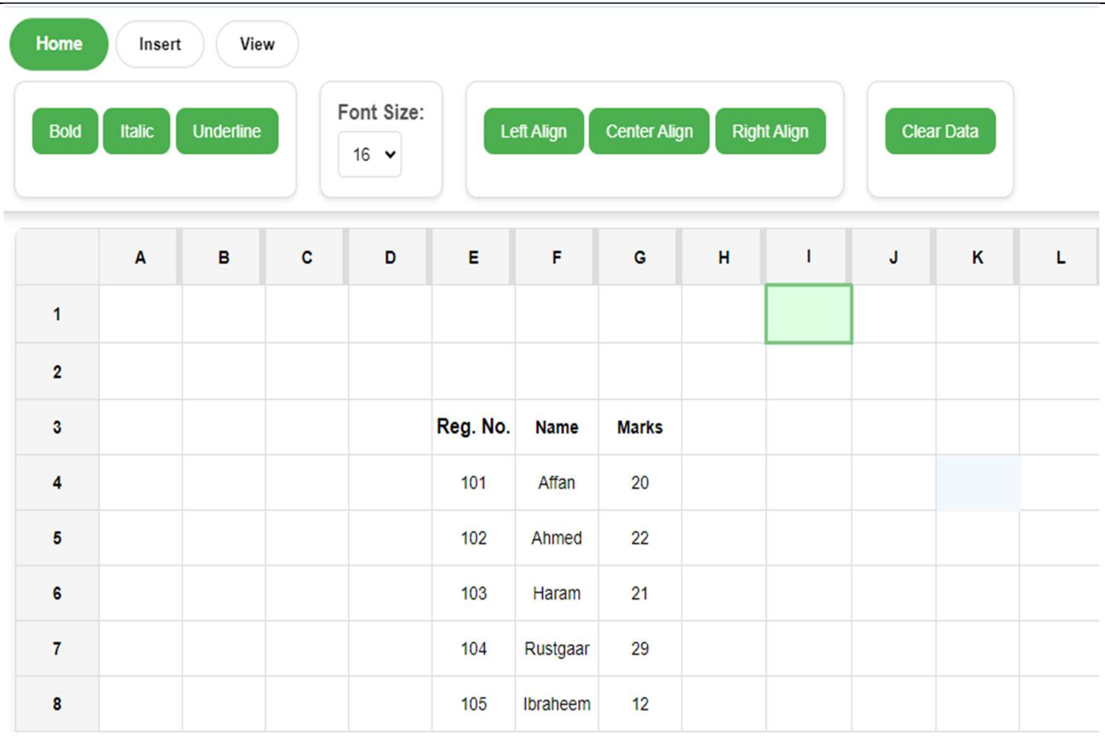
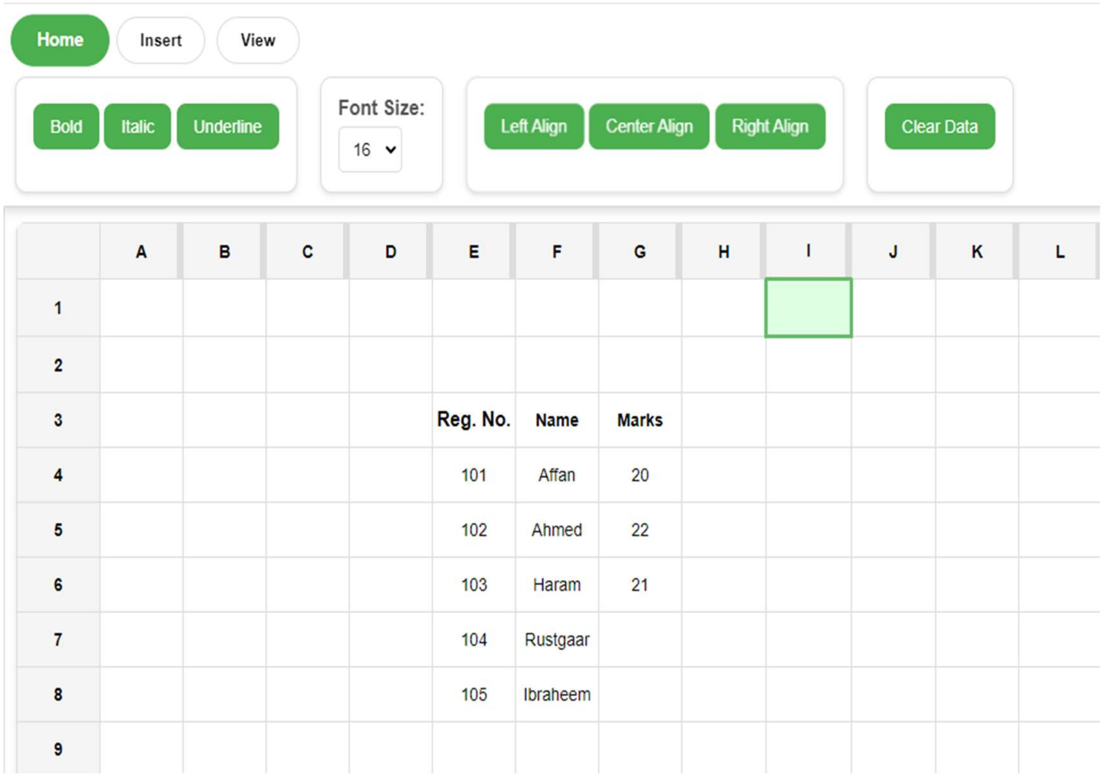
UI	 Home Insert View Bold Italic Underline Font Size: 16 Left Align Center Align Right Align Clear Data Fig. 2. Writing data in cells

