FINAL REPORT OF GROUP "PATH11"

Shortest Path Algorithm for Material Transportation

Submitted by:

Turjo Sarker (a1865095)

Project Vision:

Our original project vision recognized that in any large-scale industry, a product must go through various manufacturing stages before it is launched to the market. The path a product takes to reach its destination depends on factors like quality, requirements, and cost. To find the best way, we need a smart algorithm. This helps reduce costs, improve efficiency, and maintain product quality. Our project aims to create such an algorithm using SQL. Therefore, our objective for our final product included the following features:

1. Return 5 shortest paths for a given source and destination ordered by cost.
2. Exclude devices that are faulted or in use by another path currently running.
3. Ability to work with divergent and convergent devices to handle transporting from one source to multiple locations and vice versa.
4. Ability to reasonably scale up to larger sites with more devices.

As we progressed towards the end of our project, our initial vision remained largely consistent, with only a few minor adjustments to our requirements. These changes included considering the inclusion of loops in our systems and addressing costs associated with both the path and devices. This was made possible by our thorough examination of the project outline and the formulation of essential questions for the client, which provided us with a clearer project vision.

Customer Q&A:

We held our client meetings every alternate Friday at the end of each sprint. Therefore, we planned a virtual group meeting just before the client meeting, during which we had a 30-minute discussion to address all the issues posted on GitHub and resolve any project-related confusions. At the end of the discussion, we created a summary of vital questions for the client to provide a clearer view of our project at hand.

Here are some of the most important questions our group asked during the client meeting:

Question: “What type of factory or industry will use the final product of our project?”

Answer: "Your final algorithm should not be designed with a specific industry in mind; instead, it should be generalized for any industry dealing with product transportation from a source to a destination, where multiple possible paths, including divergent and convergent routes, can exist. Consequently, your final product should have the capability to identify the five shortest paths among these given routes.”

Addressing this question was crucial for us to gain a clearer understanding of the project vision and how to approach it.

Question: “Are loops expected in the connected paths for a given route from the source to the destination?”

Answer: “Yes, you may encounter loops in the connected paths.”

Initially, we designed an algorithm that successfully provided the shortest path from a source to a destination. However, it couldn't handle paths with loops. After consulting with our client, we gained a clearer understanding of our requirements and conditions. As a result, we decided to implement a new algorithm capable of handling loops.

Question: "When calculating the cost of the shortest path, should we consider the cost for only the path, or the cost associated to both the path and the device?"

Answer: "Yes, you must consider the cost associated to both path and device."

Initially, we accounted cost only for the path, therefore after the response to this question we decided to change our database schema and add cost for individual device as well.

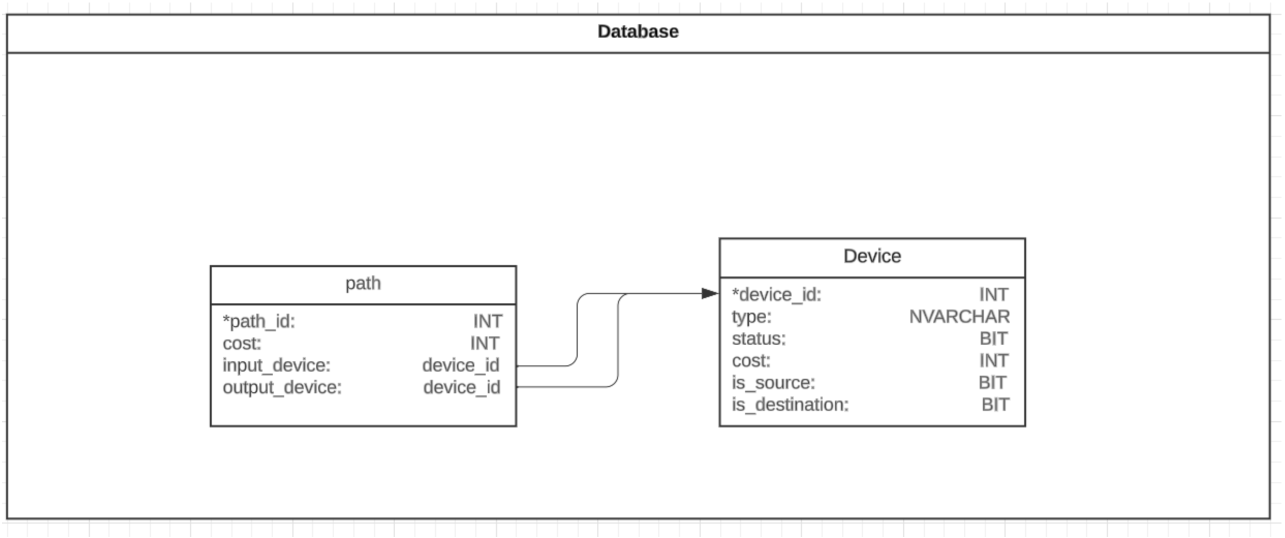
Overall, our meetings with the clients were productive and served successfully to point us in the right direction by providing a clearer view of our final product. The 30 minutes group discussion before each client meeting proved to be a very good approach where we summarized all the key points of discussion and even created a visual representation of some of the key points, we wanted to discuss which provided a clearer context to the client and leading to an efficient discussion. I am going to use this technique for my future projects as well since it showed effectiveness in our current project.

Users and User Stories (Group):

Roles	Responsibilities	Actions	User Story implementations
Transportation Planner	> Integration of real-time traffic data. > The ability to change, add, or remove nodes that stand in for routes and destinations. > Scalability to manage a high volume of cars and routes	> Update the node dataset frequently with new destinations and the state of the roads. > Provide drivers and decision-makers with optimized route data and visuals. > Routes should be continuously monitored and modified to accommodate shifting traffic patterns.	According to the responsibilities and actions discusses it meets the user story which we implemented which satisfies the following demands: 1. Ability to add or remove nodes. 2. Ability to modify nodes. 3. Provide shortest path to multiple destinations.
Network Administrator	> The capacity to depict network elements as nodes, such as servers, switches, routers, etc. > support for changing, adding, or deleting network connections and nodes. > When determining the shortest path, variables like bandwidth, latency, and dependability are taken into account. > integration for real-time updates with network monitoring tools.	> Update the node dataset often to account for network changes brought forth by expansions, new equipment installs, and device failures. > Determine the best routes for data transfer between devices. > Information to locate bottlenecks or unstable connections in order to fix network problems. > Prioritize vital data traffic	According to the responsibilities and actions discusses it meets the user story which we implemented which satisfies the following demands: 1. Ability to add or remove nodes. 2. Ability to change the status of device to identify faulty devices. 3. Provide shortest path to multiple destinations sorted by cost.

Supply Chain Analyst:	> Possibility of modeling suppliers, warehouses, distribution centers, and retail locations as nodes along the supply chain network. > Support for continuous updates to take demand, inventory levels, and supplier locations into consideration. > taking into account elements like lead times, inventory control, and the cost of transportation.	> Update the node dataset often to account for modifications to delivery capacities, inventory levels, and supplier connections. > Manage the supply chain network to keep lead times shorter, transportation costs as low as possible, and inventory levels at ideal levels. > To assess the effects of prospective modifications, such adding additional distribution centers.	According to the responsibilities and actions discusses it meets the user story which we implemented which satisfies the following demands: 1. Ability to add or remove nodes. 2. Ability to modify nodes. 3. Provide 5 shortest path to destination sorted by cost.

Software Architecture:



The database structure chosen for this project is illustrated in the database schema shown above. This database is designed using a graph format, where information is structured as nodes, and the connections between nodes are indicated by the device and path entities. This architecture allowed us to effectively model the factory's layout, which in turn helped us in the experimentation and development of a suitable search algorithm for our project.

The device entity comprises six fields. The device_id serves as a unique identifier to distinguish each device in the database. The type field defines the device's category, such as "divergent," "source," etc. The status field contains a Boolean value that indicates the device's operational status, with "false" denoting a faulty device. The cost field represents the current device cost as an integer value. The "is_source" and "is_destination" fields specify whether the device functions as a source and are represented using a binary format, where 0 signifies false and 1 signifies true.

The path entity comprises four fields. The path_id serves as a unique identifier to distinguish each path. The cost field represents the current cost of the path as an integer value. The input_device and output_device fields refer to the devices connected by the path. The input_device and output_device fields serve as foreign keys referencing the device_id field of the device entity.

