**Updated Final Project Design Document**

**Program Requirements:**

**Assignment Description**

Create a real-world project that demonstrates the shortest path problem. Your project will be composed of a front-end application that makes use of a back-end. Your back end will solve the shortest path algorithm on a directed acyclic graph. The front-end will show your back-end works.

**Real-World Proposal to demonstrate the shortest path problem:**

There is a new uber service coming to Florence, Alabama! They call themselves “Fluber”. This service caters directly to students at UNA. The purpose of our program is to find the shortest path for “Fluber” to take the students where they need to go.

Every UNA student needs to access certain amenities on a daily basis. They need to be able to reach places such as Restaurants, Gyms, Libraries, Grocery Stores, Health Services and Entertainment.

Fluber will be their friend when it comes to being able to get to these places as efficiently as possible. If a student needs to go to the nearest gym, restaurant, grocery stores etc. It’ll show them the nearest possible options and display the shortest path to whatever selection is made by them.

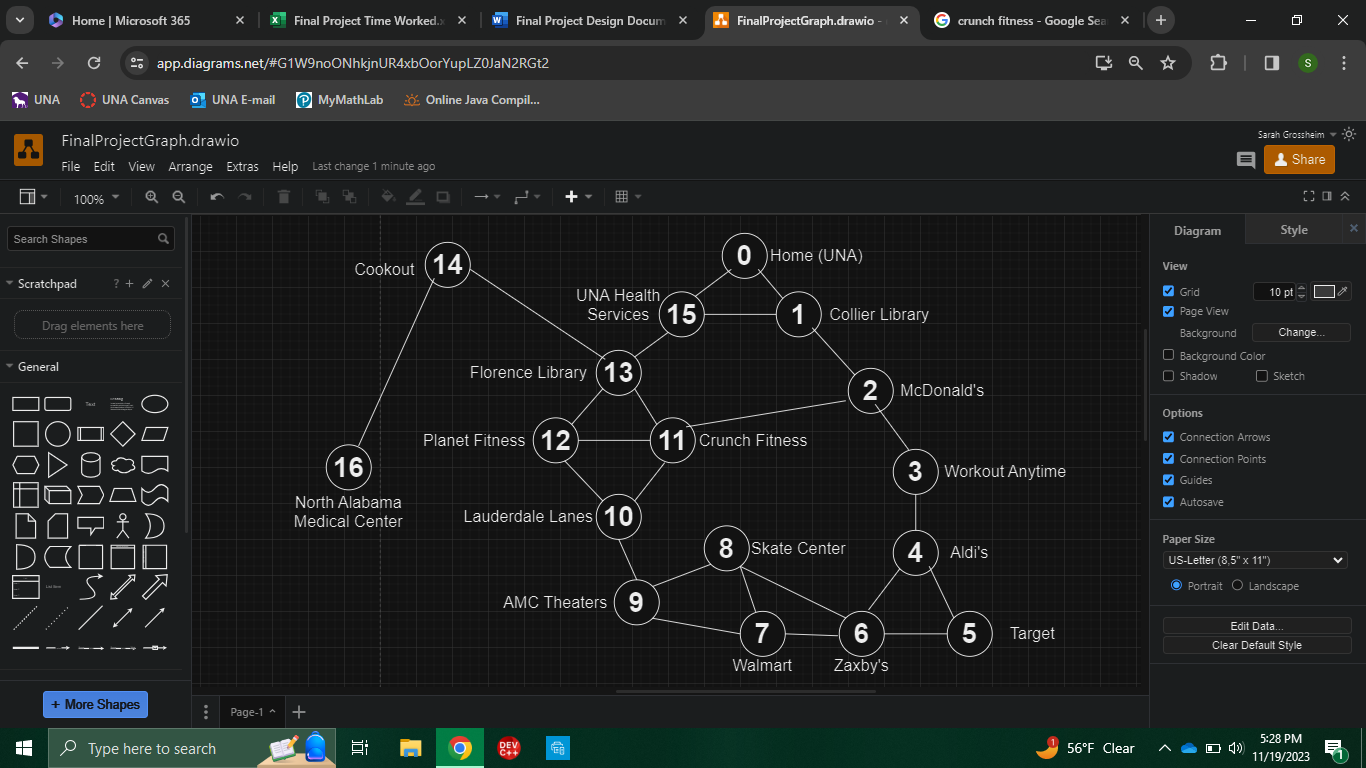
**Data Structure Choice:**

In this code, an adjacency list is used to represent the connections between nodes in the graph. This choice is suitable because the graph is not densely connected – meaning, not every node is directly linked to every other node. An adjacency list is more memory-efficient in such cases because it only stores information about existing connections

Think of the adjacency list to list each node's friends (neighbors) directly. For each node, you have a list that tells you which other nodes it is connected to. This is practical for algorithms like Dijkstra's, implemented in the code, where you need to efficiently explore the neighbors of each node to find the shortest path.

To implement this code, we imported an adjacency matrix from a .txt file. This used to represent the connected nodes. Below you can see a picture of what our graph looks like and the adjacency matrix that is used to represent it.

