CSE 110 Warmup Submission

Team Number: 30
Team Name: Echo: 30
Github Repo Link: https://github.com/cse110-sp24-group30/warmup-exercise
Youtube Link: https://youtu.be/DcuYq5LKROc?si=k2MqD_VKYMJVTgMo

SWOT Analysis:

Strength

- Team's proficiency in HTML, CSS, and JavaScript was a significant strength, enabling us to implement features like task editing, deletion, and dynamic rendering without relying on external libraries
- The use of a straightforward technology stack without libraries or frameworks reduced complexities associated with dependencies and versioning
- Specialized roles in UI/UX design, backend logic, and front-end styling (structured across different files like app.js, index.html, and styles.css) helped maintain an organized development environment
- The project facilitated a better understanding of how team members work together, allowing the group to effectively gauge individual skill sets and preferences

Weakness

- Since the data is hardcoded from JSON files, the widget lacks dynamic interaction with real-time data, which could limit its practicality beyond academic purposes
- Due to time constraints, comprehensive documentation and thorough testing were not as prioritized as they could have been
- Widget is designed for the web, so UI may be hindered on mobile devices

Opportunity

- Offers opportunity to delve deeply into the basics of web development and to practice clean, maintainable coding practices without relying on frameworks
- Each team member had the chance to work on areas that piqued their interest, whether it was UI design, backend processing, or user interaction which not only enhanced individual skill sets but also increased overall job satisfaction and engagement with the project
- There is scope to expand the project by integrating more complex functionalities such as syncing with calendars, adding user authentication, or using local storage to save task states

Threat

- Relying solely on vanilla JavaScript, HTML, and CSS may limit our ability to implement more complex features efficiently, which could be better handled by modern frameworks or libraries
- The limited timeframe may have led the team to prioritize certain functionalities over others, potentially impacting the overall quality and depth of the final product

- As team members are new to working with each other, there's a risk associated with the initial learning curve and establishing effective communication patterns which took sometime to get used to

Reflective Summary

Throughout the development of the Task List Widget, our team faced a variety of challenges that tested our technical abilities and collaborative skills. This project was not just about coding; it was a comprehensive exercise in teamwork, problem-solving, and time management, all within the structured confines of our course objectives.

From the outset, we recognized the importance of clear communication and role clarity. As a new group working together for the first time, establishing effective communication channels was our first hurdle. We addressed this through meetings after-class, and updates through our Slack workspace, which helped us stay aligned on our goals and progress. These practices not only kept the project on track but also fostered a supportive environment where team members felt comfortable sharing ideas and feedback.

One limitation we encountered was ensuring that all team members were as involved and engaged as they could be. Given the varied skill levels and the nature of the project tasks, some members were less active than others. This highlighted the need for better role distribution and task allocation tailored to individual skills and learning goals, ensuring everyone could contribute meaningfully and gain from the project.

Technically, the decision to use only HTML, CSS, and JavaScript without any external libraries was initially seen as a challenge, in terms of implementing features that are typically more straightforward with frameworks, such as dynamic content updates. This limitation, however, turned into a valuable learning opportunity. It pushed us to delve deeper into the nuances of vanilla JavaScript and explore creative solutions to overcome these obstacles. For example, we developed a deeper understanding of how to manage state manually and manipulate the DOM effectively, which are crucial skills for any developer.

Reflecting on our project, one of the key lessons learned was the importance of adaptability. We often had to pivot our approaches based on technical feasibility and time constraints, which taught us the value of flexibility in software development. Additionally, this project highlighted the critical role of thorough testing and documentation—areas that, due to time constraints, we felt could have been more robust. This will serve as a poignant takeaway for future projects, emphasizing the need to allocate sufficient time and resources to these aspects to ensure maintainability and scalability.

Nevertheless, this exercise equipped us with firsthand insights into the complexities of software development, from sketching out our initial skeleton design to constructing the final implementation, all while enhancing our abilities to work effectively as a team. These lessons will not only help guide our approach to when the final project arises, but will further extend across our ongoing academic and professional development.