	A	B	C	D	E	F	G	H	I	J	K	L
1												
2												
3					Reg. No.	Name	Marks					
4					101	Affan	20					
5					102	Ahmed	22					
6					103	Haram	21					
7					104	Rustgaar	29					
8					105	Ibraheem	12					

2.3 Undo Redo

Table 4. Undo Redo

Use Case Id	U03
Name	Undo Redo
Actor	User
Description	The user can perform undo and redo by using ctrl+z and ctrl + y respectively

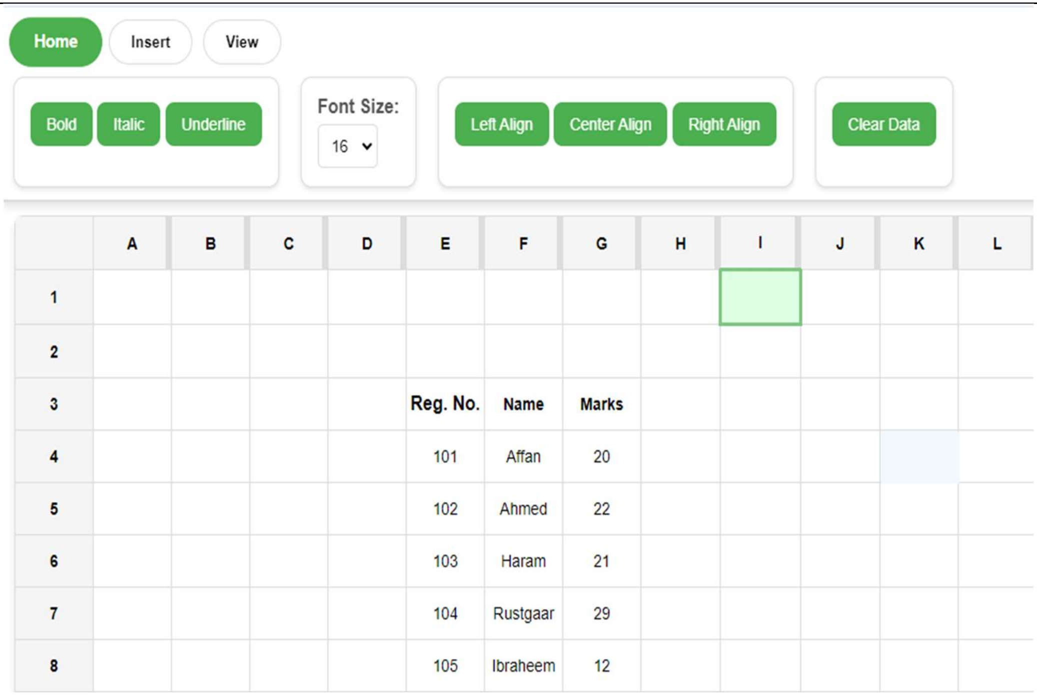
UI	 Fig 3. Simply data writing  Fig 4. After undo

Font Size: 16

	A	B	C	D	E	F	G	H	I	J	K	L
1												
2												
3					Reg. No.	Name	Marks					
4					101	Affan	20					
5					102	Ahmed	22					
6					103	Haram	21					
7					104	Rustgaar	29					
8					105	Ibraheem	12					

Font Size: 16

	A	B	C	D	E	F	G	H	I	J	K	L
1												
2												
3					Reg. No.	Name	Marks					
4					101	Affan	20					
5					102	Ahmed	22					
6					103	Haram	21					
7					104	Rustgaar						
8					105	Ibraheem						
9												

