Final Report: Wildcat Ways

Github: https://github.com/MrOliverpt/WildcatWays

Group 5: Taylor Hopkin, Davin Lim, Oliver Tipton, Kiko Lancastre

April 30th 2024

1. Introduction

- What is the title of your project?
    - Wildcat Ways
- What is the goal of the project?
    - The goal of the project is to create a web-based application that finds the fastest, most optimal route to get from place to place on Davidson College's campus
- What is the motivation for this project?
    - The motivation for this project originated from us arguing over which path is the fastest way to get back to our dorms after class in Watson.
    - We also got motivation from competitor apps, such as Google Maps and Apple Maps, because they do not support functionalities that display optimal and precise walking paths around Davidson College's campus.
- Who are the customers/users?
    - Davidson College students, faculty, and staff
    - Prospective students or athletes on official visits
    - Davidson student tour guides that work for the Davidson's admissions office
    - Residents of the town of Davidson

- What development process/method did you use for your project (e.g., Agile-Scrum, Waterfall, etc.)?
    - We initially used Agile-Scrum for our development process, but at the end of our project deadline, we accidentally switched over to a plan-driven development process due to time constraints and other unforeseen factors.

2. Novelty

Wildcat Ways is designed to be an interactive, dynamic route optimization software for members of the Davidson Community. The project is novel in a number of ways:

1. Scope/Specific Customer Base: Large mapping services such as Google Maps, Apple Maps, and Waze all are designed to provide their service on a global scale. In doing so, they sacrifice accuracy in smaller, more niche paths and roads. Davidson college is a small college filled with small paths that cater to its scenic campus. The majority of these paths are not recognized on these global mapping services, and those that are, are often slightly misplaced, not considered as viable walking paths, or their distance and route calculation weight is off. This means that for customers trying to utilize these services, their ability to find the most optimal route to and from locations specifically on Davidson campus is nonexistent. WildcatWays considers every small path on Davidson's campus and in doing so, truly gives the user the best estimate for the most optimal route.
2. Unique implementation of Dijkstra's Algorithm: Dijkstra's algorithm is primarily used in weighted networking graphs. It is designed to find the minimum weighted connection between nodes. However, it is rarely used in mapping contexts due to lack of specific and accurate information between intersection points. Because Davidson's campus is relatively small, we were able to gather a tremendous amount of information (nearly 50,000 points of information) to build out a weighted graph of Davidson's campus where every node represents an intersection and every edge (weighted according to walking time) represents connections between intersections. This implementation of Dijkstra's algorithm provides our mapping service with a unique ability to provide extremely

accurate route optimization paths in ways large scale mapping services cannot provide due to a lack of information.

Competitors: Wildcat Ways has a number of competitors. The three largest competitors are Google Maps, Apple Maps, and Waze. As mentioned before, these competitors operate on a global scale and have a customer base which primarily use their services for driving. As such, they fail to properly account for walking paths, and generally calculate inaccurate walking times. Because the amount of data that exists on small scale walking paths is extremely limited, these services, while competitors in a conceptual sense, are not going to be formidable competitors in a practicality/scope sense.

3. Customer

3.1 Primary, Secondary, and Any Other Stakeholders

- Our primary customers outside the team are Davidson College students, faculty, and staff.
- Our main stakeholders are three Davidson students: Maggie Woodward, Matthew Bernard and Cate Goodin (two of which are student tour guides who work for the Admissions office)
- **Maggie Woodward**: Her first suggestion was that we add a drop down menu with all the different building types for convenience. She also wanted us to include important locations within buildings, such as Union, for example. She argued that Union contained a lot of different important places such as the career center, SGA, Student activities, Union board, etc… This same logic could be applied to every other big building on campus. Maggie also asked that we include accessible paths. Maggie is a tour guide here at Davidson and sometimes she has to give accessible tours. She said that she already knows the tours by heart but when she first started, she struggled to know which paths to take and which paths not to take. By adding this accessible path feature, we would make it more inclusive and make it so anyone can use it. We thought this idea was phenomenal and agreed to set it as one of our top priorities.

- **Matthew Bernard**: At first, we were thinking of adding a small game component to our software, something like a scoreboard keeping track of who has walked the most. Matthew was adamant that he felt as though this feature was a little out of place and generally unnecessary. He argued that if he was going to use our software, it would be to get from one place to another, not to try and walk more miles than other individuals. We appreciated his feedback and removed that idea from our plan. Matthew also really wanted us to include parking lots in our software so he could check which one is closer to him. This is something we had been considering but hearing his approval certified the need.
- **Cate Goodin**: Cate wanted us to include all the different entrances to all the buildings. She explained that some entrances would take longer to get to than others and our software should decide which entrance to go to. She also wanted us to include other buildings outside of campus. She mentioned buildings like PickledPeach, Harris Teeter and CVS. Although her idea was good, given our data collection capabilities and overall scope of our project, we had to turn down this idea. She also wanted our software to include parking lots like Matthew for the same reasons. Her last suggestion for our project was for us to include CatCard Accessible buildings along with the times in which they are only accessible to people with CatCards (buildings such as library, Wall, Watts, etc…). At first we thought this idea was good but given the scope of our project, we had to remove it.

3.2 Problems and Requirements

3.2.1 Meeting Log

Date	Description of the Customer	Description
March 18th	Maggie Woodward (Tour Guide)	First time meeting with Maggie. Introduced her to our project and asked what she

		would like to see. She told us she wanted a drop down menu with all the buildings divided into categories. She wanted us to include rooms within buildings on campus like Union and the ability to have accessible paths only.
March 19th	Cate Goodin (Tour Guide)	First time meeting with Cate. Introduced her to our project and asked what she would like to see. She told us she wanted us to include parking lot locations, buildings outside of campus and CatCard accessible buildings.
March 21st	Matthew Bernard (Student)	First time meeting with Matthew. Introduced him to our project and asked what he would like to see. He told us he thought our leaderboard idea for “who has walked the most” was not necessary and told us we should scrap it. He also told us to include parking lots.
April 10th	Maggie Woodward (Tour Guide)	Second time meeting with Maggie. Updated her on the process of our project. Asked

		her if she had any more ideas for the project to which she said no. She maintained her previous ideas.
April 10th	Cate Goodin (Tour Guide)	Second time meeting with Cate. We told her we weren't going to include buildings outside of campus. At first she thought that was a shame but after explaining to her that our project is focused solely on Davidson, she understood. She also gave us the idea of adding all the cross country trails into our software. We had to deny that idea due to the scope of our project.
April 12th	Matthew Bernard (Student)	Second time meeting with Matthew. Updated him on our progress and asked him if he had any more ideas. He said he didn't.
April 23rd	Maggie Woodward (Tour Guide)	Third time meeting with Maggie. Let her know about our struggles finishing up our software. We didn't go into too much detail given that it's problems with Front-end and

		Back-end integration but she understood that our project would likely not be done this semester.

