MVP

Name: Working GUI

Actor: User

Stakeholders and Interests:
- User: Wants to interact with the application to execute the main functionalities of a spreadsheet
- System: Needs to ensure communication of user input with the backend system.

Preconditions: The user is logged into the application, either looking to load a spreadsheet or currently editing a spreadsheet

Postconditions: The system processes the input accurately and displays its response accordingly.

Basic Flow:
1. GUI loads and displays depending on different parts of the application: startup, in-editor, etc.
2. The user decides to click on a button displayed by the GUI
3. The system processes the user input, and communicates with the back-end, for example: to save the project.
4. The system executes the input and passes the GUI the status of the command, success/error.
5. The GUI displays the result with a message such as: “Saved to: ‘home/cs4530’ successfully”

Extensions (Alternative Flows):
1. Executing certain commands displayed available by the GUI can also be triggered through shortcuts:
    a. Ex: New document: ^ + lshift + n

Special Requirements:
1. The application should be able to handle “spamming” of buttons from a user
2. The GUI should be able to scale in proportion to the user’s screen size

Open Issues:

1. If the GUI becomes unresponsive, what is the solution to restart + restore the data of the user’s current session.

Name: Store Sheets Over Multiple Sessions

Actor: User

Stakeholders and Interests:
User: Aims to access spreadsheet data across multiple sessions.
System: Must reliably store, protect, and retrieve spreadsheet data for the user.

Preconditions: The user has previously saved a spreadsheet in the application.

Postconditions: The system saves the spreadsheet data persistently, allowing the user to access it in subsequent sessions.

Basic Flow:

1. The user opens the application.
2. The user loads a previously saved spreadsheet.
3. The user interacts with the spreadsheet, making changes as needed.
4. The user closes the application.

Extensions (Alternative Flows):
- If the user opens the application without any previously saved spreadsheet, they are prompted to either create a new one or locate an existing sheet.
- If the user attempts to close the application without saving changes to the current spreadsheet, they are prompted to confirm their action.
-Potentially the system may auto-save the user's work.

Special Requirements:
- The application must have reliable data storage to ensure the reliability and security of the spreadsheet data.
- The user can set the default save locations.

Open Issues:
- How to handle save conflicts when multiple instances of the application are accessing and modifying the same spreadsheet simultaneously.
- Ensuring data integrity of the spreadsheet data.

Name: Basic Editing Features like Insert, Delete, and Update Cell Value

Actor: User
Stakeholders and Interests:
User: Desires tools for manipulating spreadsheet data.
System: Must execute user commands to modify cell content responsive to user input.

Preconditions: The user has opened a spreadsheet in the application.
Postconditions: Changes made by the user to cell values are reflected in the spreadsheet, and the system ensures correctness of the data.

Basic Flow:
1. The user selects a cell (or range of cells) to edit.
2. The user inputs new data or modifies existing data in the selected cells.
3. The user confirms changes by pressing enter or clicking outside the cell.
4. The system updates the cell values.

Special Requirements:

-The application should support data types such as strings and numbers.
-Inserting or deleting a cell should not disrupt the structure of the spreadsheet
-Inserting or deleting should also update formulas accordingly.

Extensions:

-If the user selects an incorrect region in the GUI, their input will not be received/detected.

Open Issues:

- If the user wishes to undo a recent edit, they can use the undo command

Name: Download spreadsheet data
Actor: User
Stakeholders and Interests:

- **User:** Wants to download a local copy of the data in the spreadsheet with the freedom to choose a specific file format
- **System:** Needs to ensure that a given spreadsheet can be converted to a specified file format without issue

Preconditions: The user is logged in to the application and has opened a spreadsheet
Postconditions: The spreadsheet has been downloaded to the user's local directory in the specified file format
Basic Flow:

1. **Indicate intention to download:** The user hovers over a download button that offers different format options
2. **Select file format for download:** The user specifies the format in which they want to receive the data
3. **File conversion:** The system converts the data for the current spreadsheet into the specified format
4. **Download Locally:** The system downloads the resulting file to the user's local directory

Extensions:

1. **File conversion error:** If there is an issue that arises when converting the spreadsheet data to the chosen file format, alert the user with a notification that prompts the user to select a different file format

Special Requirements:

- The application should be able to convert and download spreadsheets of all sizes

Open Issues:

- What to do in the case that all file formats fail to download?

