Design Document: PathLess

Introduction

With each major personal project that I complete, I have been creating "thought process" documents to explain how they work. For this project, the document will be a little different. It will be like a functional/technical specification of PathLess, modified to be somewhat in the style of the thought process documents with my thoughts and explanations. Additionally, I will not be going into the same depth when it comes to the functionality of the algorithm.

Consider the user story that explains PathLess. I started to bike outside for a few miles often, but seeing the same things became predictable and took away part of the fun. That is where this application comes in – to create a route to follow given a certain distance and location randomly.

I looked through the internet to see if I could find something for this. The major map applications I had on my phone did not have this as a feature (at the time of writing this, at least). I found a few possible implementations online, but they simply were not good. Most of the time, they were wildly inaccurate. While I am unsure, my guess is that these took about 5 points, evenly spaced in all directions the given distance away from the user and then created a path through these points for the user. Of course, that means that the actual distance the user travels was often nowhere near the given distance.

This was the challenge – it was not sufficient to create random routes; routes had to be accurate to what the user wanted. Additionally, I wanted the app to be as user centric as possible. Instructions, an about page, relatively fast loading, high accuracy, and little to no technical glitches was the goal.

With this preamble out of the way, I hope the background and idea for the application makes sense. The goal behind creating products lies in solving solutions for the user, so this section is perhaps the most part of the entire project since it defines what it means for the app to be successful.

I would also highly recommend trying out the app if you have not already – go to pathless.vercel.app online, on your phone/tablet/desktop/device, and have fun trying different locations and distances to look at the possible routes.

I have split this document into a few sections: user stories, goals and stretch goals, screen specifications, wireframes, and the algorithm (briefly)[1]. Please note that these specs are not complete; it is simply meant to show the process that went into designing the application.

User Stories

As I mentioned earlier, the primary purpose of products is to address user problems. This is what defines what it means for the result to "work", so it is useful to play through a couple scenarios of when the product could be used to help shape the goals and specifications that follow. These are not meant to

[1] Full disclosure: After doing some research online, I came across a sample functional specification that heavily inspires the format/structure of this document. I encourage you to read it as well if these sorts of things interest you: https://www.joelonsoftware.com/whattimeisit/

be completely exhaustive or mutually exclusive of any or all user stories when using the product; instead, these are just possible scenarios that are slightly different.

Scenario 1: The Competitive Runner

The competitive runner is looking to train – perhaps for a 5K they have coming up, for school, or a track meet. They know the distances that they are going to run each day and the times to aim for. It is very precise and planned out well in advance, perhaps for different locations other than the current one. Having random routes is nice, but not a necessity. However, the routes must be precise.

Scenario 2: The Casual, Relaxed Biker

The casual, relaxed biker (or walker) is not necessarily looking to improve their mile time; rather, it is to go outside and do it perhaps as a part of a general health routine. In these cases, the routes are rarely, if ever, planned; speed of generating the route is far more important and will often use the phone's current location. The interface must be simple and easy to use, since these routes will be generated on the fly.

As can be seen from just two scenarios, there are a lot of different uses and reasons for each feature of the product. Between these two archetypes, most requirements of the app should be met; any further ones can be added in future iterations after getting the minimal components ready and prepared.

Goals and Stretch Goals

This section will feature a short list of goals and stretch goals, so that it is clear what the aims of the app are and are not, partly informed by the user stories. The stretch goals are great to have but are not as essential to the app or pose challenges in implementation currently justifying postponement.

Goals:

1. Create a randomly generated route as fast as possible while being precise
2. Have an intuitive and well-designed user interface that is mobile-friendly
3. Allow search for different locations
4. Request to get current location to use from the user
5. Have an about page to have further information about uses for the app

Stretch Goals:

1. Allow for pre-planned stops (adds complexity and increases both number of API requests and time needed)
2. Have different themes applied to the map
3. Export to other map applications to view the route externally from the app

Screen Specifications

For the sake of simplicity, there are only two screens in the entire application: the home screen which will feature most of the app, and an about screen that will have some additional information for interested users. Both pages will have a horizontal navigation bar at the top with links to both screens on the right and the name of the app on the left part of the bar which links to the home page.