3.2.2 Product Backlog

Sprint 1

Route Optimization:

- Provide a written description of optimal paths to user

Sprint 2

Functional UI

- Search Bar
- Drop Down Menu
- Working Map that is displayed

Sprint 3

Functional UI

- Enhance User experience on UI
- Map to display the optimal route that was outputted in Sprint 1

Route Optimization Features

- Speed Estimates
- Accessibility Features
- Scenic Routes

Future Sprints

App Service

- For all mobile devices

Web Service

- Provide a downloadable feature that allows the user to take outputs on the go

Route Optimization

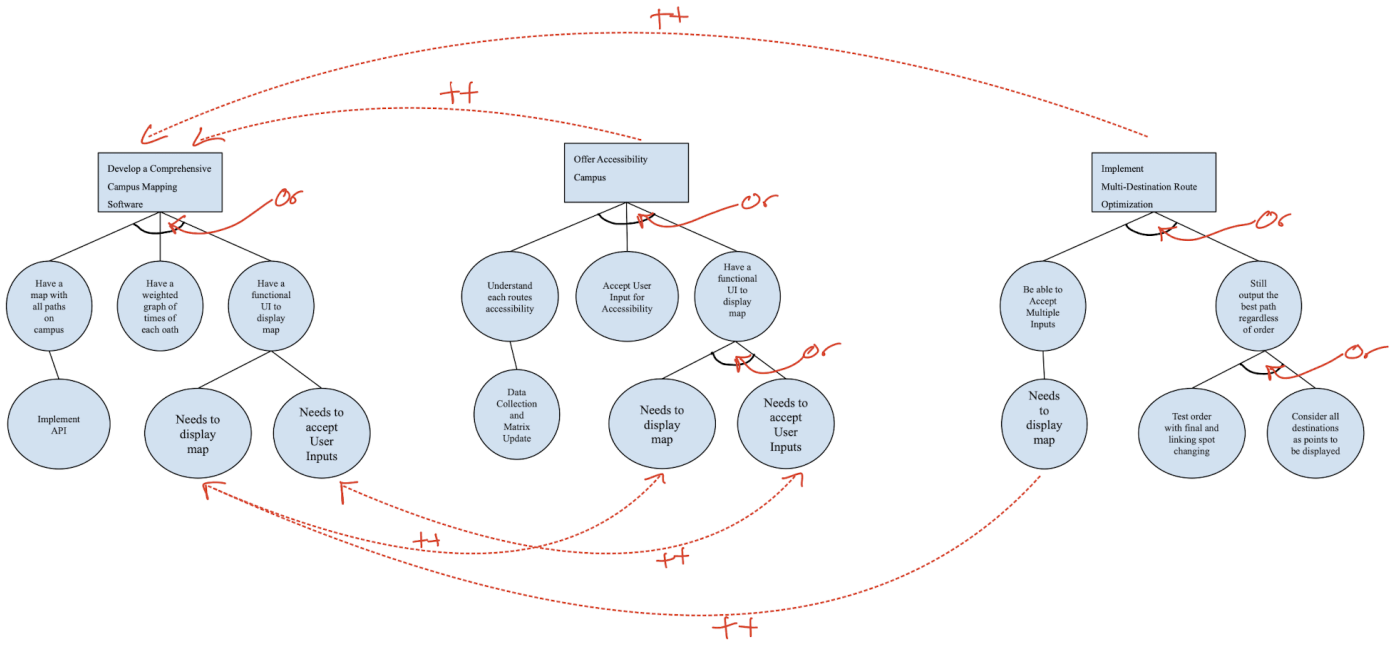
- Expand the scope of the map to include Davidson Main Street
- Expand the scope of the optimization features to include inside of building data
- Increase the Scenic Route Options
- Add a Tour Guide Feature

3.2.3 Overall Customer Experience

Our customers desired an app experience similar to mainstream GPS applications such as Google Maps. However, our app distinguishes itself by tapping into paths not typically accessible via standard Map APIs. Consequently, it offers a broader range of possibilities and more optimized routes from point A to point B. Users also have the flexibility to include multiple destinations if they need to make additional stops along the way. Upon opening our app, users are greeted with a map centered on Davidson Campus. Adjacent to the map, a dropdown menu neatly categorizes all on-campus buildings into various categories such as food, academic, social, dormitories, and more. Additionally, a search bar is conveniently provided for users who prefer not to navigate through the dropdown menu. When the user puts down their initial location and their destination, the most optimal path is shown in the Map API alongside the time it'll take to complete the trip.

Currently, our customers are pleased with the direction of our product. They appreciate the visually appealing website design and find the dropdown menu featuring all the buildings, along with the option to add different paths, to be beneficial. However, they expressed some dissatisfaction with the optimal path being displayed in written form rather than directly on the map. Nonetheless, they understand that this functionality will be addressed in the upcoming sprint.

4. SMART Goals



- Goal 1: Develop a Comprehensive Campus Mapping Software
    - S: Create an interactive map of Davidson College that includes all paths, including small, niche walking paths not recognized by major mapping services.
    - M: Compare time and route predictions to that of global map providers to ensure we have created an improved service.
    - A: Utilize student volunteers to gather specific walking times for each path and intersection on campus.
    - R: We will focus on the unique campus environment to offer a specialized service that global map providers cannot.
    - T: Set weekly milestones for path data collection and integration into the map.
- Goal 2: Offer Accessibility Routes
    - S: Develop and integrate features within WildCat Ways that take into account the accessibility needs of users, including pathways suitable for wheelchairs and other mobility aids.

    - M: Ensure that every path considered is denoted as either accessible or inaccessible.
    - A: Utilize student volunteers that use wheelchairs and other mobility aids to confirm that each path is being correctly labeled.
    - R: Providing accessible route options is crucial for inclusivity, ensuring that all members of the Davidson community can navigate the campus effectively.
    - T: Set weekly milestones for path data collection and integration into the map and have accessibility path calculation done in the iteration following initial map deployment.
- Goal 3: Implement Multi-Destination Route Optimization
    - S: Enable WildCat Ways to support route optimization for multiple destinations, allowing users to plan efficient paths through multiple points on campus in a single trip.
    - M: Compare time and route predictions to that of global map providers to ensure we have created an improved service.
    - A: Enhance the algorithm to consider multiple nodes and stops in the optimization process, using additional computational resources and advanced route planning techniques.
    - R: Multi-destination routing is a valuable feature for users with complex schedules or those looking to maximize efficiency in their campus navigation.
    - T: Set weekly milestones for algorithm and map integration and have accessibility path calculation done in the iteration following initial map deployment.

5. Sprint Backlog

Sprint 1:

- Define project scope, objectives, and requirements.
- Conduct initial research and analysis.
- Develop wireframes and design mockups.

Sprint 2:

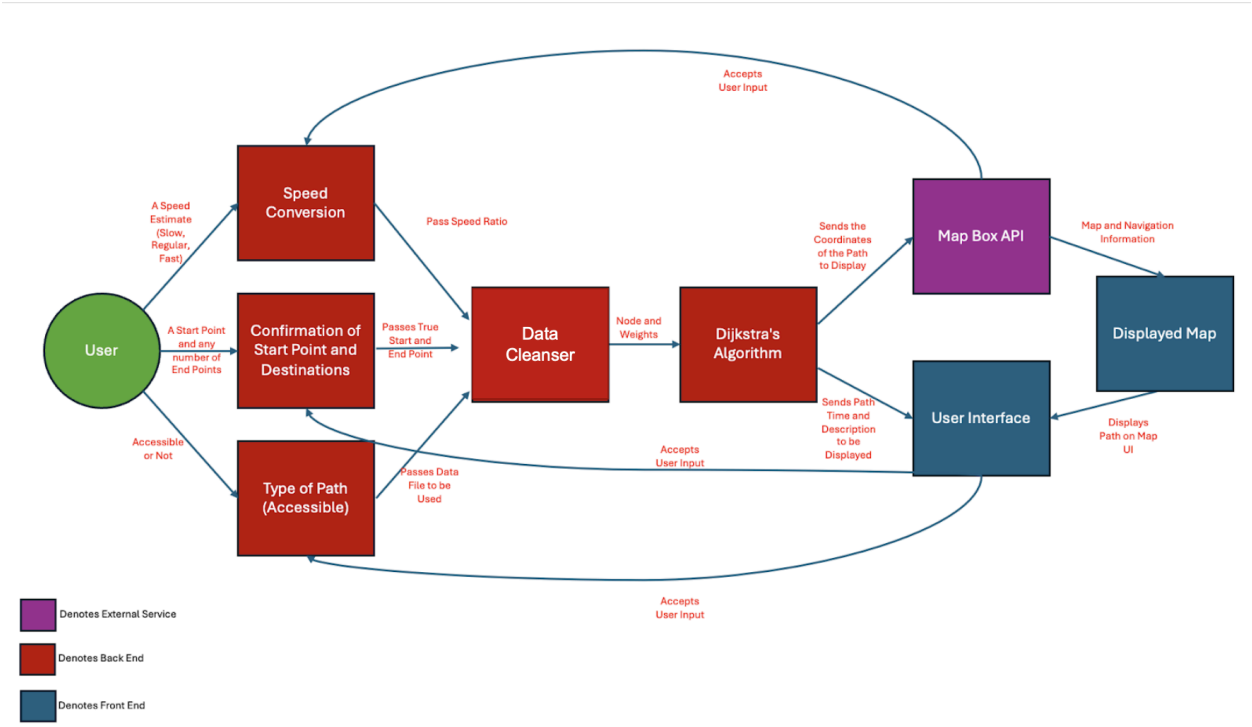
- Data collection
- Implement Djisktra’s fastest path finding algorithm
- Fully integrate data with Dijkstra's algorithm.
- Create wireframe for the UI
- Meet with stakeholders

Sprint 3:

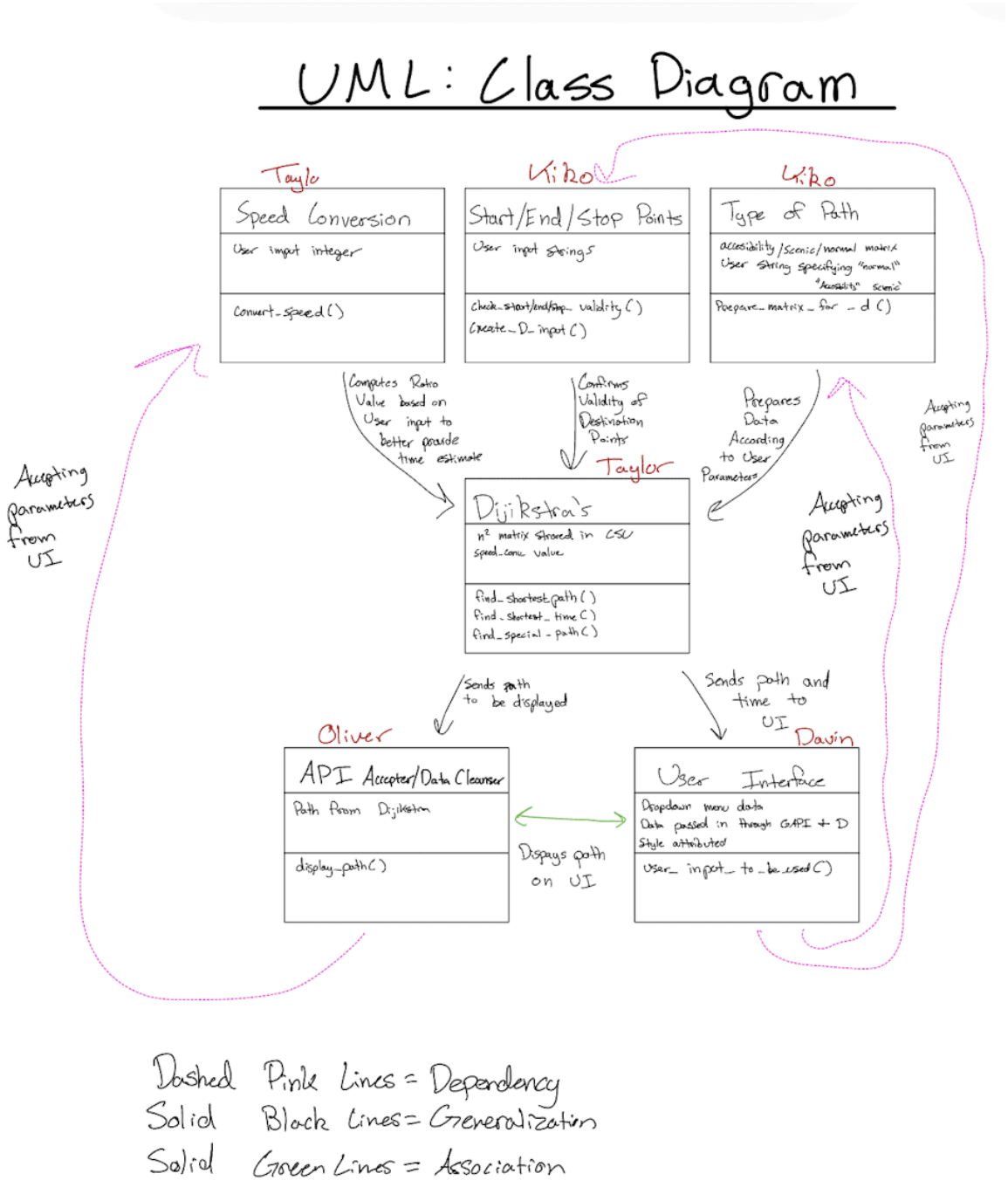
- Meet with stakeholders
- Integrate Dijkstra algorithm with small data set from our total data set
- Integrate Maps API with UI
- Python server using Flask
- Integrate Python server with front-end to get fully functional demo

