

Automating Lab Product Stability Tracking: Reducing Task Time by 70% in One Week

Ben Christiansen-Developer & Designer

Built a JavaScript-based app that automated the manual calculation of lab product stability, reducing a 20-minute task to 6 minutes and freeing MLS staff to focus more on patient sample analysis.

Outcome Statement

Developed an app that automatically calculates remaining stability hours for lab products, cutting a 20-minute manual task down to 6 minutes — a 70% faster process. Staff gained time back immediately after launch, boosting overall productivity.

Context

Medical Laboratory Scientists (MLS) are required to quickly analyze and report results for a high volume of patient specimens.

They are responsible for instrument maintenance and quality control tests. These duties are time consuming and tools are needed to make them more efficient.

For example, we discussed the current manual workflow for calculating remaining stability of lab products, taking 20 minutes on average. A way to automate the process would help free up time to focus on patient samples.

My role in the project

Collaborating closely with MLS to understand the features they would find most useful, I applied my HTML, CSS, and JavaScript skills to design and code the calculation app from scratch to automate the calculation of remaining product stability. Upon completion, I collaborated with the lab team to test the app's accuracy and ease of use.

The mandate: 50%+ reduction in task time in 1 week

MLS staff were tasked with maintaining stability of reagents (lab products) on two specimen analyzers, with each analyzer used on alternating days for testing. MLS manually calculates remaining reagent stability hours to prepare for the next day. We needed a solution to reduce the 20-minute manual process, improving efficiency and freeing up time for patient sample analysis.

Note:

All tools were used to assist with research, debugging, and code explanation. All design decisions, implementation, and final testing were completed by me.

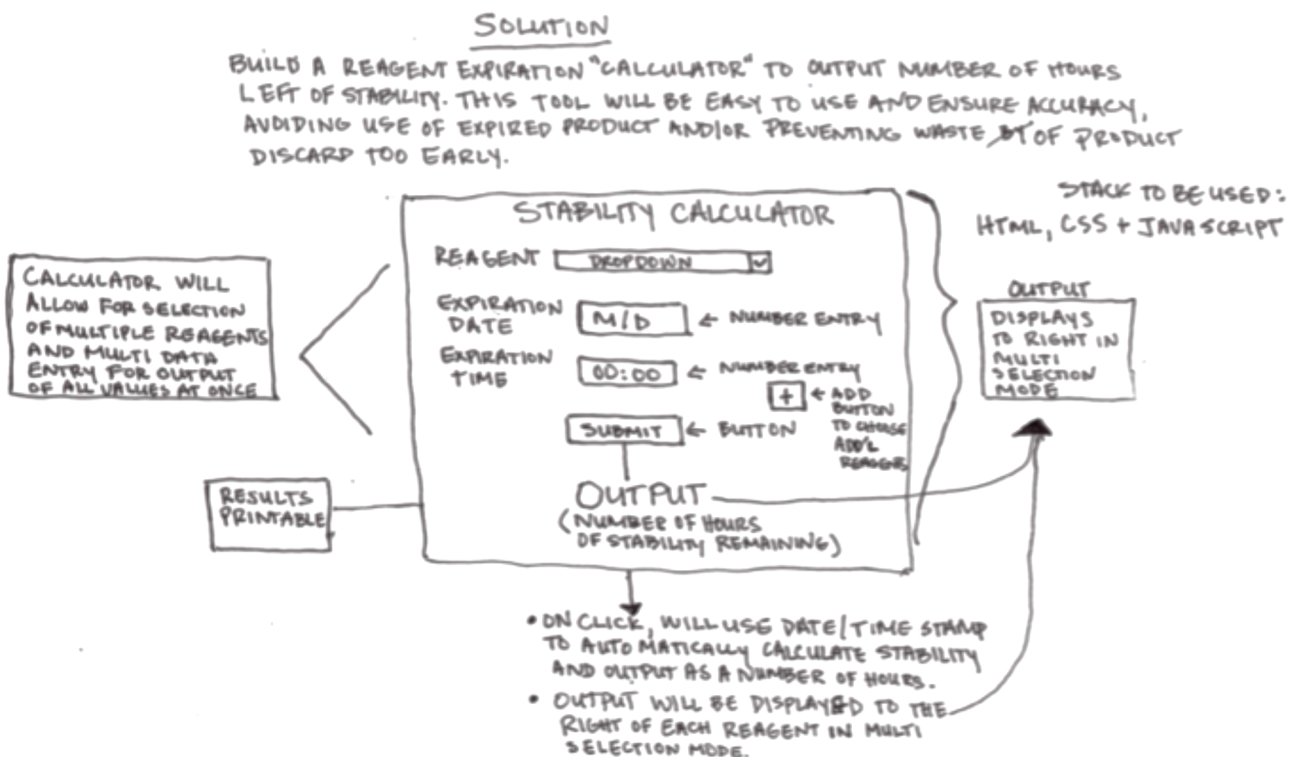
Day 1-2:

I spoke with Anny, an MLS, to understand what features and functions the app would require. We discussed the importance of it being easy to use and 100% accurate.

