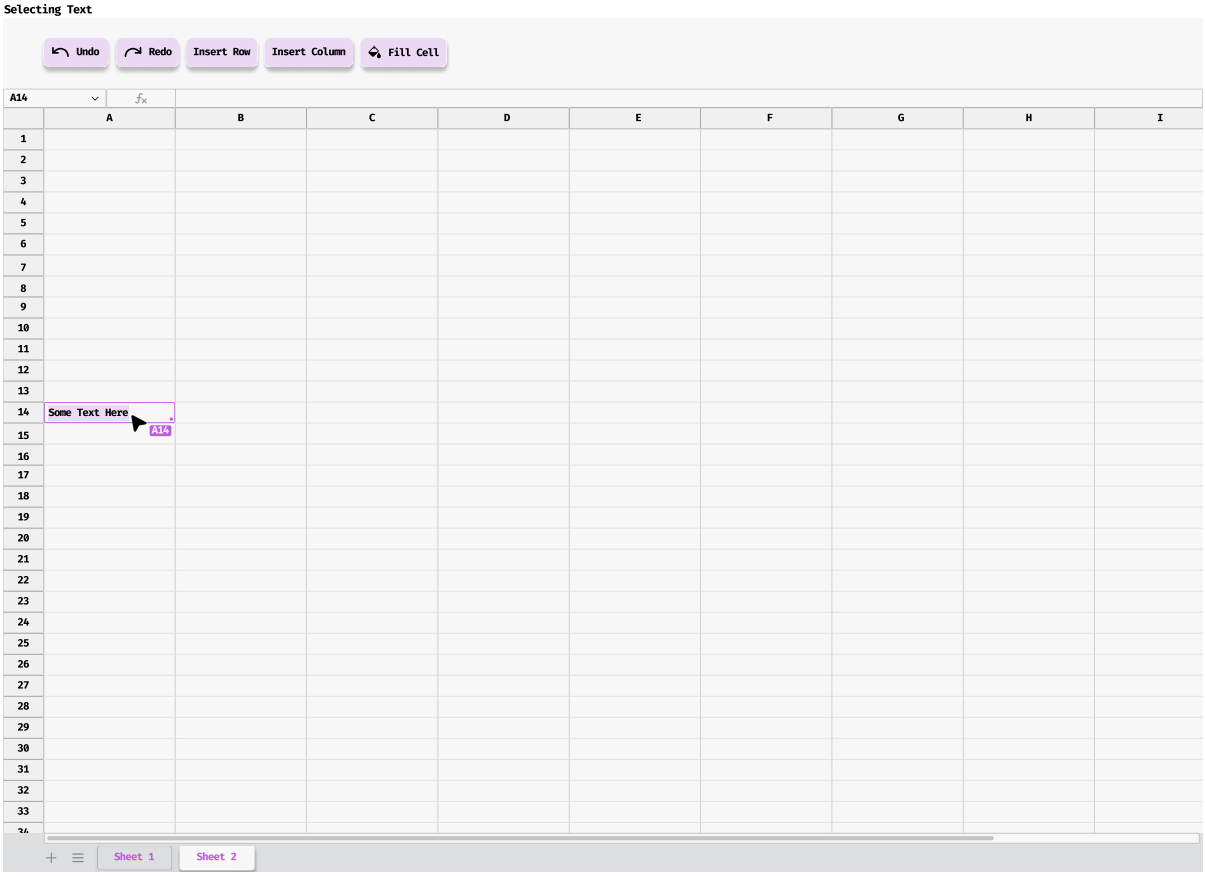
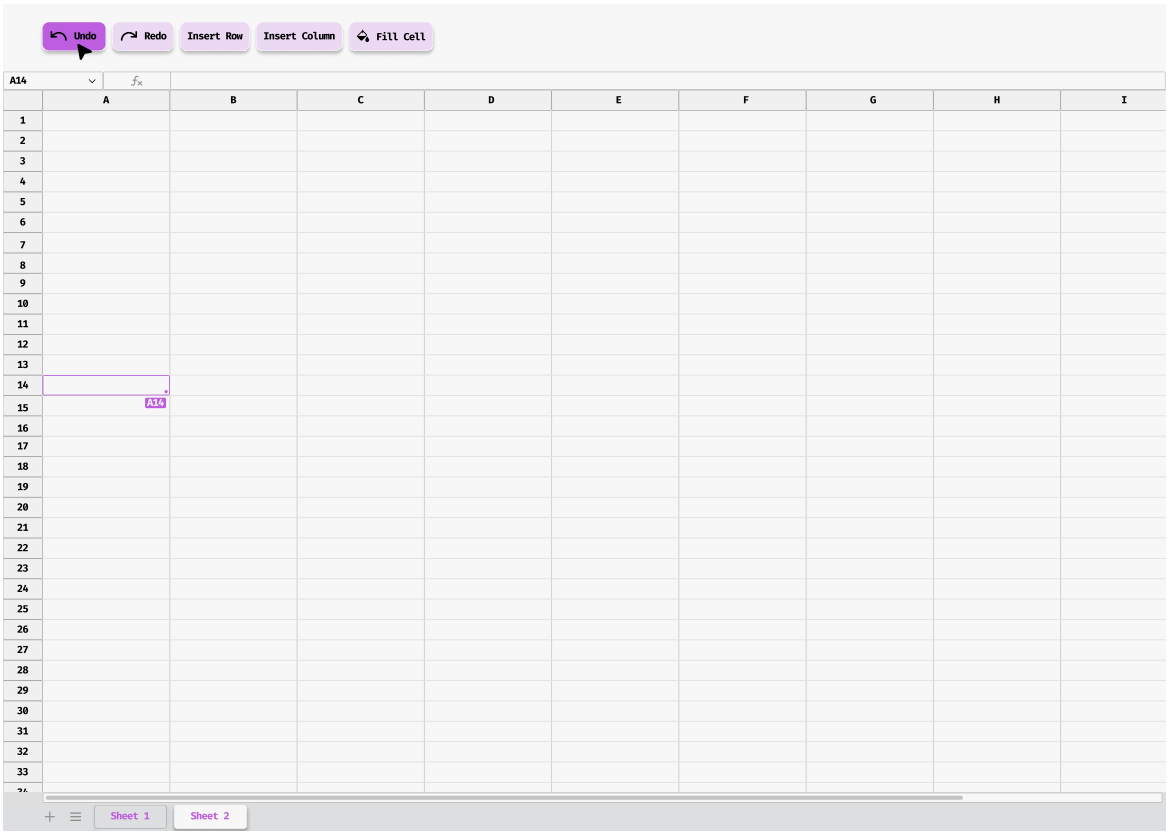
