Graphs.

1. Mapping with Dots

Once all the collected data was transferred to a digital map in Figma, we proceeded to mark various locations such as rooms, toilets, libraries, coworking rooms, buffets, and cafes. We represented these locations as circles with a radius of 3 pixels, forming dots on the map. Additionally, we strategically placed extra dots in the corridors to ensure smooth navigation. The routes between dots are formed by connecting each dot with its two neighboring dots, except for the first and last dot.

2. Collecting Dot Coordinates

After dotting the rooms on the maps, the next step involved collecting information about each dot. This information includes the x and y coordinates, the name of the dot, and an array of names representing the dots connected to the current dot. To ensure uniqueness, we assigned names to the dots using specific formulas. For rooms, the names followed the pattern [roomNumber] + "room". For connectors between rooms, the names consisted of [number] + [floorNumber] + "floor". Buffets, toilets, stairs, and elevators were named using the pattern "buffet" or "toilet" or "stairs" or "elevators" + [floorNumber] + "floor" + [streetName]. The street names used were "Abylaikhan," "Panfilov," "Tolebi," and "Qazybekbi." It's important to note that the dot coordinates are relative to the map's measurements and may be adjusted based on the display sizes of different devices.

3. Vertices and Edges in a Graph

Our map can be represented as a large graph consisting of vertices and edges. Each dot collected from the maps corresponds to a vertex in our graph. The vertices are connected to their neighboring vertices through edges. The graph is cyclic, undirected, simple, and connected. The edges in the graph are assigned weights equal to the distance between two vertices, which will be used for finding the shortest path in the graph.

Algorithms

1. Review algorithms for the shortest path.

During this project stage, the team conducted a comprehensive review of various algorithms for the purpose of selecting the most suitable one to be implemented in the application. The objective was to find the shortest path between two rooms within the application. Several methods and algorithms exist for achieving this, each with its own set of advantages, disadvantages, and time complexities. You can refer to the following image for a visual representation of these algorithms.

[https://miro.medium.com/v2/resize:fit:1400/format:webp/1\*rUs3PqtS5HzqnnhBgnvBbw.png](https://miro.medium.com/v2/resize:fit:1400/format:webp/1*rUs3PqtS5HzqnnhBgnvBbw.png)



The accompanying table provides an overview of the pros and cons associated with each algorithm. Considering that our application consists of approximately 1000 vertices and does not have any negative edges, it was determined that the most optimal choice for our application is the Dijkstra algorithm.

2. Implementation of the Algorithm

Our application utilizes the Dijkstra algorithm due to its suitability for our specific requirements. It was necessary to implement this algorithm on the front-end of the application to ensure its functionality in offline mode, without reliance on an internet connection or server access. The Dijkstra algorithm facilitates the discovery of the shortest path within a graph, allowing us to identify and retrieve every vertex encountered along the path from the source to the destination. This information is then displayed on the application's interface for user interaction.