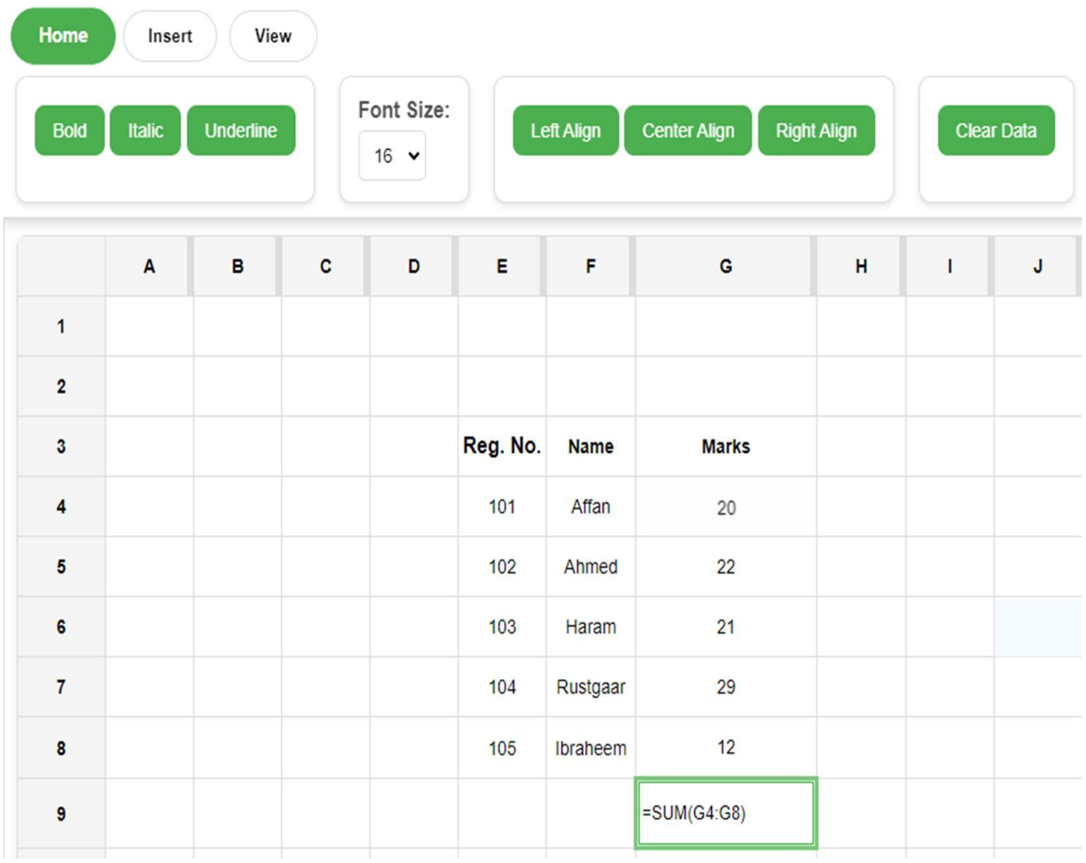
	 Fig 5. After Redo

	A	B	C	D	E	F	G	H	I	J	K	L
1												
2												
3					**Reg. No.**	**Name**	**Marks**					
4					101	Affan	20					
5					102	Ahmed	22					
6					103	Haram	21					
7					104	Rustgaar	29					
8					105	Ibraheem	12					

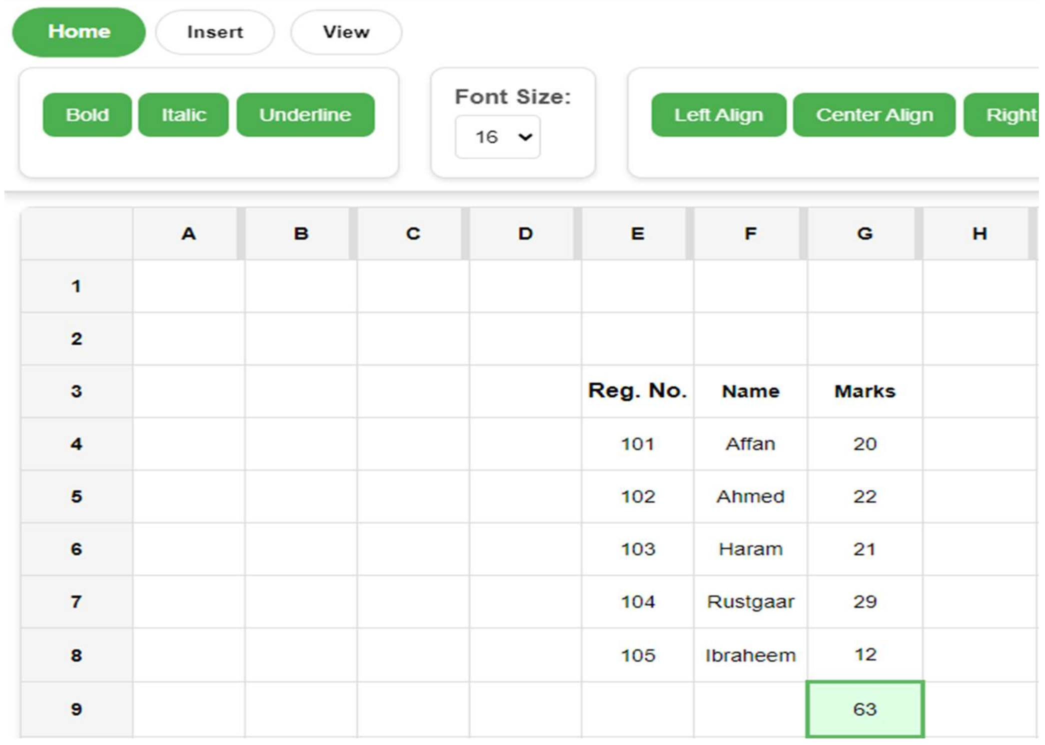
2.4 Functionalitites

Table 5. Functionalities

Use Case Id	U04
Name	Functionalities
Actor	User
Description	The user can perform several mathematical calculations on the data