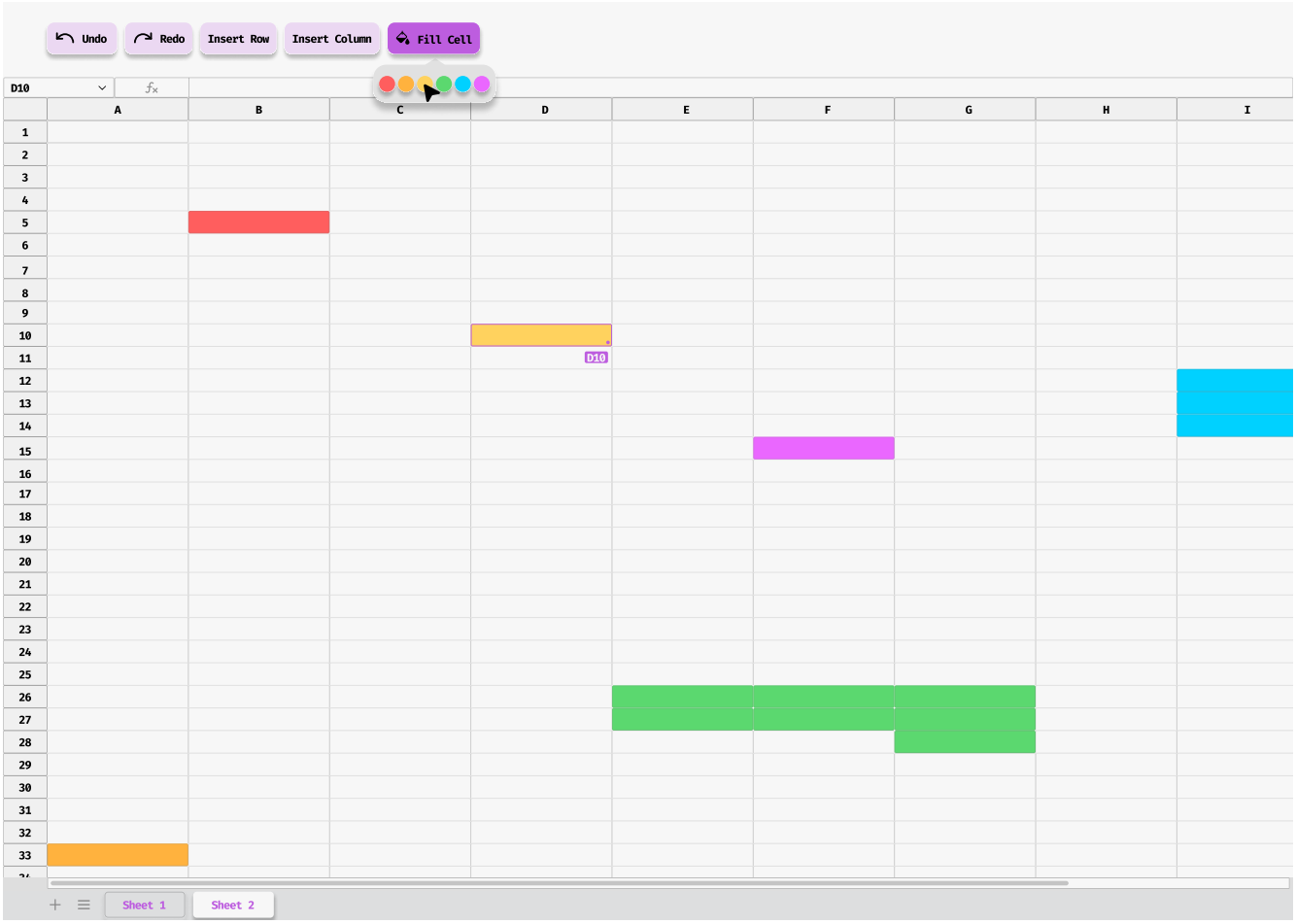
Selecting Text