Tech Stack and Standards

Our final choice for the backend development tech stack was MSSQL. Initially, we had planned to use MySQL, a platform many of our group members were already familiar with. However, after conducting research on both options, we made the decision to switch to MSSQL. Our research revealed that MSSQL is often the preferred choice in larger industries because of its comprehensive enterprise features and extensive support options. Additionally, MSSQL is well-regarded for its scalability and performance capabilities, making it suitable for handling substantial volumes of data and high transaction loads. Given that we are working on a large-scale industrial project, we opted for MSSQL as the more suitable choice.

We initially planned for a front-end tech stack as well. However, based on the deliverable requirements, which did not include building a front-end platform such as a graphical interface, we shifted our focus entirely to finding a backend solution for our project.

Throughout our project, we experimented with a variety of tools for communication and development. Our primary mode of communication was through a WhatsApp group chat. We chose WhatsApp due to its wide compatibility and user-friendly interface. For informal online meetings, we utilized Discord, which offers channels for quick and easy meetings and includes helpful features such as persistent chat histories and screen-sharing. In more formal interactions with clients, we turned to Slack, and for conducting meetings, we relied on Zoom. General files were stored and shared using OneDrive, which provides ample free storage space and easy access for the team. GitHub served as our code and version control repository. We used Google Docs to create a shared documentation file for preparing snapshots and other text documents. Initially, we opted for Google Slides for preparing our presentation slides but later switched to Canva. Canva offered greater flexibility, a better interface, and more customization options for our presentation needs.

Towards the conclusion of our project, we recognized that our communication tools could have been more optimized. One notable issue was the redundancy of sending messages in both Discord and WhatsApp. Some team members preferred WhatsApp, while others favored Discord, leading to potential miscommunication and confusion. Another challenge was the inconsistency in file uploads, as some team members uploaded files in Discord while others used OneDrive, further contributing to confusion.

In the future, I intend to streamline our communication by primarily using Discord as discord offers greater functionality and flexibility, which should help mitigate these issues and enhance collaboration within the team.

During our development phase, we experimented with various tools. Initially, we attempted to use Azure Data Studio, a highly recommended approach for large-scale industries. However, we encountered challenges with its setup process, which proved to be too complicated for some team members. Additionally, it placed a significant demand for system resources.

As a more lightweight alternative, we turned to Microsoft SQL Server Management Studio (SSMS), an integrated environment developed by Microsoft for managing SQL infrastructure. SSMS offered an intuitive interface with a wealth of useful functionalities. While one team member suggested using Visual Studio Code (VS Code), we decided to maintain our industry-standard approach and stuck with SSMS, which served us well throughout the entire project.

The coding standards we've established are as follows:

Focus on Code Readability: With a diverse team, it's crucial that all members can seamlessly pick up where others left off. Achieving this relies on making the code as readable as possible. Key points include minimizing code length, avoiding overly long lines, adhering to a naming convention of all lowercase words separated by underscores, and segmenting code blocks effectively.

Leave Comments: Comments play a vital role in comprehending potentially complex or unintuitive lines of code. To save time for team members, we encourage leaving detailed comments that explain the underlying algorithms and logic implemented.

Branch Before Updating Code: Utilizing branches allows team members to develop features and make changes within a contained area of the repository. Always branching before updating code minimizes the likelihood of merging errors and accelerates the correction of new changes.

Testing: Rigorous testing is a core practice. Each piece of code undergoes extensive testing, including checks by both the developer and another team member not involved in the code's creation. This encompasses various testing facets, such as unit testing, integration testing, and system testing, ensuring a high standard of code quality.

Group Meetings and Team Member Roles

Group Meeting Schedule:

- Virtually on every Fridays: 5pm to 5:45pm
- In-Person on every Monday: 3pm to 4pm

Meeting with Client:

- Virtually on every alternate Fridays: 5:45pm to 6pm

Preparations Before Retrospective Meetings:

• Presentation of the completed sprint and plans for the upcoming sprint.

• Seek client feedback on potential architecture changes.

• Discuss changes from the completed sprint and pose questions for the next sprint.

Communication with Tutor:

• Via Slack.

Scrum Masters by Sprint:

Sprint 1: Jacob Dorn (A1688149)

Sprint 2: Jarell Ho (A1876408)

Sprint 3: Ketong Shen (A1832586)

Sprint 4: Turjo Sarker (A1865095)

Sprint 5: Seth Ossowicz (A1771732)

I am very satisfied with the group meeting schedules we followed. Timeboxing each meeting helped us achieve a more structured execution of our project by adhering to a routine timetable. Typically, our primary discussions took place on Mondays, and we set deadlines for our tasks by Thursdays. On Fridays, we addressed any challenges we encountered and summarized any questions or concerns. We then discussed these with our clients in the following meetings.

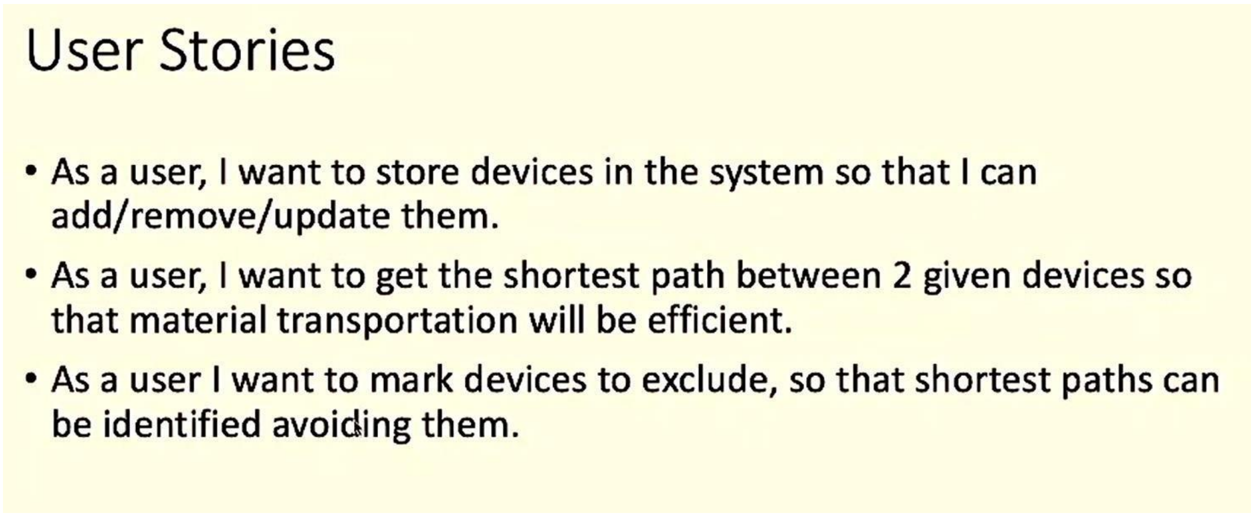
My primary role among the group was as a software tester. Throughout our project I used a Test-Driven Development methodology to ensure that our project met all the clients' requirements, and our program ran as expected. In each sprint my testing process can be broken down into 4 parts:

1. We all brainstormed in group meetings, where we identified the increment of functionality that is required.
2. Black box and white box testing were conducted:
   a. I wrote **Unit tests** which covered a wide range of inputs including the **boundary cases** and their expected outputs**.**
   b. If the Black box testing failed to meet the expected output, I then conducted **White Box** testing to look into the actual code to identify the flaws.
3. After finishing test for a certain increment and fixing any bugs, I ran regression testing, to ensure all the previous functionalities are still intact.
4. Once all the bugs were addressed and tests ran successfully, we then moved on the next functionality.

Snapshots

Snapshot 1.1:

Product Backlog and Task Board:


User Stories

- As a user, I want to store devices in the system so that I can add/remove/update them.
- As a user, I want to get the shortest path between 2 given devices so that material transportation will be efficient.
- As a user I want to mark devices to exclude, so that shortest paths can be identified avoiding them.

The Task Board

User Stories	Task to-do
As a user, I want to store devices in the system so that I can add/remove/update them	1.Create a Database of Devices. 2.Implement Function to Add Device. 3.Implement Function to Remove Device. 4.Implement Function to update Device.
As a user, I want to get the shortest path between 2 given devices so that material transportation will be efficient	1. Create Function which takes Start and End Devices, Outputs Directions for Shortest Paths from Start Device to End Device.
As a user I want to mark devices to exclude, so that shortest paths can be identified avoiding them	1. Implement Function to Exclude / Include Device from Search Algorithm.

Sprint Backlog and User Stories:

Sprint backlog

User Stories	Task to-do
As a user, I want to store devices in the system so that I can add/remove/update them	1.Determine attributes of the devices 2.Create a Database of Devices. 3.Implement Function to Add Device. 4.Implement Function to Remove Device. 5.Implement Function to update Device. 6. Refactoring, making comments etc.