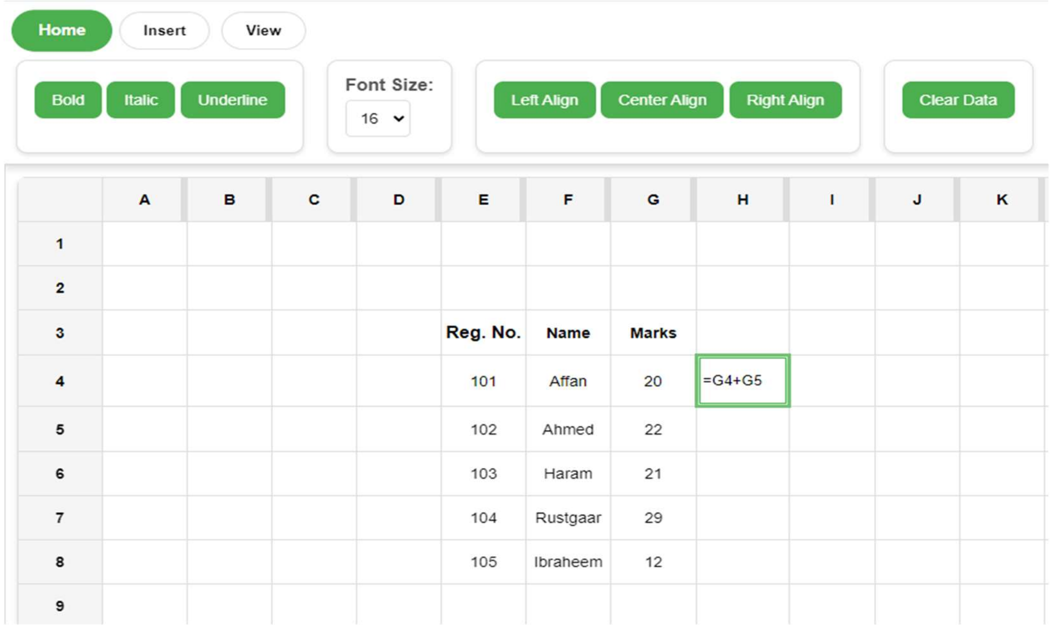
UI	 Fig 6. Applying formula

	A	B	C	D	E	F	G	H	I	J
1										
2										
3					Reg. No.	Name	Marks			
4					101	Affan	20			
5					102	Ahmed	22			
6					103	Haram	21			
7					104	Rustgaar	29			
8					105	Ibraheem	12			
9							=SUM(G4:G8)			



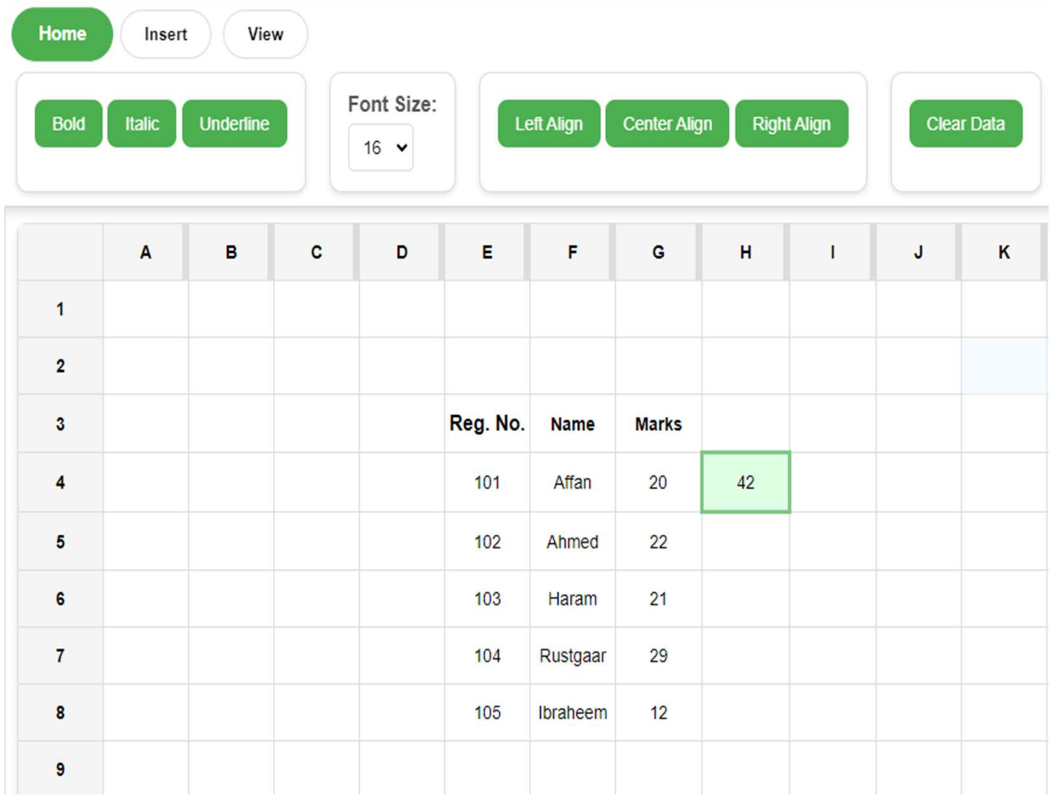
	A	B	C	D	E	F	G	H
1								
2								
3					Reg. No.	Name	Marks	
4					101	Affan	20	
5					102	Ahmed	22	
6					103	Haram	21	
7					104	Rustgaar	29	
8					105	Ibraheem	12	
9							63	