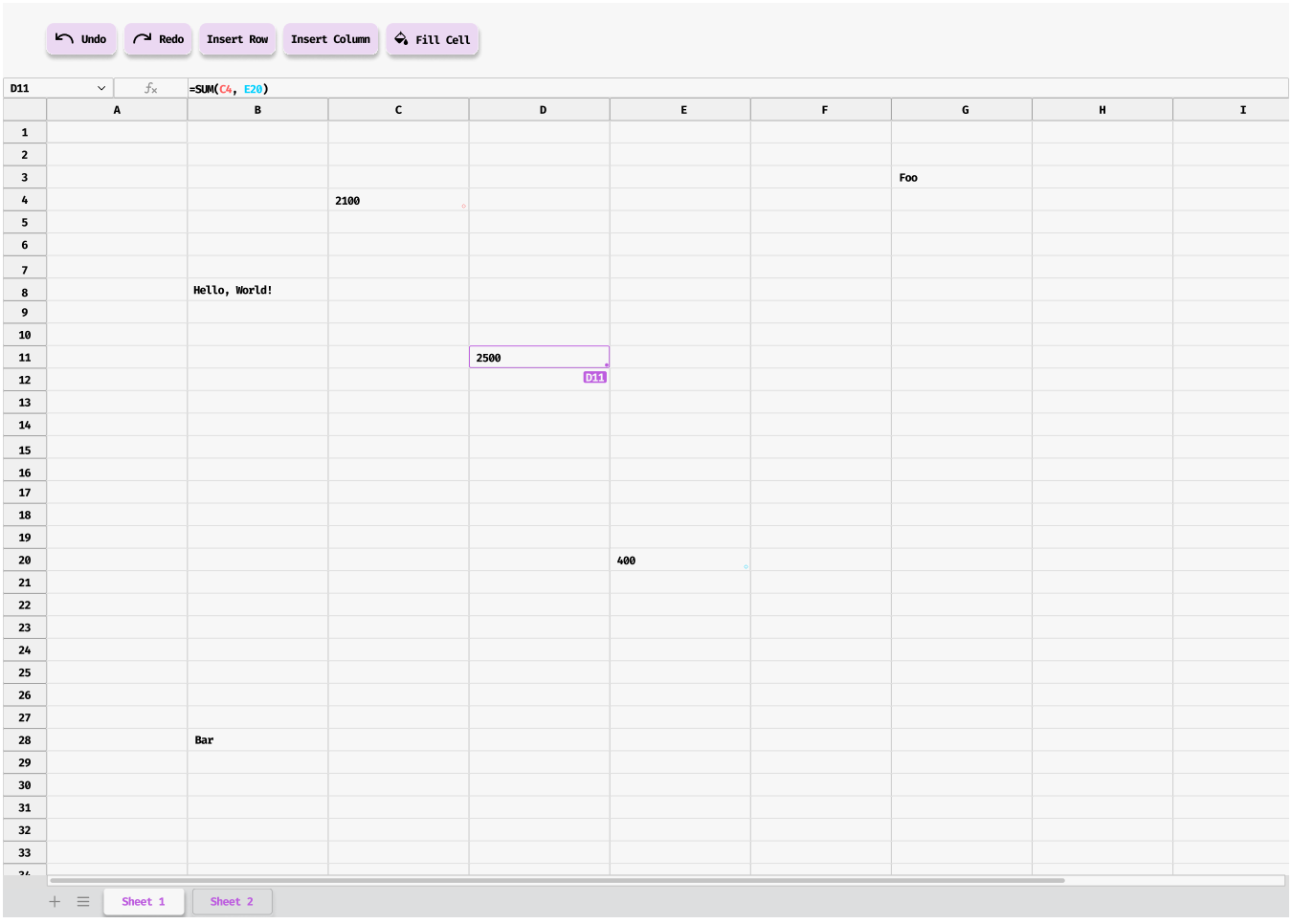
Undo Last Action



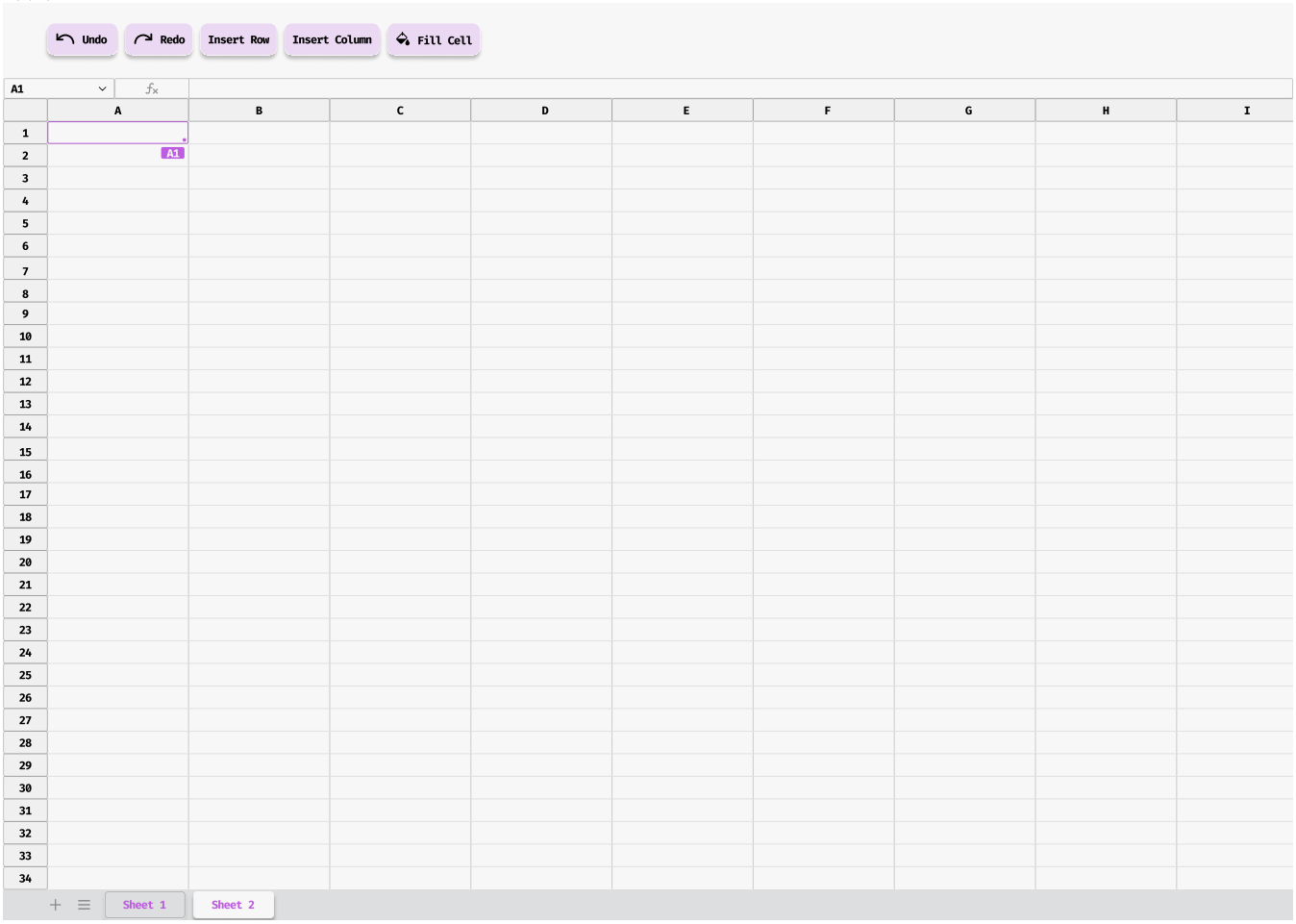
Fill Cell with Color



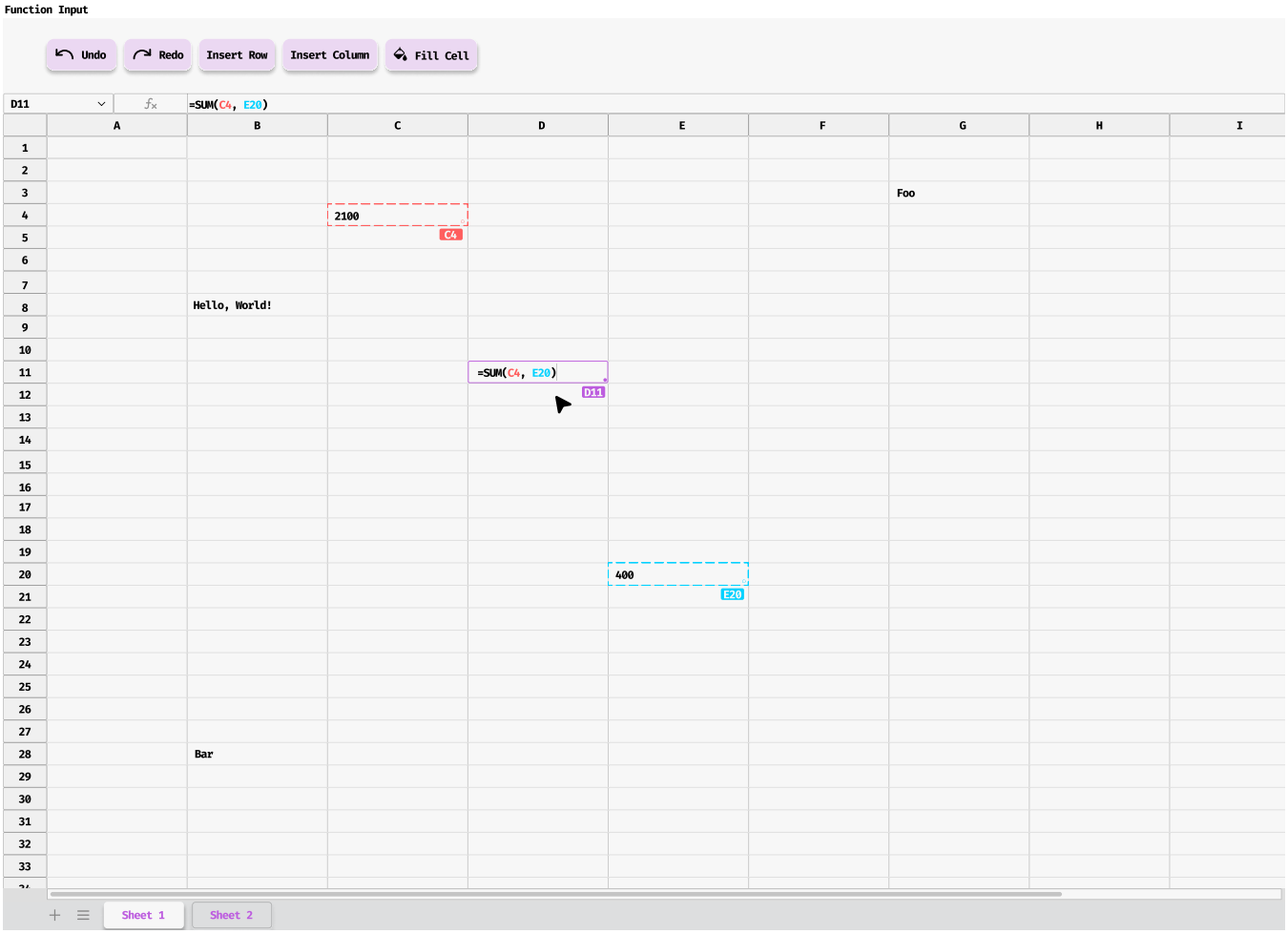
