

# **Deliverable J: User Manual**

Submitted by

A1, Team 15

Ghaleb Ghaleb, 300088667

Marshall Steele, 300142928

Dave Ly, 300132139

Sihan Wang, 8359108

Zayd Ghazal, 300112270

December 10<sup>th</sup>, 2020

University of Ottawa

## **Abstract**

---

Library way finding system is an orientation applet that guides user through out different terrains. It uses technology such as NFC beacons, QR codes and web page interfaces to achieve a budget-friendly, user-friendly and development-friendly product. By placing NFC placards across the Library and applying the web app to the beacons, the entire library becomes electronically mapped and accessible to the user. Users can scan the beacons and tap their destination to navigate. The navigation can be both audio and visual. The comparative advantage over other available product makes it unique on the market.

# Table of Contents

---

Abstract.....	2
Table of Contents.....	3
List of Figures.....	3
1 Introduction.....	5
2 User Manual.....	7
2.1 Algorithm.....	7
2.2 NFC Technology.....	10
2.3 Query String.....	11
2.4 User Interface.....	13
3 Conclusions and Recommendations for Future Work.....	16
3.1 Lessons learned.....	16
3.2 Future Work.....	16
4 Bibliography.....	17
APPENDICES.....	18
APPENDIX I:.....	18
Bill of Materials (BOM).....	18

## List of Figures

---

Figure 1: Example of creating new functions. ....	8
Figure 2: Example of adding nodes on the current function.....	8
Figure 3: This is current implementation of the library. ....	9
Figure 4: Example of using Dijkstra's algorithm to derive the function.....	9
Figure 5: An example of the output direction forms the front entrance to service desk.....	10
Figure 6: Query String .....	11
Figure 7: The Query String code .....	12
Figure 8: urlParams' code .....	12
Figure 9: Screenshot of the Home page on user interface. ....	14
Figure 10: Screenshot of the Language page.....	14
Figure 11: Screenshot of the current location page with the destination options .....	15
Figure 12: Screenshot of the output of the pathway from Front Entrance to Service Desk. ....	15

# 1 Introduction

The navigation of the library is a problem identified with students and accessibility impaired individuals. The problem is the lack of familiarity with the environment and the lack of knowledge regarding the Library space. It also increases the opportunity cost of staffing as library staffs are forced to preform navigation duties even though they have other duties that may require their attention. This is complicated by the fact that the Library changes its design once every couple of years resulting in a peak demand in navigation. All this compounds to a single need - a navigation system that is affordable, easy and fast. This is where the Library way finding system comes in.

The fundamental goal of the Library Way finding system is to increase operating capacity of both the Library and the Library staff. By providing the navigation electronically, it allows the staffs of the library to be liberated from the chores of doing manual navigation. Not only is the staff's burden lessened, but users will find themselves less reliant on other people to find what they need thus likely to increase the use of the library. The easement of the library usage will create the ability for a better flow dynamic in the library to prevent congestions and increase traffic. Fundamentally, this project is about creating and fostering a sense of community, a sense of independence and a sense of academia in the University of Ottawa through the micro-level observations. It provides utilities to the staff of the library as well as the users of the library.

The design of the library way finding system is intelligently implemented through the fact that it is built from bottoms up. This means that it is built with the problem in mind, all the way up to the solution. In other words, the solution is built to fit all situations where a similar problem is faced. The way finding system does not require regular maintenance of machines nor does it require extreme technical knowledge. It is as simple as placing NFC or QR codes on

various locations and adding mapping to those nodes. The nodes can be added or removed as well as moved to new locations and it is not limited in size nor will cost significantly increase due to scaling. The UI is extremely intuitive, and the product offers both audio and text direction for all users. Language switching is as simple as simply adding the language required thus catering to an international audience. All of these strengths combine to make a cheap and easy product that can be scaled up to address mass navigation needs.

## 2 User Manual

In this section, we will go over the key features of our software. Starting with the JavaScript programming behind our website, we will focus on how our algorithm works along with the other small features behind our website. We will show what our design can do when it comes to solving our problem. Also, we will explain how we used NFC technology as the core of our solution.