Fig 7. SUM result



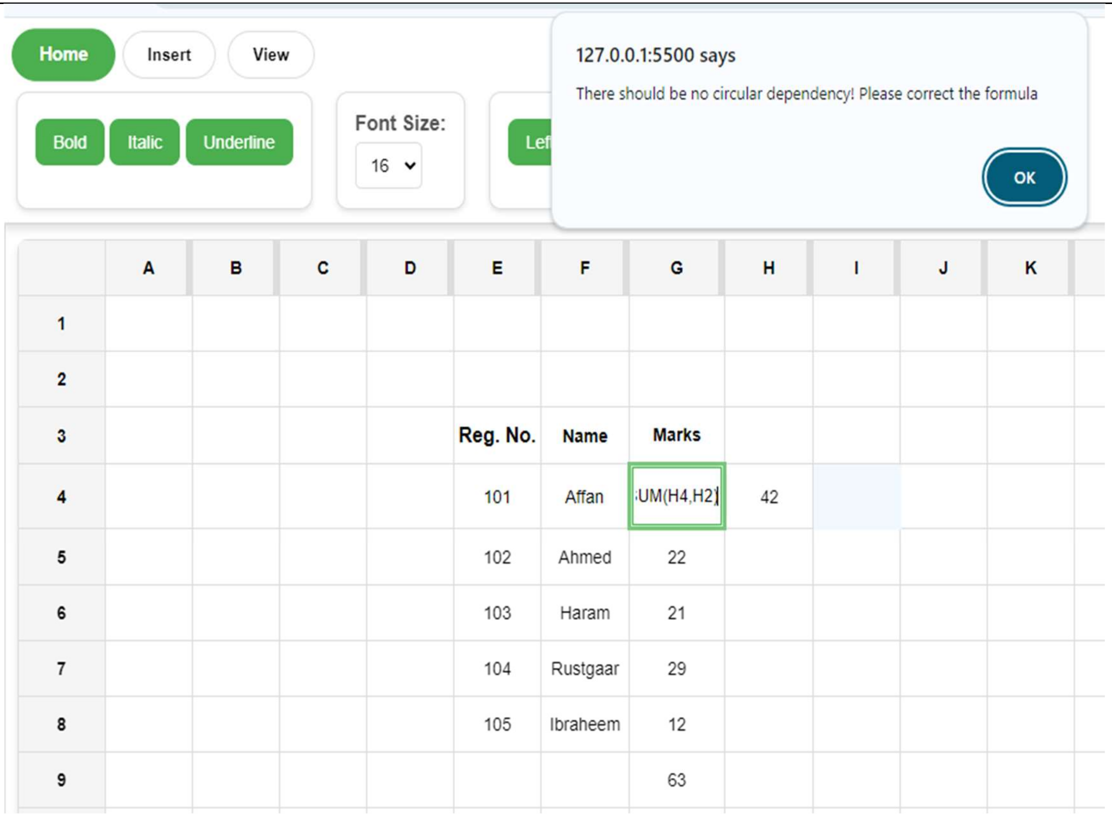
	A	B	C	D	E	F	G	H	I	J	K
1											
2											
3					Reg. No.	Name	Marks				
4					101	Affan	20	=G4+G5			
5					102	Ahmed	22				
6					103	Haram	21				
7					104	Rustgaar	29				
8					105	Ibraheem	12				
9											

Fig 8. By operators



	A	B	C	D	E	F	G	H	I	J	K
1											
2											
3					Reg. No.	Name	Marks				
4					101	Affan	20	42			
5					102	Ahmed	22				
6					103	Haram	21				
7					104	Rustgaar	29				
8					105	Ibraheem	12				
9											