Desirable Features

Name: Calculations within Cells
Actor: User
Stakeholders and Interests:

- **User:** Wants a cell to display the result of a mathematical calculation
- **System:** Needs to ensure that a given equation is formatted properly, can be interpreted, and returns the appropriate answer

Preconditions: The user is logged in to the application and has opened a spreadsheet
Postconditions: The cell the user is working in returns the proper result of the equation entered
Basic Flow:

1. **Select cell:** The user must first select a cell to work in
2. **Write equation:** The user must then write a mathematical equation that they would like the result of
3. **Compute answer:** The system validates the equation notation and computes the answer to be displayed in the same cell

Extensions:

1. **Computation error:** If there is an issue when interpreting the equation entered or finding a valid solution to the equation, alert the user with a notification that prompts the user to alter the equation to fit the capabilities of the application

Special Requirements:

- The application should not fail when dealing with mathematical exceptions such as a divide by zero error and so on

Open Issues:

- How should the application convey to the user what kind of calculations are supported?

Name: Cell Referencing
Actor: User
Stakeholders and Interests:

- **User:** Wants to be able to access the value of a selected cell when write an equation
- **System:** Needs to ensure that there is a value to be returned and that the proper value is returned

Preconditions: The user is logged in to the application and has opened a spreadsheet
Postconditions: The value inserted into the current equation is the same value that is displayed in the selected cell
Basic Flow:

1. **Select cell:** The user must first select a cell to work in
2. **Write equation:** The user must then write a mathematical equation that they would like the result of
3. **Select different cell to reference:** The user must then select another cell whose value they would like to insert into the current equation
4. **Insert value:** Insert the value corresponding to the selected cell into the equation

Extensions:

1. **Value error:** Alert the user when there is no value in the cell that they have selected

Special Requirements:

- The application should be aware of whether the type of the inserted value fits into the type context of the current cell

Open Issues:

- How should the application restrict the type of value that can be selected and subsequently inserted into a given cell?

2. **GUI**: (Troy)
    - A simple and grid interfaceToolbar: A basic toolbar with icons for common tasks like adding new sheets, formatting cells, and adjusting rows/columns.
    - Menu Options: Simplified menus focusing on file management (New, Open, Save, Close) and basic edit operations (Copy, Cut, Paste), highlighting multiple rows.

Name: Spreadsheet Interface
Actor: User

Stakeholders and Interests:

- User: User wants to interact with the application to manage and format spreadsheet data efficiently
- System: Needs to ensure smooth communication of user input with the backend to provide an intuitive user experience

Preconditions:
- The user is either looking to load an existing spreadsheet or currently editing a spreadsheet

Postconditions:
- The system processes the user's input accurately
- The system updates the interface accordingly

Basic Flow:
1. The GUI loads and displays the main interface depending on the application's state (e.g., startup, in-editor).
2. The user views the simple grid interface for data input and management.
3. The user interacts with the basic toolbar:
    - Adds new sheets by clicking the appropriate icon.
    - Formats cells by selecting formatting options (bold, italic, font size, color) from the toolbar.
    - Adjusts rows and columns as needed.
4. The user navigates the simplified menus:
    - Uses the File menu for tasks such as creating a new file, opening an existing file, saving the current file, and closing the file.
    - Uses the Edit menu for basic operations like Copy, Cut, and Paste.
5. The user highlights multiple rows by clicking and dragging or using the shift key.
6. The user applies actions (e.g., Copy, Paste) to all selected rows simultaneously.

Extensions (Alternative Flows):
- If the user encounters an error, an appropriate error message should be displayed.

Special Requirements:
- Toolbar icons should be easily distinguishable and visually distinct

OpenIssues:
- Should we have an advanced menu, or a place to use additional features that aren't displayed in the default toolbar? (if we have additional features)

3. **Security**: (Troy)
    - User login feature
    - User status: editing/viewing/suggesting
    - Data security/password

Name: User Authentication and Data Security
Actor: User

Stakeholders and Interests:

- User: Wants to securely log in to the application and have their data protected. Also wants to have different statuses (editing, viewing, suggesting) for better control over their work and collaboration.
- System: Needs to ensure that user authentication is secure and user data is protected.

Preconditions:
- The user has a registered account with the application and knows their login credentials

Postconditions:
- The user is successfully logged in.
- The system accurately reflects the user's current status (editing, viewing, suggesting).
- The user's data and password are securely handled and protected.

