Team Task Distribution

Bala

Developed the core object-oriented structure of the system by implementing the crewMember base class and extending it through the derived human and alien classes. Additionally, I implemented the file-loading functionality, which reads crew data sequentially and populates the corresponding vectors using push_back() to create fully constructed object instances.

Mitchell

Implemented the **merge sort** algorithm used for both single-field and multi-field sorting operations. This includes sorting for human and alien crew members based on criteria such as training score, mission aptitude, and alphabetical ordering.

Lucas

Implemented an alternative sorting method using **selection sort** for all single-field and multi-field sorting cases for both human and alien vectors. I also designed and developed the interactive menu system, providing the user interface for navigating sorting options and executing program features.

Test Plan Summary

To evaluate the performance and reliability of our sorting functionality, we implemented two distinct sorting algorithms:

- Selection Sort
- Merge Sort

Both algorithms were initially developed and tested; however, after analyzing their computational efficiency and scalability, Merge Sort was selected as the primary sorting method. This decision was based on its superior time complexity and significantly better performance when handling larger datasets.

Selection Sort, while simple and easy to implement, operates with a time complexity of $O(n^2)$, making it impractical for processing extensive crew data. In contrast, Merge Sort provides a consistent $O(n \log n)$ performance and ensures stable, predictable behavior across all input sizes.

As a result, *Merge Sort was adopted as the final algorithm* for all sorting operations within the system.

During testing, we evaluated how the sorting algorithm behaves when multiple objects contain identical key values (e.g., equal training scores or matching names). In such cases, the sorted output preserves the original ordering of these objects as they appeared in the input file.

Because the crew data is read sequentially from top to bottom and inserted into the vector using push_back(), the earliest entries in the file are stored at the lowest indices of the vector. Therefore, when two or more objects compare as equal during sorting, the algorithm maintains their initial order, resulting in the objects that appeared earlier in the file being displayed first in the sorted output.

Weekly Scheduling

Because our group was formed late in the project cycle, we had only two days to complete the full implementation. As a result, we divided the workload strategically to maximize efficiency and ensure timely completion.

Lucas and Mitchell focused on developing and testing the sorting algorithms both the merge sort and the selection sort implementations, while Bala concentrated on designing and implementing the class structure, including inheritance, object construction, and file-loading functionality.