Definition of Done

The project must fit the following criteria for the current user story:

- Have a Database of Devices in Microsoft SQL (MsSQL)

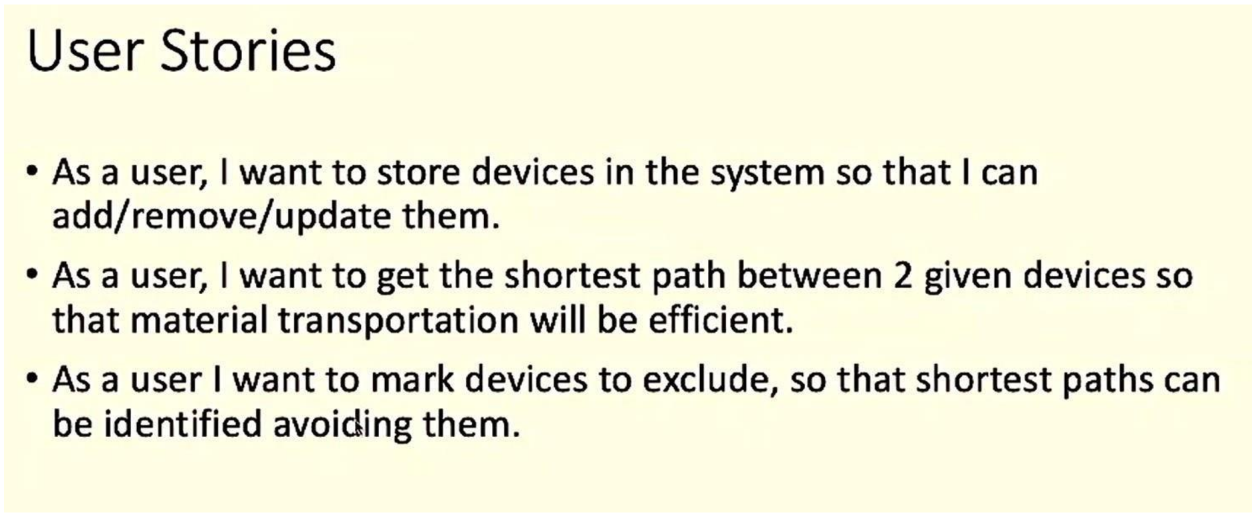
- User access to Database, including update/add/removing devices.
- Database tables have relevant columns to store device details.
- Database uploaded to GitHub and can be accessed by group members.
- Database schema changes have been documented.

Summary of Changes:

This is the first snapshot of the project. Project topics were released for the students to choose from. Clients of the project were introduced with requirements outlined for each project topics. Groups of 8 were formed and the shortest path algorithm for material transportation project was chosen. We had a short interview with the product owner who acts as a proxy to the client to gain a clearer understanding of the project. As a result, tasks were derived from the user stories for the next sprint's backlog. Communication channels were established for the group with WhatsApp being the main method of conveying information. Slack was used for formal discussion of the projects with the client and product owner. Moreover, Microsoft office and GitHub were used for our version control system. Microsoft office for general files and document sharing. Whereas GitHub for software development and version control to effectively store and manage codes for the group.

Snapshot 2.1:

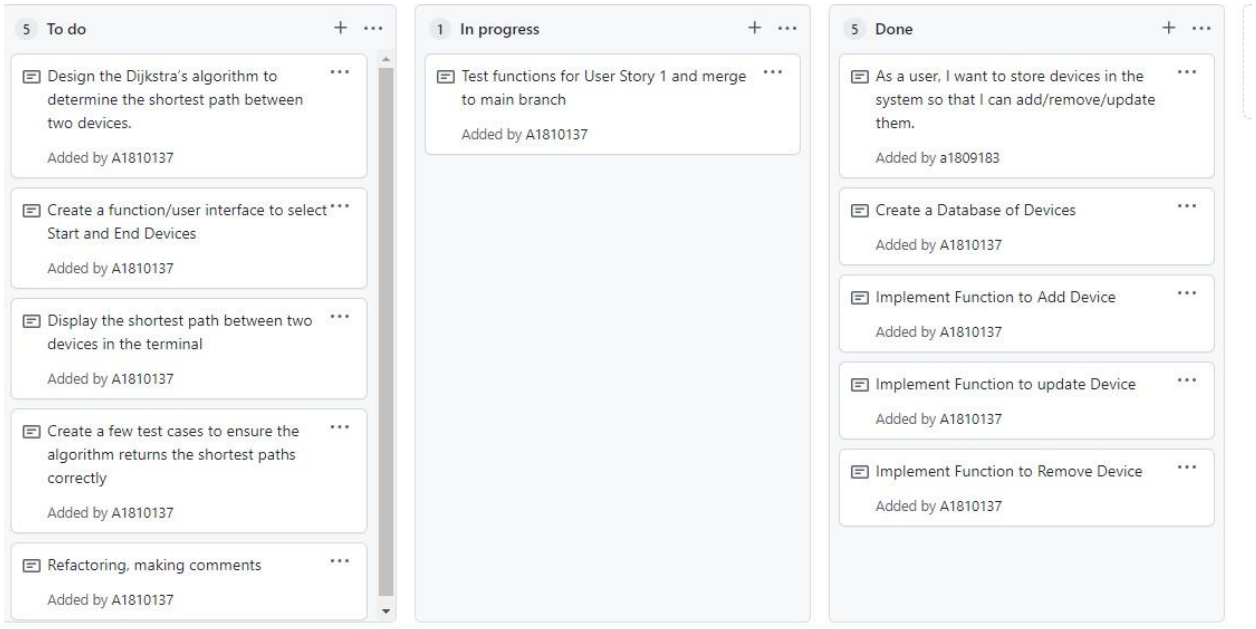
Product Backlog and Task Board:



User Stories

- As a user, I want to store devices in the system so that I can add/remove/update them.
- As a user, I want to get the shortest path between 2 given devices so that material transportation will be efficient.
- As a user I want to mark devices to exclude, so that shortest paths can be identified avoiding them.

The Task Board


5 To do

Design the Dijkstra's algorithm to determine the shortest path between two devices.
Added by A1810137

Create a function/user interface to select Start and End Devices
Added by A1810137

Display the shortest path between two devices in the terminal
Added by A1810137

Create a few test cases to ensure the algorithm returns the shortest paths correctly
Added by A1810137

Refactoring, making comments
Added by A1810137

1 In progress

Test functions for User Story 1 and merge to main branch
Added by A1810137

5 Done

As a user, I want to store devices in the system so that I can add/remove/update them.
Added by a1809183

Create a Database of Devices
Added by A1810137

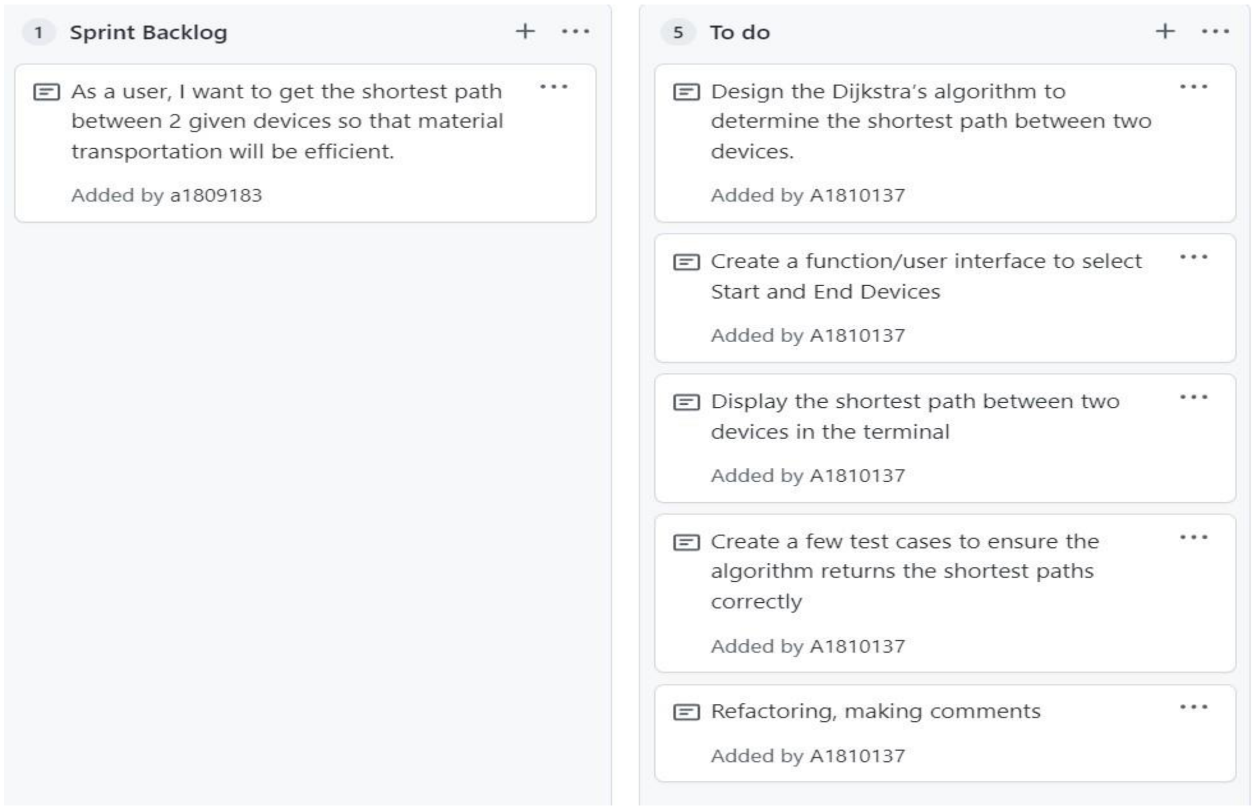
Implement Function to Add Device
Added by A1810137