Basic Flow:
1. **User Login Feature:**
    - The user opens the application and is presented with a login screen.
    - The user enters their username and password.
    - The system verifies the credentials.
    - If the credentials are correct, the user is logged in and taken to the main interface.
2. **User Status Management:**
    - The user can switch between different statuses: editing, viewing, and suggesting.
    - Editing: The user can make changes to the spreadsheet.
    - Viewing: The user can only view the spreadsheet without making changes.
    - Suggesting: The user can make suggestions for changes that can be approved or rejected by others with editing rights.
    - The system displays the current status.
3. **Data Security/Password Handling:**
    - **The system ensures that the user's password is encrypted and stored securely.**

Extensions (Alternative Flows):
- If the user enters incorrect credentials, the system provides an error message and allows for retrying

Special Requirements:
- The UI should clearly indicate the user's current status and provide easy access to change it

OpenIssues:
- What if a user forgets their login credentials?

Additional Features - Bonus

Name: Styling/Formatting Options
Actor: User

Stakeholders and Interests:

- User: Wants to make fonts larger, highlighted, and customize cell borders to draw attention to specific content
- System: Needs to make sure the cell doesn't break if text goes beyond cell capacity

Preconditions: The user is logged into the application and has opened or created a sheet.
Postconditions: The selected cells are formatted according to the user's preferences (font size, text color, background color, and cell borders).
Basic Flow:

1. **Open Formatting Options:** The user first selects one or more cells, then opens the formatting menu.
2. **Select Formatting Attributes:** The user chooses the desired font size, text color, background color, and cell borders.
3. **Apply Formatting:** The system applies the selected formatting to the chosen cells.
4. **Confirm Changes:** The system displays the formatted cells to the user.

Extensions (Alternative Flows):

- **Adjust Cell Size:** If text exceeds cell capacity, the system automatically adjusts the cell size or displays an overflow indicator.
- **Revert Formatting:** The user can revert to default formatting by selecting the default option in the formatting menu.

Special Requirements:

- The application should allow formatting for both individual cells and groups of cells.
- The formatting changes should not affect application performance or data integrity ideally.

OpenIssues:

- What default color types and font sizes should be available? Should we add different font types?
- How should the system handle conflicting formatting requests (like lets say overlapping styles)?

Name: Dynamic Filling
Actor: User

Stakeholders and Interests:

- User: User wants to select cells with a formula and apply the same math to multiple cells quickly
- System: Calculates each cell accurately following the same pattern even if there are not values, gives UNDEFINED if not possible instead of failing.

Preconditions: The user is logged into the application and has opened or created a sheet.

Postconditions: The selected cells are dynamically filled with the formula applied correctly.
Basic Flow:

1. **Select Cells:** The user selects a cell or range of cells containing a formula.
2. **Initiate Dynamic Filling:** The user drags the fill handle or uses a context menu option to apply the formula to adjacent cells.
3. **System Calculation:** The system calculates the formula for each selected cell, checking for dependencies and applying the pattern.
4. **Display Results:** The system displays the calculated values in the selected cells.

Extensions (Alternative Flows):

- **Handle Errors:** If a cell cannot be calculated (such as due to missing values), the system displays "UNDEFINED" instead of failing.
- **Undo Fill:** The user can undo the dynamic fill by using an undo action or context menu option.

Special Requirements:

- The application should handle complex formulas and large ranges of cells.
- The dynamic filling operation should be optimized for performance to prevent our sheet from lagging.

OpenIssues:

- How should the system handle circular references in formulas? Like when it is referring back to it's own cell I mean.
- What are the limitations on the number of cells that can be dynamically filled at once?

Name: Customizable UI Themes
Actor: User

Stakeholders and Interests:

- **User:** The user finds light mode to bother their eyes, it's tough when you are working on the same spreadsheet all day 😕 they want to set it to dark mode for the rest of the night until the morning.
- **System:** System needs to smoothly switch while keeping all the same cell values without losing data

Preconditions: Again, the user is logged into the application and has opened or created a sheet.
Postconditions: The user interface theme is switched to the selected mode (light, dark, or maybe System though probably not), and all data is preserved.
Basic Flow:

1. **Access Theme Settings:** The user navigates to the settings or preferences menu.
2. **Select Theme:** The user chooses the theme they are looking for (light or dark).
3. **Apply Theme:** The system applies the selected theme to the entire application's interface.
4. **Confirm Theme Change:** The system provides visual feedback that the theme has been applied.

Extensions (Alternative Flows):

- **Automatic Theme Switching:** The system can automatically switch themes based on the time of day or user preferences, similar to the system I was talking about, unsure if it will be done.
- **Custom Themes:** The user can create and apply custom themes with specific colors and styles.

