Criterion E: Evaluation

Original Success Criteria from Criterion A:

#	Main	Subsections
1	Update the day's data in Google Sheets	a. Use Google Sheets functions to create numbers from numerical and alphabetical data and import necessary data
2	Update student's total Liveschool points	a. Update on the Google Sheet by adding the points of those who came to Mathnasium on the day of the program running using JavaScript and Google Sheets API
3	Enable log-in and logout	a. Display error message if user's input is not seven digits long b. Display error message if student name and ID do not match c. Display error message if none found in database d. Change UI based on authentication status of admin or customer e. Logout button on customer and admin pages should redirect to log-in page
4	Read spreadsheet and retrieve data for ID entered	a. Utilize Google Sheets API and Fetch API in conjunction with JavaScript to read data from Google Sheets b. Store locally for use in other HTML pages or immediate use
5	Display data to customers	a. Display total liveschool points of student b. Display the five most recent comments for customer, with scroll bar if too long c. Display name of customer
6	Admin should be given ability to search, using ID, for any given student	a. Display error message if ID input not seven digits long b. Display error message if valid ID not found in database c. if ID found, display name, Liveschool points, and last five comments associated with ID d. Only change displayed data if another successful ID is entered
7	Organized UI	a. Ease of use and clean interface for a good customer experience

Evaluation:

Criterion	Evaluation
1	By testing different entries on the "Sample Gameplan" sheet together and checking if the Liveschool points update on the "Data From Today" sheet in correspondence to that, my client and I found that my project met this criterion.
2	By running the update process from Visual Studio Code multiple times while changing data values on the "Data From Today" sheet, we were able to thoroughly check whether only the kids that came for the day would get their points accurately updated on the "Data Mastersheet" Sheet. My client and I found that my project met this criterion.
3	By using multiple student names (both found and not found in database) and IDs (found and not found in database along with valid v.s. nonvalid 7 digit ID), we were able to check error messages. We also tested the log out buttons on the admin and customer pages, which both redirected us back to the login page. The students/admin accounts were authenticated regardless of casing when their credentials matched perfectly and were also redirected to their respective pages/interfaces, causing my client and I to find that my project met this criterion.
4	We assessed this criterion with the help of others, as reading the spreadsheet with the APIs is what made updating Liveschool points and data display successful. Hence, my client and I found that my project met this criterion.
5	We assessed this criterion by logging in on multiple customer accounts and finding that the Liveschool points, comments (with scroll bar if too long), and name were all perfectly displayed. My client and I found that my project met this criterion.
6	By using various seven-digit IDs found in the database, seven-digit IDs not found in the database, and non-seven-digit IDs, we found that the respective error messages displayed at the right times. We also checked whether an invalid ID entered after a customer's data is displayed would change the data: and it didn't. When a valid ID was entered, the name, Liveschool points, and last five comments (scroll bar!) displayed. Therefore, my client and I found that my project met this criterion.
7	My choice of Mathnasium colors pleased my client. He was also able to navigate through the user interface without prompting or confusion, so my client and I found that my project met this criterion.

Recommendations for Further Development:

My client and I agreed that for widespread use at Mathnasium, I would need to make the CSS styling more adaptable across all different sizes and screens of different devices. To add on, I would also need to change my instances of Fetch API to Google Sheets API, as Fetch API has a free trial limit for data requests that would likely be exceeded soon with implementation on a wide scale. Hence, converting the program fully to Google Sheets API would save Mathnasium from an extra fee.

My client also added that while my updating of the Liveschool points on Google Sheets from my own Visual Studio Code was effective and unproblematic, I could add a feature in the future

which allows an admin to update from the website itself. To implement this, I would have to figure out the aforementioned Google Sheets API more to scale it to HTML/CSS websites, or maybe even change the format of my project to an app. Of course, if my app works well in the Mathnasium system, I could add more functionality currently being divided among many apps that Mathnasium uses to streamline their work (i.e. they spread their communication to parents among apps, email, and phone). We agreed that there was even a possibility of implementing algorithms in conjunction with the instructors that can help recognize patterns of under-performing students who need more aid as in better instructors or better instructing material. The possibilities are endless following a seamless first launch.

Word Count: 263