All of our design files are uploaded to a GitHub repository, which can be found in our MakerRepo at <https://makerepo.com/Zen/uoway-gng2101-a15-library-wayfinding-system>.

### 2.1 Algorithm

To start off, we knew we had to figure out a way to get the fastest and easiest directions presented to the user based on their destination. We had the idea of having predefined data stored in the software for every specific case, but that did not allow for much innovation for the library. Instead, we did our research and found an optimal solution that still allowed the library to make changes without disrupting the pathfinding algorithm. Using a data structure in the form of a graph, we were able to implement Dijkstra's algorithm to calculate the shortest path possible from one point to another. The nodes of our graph would be the key points in the library like the washrooms, entrance, service desk, etc. For the moment, we added relative distances in-between nodes since we do not have access to the library for measuring due to COVID-19 reasons. Every node in the graph had relative edges(distance) to its neighboring nodes. So, every time the program would start, the code initializes the graph and inserts the accurate data inside of it. We also made sure that our graph is a directed graph, so showing which orientation to take to reach its neighboring node. In this snip of code, we see the constructor initializing the graph and creating the right arrays to contain the data.

```

class Graph {
  constructor() {
    this.nodes = []; //All the nodes in the graph
    this.adjacencyList = {}; //Every node and it's corresponding edge to neighbors
    this.directionList = {}; //Every node and it's corresponding orientation to neighbors
  }
}

```

Figure 1: Example of creating new functions.

Here, we can see the 2 functions used to add new nodes and edges to the graph. We thought creating functions for this is necessary in case the library decided to add more nodes or expand to another level. To add a node, make a call to the function `addNode(node)` and instead of `node`, write the name of the node, for example, `addNode("Front Entrance")`. Then, create edges between the node and its neighbors using the `addEdge` method and write the node's name, the neighbor's name, the distance and the orientation, for example, `addEdge("Front Entrance", "Mens Washroom", 15, "North")`.

```

//This function pushes a node to the Graph's collections
addNode(node) {
  this.nodes.push(node);
  this.adjacencyList[node] = [];
  this.directionList[node] = [];
}

//This function pushes the node1's edges into the Graph's collections
addEdge(node1, node2, weight, direction) {
  this.adjacencyList[node1].push({node:node2, weight: weight});
  this.adjacencyList[node2].push({node:node1, weight: weight});

  this.directionList[node1].push({node:node2, direction: direction});
  if (direction == "East"){ direction = "West";}
  else if (direction == "North"){ direction = "South";}
  else if (direction == "West"){ direction = "East";}
  else if (direction == "South"){ direction = "North";}
  this.directionList[node2].push({node:node1, direction:direction});
}

```

Figure 2: Example of adding nodes on the current function.



```

const map = new Graph();
map.addNode(" Front Entrance");
map.addNode(" Service Desk");
map.addNode(" Study rooms");
map.addNode(" Mens Washroom");
map.addNode(" Curbside Pickup");
map.addNode(" Womens Washroom");
map.addEdge(" Front Entrance", " Study rooms", 20, "West");
map.addEdge(" Front Entrance", " Service Desk", 16, "East");
map.addEdge(" Front Entrance", " Womens Washroom", 14, "North");
map.addEdge(" Womens Washroom", " Mens Washroom", 10, "East");
map.addEdge(" Womens Washroom", " Curbside Pickup", 5, "East");
map.addEdge(" Mens Washroom", " Curbside Pickup", 5, "West");
map.addEdge(" Mens Washroom", " Service Desk", 5, "South");

```

Figure 3: This is current implementation of the library.