Special Requirements:

- The theme change should be pretty close to instantaneous and should not disrupt the user's workflow in a noticeable way.
- The application should support high-contrast themes for accessibility.

OpenIssues:

- How should the system handle third-party themes or plugins that may conflict with built-in themes?
- What fallback options are available if the theme change fails?

Name: Multiple User Editing
Actor: User

Stakeholders and Interests:

- User: The user's company and coworkers have to work on the same data table in the spreadsheet but don't want to have to waste time combining them at the end of the day!
- System: Only one user should be able to edit a single cell at the same time, not allowing for ConcurrentModificationException

Preconditions:

- Users are logged into the application and have access to the shared spreadsheet.
- The spreadsheet is set up for collaborative editing.

Postconditions:

- Changes made by users are saved in real-time, and the spreadsheet reflects the latest updates from all users if this is possible.
- No data conflicts or overwrites occur due to simultaneous edits.

Basic Flow:

1. **Open Shared Spreadsheet:** Users open the shared spreadsheet from their respective devices.
2. **Real-time Collaboration:** Users can see each other's cursors and selections in real-time.
3. **Edit Cell:** A user selects a cell to edit and the system locks the cell for that user, preventing others from editing it simultaneously.
4. **Save Changes:** The user makes changes to the cell and saves them, Also, the system immediately updates the spreadsheet and unlocks the cell.
5. **Reflect Changes:** The changes are reflected in real-time on all users' screens.

Extensions (Alternative Flows):

- **Conflict Resolution:** If two users attempt to edit the same cell simultaneously, the system prompts the second user that the cell is currently being edited. The system provides options to wait, notify when the cell is available, or edit a different cell.
- **Offline Edits:** If a user goes offline while editing, the system temporarily locks the cell and saves the changes once the user reconnects.

Special Requirements:

- The application should handle high-frequency updates and large spreadsheets without a huge dropoff in performance.
- The system should provide visual indicators for locked cells and active users.
- Changes should be saved and synced in real-time with minimal latency.

OpenIssues:

- How should the system handle conflicting edits in adjacent cells that depend on each other?
- What notification mechanisms should be in place for users when another user finishes editing a cell?

Name: Auto-saving
Actor: User

Stakeholders and Interests:
- User: wants session data to be automatically saved, in case manual save is overlooked
- System: Must ensure correct data is saved and stored appropriately

Preconditions: User must be logged into application and created a sheet
Postconditions: sheet is saved with latest updates
Basic Flow:
1. User opens an application sheet and starts session
2. User manipulates data on the sheet
3. System periodically performs save operation on sheet throughout the session

Extensions:
- May be based on user data manipulation instead of time tick
- Could be toggled on/ off
- System could minimalistically update user if autosave has been applied (eg: check mark in a corner)

Special Requirements:
- May impact performance if implemented naively

OpenIssues:
4. Could conflict with undo feature depending on how many past versions can be stored.

Name: Undo (limit to 1)
Actor: User

Stakeholders and Interests:
- **User**: wants to retract change made to the sheet

- **System:** Keeps track of the preceding state

Preconditions: User must be logged in and making changes to the sheet
Postconditions: Last change made by user is retracted and sheet is restored to previous state
Basic Flow:

1. User makes update to sheet during active session
2. User invokes undo feature
3. Contents of the cell will revert to previous state; before user added the new input

Extensions:

- Could potentially remove limit of 1 undo
- Could extend it to undo functional changes

Special Requirements:

- Store previous state of the cells

OpenIssues

5. Limiting to only 1 undo can be restrictive
6. Can conflict with autosave (may cause autosave feature to be undesirable

Name: Restoring prev versions (up to 5 stored)
Actor: User

Stakeholders and Interests:

- **User:** wants to restore previous version of the sheet if new session changes are not desired
- **System:** Must keep track of previous saved versions during previous application sessions

Preconditions: User is logged in and has saved previous versions of the sheet
Postconditions: A previous version of current sheet is restored and replaces current version
Basic Flow:

1. User makes changes to sheet in the current session
2. User invokes ‘restore previous version’ feature
3. Message pops up, asking user to confirm that this feature is called
4. The data recorded in the version of the sheet saved in the previous application session replaces current data in cells

Extensions:

- Could have a recorded history of versions, allowing user to restore any previous saved version of the sheet

Special Requirements:

- May need to store data recorded in cells of previous sessions

OpenIssues

- May be difficult to undo