6. System Description

6.1 Block Diagram



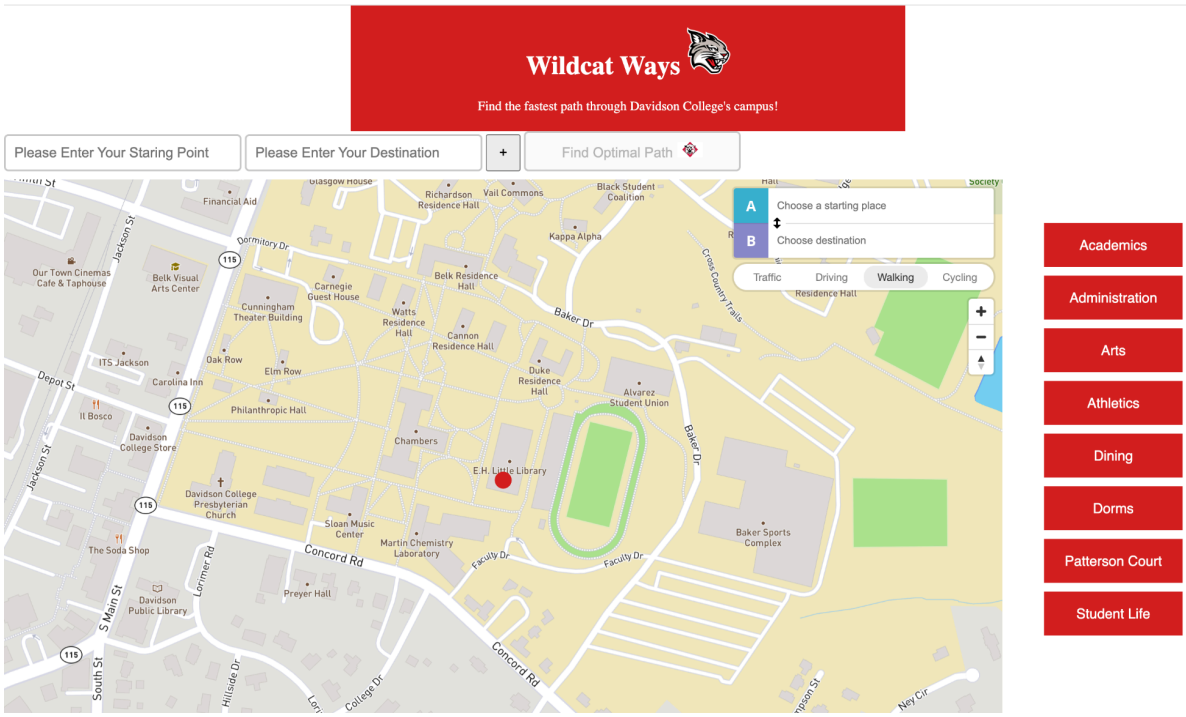
6.2 UML: Class Diagram



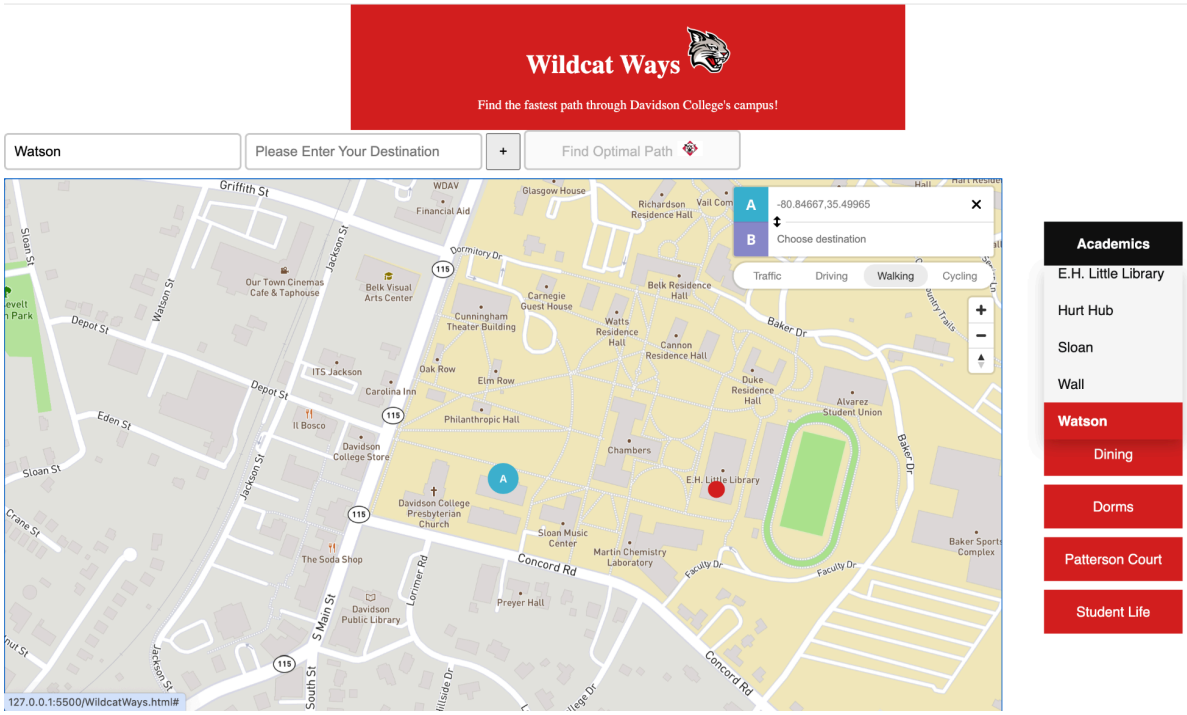
Dashed Pink Lines = Dependency
Solid Black Lines = Generalization
Solid Green Lines = Association