Fig. 9. Arithmetic result

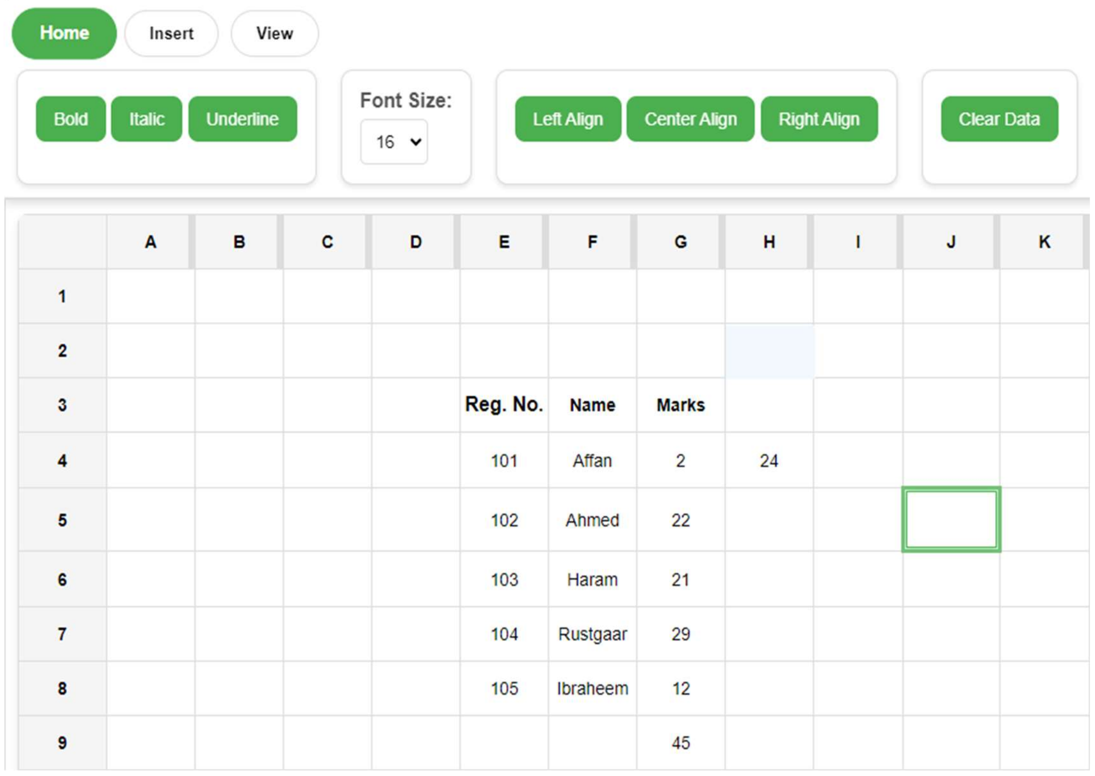


127.0.0.1:5500 says

There should be no circular dependency! Please correct the formula

	A	B	C	D	E	F	G	H	I	J	K
1											
2											
3					Reg. No.	Name	Marks				
4					101	Affan	UM(H4,H2)	42			
5					102	Ahmed	22				
6					103	Haram	21				
7					104	Rustgaar	29				
8					105	Ibraheem	12				
9							63				

Fig 11. Preventing circular dependencies



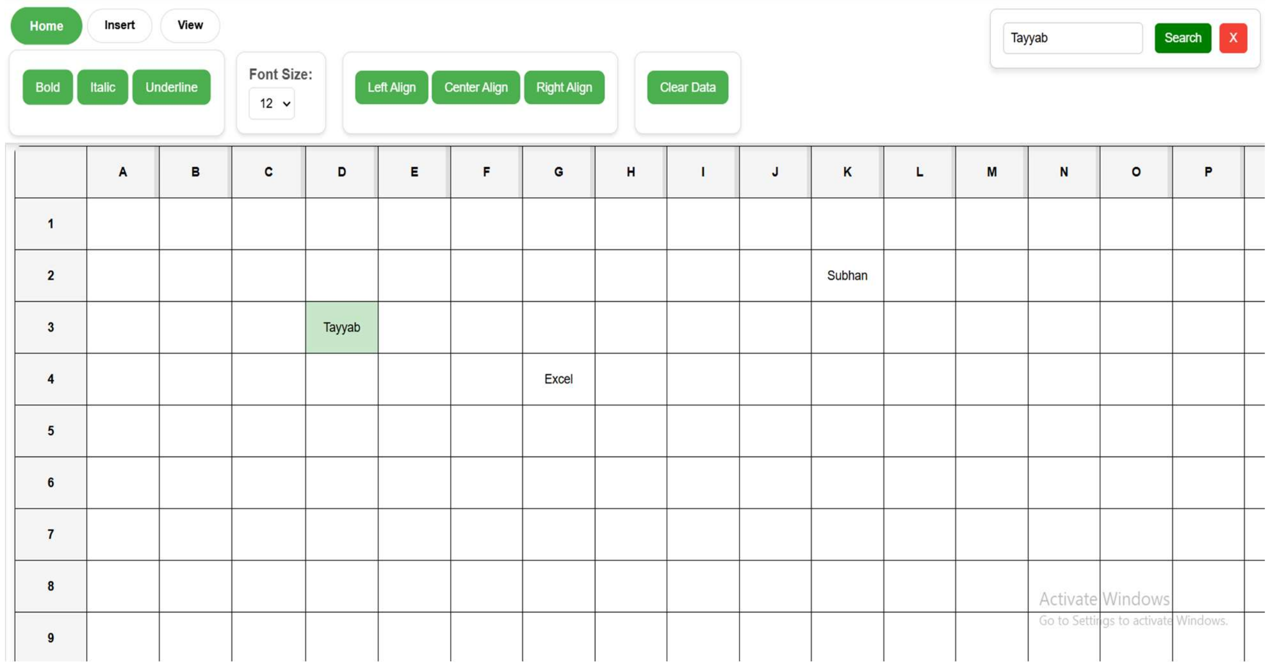
	A	B	C	D	E	F	G	H	I	J	K
1											
2											
3					Reg. No.	Name	Marks				
4					101	Affan	2	24			
5					102	Ahmed	22				
6					103	Haram	21				
7					104	Rustgaar	29				
8					105	Ibraheem	12				
9							45				

Fig. 10. Reevaluation of dependencies

2.5 Find Functionality

Table 6. Find functionality

Use Case Id	U05
Name	Find functionality
Actor	User
Description	Find data in the spreadsheet

UI	 Home Insert View Bold Italic Underline Font Size: 12 Left Align Center Align Right Align Clear Data Tayyab Search X A B C D E F G H I J K L M N O P 1 2 3 4 5 6 7 8 9 Subhan Tayyab Excel Activate Windows Go to Settings to activate Windows. Fig. 12. Searching

3. Implementation

3.1 Find Functionality

3.1.1 Linked Lists

Linked lists are utilized to represent the grid, providing a dynamic and flexible way to manage the cells. Each cell in the grid is represented as a node in the linked list, where the node contains the cell’s data and links to its adjacent cells. Specifically, every cell is connected to the cell on its left, right, above, and below, enabling easy manipulation and traversal of the grid. This structure allows for efficient row and column navigation, making it easy to access and modify the value of any cell based on its row and column number. Using a linked list helps in maintaining the grid's data efficiently, ensuring quick updates and smooth interaction between cells.