After setting up the graph, we found an implementation of Dijkstra's algorithm online written in JavaScript (1). The algorithm can be called by using the `stp(start, end)` method and it takes the starting node and end node as parameters. We then needed it to output some form of directions for the user. So, after the program is done calculating the shortest path possible from their location to their destination, it creates a sentence where the directions are explained.

```

let directions = ("Starting from" + path[0] + " take ");
var i;
for (i = 1; i < path.length; i++) {
  directions += (times[path[i]] - times[path[i-1]]) + " steps towards" + (path[i]);

  for (let j = 0; j < this.directionList[path[i-1]].length; j++) {
    if (this.directionList[path[i-1]][j].node == path[i]) {
      directions += "(" + this.directionList[path[i-1]][j].direction + ")";
    }
  }
  if (i+1 < path.length) {
    directions += " then ";
  }
}
return directions;

```

Figure 4: Example of using Dijkstra's algorithm to derive the function

As seen in the code above, calling the `stp` method will return a variable containing what will be displayed to the user. Here is an example of the output when given the front entrance and service desk as arguments:

**Here are your directions**  
**Starting from Front Entrance take 16 steps towards Service Desk(East)**

*Figure 5: An example of the output direction forms the front entrance to service desk.*

## 2.2 NFC Technology

What really makes our design unique compared to other wayfinding technologies is our use of NFC technology. In most cases, Bluetooth beacons are used for wayfinding, however one issue our clients brought up was that the walls in the library could interrupt Bluetooth signals. Our wayfinding system still needed to register the user's present location. After brainstorming, we came up with idea of NFC placards spread out throughout the library where users can simply put their phones up to it and it will launch our wayfinding software while registering their location. Every NFC placard contains the link to our website along with a string value unique to every chip. For example, the front entrance placard will contain the link to our website and a variable with "Front Entrance" stored inside of it:

<https://zayood.github.io/Library-Wayfinding/Main/?node=Front%20Entrance>

To transfer that variable from the NFC card to our software, we had to include it in the URL. Looking at the link posted above, we have [node=Front%20Entrance](#). This field is called Query String and will be discussed more in the next section. Another great feature about NFC chips is the fact that no download is required to read it. Most smartphones have NFC technology implemented inside of them as it is also used in payment applications like "Apple Pay". They also

need no source of power to function. All in all, NFC technology is the necessary hardware for our software to function properly.

## 2.3 Query String

As mentioned in the previous section, we had to use Query String in our website. If you try running the website without any Query string values, you'll run into some issues with the node starting point. We made our website so that it can only be accessed by the NFC cards since they contain the first node. Query string values are added right after the URL of the website as show in the following example. A question mark is where the Query String begins.

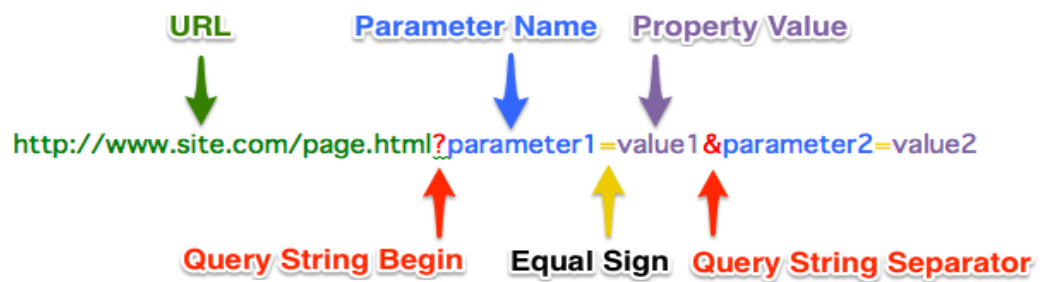


Figure 6: Query String

Query Strings are important to keep track of variables whenever changing pages on your website. For example, when the French option is selected on our website, we add “french=true” to our URL so that the website can consistently stay in French. Of course, we have to write code that will read the URL and store all the variables found in the Query String. To do so, we researched and found a good piece of code that reads and stores the data efficiently in JavaScript (2).

```

var urlParams;
var match,
    pl    = /\+/g,  // Regex for replacing addition symbol with a space
    search = /([^\&=]+)=?([^\&]*)/g,
    decode = function (s) { return decodeURIComponent(s.replace(pl, " ")); },
    query  = window.location.search.substring(1);

urlParams = {};
while (match = search.exec(query))
    urlParams[decode(match[1])] = decode(match[2]);
return urlParams;