Implement Function to update Device
Added by A1810137

Implement Function to Remove Device
Added by A1810137

Sprint Backlog and User Stories:

Sprintbacklog


1 Sprint Backlog

As a user, I want to get the shortest path between 2 given devices so that material transportation will be efficient.
Added by a1809183

5 To do

Design the Dijkstra's algorithm to determine the shortest path between two devices.
Added by A1810137

Create a function/user interface to select Start and End Devices
Added by A1810137

Display the shortest path between two devices in the terminal
Added by A1810137

Create a few test cases to ensure the algorithm returns the shortest paths correctly
Added by A1810137

Refactoring, making comments
Added by A1810137

Definition of Done

The project must fit the following criteria for the current user story:

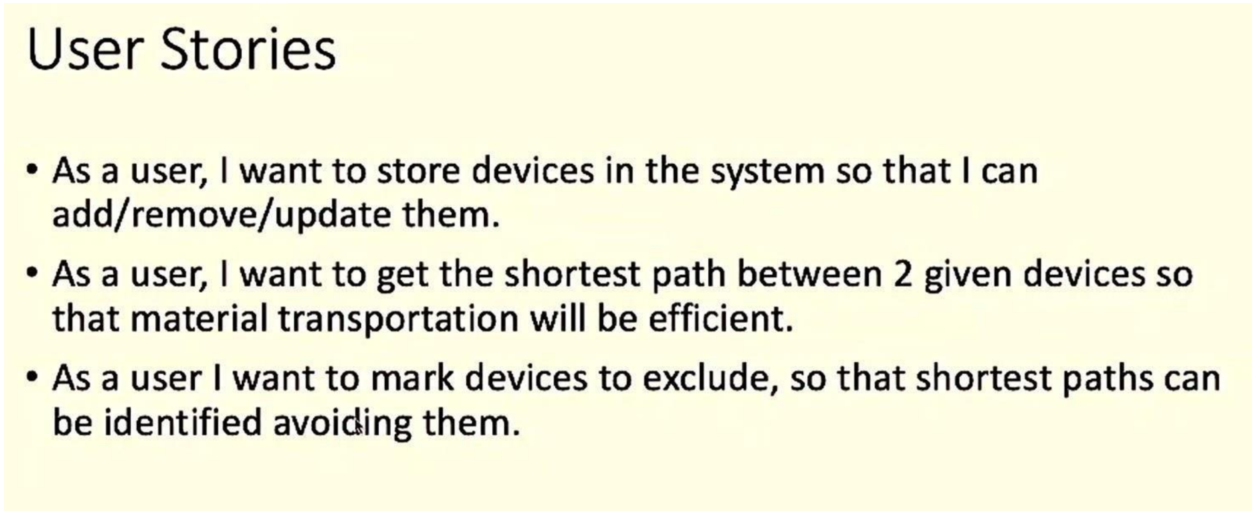
- Dijkstra's algorithm to determine the shortest path have been implemented.
- The user interface (in terminal) provides a clear way for the user to select the start and end devices to run the algorithm.
- Then, five shortest paths will be visually displayed (in terminal) in the form of a table of devices which has the start devices at the front and the end devices at the back of the array.
- Functionality has been tested under various scenarios and verified it actually works.
- Necessary documentation and comments in the code have been made.

Summary of Changes:

This is the second snapshot of the project. We have completed all the tasks for User Story 1, which states "As a user I want to store/update/remove devices in the database". We implemented the solution in Microsoft SQL and changes are being made to the GitHub repository. We are currently testing the functionality of the program to make sure everything works well before committing the codes to the main branch in the repository. This is a significant achievement as it laid down the foundation to our entire project, enabling us to proceed with more complex functionalities. This week, we have broken down the user story into a few to-do tasks: "As a user, I want to get the shortest path between two devices." We decided to implement the Dijkstra's algorithm which is a popular algorithm used to find the shortest path between two locations. This will be challenging as we have never implemented any algorithm directly inside a database.

Snapshot 2.2:

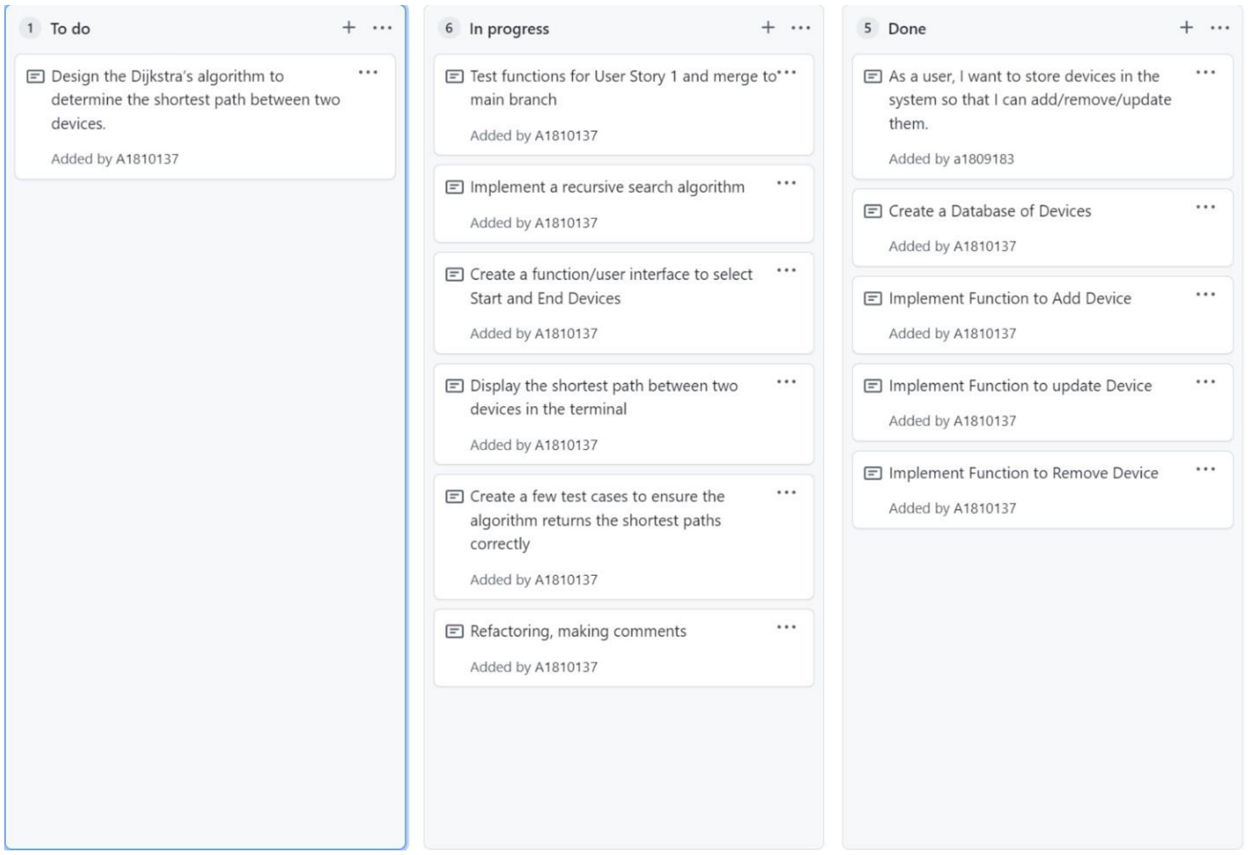
Product Backlog and Task Board:



User Stories

- As a user, I want to store devices in the system so that I can add/remove/update them.
- As a user, I want to get the shortest path between 2 given devices so that material transportation will be efficient.
- As a user I want to mark devices to exclude, so that shortest paths can be identified avoiding them.