3.1.2 Directed Acyclic Graph

A Directed Acyclic Graph (DAG) is employed in the project to model and manage cell dependencies, which is crucial for ensuring accurate calculations and preventing errors in formulas. When a cell’s value depends on other cells, such as in a formula like =A1+B1, a directed edge is created from each of the dependent cells (A1 and B1) to the calculated cell. This graph structure helps track and manage these dependencies, ensuring that the necessary calculations occur in the correct order. The use of a directed acyclic graph is particularly effective because it prevents the creation of circular dependencies, where a cell might depend on itself either directly or indirectly.

3.1.3 Stacks

Stacks are used extensively throughout the project, particularly for implementing undo and redo functionalities, which enhance the user experience by providing the ability to revert changes and restore previous states. Every time a user performs an action—such as editing a cell's value, changing its formatting, or making other modifications—the action is pushed onto an undo stack. This stack operates based on the Last In First Out (LIFO) principle, meaning that the most recent action can be undone first. When an action is undone, it is moved to the redo stack, where it can be reapplied if the user decides to restore it. This dual stack structure makes it easy to navigate backward and forward through changes, allowing users to manage their edits effectively. Additionally, stacks are also used in methods that involve calculations, such as those using postfix notation, to simplify and organize the sequence of operations, ensuring that the calculations are performed in the correct order.

3.1.4 Queues

Queues follow the first in first out and this feature of queues is proved useful in techniques like topological sort which is used as detecting cycles in the Directed Acyclic Graph to prevent the happenings of circular dependencies.

3.1.5 Binary Search Trees

Binary Search Trees (BST) are used in the project to optimize the process of searching for data within the grid. BSTs are a highly efficient data structure for searching, insertion, and deletion operations, offering much better time complexity of O (log N) in best case than linear data structures like arrays or linked lists. In the context of the mini Excel application, the data entered into cells is stored within binary search trees, which allows for quick searching of specific values within the grid. By organizing the data in a hierarchical structure, BSTs enable faster access and manipulation, especially when dealing with large datasets.

3.2 Initialization:

3.2.1 Creation of Grid

The grid structure in the mini-Excel application is initialized by creating nodes one at a time and systematically inserting them into the linked list. Each node represents a cell in the grid and is connected to its neighboring nodes to establish a well-defined structure. The process involves iterating through all rows and columns, and for each cell located at position (i,j), a new node is dynamically created and inserted into the ternary linked list using the insertNode(i, j) method. This method ensures that the nodes are appropriately linked, maintaining connections to their left, right, upper, and lower neighbors wherever applicable.

3.2.2 Bidirectional Link

Each node in a linked list is connected to its dom element (td element) and each td element (cell) is connected back to the node of the linked list. This ensures easy access to the cells and nodes and makes updates in the cells easy. The appropriate event listeners are also added for the cells to enable the data to be written inside the cells.

3.2.3 Ribbon

The ribbon is a crucial component of the mini-Excel application, designed to provide users with a convenient and accessible toolbar for managing and formatting their data. Located at the top of the window, the ribbon offers a wide range of functionalities that simplify the process of organizing and decorating data within the spreadsheet. Users can easily apply formatting options like bold, italic, and underline to text, change font sizes to improve readability, and manage the layout by inserting new rows and columns as needed.

3.2.4 Initialization of Objects

At the start of the application, several critical objects are initialized to provide the foundational functionality of the mini-Excel application. These include objects for the dependency graph, binary search trees, and undo and redo stacks. The dependency graph, represented as a Directed Acyclic Graph (DAG), is created to manage cell dependencies, ensuring that formulas referencing other cells are calculated correctly and without circular dependencies. Binary search trees are initialized to store and retrieve data efficiently, particularly for search operations within the grid. The undo and redo stacks are set up to record user actions, allowing for the implementation of reversible operations.

3.3 Working:

3.3.1 Cell Selection and Navigation

Each cell in the grid can be selected by clicking on it. The selected cell is highlighted, making it easy for users to identify which cell they are working on. Keyboard navigation is enabled through the use of arrow keys (Arrow Up, Arrow Down, Arrow Left, Arrow Right), allowing users to move between cells efficiently without requiring a mouse. The interaction is supported by a ternary linked list where each node represents a cell. This linked structure ensures that neighboring cells are directly accessible, enabling smooth traversal and updates.

3.3.2 Cell Editing

Upon selecting a cell, users can input data or formulas directly. If the cell does not already contain an input element, one is dynamically created to allow for inline editing. This approach prevents unnecessary DOM elements and ensures a responsive interface. When a cell is clicked or navigated to via the keyboard, it becomes highlighted. This visual representation of the cell helps having understanding of whatever is happening within the grid. The focused cell can also be edited immediately, ensuring smooth interaction without extra clicks.