We collaborated on a wireframe to visualize the design and plan the UX/UI components. This included multiple value inputs to calculate and display the remaining stability hours, with an option to print the results.

Project wireframe: used to complete initial HTML coding and CSS styling.

- 2 DIFFERENT ANALYZERS
- SHARED REAGENTS - EACH WITH IT'S OWN STABILITY CRITERIA (EX: ^{SOME} EXPIRE 48 HRS FROM OPEN DATE/TIME, OTHERS 120 HRS)
- CURRENT PROCESS: MED TECH MANUALLY CALCULATES NUMBER OF HOURS LEFT OF STABILITY AND ENTERS VALUE INTO ANALYZER THE REAGENT IS BEING MOVED TO.



Day 1-2 Continued:

Initial prototype:

The screenshot shows a web application titled "Remaining Stability Calculator". It features a central image of purple and blue spheres. Below the image is a form with three input fields: "Select Reagent:" with a dropdown menu showing "Neoplastine C1 Plus", "Expiration Date:" with a date picker showing "mm/dd/yyyy", and "Expiration Time:" with a time picker showing "--:--". To the right of these fields is a green "Submit" button with a plus sign. Below the form is a green "Print" button.

Day 3-4:

I shared the app prototype with Anny and we discussed the following updates:

- Submit button to be moved below the inputs.
- A new input for reagent volume.
- A "-" button to delete unintentional entries.
- A "reset" button to clear entries was needed.
- Output rounded down to the nearest whole hour.

Version 2 wireframe:

- 10/12/25 - MEETING WITH ANNY P. MLS. VERSION 2 CHANGES/ADDITIONS
- ① MOVE SUBMIT BUTTON.
 - ② ADD INPUT (NUMBER TYPE) FOR VOLUME ENTRY. NO CALCULATION NECESSARY FOR THIS FIELD. SIMPLY NEEDED FOR PRINTOUT OF REMAINING STABILITY HOURS OUTPUT.
- OUTPUT/PRINTOUT
- OUTPUT HOURS SHOULD ROUND DOWN TO NEAREST WHOLE NUMBER.
 - PRINT VERSION SHOULD HAVE ALTERNATING GRAY/WHITE ROWS FOR READABILITY.

The wireframe shows the "Remaining Stability Calculator" with a central image of purple and blue spheres. Below the image is a form with three input fields: "Select Reagent:" with a dropdown menu showing "Neoplastine C1 Plus", "Expiration Date:" with a date picker showing "mm/dd/yyyy", and "Expiration Time:" with a time picker showing "--:--". To the right of these fields is a green "Submit" button with a plus sign. Below the form is a green "Print" button. A handwritten "VOLUME:" label with a circled "2" is next to a new input box. A circled "1" is next to the "Submit" button. Arrows point from the "VOLUME:" input box and the "Submit" button to the "Print" button.

* MLS TIMED TASK AND DOING IT MANUALLY, TAKES ABOUT 20 MIN.

Day 3-4 continued:

HTML and CSS were updated to accommodate the layout changes and button additions.

JavaScript code was written using reagent, expiration date and expiration time inputs and comparing the values to a timestamp from the user's computer in ISO 8601 format / UTC time. This calculated the difference in hours and output the remaining stability rounded down to the nearest whole hour.

Version 2 completed: Revised prototype with new buttons and improved usability.

The screenshot shows the 'Remaining Stability Calculator' interface. At the top, the title 'Remaining Stability Calculator' is displayed in green. Below the title is a decorative banner with colorful, abstract shapes. The main form area contains four input fields: 'Select Reagent:' with a dropdown menu showing 'Neoplastine C1 Plus', 'Expiration Date:' with a date picker showing 'mm/dd/yyyy', 'Expiration Time:' with a time picker showing '--:--', and 'Volume:' with a text input field. To the right of the Volume field is a '+' button. Below these fields is a green 'SUBMIT' button. At the bottom of the form is a trash icon button.