7. Current Status

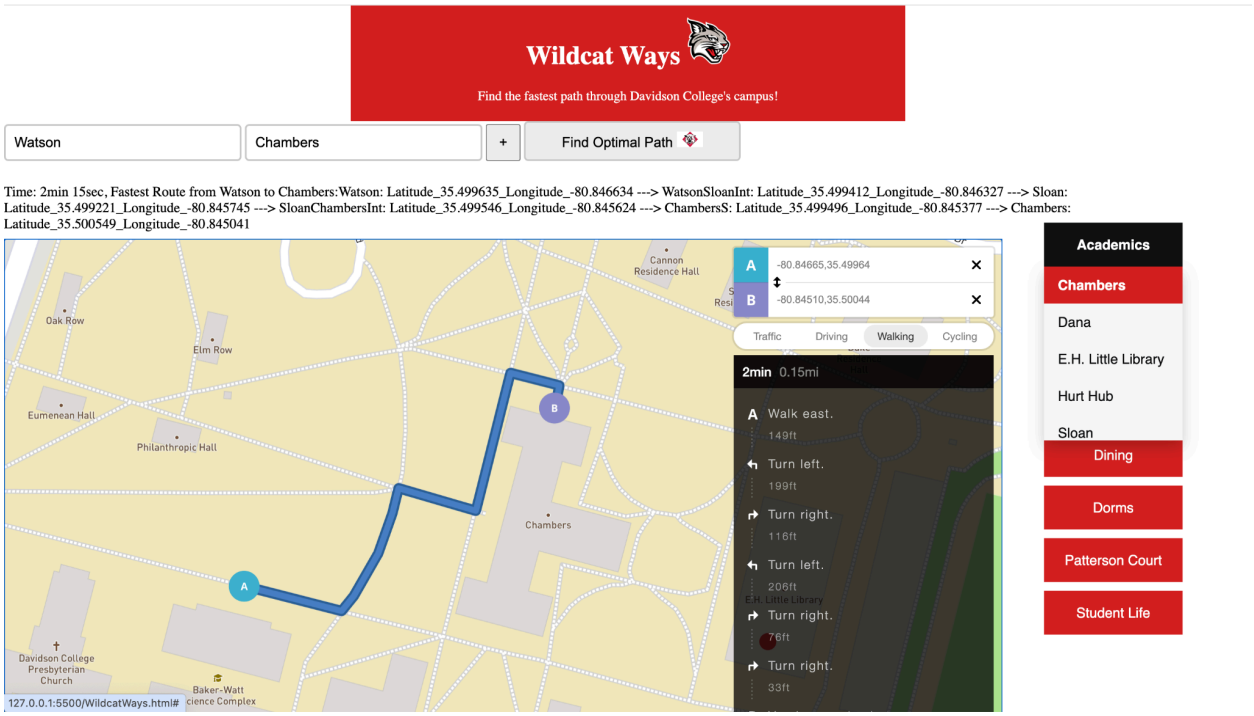
7.1 Screenshots



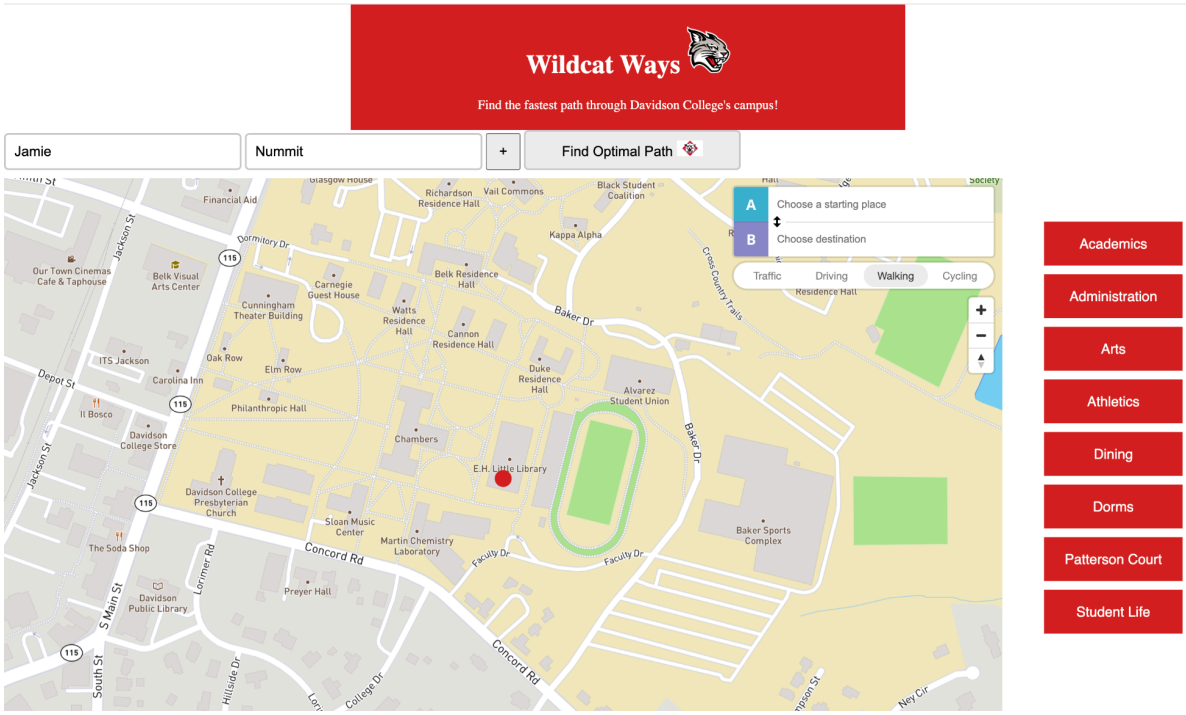
This is our main UI when you first load the website. This part belongs to the UI in the system description. In this current screenshot of the UI, the user has not inputted anything and thus, the system has not output anything. The user can enter their starting point and their destination.The basic UI of the system is very important because without it, the user won't be able to find the fastest route from point A to point B.



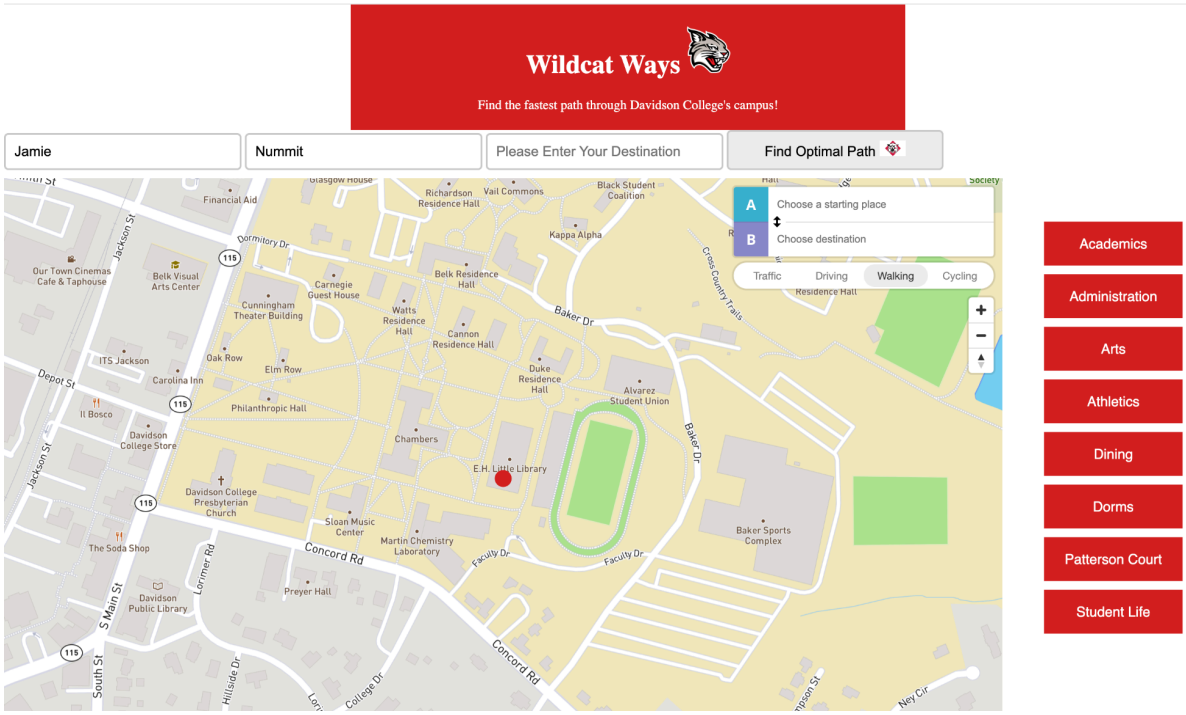
In this screenshot, the user has inputted Watson as their starting point by clicking on the side-bar menu and selecting Watson from the Academics category. This part of the screenshots belongs to the UI, Mapbox API, Displayed Map, and “Confirmation of Start Point and Destinations” in the system description. The input of the system is “Watson” for the starting point. The user also selects Watson as their starting point on the map, which is another input. Currently, in this screenshot, there are no outputs by the system. This part of the system important is because it demonstrates how the user can select an input