The Task Board


1 To do

Design the Dijkstra's algorithm to determine the shortest path between two devices.
Added by A1810137

6 In progress

Test functions for User Story 1 and merge to main branch
Added by A1810137

Implement a recursive search algorithm
Added by A1810137

Create a function/user interface to select Start and End Devices
Added by A1810137

Display the shortest path between two devices in the terminal
Added by A1810137

Create a few test cases to ensure the algorithm returns the shortest paths correctly
Added by A1810137

Refactoring, making comments
Added by A1810137

5 Done

As a user, I want to store devices in the system so that I can add/remove/update them.
Added by a1809183

Create a Database of Devices
Added by A1810137

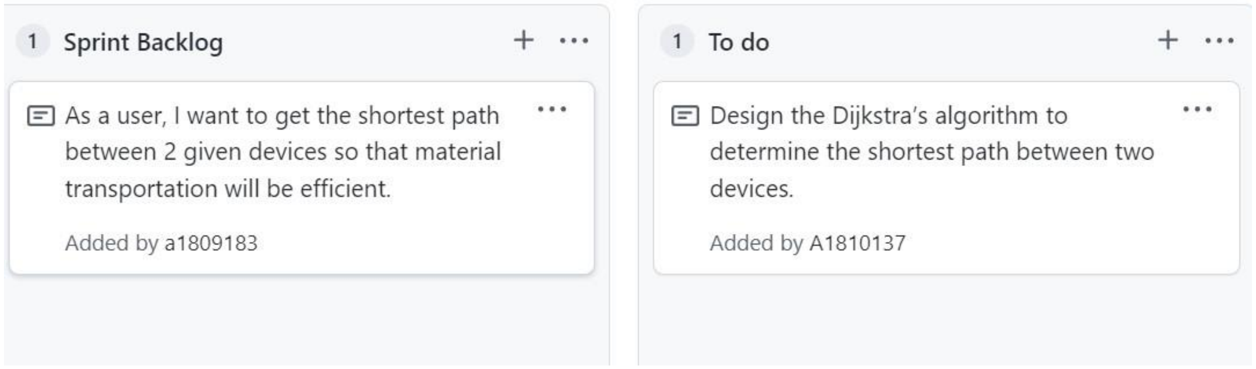
Implement Function to Add Device
Added by A1810137

Implement Function to update Device
Added by A1810137

Implement Function to Remove Device
Added by A1810137

Sprint Backlog and User Stories:

Sprintbacklog


1 Sprint Backlog

As a user, I want to get the shortest path between 2 given devices so that material transportation will be efficient.
Added by a1809183

1 To do

Design the Dijkstra's algorithm to determine the shortest path between two devices.
Added by A1810137

Definition of Done

The project must fit the following criteria for the current user story:

- Dijkstra's algorithm to determine the shortest path have been implemented otherwise other algorithm could be implemented.

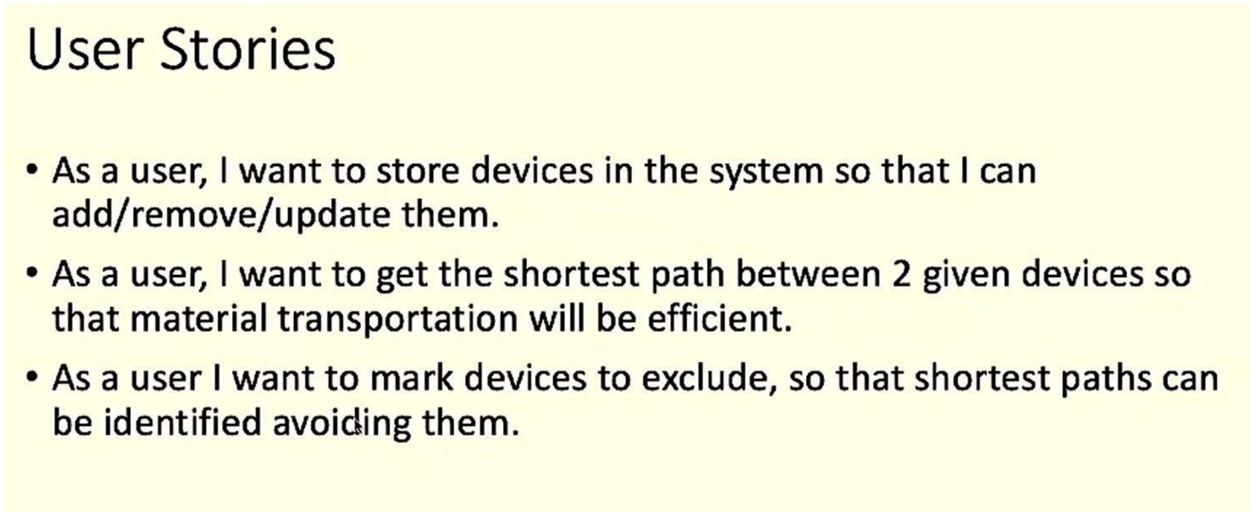
- The user interface (in terminal) provides a clear way for the user to select the start and end devices to run the algorithm.
- Then, five shortest paths will be visually displayed (in terminal) in the form of a table of devices which has the start devices at the front and the end devices at the back of the array.
- Functionality has been tested under various scenarios and verified it works.
- Necessary documentation and comments in the code have been made.

Summary of Changes:

The functions/queries of User story 1 have already been tested by group members and confirming that the functions work as expected. However, the product owner has provided us with new information on the devices available. We now have a better understanding of the layout, map, and connections of the devices. Hence, further changes to the database schema might be required and testing will be done again to ensure its reliability and functionality. This testing phase allowed us to gain an insight to Microsoft SQL and database query, laying a solid foundation for our project. Besides that, we modified our approach for User Story 2, which is finding the five shortest paths between two devices. Originally, we considered implementing Dijkstra's algorithm but its complexity within MSSQL led us to adopt a recursive search algorithm instead. This recursive search algorithm serves as a proof of concept before developing the more complex Dijkstra's algorithm.

Snapshot 3.1:

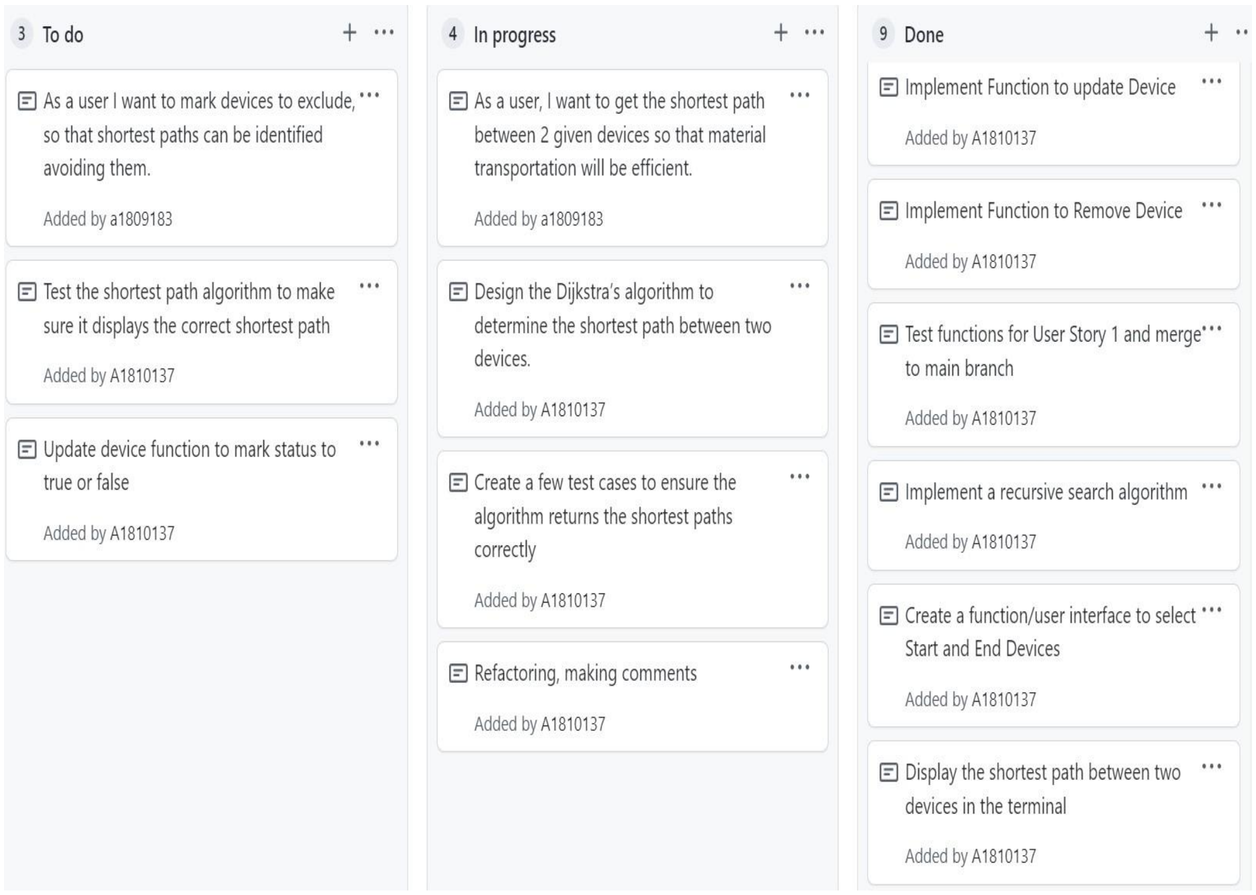
Product Backlog and Task Board:



User Stories

- As a user, I want to store devices in the system so that I can add/remove/update them.
- As a user, I want to get the shortest path between 2 given devices so that material transportation will be efficient.
- As a user I want to mark devices to exclude, so that shortest paths can be identified avoiding them.

The Task Board



3 To do

As a user I want to mark devices to exclude, so that shortest paths can be identified avoiding them.

Added by a1809183

Test the shortest path algorithm to make sure it displays the correct shortest path

Added by A1810137

Update device function to mark status to true or false

Added by A1810137

4 In progress

As a user, I want to get the shortest path between 2 given devices so that material transportation will be efficient.

Added by a1809183

Design the Dijkstra's algorithm to determine the shortest path between two devices.

Added by A1810137

Create a few test cases to ensure the algorithm returns the shortest paths correctly

Added by A1810137

Refactoring, making comments

Added by A1810137

9 Done

Implement Function to update Device

Added by A1810137

Implement Function to Remove Device

Added by A1810137

Test functions for User Story 1 and merge to main branch

Added by A1810137

Implement a recursive search algorithm

Added by A1810137

Create a function/user interface to select Start and End Devices

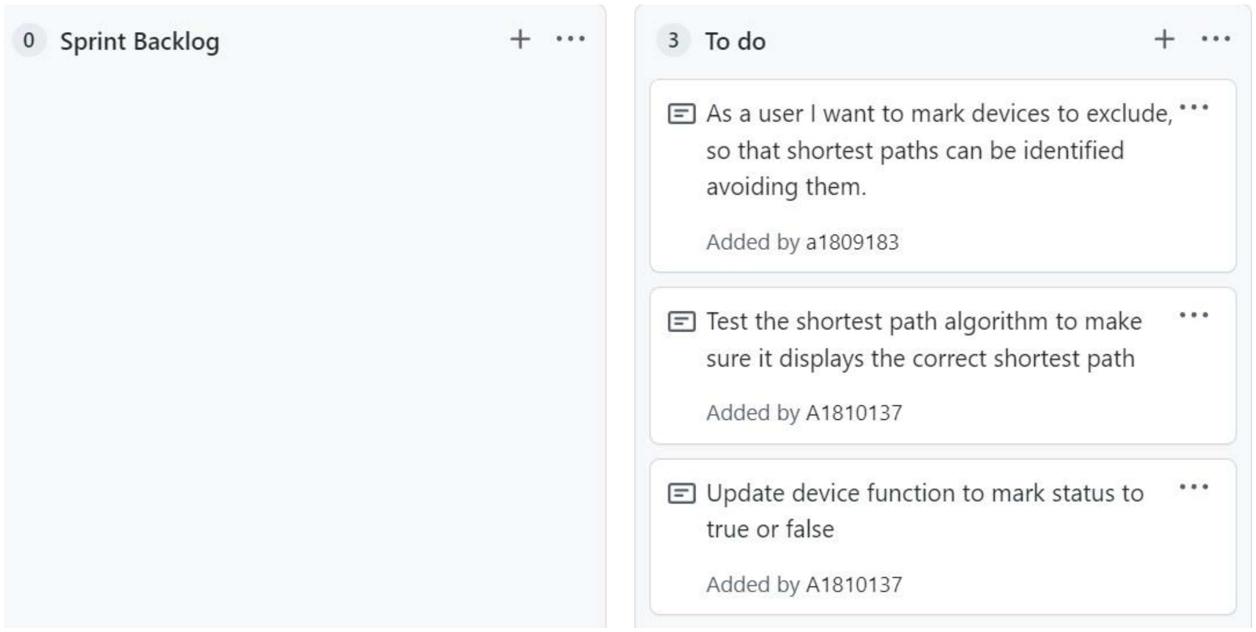
Added by A1810137

Display the shortest path between two devices in the terminal

Added by A1810137

Sprint Backlog and User Stories:

Sprint backlog



0 Sprint Backlog

3 To do

As a user I want to mark devices to exclude, so that shortest paths can be identified avoiding them.

Added by a1809183

Test the shortest path algorithm to make sure it displays the correct shortest path

Added by A1810137

Update device function to mark status to true or false

Added by A1810137

Definition of Done

The project must fit the following criteria for the current user story:

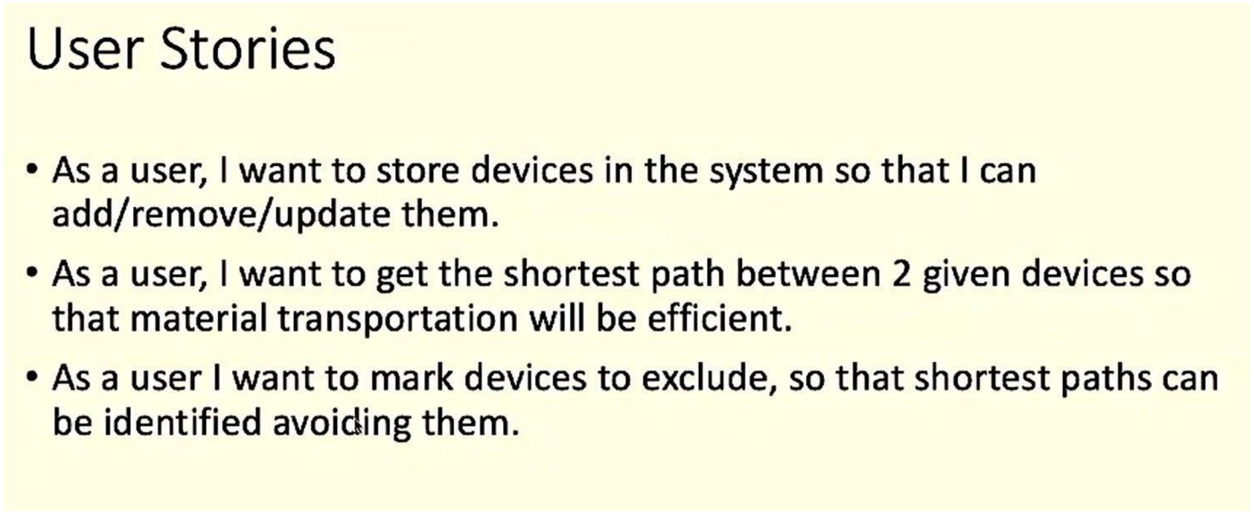
- Update the status of devices: 1 for true and 0 for false
- When search function is run again, the path with the excluded devices will be ignored
- Correctly return five shortest paths that exclude the devices status of 0
- Functionality has been tested under various scenarios and verified it works
- Necessary documentation and comments in the code have been made

Summary of Changes:

The functions/queries of User story 2 have already been tested by group members. However, it was tested with only a few simple cases. In-depth testing still needs to be done to fully verify the functionality. The database schema has been altered to fit the new requirements of the devices to include cost and whether they are a source or destination type devices. An additional query will be pushed to alter any existing local database to include the corresponding schema. This is so that group members do not need to recreate their database to fit the new criteria. Moreover, we have started to work on user story 3 which is to exclude devices so that the query will avoid them when returning the shortest path. We plan to modify an existing function/query "UpdateDevice.sql" to include an extra functionality to mark device's status.