Home Screen

Upon first load, the home screen will load centered on the University of California, Berkeley while requesting user location access. If this access is granted, the map recenters to the new location. Further, if any location is searched in the search bar, the map recenters on the searched location. The control dialog above the map will have an input to change the distance (which will have a default of 3).

Upon clicking load new path, that triggers the algorithm to generate the route which will then be shown on the map. An alert is shown informing the user the process is underway, and after completion, the trip duration, trip distance, and instructions are displayed.

About Screen

This is a static page with explanatory information about PathLess (headers are in the wireframe).

Wireframes

Below are both the phone and desktop wireframes[2]. They were both created in Google Drawing, which I found to be quite convenient for rapid prototyping.

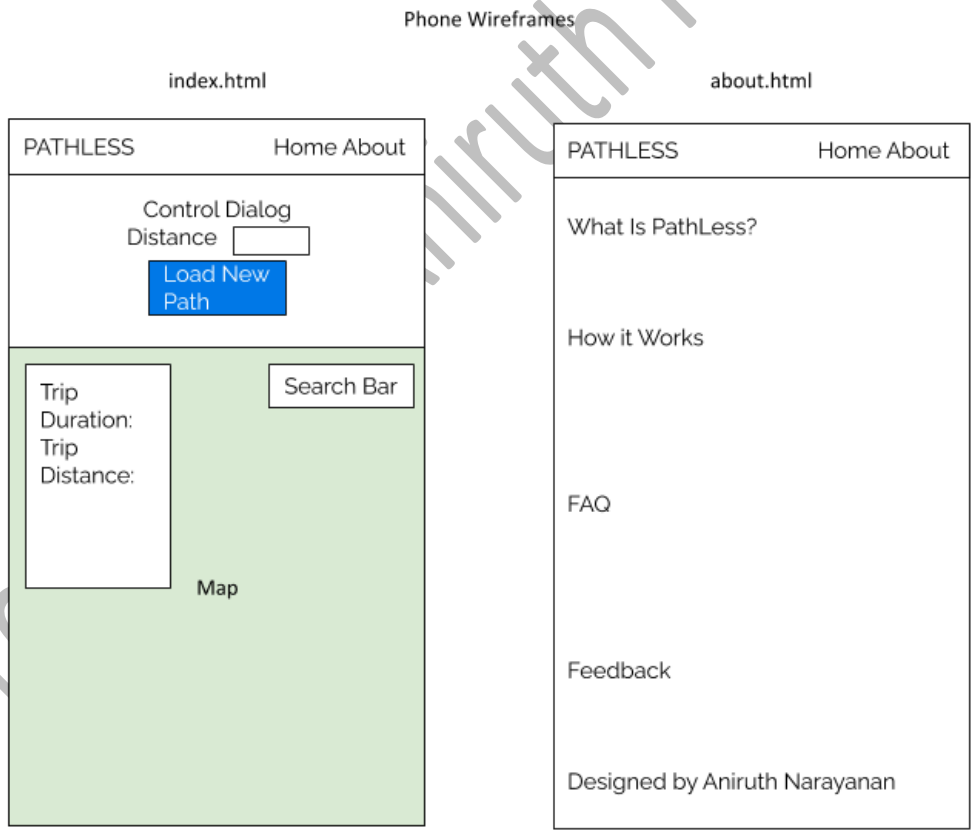


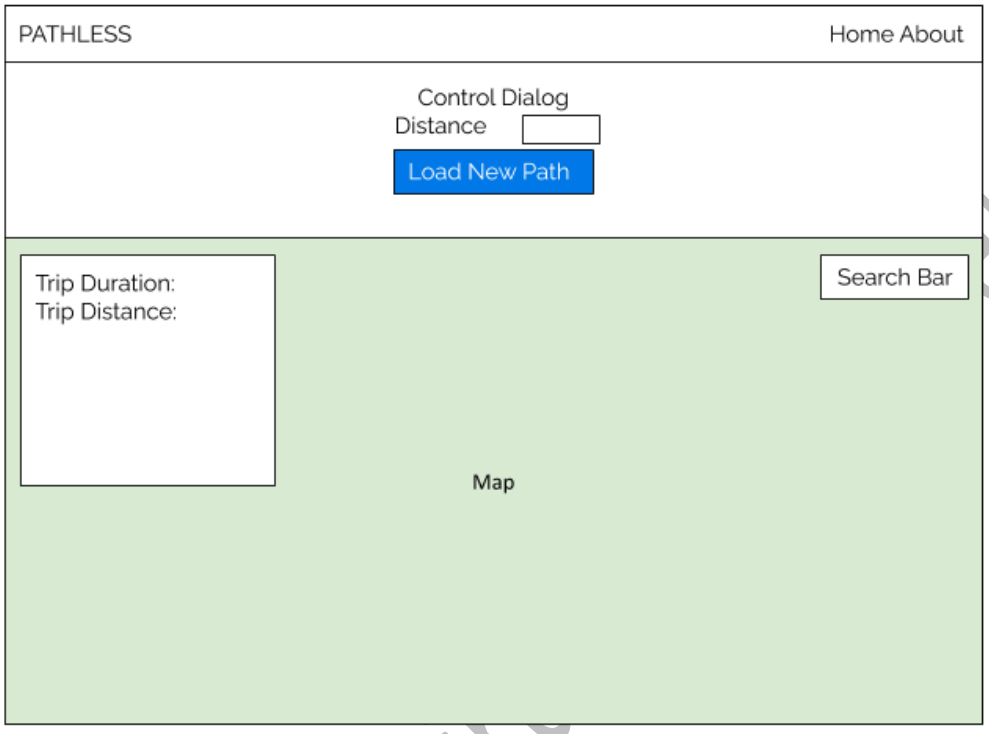
Figure 5-1

Both screens are shown above on the mobile layout, with the map changing as needed.

[2] The index.html wireframe refers to the home screen.

Desktop Wireframes

index.html



about.html

Figure 5-2

The map screen is noticeably wider in the desktop view of the screens.

These wireframes are for the most part what was used in the final version of the application; however, a small modification with brief instructions was added to the top of the control dialog in the mobile view and to the left of the control dialog in the desktop view to assist first-time users. Of course, actual text content is filled in on the actual website that is not shown on the wireframes.

The Algorithm

As mentioned earlier, the algorithm is not the focus of this document. However, one of the key goals was ensuring the algorithm was fast and more precise than the few existing solutions. So, I will briefly outline the procedure of the algorithm, although there are not any graphics so it will be a bit dense.

The initial step is to set a few variables – create a default starting location and distance. Then, the map is added using this initial location to the webpage. The search bar is linked to the start coordinates, so if it changes, so does that location. The user sees a request to access their current location upon first loading the page, to have the map recenter around their location and change the start.

When the button for loading a new path is selected, the algorithm proper begins. It takes a bit of time because it uses synchronous requests[3], so an alert pops up to the user to buy a little bit of time since the entire webpage freezes during this. Broadly speaking, the algorithm has two components. First, it generates points, ensuring they are separated from each other but still close to the start, and second, it checks distance of the resulting route. Points are added to increase the overall distance, and if the distance gets too high, the process begins again.

Then, once the distance is close enough to the initial input, then the route and arrows are plotted on the map, the trip instructions are loaded, and then the instructions overlay has the step-by-step information with distance and duration.

While not related to the algorithm, something worth mentioning is how the elements change when used on both desktop and mobile. Some parts of the map's formatting is out of my control – so size and position of elements are recalculated often to ensure mobile compatibility across all devices after testing.

[3] These are not recommended and are deprecated; however, the nature of the algorithm that cannot proceed until the data is fetched (i.e., it actively uses the results of the request) so this is forced. It is also why the webpage freezes sometimes.