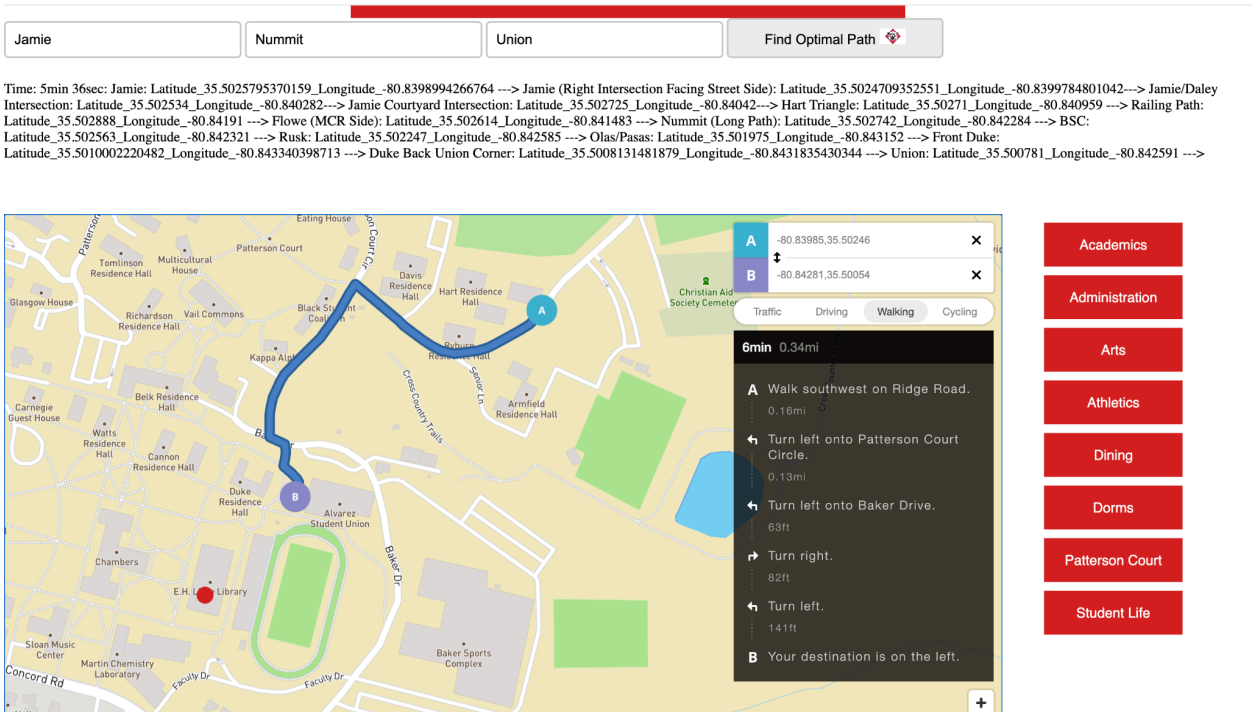
In this screenshot, the user has inputted “Chambers” as a destination point and has clicked the Find Optimal Path button. The user selected Chambers from the side-bar menu in the Academics category to select their destination. After the user hits the Find Optimal Path button, the system displays the time it takes to walk the shortest route and the nodes (with the GeoJSON coordinates) that the user has to traverse to get to the destination. This part belongs to the UI, Dijkstra’s Algorithm, Mapbox API, Displayed Map, and Confirmation of Start Point and Destinations. The input of the system is the starting point and destination as well as the map inputs in which the user selected the starting point and destination on the map. The output of the system is the shortest route from the starting point and the destination and the time it takes to walk that route. Additionally, the map UI outputs the shortest route from point A to point B. This part of the system is important because it shows how given a user’s input for the starting point and destination, the back-end server retrieves the inputs and calculates the shortest route and time using Dijkstra’s. Finally, the back-end server pushes the output of the shortest route and time to the front-end UI to be displayed.



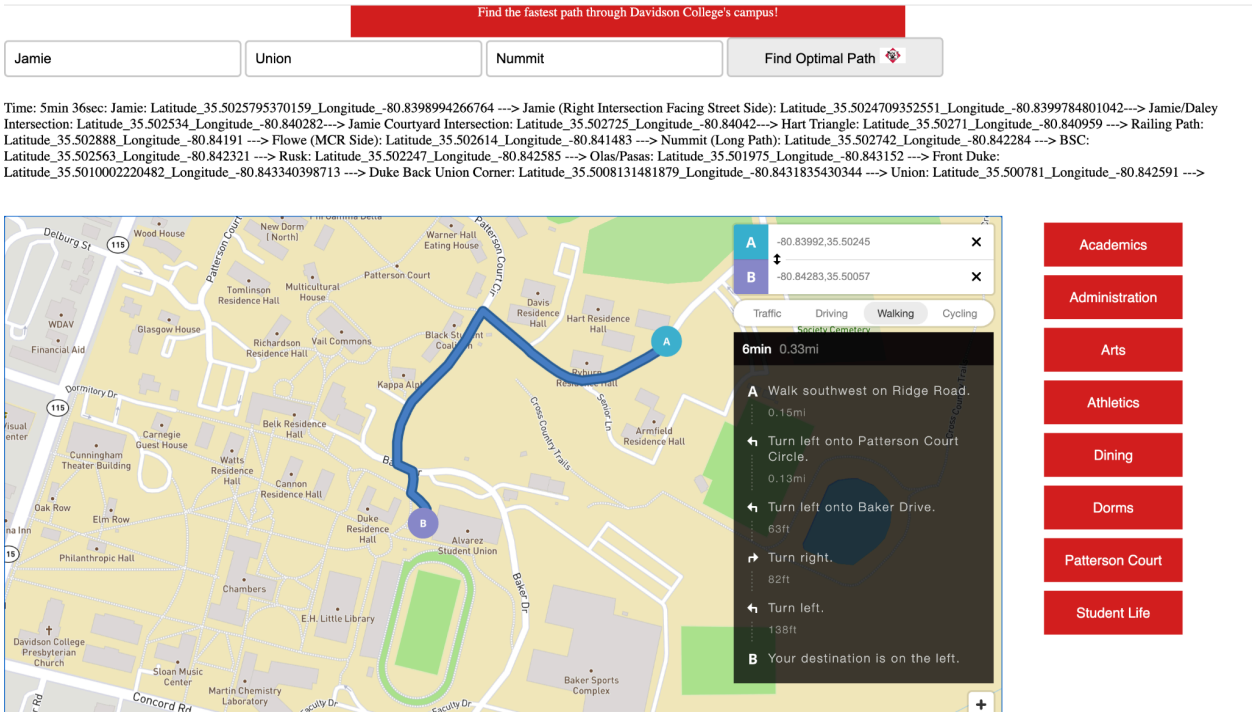
In this screenshot, the user has inputted Jamie and Nummit as the starting point and destination. We want to show how (in the next screenshot) when the user clicks on the plus “+” button, the user can add a stopping point before the destination. This part of the screenshots belongs to the UI and “Confirmation of Start Point and Destinations” in the system description. The inputs of the system are the starting point and the destination by the user. The system has not outputted anything in the UI. This part of the system is important because it shows how the user will be able to add an extra starting point to go from point A to point B to point C.