```

Figure 7: The Query String code

After storing the values in the collection, the rest of the code can now access any variables needed to complete their functions. For example, when our wayfinding algorithm needs to get the starting node and final node, it can access it from our urlParams collection.

```

path = wayfind(urlParams["start"], urlParams["end"]);

```

Figure 8: urlParams' code

## **2.4 User Interface**

For our user interface, we wanted it to be as simple as possible and with contrasted colors so that any users with visual impairments can have a better experience operating the website. After scanning an NFC chip, it would bring the user to the main page where they will be prompted to answer a few questions about what kind of directions they would like, what language they would like the page to be in and where they are looking to go.

# Welcome to the uOttawa Library Wayfinding!



uOttawa  
What kind of directions?  
Quel type de directions?

Text

Audio

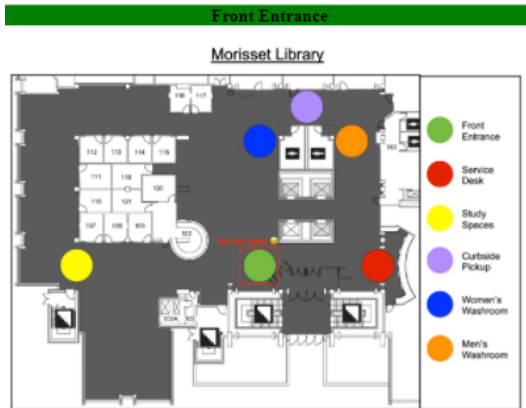
Figure 9: Screenshot of the Home page on user interface.

## Choose your language

English

French

Figure 10: Screenshot of the Language page.



Study Room

Service Desk

Women's Washroom

Men's Washroom

Figure 11: Screenshot of the current location page with the destination options

## Here are your directions

Starting from Front Entrance take 16 steps towards Service Desk(East) then 5 steps towards Mens Washroom(North)

tts

Figure 12: Screenshot of the output of the pathway from Front Entrance to Service Desk.

## **3 Conclusions and Recommendations for Future Work**

### **3.1 Lessons learned**

The main lesson learned is that lack of real-world testing can be negative towards project development. As the restriction of the pandemic was still in place, there was no option to physically access the space to get a feel of how the product work. Some features are incomplete from our initial testing such as the ability to remember users as they used the space. Other features such as audio navigation could not be tested for their accuracy. We were able to pick up on the use of a novel algorithm to achieve the way finding but that also means the algorithm is not specifically developed for the purpose of way finding in a library. To improve the algorithm would take immense time and experiences that we do not currently have but the current implementation is adequate enough for generalized purpose.

### **3.2 Future Work**

Our design is nowhere near perfection, there is still so many features that we had planned to add but did not get the chance and there are still known bugs. Some features we had planned to implement were the use of cookies to store the options of users to prevent asking the same questions every time. We would've liked to include a speech to text option for inputs from the users. Also, a visual map displaying the directions would've been a nice asset. Most importantly, the website needs a lot of work on the esthetics to make it more mobile compatible. Of course, our code can be optimized even further, and some small bugs need to be fixed.



## 4 Bibliography

1) Johnson, Adrienne (2018) Dijkstra's Algorithm (in JavaScript!) [Source code].

<https://medium.com/@adriennetjohnson/a-walkthrough-of-dijkstras-algorithm-in-javascript-e94b74192026>


2) Andy, E (2020) Getting Query String values in JavaScript [Source code].

<https://stackoverflow.com/questions/901115/how-can-i-get-query-string-values-in-javascript>

## APPENDICES

### APPENDIX I:

Table 1: Bill of Materials for the final prototype

Bill of Materials (BOM)					
Product	Quantity	Reason	Price	Picture	Website link
NFC chips	10	Set the NFC chips on the specific locations of the library, users can scan chip to access the uOttawa library wayfinding web-app.	\$ 9.99 without shippment		<a href="https://tinyurl.com/y2kd4ntt">https://tinyurl.com/y2kd4ntt</a>