**Reflective Paper**

For this project, I developed an advanced **Todo App** designed to help users manage their tasks effectively. The app includes essential features like adding, editing, and deleting tasks, as well as advanced functionalities such as sorting, searching, highlighting overdue tasks, and tracking progress. Additionally, it incorporates persistent local storage to ensure tasks remain available across sessions and a theme toggle to allow users to switch between light and dark modes.

**What I Built and How I Approached It**

The app was built step by step, starting with the basic structure and functionality. My initial focus was on creating a clean HTML framework and ensuring all the necessary inputs and buttons were present. Once the basics were functional, I moved on to implementing task management features. For example, I wrote a function to handle adding tasks and saving them to localStorage. From there, I enhanced the app by introducing features like sorting tasks by due date, priority, and name, and filtering tasks through a search bar.

For the theme toggle, I used event listeners to add or remove a dark-theme class to the body and saved the user’s preference in localStorage. This way, the chosen theme persists across page reloads. Ensuring seamless integration between these features required me to break the functionality into modular functions and test them rigorously.

**Challenges and How I Overcame Them**

One of the biggest challenges I faced was implementing the overdue task highlighting feature. Initially, comparing dates in JavaScript caused issues because of inconsistent formatting. After some research, I standardized the dates by using new Date() and ensured proper comparisons. Debugging this helped me understand how to work with JavaScript’s Date object more effectively.

Another challenge was ensuring the progress tracker updated dynamically. At first, the percentages were incorrect because the calculations didn’t account for edge cases, such as having no tasks or deleting all tasks. Adding conditional checks resolved this issue and ensured the progress tracker was accurate.

Cross-browser compatibility was another area that required attention. Certain features, like the CSS animations for adding and removing tasks, behaved differently in various browsers. I used browser developer tools to identify the discrepancies and made adjustments to ensure a consistent experience.

**What I Learned**

This project reinforced my understanding of JavaScript, particularly in handling arrays, manipulating the DOM, and working with local storage. I also gained insights into CSS animations and transitions, which were used to enhance the user experience when tasks were added or removed.

Debugging and testing became an integral part of the process, and I learned to use browser developer tools effectively to trace and resolve issues. Another key takeaway was the importance of planning and modularizing code. Breaking down the project into smaller components made it easier to manage and debug.

**Feedback and Ideas for Improvement**

While the app works well, there are areas I’d like to improve. For instance, implementing reminders or notifications for upcoming tasks would make the app more practical for users with busy schedules. Another idea is to introduce task categories or tags for better organization. Adding responsive design features, such as drag-and-drop task reordering, could also enhance usability.

Finally, integrating a backend system to sync tasks across devices would make the app more robust. This would also allow for user authentication and further customization options, such as personalized themes.