Selecting a Cell



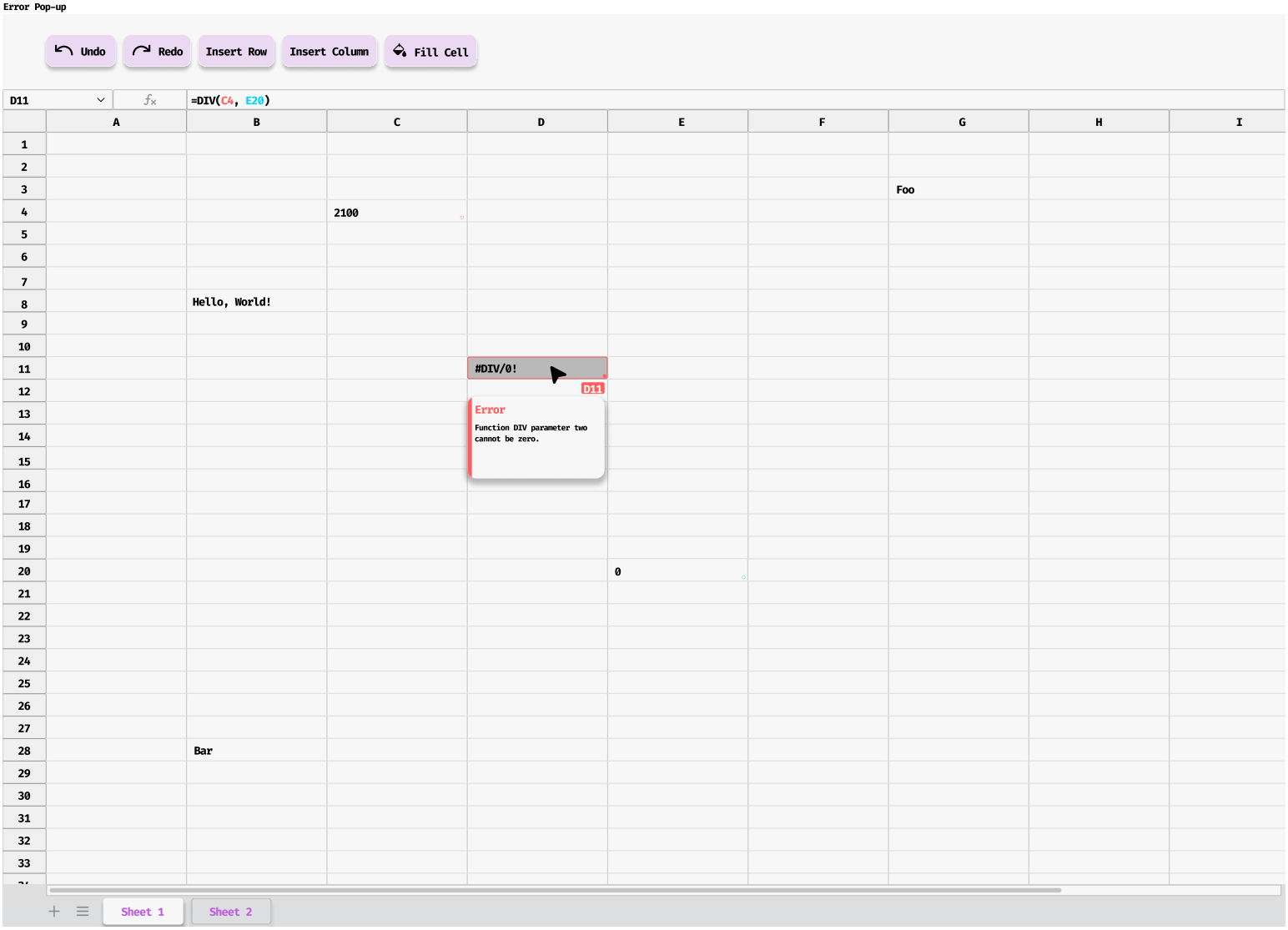
Empty Spreadsheet

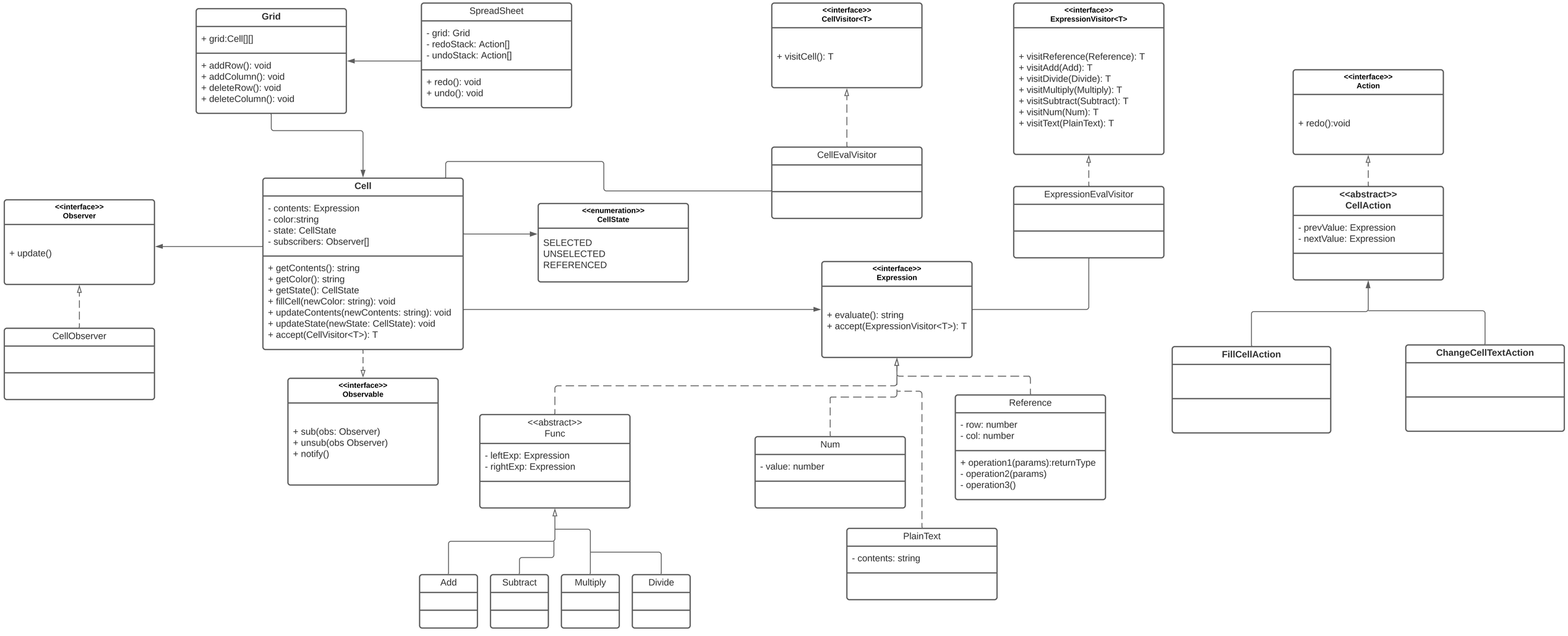


Function Input



Error Pop-up


Undo
Redo
Insert Row
Insert Column
Fill Cell
D11
=DIV(C4, E20)
A
B
C
D
E
F
G
H
I
Foo
2100
Hello, World!
#DIV/0!
D11
Error
Function DIV parameter two cannot be zero.
0
Bar
Sheet 1
Sheet 2