**Basic Graph**



**Adjacency Matrix**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 2 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 11 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 15 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

**List of places with their corresponding vertex #:**

1. 0 – Home (UNA)
2. 1 – Collier Library
3. 2 – Mcdonald's
4. 3 – Workout Anytime
5. 4 – Aldi's
6. 5 – Target
7. 6 – Zaxby's
8. 7 – Walmart
9. 8 – Skate Center
10. 9 – AMC Theaters
11. 10 – Lauderdale Lanes
12. 11 – Crunch Fitness
13. 12 – Planet Fitness
14. 13 – Florence Public Library
15. 14 – Cookout
16. 15 – UNA Health Services
17. 16 – North Alabama Medical Center

Test plan for Sunday:

* First, I wake up and knock out an early morning workout at **Planet Fitness**.
* The only snack I like to eat is pork rind cereal, but I ran out this morning. So now I must go grocery shopping. Head to **Aldi's**!
* Since I had to buy milk for my Porkio's, I must now take my groceries **home**.
* I have a test tomorrow, so I will go to the **Collier Library** to have a study sesh.
* My roommate is hungry, and they invited me to go out to eat. Head to **Zaxby's**.
* Finally, go back **home** for a good night's sleep.

Test plan for Monday:

* First, I wake up and go to the **Florence Public Library** to study for my test.
* Then, I go back **home** (to UNA), to take my test.
* I wasn't feeling well during my test, so after, I visited **UNA Health Services.**
* Uh oh, it is worse than I expected. I must go to the hospital. Go to **North Alabama Medical Center**.
* Turns out I got food poisoning last night from Zaxby's. Go to **Walmart** to pick up my prescription.
* Go back **home** to rest for the rest of the day. Distance = 6

Test plan for Tuesday:

* I woke up this morning and my classes for today were cancelled. Let's have some fun today! Head to **Lauderdale Lanes** for a game of bowling!
* I wanted to continue the physical activity, so I went to **Workout Anytime** to break a sweat.
* After all that physical activity, I was craving a corn dog. Go to **Cookout**!
* I have been wanting to see the new Hunger Games movie, so go to **AMC theaters**.
* My girlfriend just got out of class and wants me to take her on a date. Go to **Target** to surprise her with a gift.
* I met up with her (she loved her gift), now take her skating at the **Skate Center**.
* Take her out to eat but only let her order off the dollar menu at **Mcdonald's**.
* You are pooped. It is time to go **home**.

**Three Day Test Plan:**

|  |  |  |
| --- | --- | --- |
| **Day of the week** | **Nodes visited & Respecting Vertex #** | **Shortest Path**  **D = Distance** |
| **Sunday’s schedule** | **1. Planet Fitness (12)**  **2. Aldi’s (4)**  **3. Home (0)**  **4. Collier Library (1)**  **5. Zaxby’s (6)**  **6. Home (0)** | **1. 0->15->13->12 D = 3**  **2. 12->11->2->3->4 D = 4**  **3. 4->3->2->1->0 D = 4 4. 0->1 D = 1**  **5. 1->2->3->4->6 D = 4**  **6. 6->4->3->2->1->0 D = 5** |
| **Monday’s schedule** | **1. Florence Public Library (13)**  **2. Home (0)**  **3. UNA Health Services (15)**  **4. North Alabama Medical Center (16)**  **5. Walmart (7)**  **6. Home (0)** | **1. 0->15->13 D = 2**  **2. 13->15->0 D = 2**  **3. 0->15 D = 1**  **4. 15->13->14->16 D = 3**  **5. 16->14->13->12->10->9->7**  **D = 6**  **6. 7->6->4->3->2->1->0**  **OR**  **7->9->10->11->13->15->0**  **D = 6** |
| **Tuesday’s schedule** | **1. Lauderdale Lanes (10)**  **2. Workout Anytime (3)**  **3. Cookout (14)**  **4. AMC Theaters (9)**  **5. Target (5)**  **6. Skate Center (8)**  **7. Mcdonald’s (2)**  **8. Home (0)** | **1. 0->15->13->11->10 D = 4**  **2. 10->11->2->3 D = 3**  **3. 3->2->11->13->14 D = 4**  **4. 14->13->12->10->9 D = 4**  **5. 9->7->6->5 D = 3**  **6. 5->6->8 D = 2**  **7. 8->9->10->11->2**  **OR**  **8->6->4->3->2 D = 4**  **8. 2->1->0 D = 2** |

**Algorithm In Pseudocode:**

Program working:

Categories = [Restaurants, Gyms, Grocery Stores, Libraries, Healthcare, Entertainment]

1. The Welcome page is displayed to the user
2. The program verifies if the user is a UNA student
3. The program prompts the user to select the category of the place they’re trying to visit.
4. From the selected category, the program then provides the user with all the available choices of the places the user can visit.
   1. The program then displays the shortest path and distance to the selected destination, from the starting position.
   2. Update the current position of the user to their current location on the graph and present them with the options to continue traveling elsewhere or give them the option to go home or to exit the program.

Djikstra’s Algorithm (Finding the Shortest Path) in Pseudocode:

1. Make a loop that keeps repeating until all vertices have been visited.
2. In the loop, visit an unvisited vertex with the shortest distance from the starting vertex.
3. Make another loop for each unvisited neighbor of the current vertex.
   1. Calculate the distance from the start.
   2. If calculated distance is < known distance:
      1. Update the shortest path/distance to the vertex.
      2. Update the previous vertex with the current index
4. Finally add current vertex to the visited list.
5. Stop when there are no more unvisited vertices.

(This algorithm begins with the root or your “home”. When the person decides to travel to a new destination, instead of turning around and going back home and then traveling somewhere else, the shortest path will be calculated from the new current root to find the shortest path to their new destination.)