Day 5:

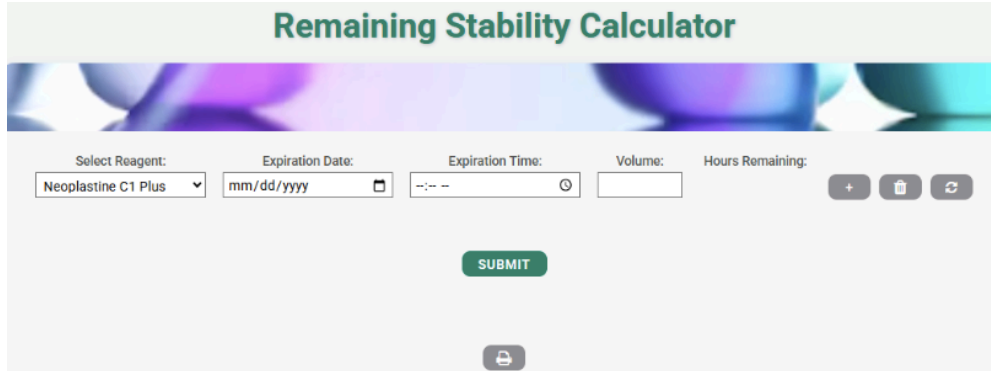
I added CSS and JavaScript event listeners for the Add, Minus, and Reset buttons. The Add button appends a new row of input fields, the Minus button removes the selected row, and the Reset button clears all input fields in all rows. Clicking the Submit button calculates and displays all outputs.

Version 3 wireframe:

The wireframe is titled 'VERSION 3' and shows the calculator interface with handwritten annotations. The form fields are the same as in Version 2. Handwritten notes include: '1. ADD A MINUS "-" OR TRASH BUTTON TO ENABLE DELETION OF ENTRIES.' and '2. ADD A RESET BUTTON TO CLEAR ALL ENTRIES/ DATA.' The wireframe shows a trash icon button and a 'RESET' button with a circular arrow icon, both with circled numbers 1 and 2 next to them. The 'SUBMIT' button is also present.

Day 5 Continued:

Version 3 completed: Refined design supporting multiple input rows and streamlined workflow.

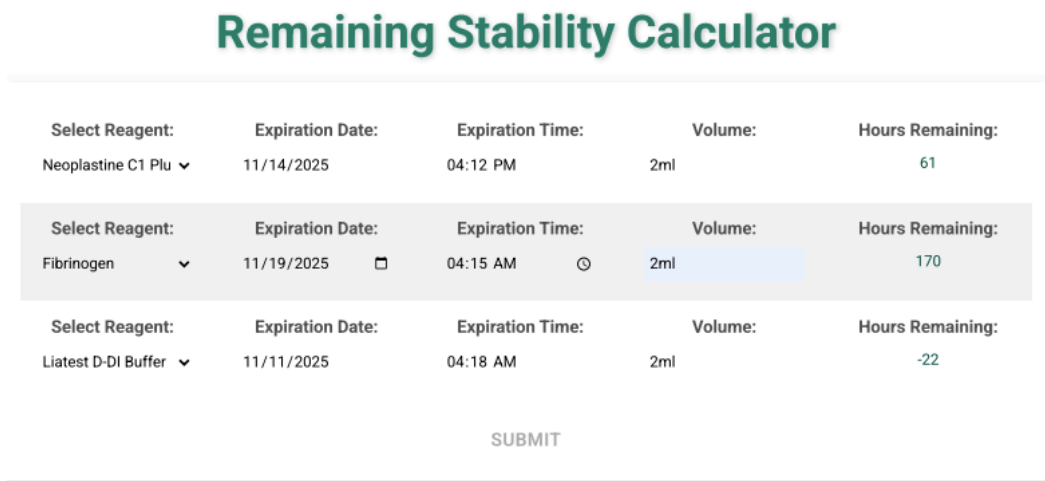


The screenshot shows the 'Remaining Stability Calculator' interface. It features a header with the title 'Remaining Stability Calculator' in green. Below the header is a form with five input fields: 'Select Reagent:' (a dropdown menu showing 'Neoplastine C1 Plus'), 'Expiration Date:' (a date picker showing 'mm/dd/yyyy'), 'Expiration Time:' (a time picker showing '--:--'), 'Volume:' (an empty text input), and 'Hours Remaining:' (an empty text input). To the right of these fields are three buttons: a plus sign (+), a trash can icon, and a refresh icon. Below the form is a green 'SUBMIT' button and a print icon at the bottom center.