Grid
+ grid:Cell[][]
+ addRow(): void
+ addColumn(): void
+ deleteRow(): void
+ deleteColumn(): void
SpreadSheet
- grid: Grid
- redoStack: Action[]
- undoStack: Action[]
+ redo(): void
+ undo(): void
<<interface>>
CellVisitor<T>
+ visitCell(): T
<<interface>>
ExpressionVisitor<T>
+ visitReference(Reference): T
+ visitAdd(Add): T
+ visitDivide(Divide): T
+ visitMultiply(Multiply): T
+ visitSubtract(Subtract): T
+ visitNum(Num): T
+ visitText(PlainText): T
<<interface>>
Action
+ redo():void
CellEvalVisitor
ExpressionEvalVisitor
Cell
- contents: Expression
- color:string
- state: CellState
- subscribers: Observer[]
+ getContents(): string
+ getColor(): string
+ getState(): CellState
+ fillCell(newColor: string): void
+ updateContents(newContents: string): void
+ updateState(newState: CellState): void
+ accept(CellVisitor<T>): T
<<enumeration>>
CellState
SELECTED
UNSELECTED
REFERENCED
<<interface>>
Observer
+ update()
CellObserver
<<interface>>
Observable
+ sub(obs: Observer)
+ unsub(obs Observer)
+ notify()
<<interface>>
Expression
+ evaluate(): string
+ accept(ExpressionVisitor<T>): T
<<abstract>>
CellAction
- prevValue: Expression
- nextValue: Expression
FillCellAction
ChangeCellTextAction
<<abstract>>
Func
- leftExp: Expression
- rightExp: Expression
Num
- value: number
Reference
- row: number
- col: number
+ operation1(params):returnType
- operation2(params)
- operation3()
PlainText
- contents: string
Add
Subtract
Multiply
Divide

Scenario Number	1	**Actors**	Spreadsheet User
Creation Date	10/13/21	**Last Update**	10/13/21
Description			
This scenario describes normal spreadsheet operation when the user wants to input text into an empty cell of the spreadsheet.			
Pre-conditions	User has the spreadsheet application open, displaying the document they wish to edit.		
Step	**Action**		
1	User left-clicks on an empty cell of the spreadsheet.		
2	The cell's border is highlighted.		
3*	User double clicks the same cell of the spreadsheet.		
4	The cell enters insert mode and displays a blinking cursor.		
5	User types desired text input into cell.		
6	The cell displays the text in real-time.		
7*	User hits Enter/Return on the keyboard.		
8*	The text is saved in the cell and the cell directly below is highlighted.		
Variants			
3.1	User types desired text input into cell.		
7.1	User left-clicks another cell of the spreadsheet.		
8.1	The text is saved in the cell and the newly clicked cell is highlighted.		
7.2	User hits Tab on the keyboard.		
8.2	The text is saved in the cell and the cell directly to the right is highlighted.		
7.3	User hits Escape on the keyboard.		
8.3	The cell returns to its state before the user entered any text, i.e. highlighted and empty.		
7.4	User hits Shift + Enter/Return on the keyboard		
8.4	The text is saved in the cell and the cell directly above is highlighted.		
Exceptions			
1	User inputs an invalid formula as the text into the empty cell, the application will display an error informing the user to input a valid formula or the escape character.		
Priority	High		

Scenario Number	2	**Actors**	Spreadsheet User
Creation Date	10/13/21	**Last Update**	10/13/21
Description			
This scenario describes normal spreadsheet operation when the user wants to overwrite previously entered text.			
Pre-conditions	User has the spreadsheet application open, displaying the document they wish to edit. At least one cell has text in it.		
Step	Action		
1	User left-clicks the cell of the spreadsheet they wish to overwrite.		
2	The selected cell is highlighted.		
3*	User types desired text input into the cell.		
4	The previous text in the cell is removed and the new text input is displayed in real-time.		
5	User hits Enter/Return on the keyboard.		
6	The new text is saved in the cell, and the cell directly below is highlighted.		
Variants			
3.1	User double clicks the highlighted cell, deletes the current text, and types the desired text input into the cell.		
Exceptions			

1	If the user hits the escape button after overwriting text in the cell, the saves will not be saved. Instead, the original text will remain in the cell.
2	If the user fails to select all text in the cell they wish to overwrite, the newly inputted text will be appended to the unselected original text in the cell.
Priority	High

Scenario Number	3	**Actors**	Spreadsheet User
Creation Date	10/13/21	**Last Update**	10/13/21
Description			
This scenario describes normal spreadsheet operation when the user wants to enter a formula into a cell.			
Pre-conditions	User has the spreadsheet application open, displaying the document they wish to edit.		
Step	**Action**		
1	User double clicks on a cell of the spreadsheet.		
2	The cell is highlighted, enters insert mode, and displays a blinking cursor.		
3	User types the equal sign (=) into the cell.		
4	The cell recognizes that a formula is desired and displays the equal sign.		
5	User types out the desired formula.		
6	The cell displays the typed formula verbatim.		
7	User hits Enter/Return on the keyboard.		
8	The formula is applied and the cell displays the output. The cell directly below is highlighted.		
Exceptions			
1	If the user enters a formula without prepending an equal sign, the application will not error, but will not apply the formula. Instead, the raw text input will be displayed in the cell. E.g. “SUM(A10, B12)” would be displayed in the cell, rather than the output from “=SUM(A10, B12).”		
2	If the user enters the equal sign but does not append anything, the application will throw an error informing the user to input a valid formula or escape character before the equal sign.		
3	If the user enters an invalid formula following an equal sign, the application will throw an error informing the user to input a valid formula or escape character before the equal sign.		
Priority	High		

Scenario Number	4	**Actors**	Spreadsheet User
Creation Date	10/13/21	**Last Update**	10/13/21
Description			
This scenario describes normal spreadsheet operation when the user wants to insert a new row.			
Pre-conditions	User has the spreadsheet application open, displaying the document they wish to edit.		
Step	**Action**		
1*	User right-clicks any cell in the row they wish to add a new row above.		
2*	The cell is highlighted, and the right-click context menu pops up.		
3	User selects “Insert Row” from the context menu.		
4	An empty row is added above the row in which the highlighted cell exists.		
Variants			
1.1	With a cell in the row the user wishes to add a new row above selected, the user clicks the “Insert Row” button in the top bar.		
2.1	An empty row is added above the row in which the highlighted cell exists.		
Exceptions			
1	If the user reaches the maximum row count, attempting to insert a new row will have no effect on the spreadsheet.		
2	If the user clicks a row expecting to add a new row below it, following steps one through four will add a new row in an unexpected location of the spreadsheet (since the new row is added above the selected cell’s row)..		