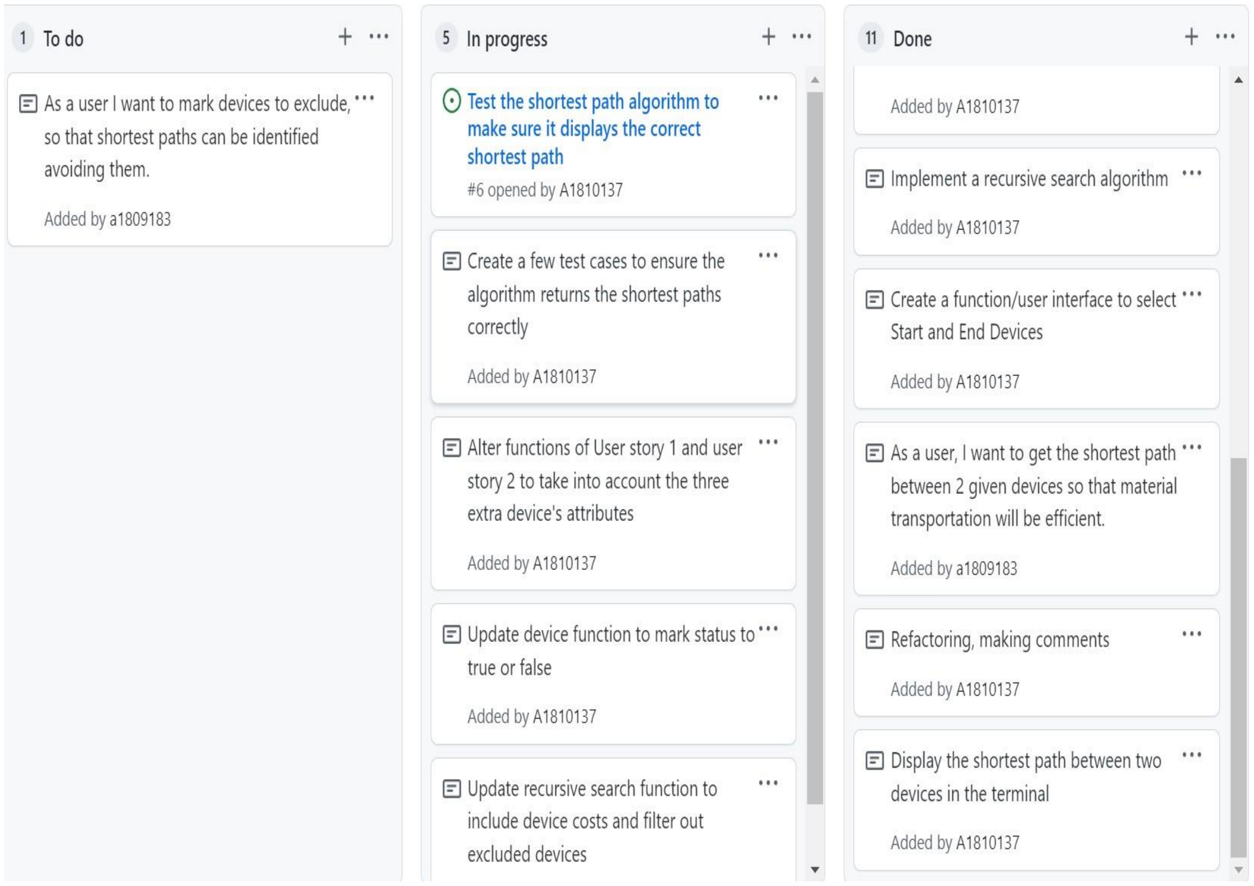
Snapshot 3.2:

Product Backlog and Task Board:


User Stories

- As a user, I want to store devices in the system so that I can add/remove/update them.
- As a user, I want to get the shortest path between 2 given devices so that material transportation will be efficient.
- As a user I want to mark devices to exclude, so that shortest paths can be identified avoiding them.

The Task Board


1 To do

As a user I want to mark devices to exclude, so that shortest paths can be identified avoiding them.

Added by a1809183

5 In progress

Test the shortest path algorithm to make sure it displays the correct shortest path

#6 opened by A1810137

Create a few test cases to ensure the algorithm returns the shortest paths correctly

Added by A1810137

Alter functions of User story 1 and user story 2 to take into account the three extra device's attributes

Added by A1810137

Update device function to mark status to true or false

Added by A1810137

Update recursive search function to include device costs and filter out excluded devices

11 Done

Added by A1810137

Implement a recursive search algorithm

Added by A1810137

Create a function/user interface to select Start and End Devices

Added by A1810137

As a user, I want to get the shortest path between 2 given devices so that material transportation will be efficient.

Added by a1809183

Refactoring, making comments

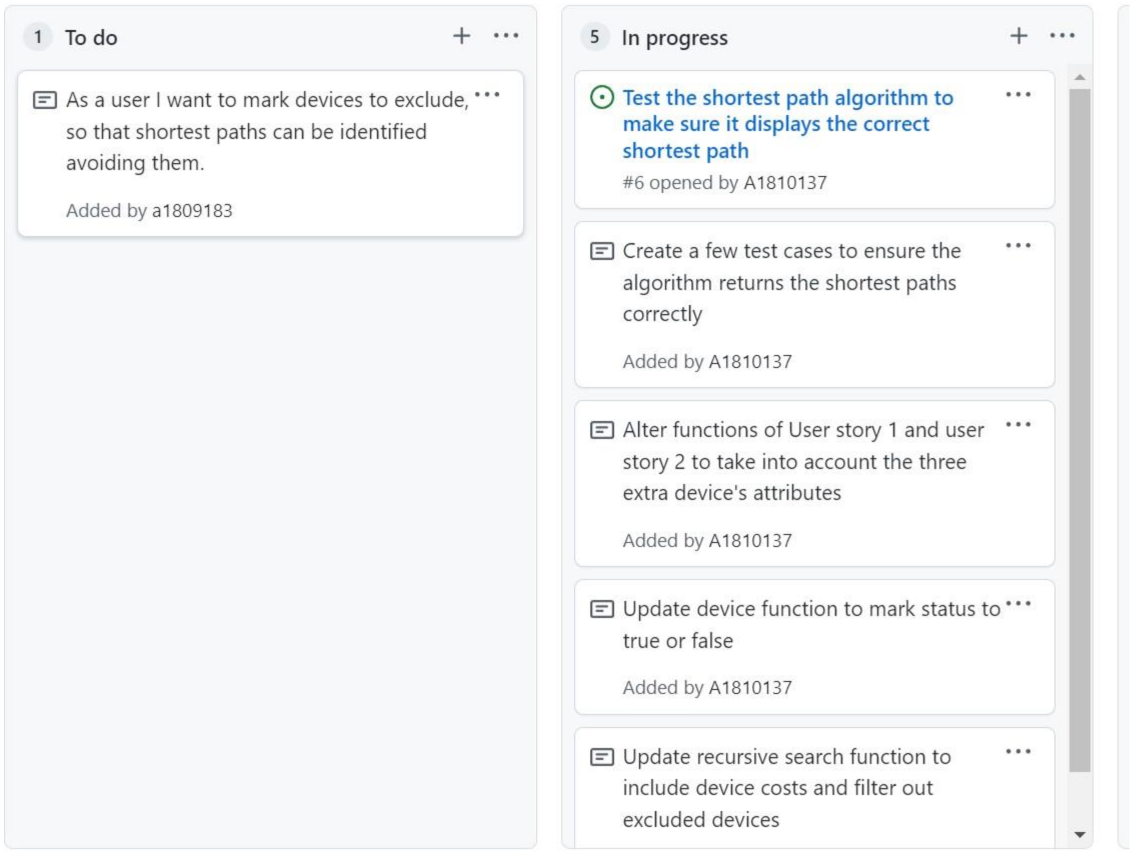
Added by A1810137

Display the shortest path between two devices in the terminal

Added by A1810137

Sprint Backlog and User Stories:

Sprint backlog


1 To do

As a user I want to mark devices to exclude, so that shortest paths can be identified avoiding them.