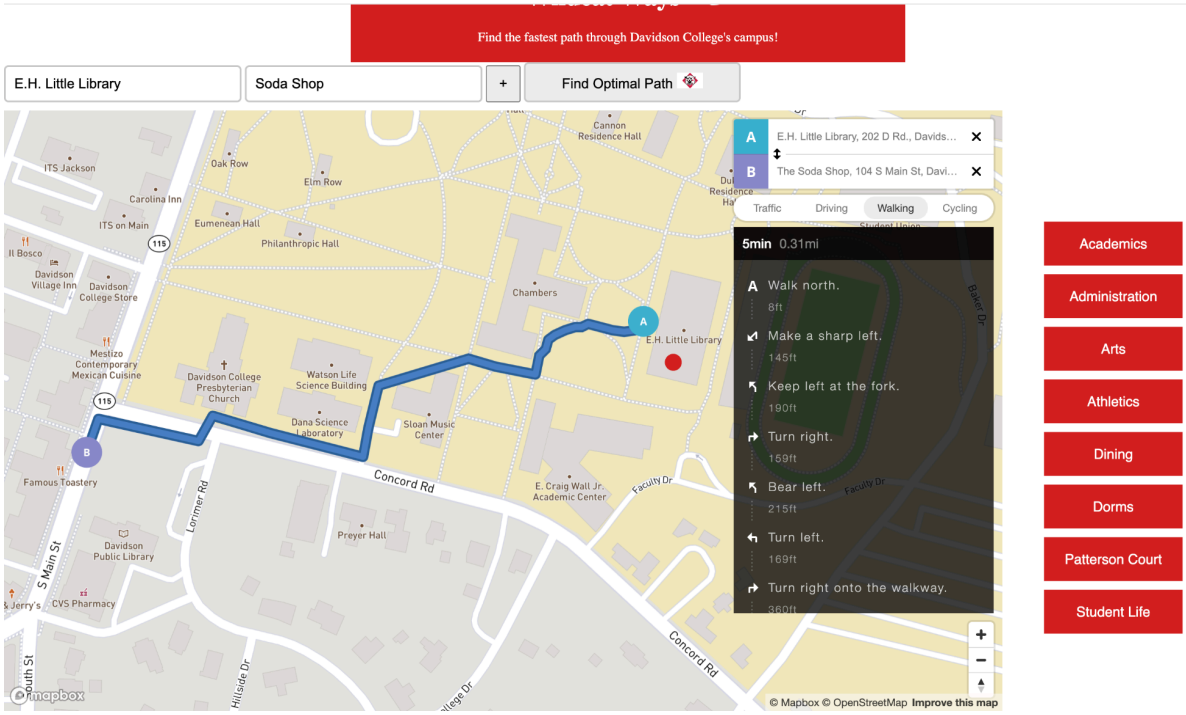
This screenshot shows how the user has pressed the plus "+" button to add an extra stopping point, which is Nummit. Now the user can enter their destination so they can find the fastest path from Jamie to Nummit to their selected destination. This part of the screenshot belongs to the UI and "Confirmation of Start Point and Destinations". The inputs are the starting point and stopping point. The system has not outputted anything because the user hasn't put in a destination yet. This part of the system is important because it shows how the plus "+" button is working by adding another user input div so the user can select a stopping point besides the start and destination.



This screenshot shows the shortest route from Jamie to Nummit to Union and the time it takes to get from the starting point through the stopping point and then the destination. This part belongs to the UI, Dijkstra's Algorithm, Mapbox API, Displayed Map, and Confirmation of Start Point and Destinations in the system description. The inputs of the system are Jamie as a starting point, Nummit as a stopping point, and then Union as the final destination. The input on the map is Jamie to Union (but passing through Nummit). The system outputs the shortest path from Jamie to Nummit to Union and the time it takes to walk that path. The map UI outputs the fastest route from Jamie to Nummit to Union and the directions. This part of the system is important because it shows how the system can handle more than two inputs and display the shortest route given a stopping point between the start point and destination.



Similar to the previous screenshot, this screenshot shows the shortest route from Jamie to Union to Nummit and the time it takes to get from the starting point through the stopping point and then the destination. This part belongs to the UI, Dijkstra's Algorithm, Mapbox API, Displayed Map, and Confirmation of Start Point and Destinations in the system description. The inputs of the system are Jamie as a starting point, Union as a stopping point, and then Nummit as the final destination. The input on the map is Jamie, Nummit, and Union. Since Dijkstra's algorithm finds the shortest path to traverse all three points, the system optimizes the route to start at Jamie and go to Nummit first instead of Union. Thus, the system outputs the shortest path from Jamie to Nummit to Union and the time it takes to walk that path. The map UI outputs the fastest route from Jamie to Nummit to Union and the directions. This part of the system is important because it shows how the system can handle more than two inputs and display the shortest route given a stopping point between the start point and destination. Also, it demonstrates how Dijkstra's algorithm optimizes the path between three points.



In this screenshot, we are trying to show how the Mapbox API can find the fastest route from E.H. little library to the Soda Shop, an off-campus restaurant on Main Street. This part belongs to the Mapbox API and Displayed Map in the system description. The inputs are E.H. Little Library as a starting point and the Soda Shop as the destination. The Mapbox API outputs the fastest route from the library to the soda shop on the Map UI with the directions and time it takes to get there. This part of the system is important because it shows that the Mapbox API has the capabilities of finding the shortest route to off-campus locations on Main Street, which are often visited by Davidson college students and faculty.

7.2 Software Testing

7.2.1 Test Cases

1. Case 1: Updating available paths. When a new path is made on campus, it will become an available option for the optimal path algorithm. Also, when a path is not available for some reason, it will be removed from the optimal path algorithm.
    a. Justification: This feature is crucial because it ensures that the software always provides users with the most accurate and up-to-date navigation routes. Testing this feature ensures that the software accurately incorporates new paths and removes unavailable ones, preventing users from being directed through inaccessible or non-existent routes.
    b. Current Status: We were able to place new paths on our map image, right now this feature is only available to “admins” or people who have access to the path updating, though, this makes sense, we don’t want people to add random paths.

2. Case 2: Time to walk 10 meters. When users input the time it takes for them to walk 10 meters, the time in the displayed optimal path will be adjusted accordingly.
    a. Justification: Walking speed can vary significantly among users, and it's essential for our software to provide personalized optimal path times based on the user's walking speed. By allowing users to input their walking speed, the software can calculate more accurate estimated travel times for each segment of the route. Testing this feature ensures that our software correctly adjusts the time it takes for the optimal path based on the user's specified walking speed.
    b. Current Status: Right now we do have the ability to account for user speed, this allows us to provide the most accurate estimate of journey speed.

3. Case 3: Optimal path, 2 inputs (starting location, destination). When given a starting location and a destination, the algorithm will show the optimal path to take.
    a. Justification: This feature forms the core functionality of our software, providing users with the most efficient route between a starting location and a single

destination. It's crucial to test this separately to ensure the algorithm's accuracy in determining the optimal path, as it directly impacts the usability and reliability of the software. Testing this feature ensures that users can rely on our software to provide accurate and efficient navigation for their intended destinations.

b. Current Status: Right now we have this functioning both on front end and back end.

4. Case 4: Optimal path, 3 or more inputs. When given a starting location and various different destinations, the algorithm will show the least time consuming path between each different destination
    a. Justification: This feature extends the functionality of number 3 by allowing users to plan multi-stop journeys within the campus. Users may have multiple destinations to visit consecutively, such as attending classes in different buildings or running errands across various locations. Testing this feature separately ensures that the software can effectively optimize routes for multi-stop journeys, minimizing overall travel time and providing users with practical navigation solutions.
    b. Current Status: Current Status: Right now we have this functioning both on front end and back end.

5. Case 5: Accessibility flag. When this flag is turned on, the algorithm will only pick paths that are accessible to every user.
    a. Justification: This feature is essential for ensuring inclusivity and accessibility within the campus environment. By allowing users to toggle an accessibility flag, our software will only use paths that accommodate individuals with disabilities or mobility challenges. This feature is crucial for users who rely on wheelchair accessibility, ramps, elevators, or other accommodations to navigate the campus safely and efficiently.
    b. Current Status: Right now this feature exists, the user can specify accessibility paths on the front end and back end though the paths do not line up correctly. This would need further testing to link the two and ensure its accuracy.