Priority	High

Scenario Number	5	**Actors**	Spreadsheet User
Creation Date	10/13/21	**Last Update**	10/13/21
Description			
This scenario describes normal spreadsheet operation when the user wants to delete an existing column.			
Pre-conditions	User has the spreadsheet application open, displaying the document they wish to edit. At least one cell in the spreadsheet has content.		
Step	**Action**		
1	User right-clicks a cell in the column they wish to delete.		
2	The cell's border is highlighted, and the right-click context menu pops up.		
3	User selects "Delete Column" from the context menu.		
4	The column containing the highlighted cell is removed from the spreadsheet.		
Exceptions			
1	If the user reaches the minimum column count, attempting to delete a column will have no effect on the spreadsheet.		
2	If the user selects a cell in a column they don't wish to delete and continues with the above steps, an incorrect row will be deleted from the spreadsheet.		
Priority	High		

Scenario Number	6	**Actors**	Spreadsheet User
Creation Date	10/13/21	**Last Update**	10/13/21
Description			
This scenario describes normal spreadsheet operation when the user wants to concatenate two strings.			
Pre-conditions	User has the spreadsheet application open, displaying the document they wish to edit.		
Step	**Action**		
1	User left-clicks a cell of the spreadsheet.		
2	The cell's border is highlighted.		
3	User types the equal sign (=) followed by two strings they would like to concatenate, following the valid syntax: "string1" + "string2"		
4	The cell displays the text input verbatim; e.g. ="string1" + "string2"		
5	User hits Enter/Return on the keyboard.		
6	The cell displays the concatenated string. E.g. "string1string2"		
Exceptions			
1	If the user fails to prepend the equal sign, the string concatenation will not be completed. Instead, the cell will display the literal form of the text input; e.g. "string1" + "string2," instead of string1string2.		
2	If the user creates an addition formula using a string and a value of another type, string concatenation shall be used rather than number addition. For example, = "hello" + 3 → hello3.		
Priority	High		

Scenario Number	7	**Actors**	Spreadsheet User
Creation Date	10/13/21	**Last Update**	10/13/21
Description			
This scenario describes normal spreadsheet operation when the user wants to reference another cell in the same spreadsheet.			
Pre-conditions	User has the spreadsheet application open, displaying the document they wish to edit.		
Step	**Action**		
1	User left-clicks a cell of the spreadsheet.		

2	The cell's border is highlighted.
3	User types the equal sign (=) into the cell.
4	The cell recognizes that a formula is desired and displays the equal sign.
5	User enters REF(<the cell they wish to reference>)
6	The cell displays the entered text verbatim.
7	User hits Enter/Return on the keyboard.
8	The reference formula is applied and the cell's text reflects that of the referenced cell.
Exceptions	
1	If the user references a cell that does not exist, the cell will display a warning that the referenced cell does not exist.
Priority	High

Scenario Number	8	**Actors**	Spreadsheet User
Creation Date	10/13/21	**Last Update**	10/13/21
Description			
This scenario describes normal spreadsheet operation when the user wants to compute the average across a selected range of cells.			
Pre-conditions	User has the spreadsheet application open, displaying the document they wish to edit. At least one cell in the spreadsheet has content.		
Step	**Action**		
1	User left-clicks a cell of the spreadsheet.		
2	The cell's border is highlighted.		
3	User types the equal sign (=) into the cell.		
4	The cell recognizes that a formula is desired and displays the equal sign.		
5	User enters AVERAGE(<starting cell>, <ending cell>).		
6	The cell displays the entered text verbatim.		
7	User hits Enter/Return on the keyboard.		
8	The average formula is applied and the cell's text reflects the computed average across the range of specific cells.		
Exceptions			
1	If the user inputs two numbers rather than two cell locations, the average between the two numbers will be computed.		
2	If the user inputs a cell location that references an empty cell, the average function will interpret the empty cell's value as 0.		
3	If the user inputs a cell location that references a cell with a non-number input (e.g. a string), the average function will interpret the empty cell's value as 0.		
Priority	High		

Scenario Number	9	**Actors**	Spreadsheet User
Creation Date	10/13/21	**Last Update**	10/13/21
Description			
This scenario describes behavior for one of the three additional features: a Spreadsheet User undoing adding text to a cell by clicking the undo button.			
Pre-conditions	User has the spreadsheet application open, displaying the document they wish to edit. At least one cell in the spreadsheet has content.		
Step	**Action**		
1	User double-clicks a cell with content in it.		
2	The cell is highlighted, enters insert mode, and displays a blinking cursor.		
3	User selects all text in the cell and deletes it.		
4	The cell's text is removed, leaving the selected cell empty and still in insert mode.		
5	User clicks outside the currently selected cell.		
6	The cell's state (current text) is saved and the border of the cell the user just selected is highlighted..		