3.3.3 Undo and Redo

Each state of the cell is being saved in the undo stack as the focus on the cell is removed. The undo and redo functionality is triggered by Ctrl+z and Ctrl+y respectively. Whenever an undoAction() is triggered. The state of the cell is popped from the undo stack and pushed to the redo stack and the cell then represents the state that was popped from the undo stack. Similarly, whenever redo event is triggered the state of the cell is popped from the redo stack and pushed to the undo stack and the cell then represents the state that was popped from the redo stack.

3.3.4 Formula Functionalities

The user can perform formula functionalities by simply using functions like SUM, MUL, MAX, MIN followed by an equal (=) sign and the two cells should be passed as arguments to these functions. If both cells are separated by colon (:) in the argument, then it will perform calculations on the range of cells between these two cells including both of those cells. If both cells are separated by a comma (,) then the calculation will be performed only those 2 cells. It can also perform simple mathematical operations like =23*31 or =A1/9. If a cell is dependent on other cells, then updating the dependent cell will also change the value present in the cell.

3.3.5 Find Functionality

Whenever the focus on the cell is removed. The binary search tree is updated. You can find it in the grid by using CTRL+F and then typing in the input box. The cell with specified input will be highlighted.

3.3.6 Insert Row and Column

In the insert tab, new rows and columns can be inserted by simply clicking on the insert row and insert column respectively.

3.3.7 Other Functionalities

In the toolbar, there are functions like bold, italic, underline, left align, center align and right align. You can use them either by clicking on them or by using shortcut keys for bold, italic and underline. (CTRL+I) for italic, (CTRL+B) for bold, (CTRL+Q) for underline.

4. Test Cases

Test Case Number	Description	Steps	Expected Result
1	Verify that a user can edit a cell by entering data.	1. Click on a cell. 2. Enter data. 3. Press Enter or click outside the cell.	The data is saved in the cell and displayed properly.
2	Verify that the spreadsheet can calculate formulas correctly.	1. Enter =SUM (A1:A3) in cell A4. 2. Enter values in cells A1, A2, and A3. 3. Check the result in A4.	The correct sum of A1, A2, and A3 is displayed in A4.
3	Verify that the undo/redo functionality works correctly.	1. Edit a cell and press Enter. 2. Press CTRL+Z to undo the change. 3. Press CTRL+Y to redo the change.	The cell reverts to its previous state on undo and restores the change on redo.
4	Verify that the application prevents circular dependencies.	1. Enter a formula in A1 that references B1. 2. Enter a formula in B1 that references A1.	An error message is displayed, and the formula is not applied.
5	Verify that the find functionality locates specific data.	1. Press CTRL+F. 2. Enter the search term. 3. Click Find.	The matching cell/cells are highlighted.
6	Verify that the formatting options work.	1. Select a cell. 2. Apply bold, italic, underline or aligning formatting using the toolbar or shortcuts (CTRL+B, CTRL+I, CTRL+Q for Underline).	The text in the cell is formatted as selected.
7	Verify that rows and columns can be inserted dynamically.	1. Click the Insert Row/Column button. 2. Verify the new row/column appears at the specified location.	The row or column is added successfully without disrupting existing data.
8	Verify that errors in formulas are handled appropriately.	1. Enter an invalid formula, such as \`=A1+B\`. 2. Press Enter.	An error message is displayed on the cell as '#NAME?'.

9	Verify that theme and gridlines are toggling perfectly.	Click Toggle Theme or Toggle Gridlines button.	The result should be the opposite of previous one.
10	Verify that the data is cleared correctly.	1.Enter data in some cells. 2. Click Clear Data button.	The data should be cleared from the whole table.

5. Challenges

The challenges we faced during the project are as follows:

5.1 Complexity of Data Structures:

Implementing data structures like linked lists (with four pointers), Directed Acyclic Graphs (DAGs), and Binary Search Trees required deep understanding and adjustments according to the projects.

5.2 Evaluating Formula:

Evaluating the formula was quite challenging, as first we must find values according to the references given and then convert the infix formula to postfix and then formula is evaluated.

5.3 Cell Dependency:

Cell Dependency was also one of the things that we faced problem with as it uses the DAGs to check the circular dependency and do further operations.

5.4 Undo Redo:

Undo Redo is the most challenging thing in the project, but we implemented it with collaborative effort.

6. Future Improvements

The improvements which we can make in the future are as follows.

6.1 User Interface

Add a more modern and user-friendly interface using libraries like React.js or Next.js.

6.2 Charting

Help the user to create charts (bar, pie, line, etc.) directly from the data.

6.3 Other Functionalities of Excel

There are many other functionalities like inserting pictures, merge cells etc. which we would like to add in the future.

7. Conclusion

In conclusion, working on the Mini Excel project was a learning journey that helped us to understand how the theoretical things are applied practically. Through this project, we learnt key concepts of DSA e.g. BSTs, DAGs etc.

Implementing features like formula evaluation, cell formatting, cells dependency and detecting circular dependency helped us understand how real-world software operates.

Overall, this project not only improved our technical skills but also gave us a sense of how to manage more complex projects in the future.