7.2.2 Combinatorial Testing

Factor 1: Path Availability

- Values:
    1. Path Available
    2. Path Unavailable

Factor 2: User Walking Speed

- Values:
    1. Slow (e.g., 20 meters per minute)
    2. Average (e.g., 60 meters per minute)
    3. Fast (e.g., 100 meters per minute)

Factor 3: Number of Inputs (Destinations)

- Values:
    1. Single Destination
    2. Multiple Destinations

Factor 4: User Role

- Values:
    1. Admin
    2. General User

Factor 5: Accessibility Requirement

- Values:
    1. Accessibility Required
    2. No Accessibility Required

Factor 6: System Interface

- Values:

1. Front End
2. Back End

Factor 7: Current Path Status

- Values:
    1. Correctly Aligned Paths
    2. Misaligned Paths

Each of these factors can influence the system's functionality and its response to different scenarios, making them critical to combinatorial testing. The values for each factor allow for testing various combinations of system states and user inputs to ensure comprehensive coverage and robustness in functionality.

8. Project Management

Integrated data into the matrix for Djakostra's algorithm.	March 30th
Added feature that allows users to include more than 1 destination in the algorithm	March 30th
Created basic UI for Wildcat Ways website	April 1st
Added drop down menu for different categories of locations	April 1st
Integrated the Google Maps API into the website	April 1st
Began working on customized paths/ walkways/buildings to the Google Maps API for Davidson College's campus	April 1st

Switched from Google Maps API to Mapbox API	April 7th
Integrated Coordinate Data into the matrix for Djakostra's algorithm.	April 11th
Introduced Speed Factor	April 15th
Introduced Accessibility Factor	April 18th
Added UI connection to back end enabling the user to select inputs to be implemented in backend software	April 15th
Added verbal description in addition to map	April 21st

9. Review and Retrospective

9.1 Sprint Review

- Are there any additional customer needs?
    - Cate Goodin (a student tour guide) requested that we include other buildings, locations, and shops outside of campus. She mentioned places like the PickledPeach, Harris Teeter, CVS, and other Main Street buildings. Although her suggestion was insightful and practical, given the overall scope of the project, we did not collect data on these places. However, Mapbox's Navigation, Search, and Directions API already had pre-existing data on the vast majority of the main street buildings, including Harris Teeter. Thus, we were able to successfully implement the addition of mainstreet buildings and other relevant off-campus locations into our search and navigation feature, despite not having collected data on those specific locations.
- What are some of the customer(s) requests that you could not accomplish?
    - Cate Goodin requested that we include all the different entrances/exits to each building on campus because some entrances/exits take longer to get to than others. We did accomplish this task to a certain extent as we identified different nodes for the four entrances for Chambers Hall (North, South, East, and West). However, we quickly realized that repeating the same process for every single building on campus would be extremely time-consuming and laborious since there are many unknown or less used entrances and exits for various buildings. Therefore, we decided not to accomplish this customer request.
    - Matthew Bernard, one of our other customers, requested that we include parking lots and spots in our project so students and faculty can easily check where the closest place to park is relative to their current location. We thought this was a great suggestion that holds a lot of potential for practical use, but given our limited resources in time, we decided to not implement this feature as there are many different parking lots and spots scattered throughout campus.

    - Finally, Maggie Woodward (one of the other tour guides and our customer) requested that we add minor details on each building such as the Student Union, which houses the Career Center, Student Government Association, Post Office, Student Activities Office, Walt 1610, and Union board. We thought this was also a great idea that could be implemented since a lot of students do not know where each office or student organization is located. However, we could not accomplish implementing this feature at the end due to limited time and resources.
- Any comments and feedback from the customer(s)?
    - All three of our customers were very satisfied with our final product as it outputs the most optimal path given a set of inputs (starting, stop, and destination points). Moreover, they were impressed by the Map UI that displays the shortest path on Davidson College's campus given a user's selection of the start and finish points. They commented that it would be even better and more sophisticated if we can add a feature in which the user can select their walking speed to adjust the time it takes for them to walk a given path, so they won't underestimate the time it takes to get somewhere. They also said having a bicycle path feature for the map is super helpful as there is a good amount of students who either use a scooter or bicycle to get around campus.

9.2 Sprint Retrospective

- What went well?
    - We successfully implemented our front-end UI and back-end server. Additionally, we completed the data collection process and successfully implemented our Dijkstra's algorithm.
- What didn't go well?
    - We failed to complete the integration of our back-end with our map feature in the front-end. We also failed to customize Mapbox's Directions and Navigations API so that we can add paths and locations that do not exist on Mapbox's map GeoJSON database. These were goals of this sprint though they were not completed in their entirety.

    - Similarly, we changed from an iterative process to a plan driven process mid-sprint. This was a complete oversight from our group. We justified this decision because the purpose of our project was not truly to serve customers and instead was for a class. We thought given the extra day we would be able to accomplish all of our test cases and because a group was going to have to present a day later, we tried to take advantage of this situation. However, we acknowledge that in a professional setting this would be inappropriate to do and instead we should have gone into the presentation knowing that we had not finished all of the test cases and they should have just been added to the next sprint. This provided a good learning experience for us.
- For the goals that were not met, what were the issues?
    - We didn't realize early enough that connecting our back-end (data + Dijkstra's) to Mapbox's Navigation and Directions API (used for the front-end map feature) was a very complicated and difficult process. Also, Mapbox's APIs didn't allow us to customize the path optimization algorithm with our Dijkstra's algorithm. It already had two pre-built-in path optimization algorithms that could be used.
- How could you have done things differently?
    - We should have taken a different approach at the onset of the project by focusing on the map API first to make sure we can customize the map and tailor it specifically to Davidson college's campus. We also could have displayed the most optimal path outputted by Dijkstra's on a static image of the campus map instead of a dynamic, interactive map.

10. Team Management

- What were the team roles?
    - Taylor: Scrum Master / Developer
    - Davin: Product Owner / Developer
    - Oliver: Developer
    - Kiko: Developer
- What did each team member contribute?

    - Davin designed and coded the front-end UI for the Wildcat Ways website. He also implemented the custom interactive map on the website using Mapbox's Directions API, Navigation API, and Map API.
    - Oliver used Python and Flask to set up the back-end server and connected the back-end to the front-end UI.
    - Taylor spearheaded the collection of our data for Dijkstra's and implemented Dijkstra's algorithm for route optimization. He also combined all of our data into one spreadsheet and formatted it to make sure it was compatible with Dijkstra's algorithm.
    - Kiko flagged nodes and pathways on campus that are not accessible to create the accessibility functionality for our project. He also set up meetings with our stakeholders and kept them updated on the progress of our project.
    - Each member of the team collected their own set of data for their designated area on campus (we split campus into four sections)
- What were the challenges regarding team management, e.g., regular meeting, etc.?
    - The most challenging part in terms of team management was finding a consistent weekly meeting time that worked with everyone's schedules. All of our team members had very different schedules with different availability and extracurricular commitments. Therefore, finding a time that consistently worked for everyone was quite difficult.
- What are the plans to overcome the challenges?
    - To overcome the challenges of finding a time that works for everyone to hold our weekly meetings, we decided to meet briefly before and after class on Tuesdays and Thursdays to review our current sprint and delegate tasks for our next sprint. In addition to those meetings, we shared our individual schedules (on Outlook and Google Calendar) with each other to make sure we can block out commonly available times for sprint reviews and weekly meetings.
- If you were the third party who knows very well about your team, what suggestions would you give to your team?
    - Find a more effective way of communicating within the team for weekly meetings and progress checks

- Use an Agile project management tool like Jira to organize sprint backlogs and keep track of each team member’s progress on their delegated tasks