a. Meharvir Randhawa and Chris Ingram
b. We encountered problems related to memory management and implementing pointers in our program. Also, we had issues with file input parsing since parsing the input from the file and populating the graph with the provided data required precise handling to avoid messing up test cases or causing possible bugs/errors in our program.
c. We intended to create a more optimized error management system, however, we ended up implementing a simple error-handling system due to time constraints.
d. No code from external books or websites was directly used in our program however we used the site "https://www.freecodecamp.org" to get a better idea of how Dijkstra's algorithm functions.
e. We made several design decisions to ensure efficiency and clarity in our program. For example, the adjacency matrix was chosen to represent the graph due to its simplicity and ease of implementation. Additionally, dynamic memory allocation was utilized to handle matrices of variable sizes. Also, A Graph class was designed to encapsulate the graph-related functionalities, including adding edges and retrieving adjacency information. Lastly, Dijkstra's algorithm was employed to find the shortest paths from a given vertex to all other vertices in the graph. The algorithm was implemented efficiently using priority queues to optimize the search for the minimum distance vertex. All in all, the design aimed for a balance between simplicity, efficiency, and readability, while ensuring the functionality meets the project requirements.

Works Cited

Navone, Estefania Cassingena. "Dijkstra's Shortest Path Algorithm - a Detailed and Visual Introduction." *freeCodeCamp*, freeCodeCamp.org, 3 Feb. 2022, www.freecodecamp.org/news/dijkstras-shortest-path-algorithm-visual-introduction/.