7	User clicks the “Undo” button in the Top Bar.
8	The removal of the text is undone, restoring the cell’s state (text) back to what it was.
Exceptions	
1	If no actions have been completed, the Undo button will do nothing when clicked. Since there is nothing to undo, the button will have no effect on the overall state of the spreadsheet application.
2	If the user selects the undo button, performs an action, and attempts to reset state to that before clicking the undo button, the undo operation will not perform as expected, so previous actions may be undone that are not desired to be undone. This is easily fixable by clicking the Redo Button if the User undoes too many actions.
Priority	Medium

Scenario Number	10	**Actors**	Spreadsheet User
Creation Date	10/13/21	**Last Update**	10/13/21
Description			
This scenario describes behavior for one of the three additional features: Detailed error information popup in the case of user error being hovered over.			
Step	**Action**		
1	User inserts, edits, or deletes the value of a cell that results in an error.		
2	System replaces the contents of the cell with an error display.		
3	User hovers over the cell with the error using their mouse.		
4	System displays an error pop-up next to the erroring cell, displaying further detail and explanation as to why the error has occurred.		
5	User moves their mouse away from the erroring cell.		
6	The application hides the error pop-up window.		
Priority	Medium		

Scenario Number	11	**Actors**	Spreadsheet User
Creation Date	10/13/21	**Last Update**	10/13/21
Description			
This scenario describes behavior for one of the three additional features: the Spreadsheet User wants to fill a cell with a background color.			
Pre-conditions	User has the spreadsheet application open, displaying the document they wish to edit.		
Step	**Action**		
1	User left-clicks a cell of the spreadsheet.		
2	The cell’s border is highlighted.		
3	User hits the “Fill Cell” button in the top bar.		
4	The color selection dropdown modal is displayed below the “Fill Cell” button.		
5	User left-clicks the desired color with which to fill the highlighted cell.		
6	The cell’s background color is changed to the selected color.		
Exceptions			
1	If the user selects the wrong color in the color selection dropdown modal, the cell will be filled with the wrong color. The user can remedy this by repeating steps one through six, this time selecting the correct desired color.		
2	If the user fails to select a cell before completing steps three through six, no change to any cell color will occur. Instead, the mouse clicks will not have any effect on the spreadsheet.		
Priority	Medium		

Code Outline:

```
interface Observer {
   update(): void;
```

```
}

class CellObserver implements Observer {
    update(): void {
        throw new Error("Method not implemented.");
    }
}

interface Observable {
    sub(obs: Observer): void;
    unSub(obs: Observer): void;
    notify(): void;
}

class Cell implements Observable {
    private _contents: Expression;
    private _color: string;
    private _state: CellState;
    private _subscribers: Observer[];

    constructor() {}

    sub(obs: Observer): void {
        throw new Error("Method not implemented.");
    }
    unSub(obs: Observer): void {
        throw new Error("Method not implemented.");
    }
    notify(): void {
        throw new Error("Method not implemented.");
    }

    accept<T>(cv: CellVisitor<T>): T {
        throw new Error("Method not implemented.");
    }
}

enum CellState {
    SELECTED,
    UNSELECTED,
    REFERENCED,
}

class SpreadSheet {
    private _grid: Grid;
    private _redoStack: Action[];
    private _undoStack: Action[];

    constructor() {}
```

```
    redo(): void {
        throw new Error("Method not implemented.");
    }
    undo(): void {
        throw new Error("Method not implemented.");
    }
}

class Grid {
    private _grid: Cell[][];

    constructor() {}

    addRow(): void {
        throw new Error("Method not implemented.");
    }
    deleteRow(): void {
        throw new Error("Method not implemented.");
    }
    addColumn(): void {
        throw new Error("Method not implemented.");
    }
    deleteColumn(): void {
        throw new Error("Method not implemented.");
    }
}

interface CellVisitor<T> {
    visitCell(c: Cell): T;
}

class CellEvalVisitor implements CellVisitor<string> {
    visitCell(c: any): string {
        throw new Error("Method not implemented.");
    }
}

interface ExpressionVisitor<T> {
    visitReference(r: Reference): T;
    visitAdd(a: Add): T;
    visitSubtract(s: Subtract): T;
    visitMultiply(m: Multiply): T;
    visitDivide(d: Divide): T;
    visitPlainText(t: PlainText): T;
    visitNum(n: Num): T;
}

class ExpressionEvalVisitor implements ExpressionVisitor<string> {
    visitReference(r: any): string {
        throw new Error("Method not implemented.");
```

```
    }
    visitAdd(a: any): string {
        throw new Error("Method not implemented.");
    }
    visitSubtract(s: any): string {
        throw new Error("Method not implemented.");
    }
    visitMultiply(m: any): string {
        throw new Error("Method not implemented.");
    }
    visitDivide(d: any): string {
        throw new Error("Method not implemented.");
    }
    visitPlainText(t: PlainText): string {
        throw new Error("Method not implemented.");
    }
    visitNum(n: any): string {
        throw new Error("Method not implemented.");
    }
}

interface Expression {
    evaluate(): string;
    accept<T>(ev: ExpressionVisitor<T>): T;
}

class Num implements Expression {
    value: number;

    constructor() {}

    evaluate(): string {
        throw new Error("Method not implemented.");
    }
    accept<T>(ev: ExpressionVisitor<T>): T {
        throw new Error("Method not implemented.");
    }
}

class PlainText implements Expression {
    contents: string;

    constructor() {}

    evaluate(): string {
        throw new Error("Method not implemented.");
    }
    accept<T>(ev: ExpressionVisitor<T>): T {
        throw new Error("Method not implemented.");
    }
```

```
}

class Reference implements Expression {
    row: number;
    col: number;

    constructor() {}

    evaluate(): string {
        throw new Error("Method not implemented.");
    }
    accept<T>(ev: ExpressionVisitor<T>): T {
        throw new Error("Method not implemented.");
    }
}

abstract class Func implements Expression {
    leftExp: Expression;
    rightExp: Expression;

    constructor() {}

    evaluate(): string {
        throw new Error("Method not implemented.");
    }
    accept<T>(ev: ExpressionVisitor<T>): T {
        throw new Error("Method not implemented.");
    }
}

class Add extends Func {}
class Subtract extends Func {}
class Divide extends Func {}
class Multiply extends Func {}

interface Action {
    redo(): void;
}

abstract class CellAction implements Action {
    prevValue: Expression;
    nextValue: Expression;

    abstract redo(): void;
}

class FillCellAction extends CellAction {
    redo(): void {
        throw new Error("Method not implemented.");
    }
```

```
}

class ChangeCellTextAction extends CellAction {
    redo(): void {
        throw new Error("Method not implemented.");
    }
}
```