Day 6:

After further consideration and confirming with Anny, I changed the JavaScript functionality by making the Reset button clear only the input fields and output for the row in which it is clicked. This prevents MLS from having to start over when only one row needs to be deleted. I added an event listener to the Print button to open the printer dialog. I also implemented a print-specific CSS media query to format the report for readability, which included zebra striping on alternate rows and hiding buttons and images to ensure a clean, professional layout when printing.

Report with CSS formatting: Final print layout showing zebra striping and clean formatting.



The screenshot shows the 'Remaining Stability Calculator' interface with a table of three rows. The table has five columns: 'Select Reagent:', 'Expiration Date:', 'Expiration Time:', 'Volume:', and 'Hours Remaining:'. The first row is for 'Neoplastine C1 Plu' with an expiration date of 11/14/2025, time of 04:12 PM, volume of 2ml, and 61 hours remaining. The second row is for 'Fibrinogen' with an expiration date of 11/19/2025, time of 04:15 AM, volume of 2ml, and 170 hours remaining. The third row is for 'Liatest D-DI Buffer' with an expiration date of 11/11/2025, time of 04:18 AM, volume of 2ml, and -22 hours remaining. Below the table is a 'SUBMIT' button.

Select Reagent:	Expiration Date:	Expiration Time:	Volume:	Hours Remaining:
Neoplastine C1 Plu	11/14/2025	04:12 PM	2ml	61
Fibrinogen	11/19/2025	04:15 AM	2ml	170
Liatest D-DI Buffer	11/11/2025	04:18 AM	2ml	-22

Day 7:

Finalized the HTML, CSS, and JavaScript files in Visual Studio Code and pushed all project files to a GitHub repository, including a detailed README documenting the project.

Automated Testing

To further verify the calculator's reliability, I implemented automated tests using Cypress. While the MLS team and I had already tested the application manually, automated testing allowed the calculator's functionality to be validated repeatedly and consistently.

The Cypress test suite simulates real user interaction with the interface by entering reagent data, selecting expiration dates and times, and triggering the calculation.

Test Scenarios

The automated tests verify several core behaviors of the application:

- Confirming the calculator page loads successfully.
- Verifying that reagent inputs and expiration values produce the correct remaining stability calculation.
- Testing how the application behaves when required inputs such as expiration date or time are missing.
- Checking edge cases including past expiration dates and expiration dates far in the future.

Ensuring Consistent Time-Based Calculations

Because the calculator compares expiration values to the user's system time, the tests use Cypress's clock control feature to freeze time during execution. This ensures that the calculation results remain consistent each time the tests are run.

Testing Workflow

1. Cypress loads the deployed calculator application.
2. The test simulates a user entering reagent and expiration data.
3. The calculation is triggered by clicking the submit button.

Day 7 Continued:

- Cypress verifies that the resulting output appears and matches the expected result.

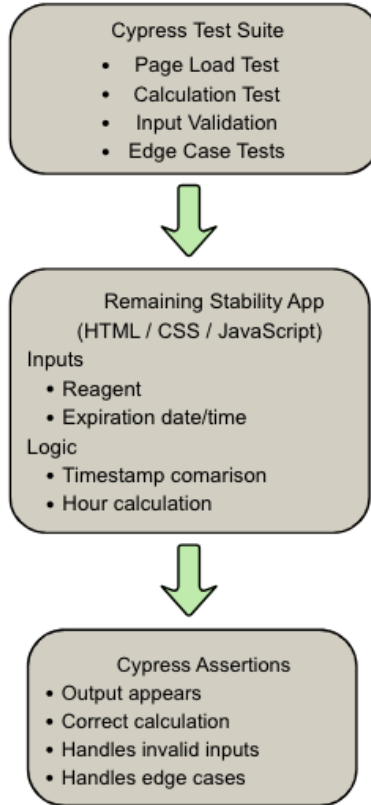


Figure: Automated testing workflow using Cypress.

Cypress tests simulate user interaction with the calculator by entering reagent data, triggering the calculation, and verifying that the resulting output behaves as expected.

Outcome

The addition of automated tests improved confidence in the calculator's accuracy and stability. The Cypress test suite now provides a repeatable way to confirm the application behaves correctly if updates are made in the future.

Final Verification

After completing the automated testing, MLS staff and I performed a final review of the calculator to confirm that the interface was easy to use and that all outputs were accurate. The application successfully met the project goal of reducing the manual calculation process by more than 50%.

Project Reflection:

This project deepened my understanding of JavaScript and front-end development while highlighting the impact of close collaboration with end users. Working alongside MLS staff showed me how thoughtful design and automation can significantly improve efficiency. Seeing the tool reduce task time by over 70% reinforced my motivation to build solutions that make everyday work easier and more effective.