Added by a1809183

5 In progress

Test the shortest path algorithm to make sure it displays the correct shortest path

#6 opened by A1810137

Create a few test cases to ensure the algorithm returns the shortest paths correctly

Added by A1810137

Alter functions of User story 1 and user story 2 to take into account the three extra device's attributes

Added by A1810137

Update device function to mark status to true or false

Added by A1810137

Update recursive search function to include device costs and filter out excluded devices

Definition of Done

The project must fit the following criteria for the current user story:

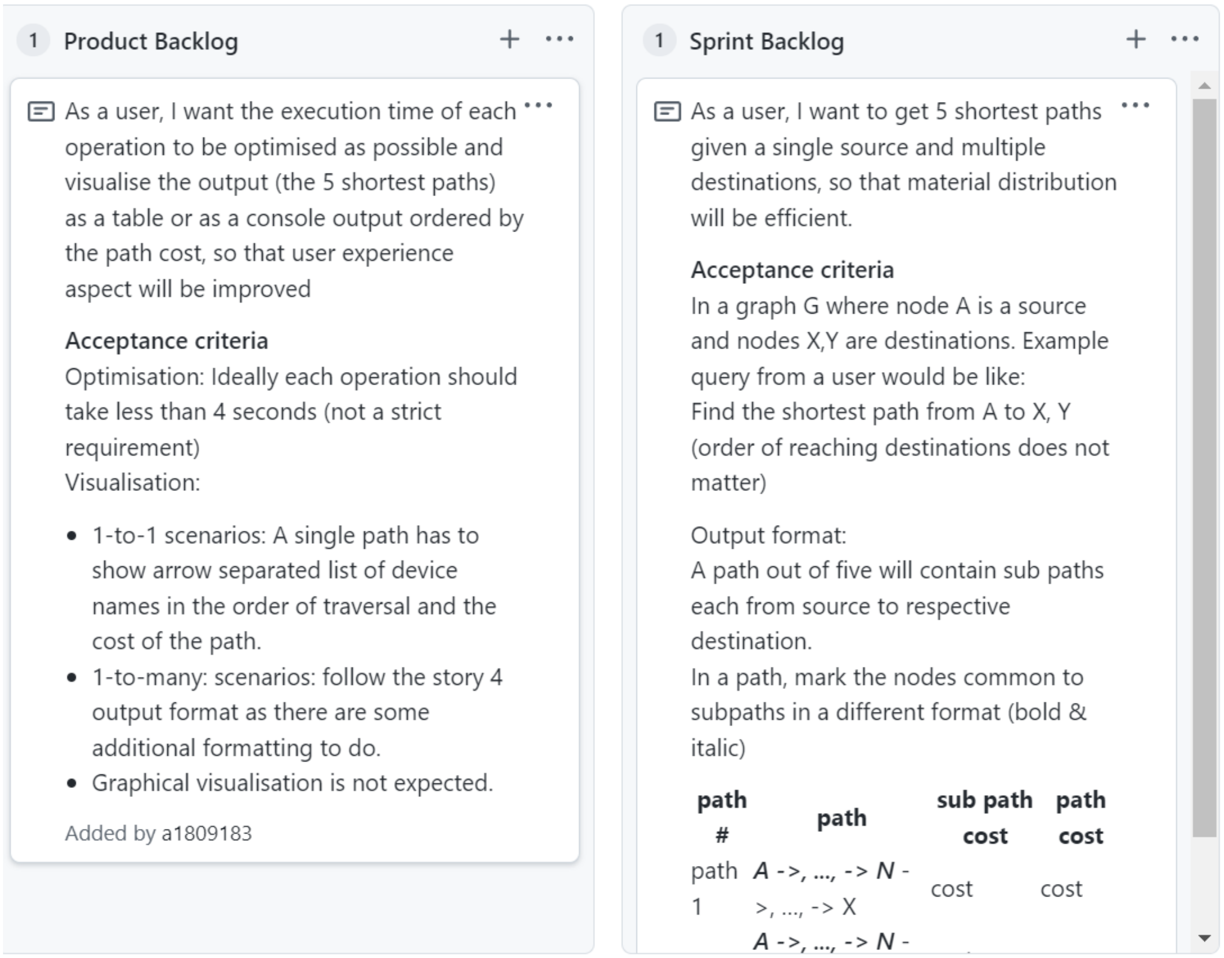
- Update the status of devices: 1 for true and 0 for false
- When search function is run again, the path with the excluded devices will be ignored
- Correctly return five shortest paths that exclude the devices status of 0
- Functionality has been tested under various scenarios and verified it works
- Necessary documentation and comments in the code have been made

Summary of Changes:

User story 3 to exclude the devices that been marked to status ‘0’ when returning a list of shortest paths has been implemented easily. However, we are still working on verifying that the function fully works as expected before pushing to the main branch. A set of new issues have arisen due to the changing requirements of database schema. The original user story 1 to add devices must now account for cost and whether it’s a source or destination. The recursive search function was updated to also include device costs and filter out excluded devices, as it currently only takes path cost into account and assumes all devices are active. These issues have been created in GitHub so that group members can solve them.

Snapshot 4.1:

Product Backlog and Task Board:



1 Product Backlog

As a user, I want the execution time of each operation to be optimised as possible and visualise the output (the 5 shortest paths) as a table or as a console output ordered by the path cost, so that user experience aspect will be improved

Acceptance criteria
Optimisation: Ideally each operation should take less than 4 seconds (not a strict requirement)
Visualisation:

- 1-to-1 scenarios: A single path has to show arrow separated list of device names in the order of traversal and the cost of the path.
- 1-to-many: scenarios: follow the story 4 output format as there are some additional formatting to do.
- Graphical visualisation is not expected.

Added by a1809183

1 Sprint Backlog

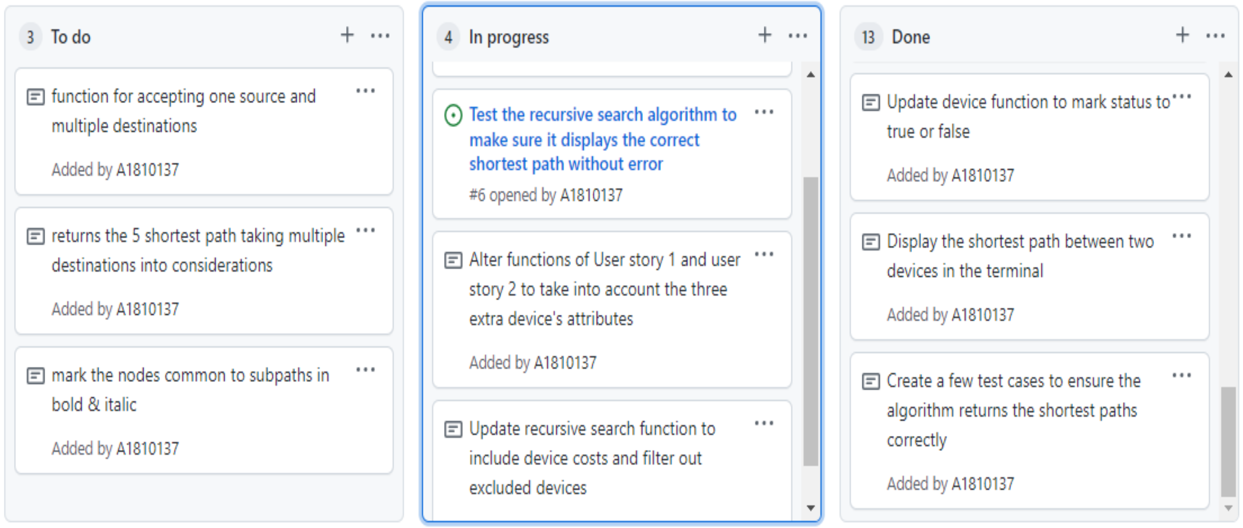
As a user, I want to get 5 shortest paths given a single source and multiple destinations, so that material distribution will be efficient.

Acceptance criteria
In a graph G where node A is a source and nodes X,Y are destinations. Example query from a user would be like:
Find the shortest path from A to X, Y (order of reaching destinations does not matter)

Output format:
A path out of five will contain sub paths each from source to respective destination.
In a path, mark the nodes common to subpaths in a different format (bold & italic)

path #	path	sub path cost	path cost
path 1	*A* ->, ..., -> *N* ->, ..., -> X	cost	cost
	A ->, ..., -> *N* -		

The Task Board


3 To do

function for accepting one source and multiple destinations

Added by A1810137

returns the 5 shortest path taking multiple destinations into considerations

Added by A1810137

mark the nodes common to subpaths in bold & italic

Added by A1810137

4 In progress

Test the recursive search algorithm to make sure it displays the correct shortest path without error

#6 opened by A1810137

Alter functions of User story 1 and user story 2 to take into account the three extra device's attributes

Added by A1810137

Update recursive search function to include device costs and filter out excluded devices

13 Done

Update device function to mark status to true or false

Added by A1810137

Display the shortest path between two devices in the terminal

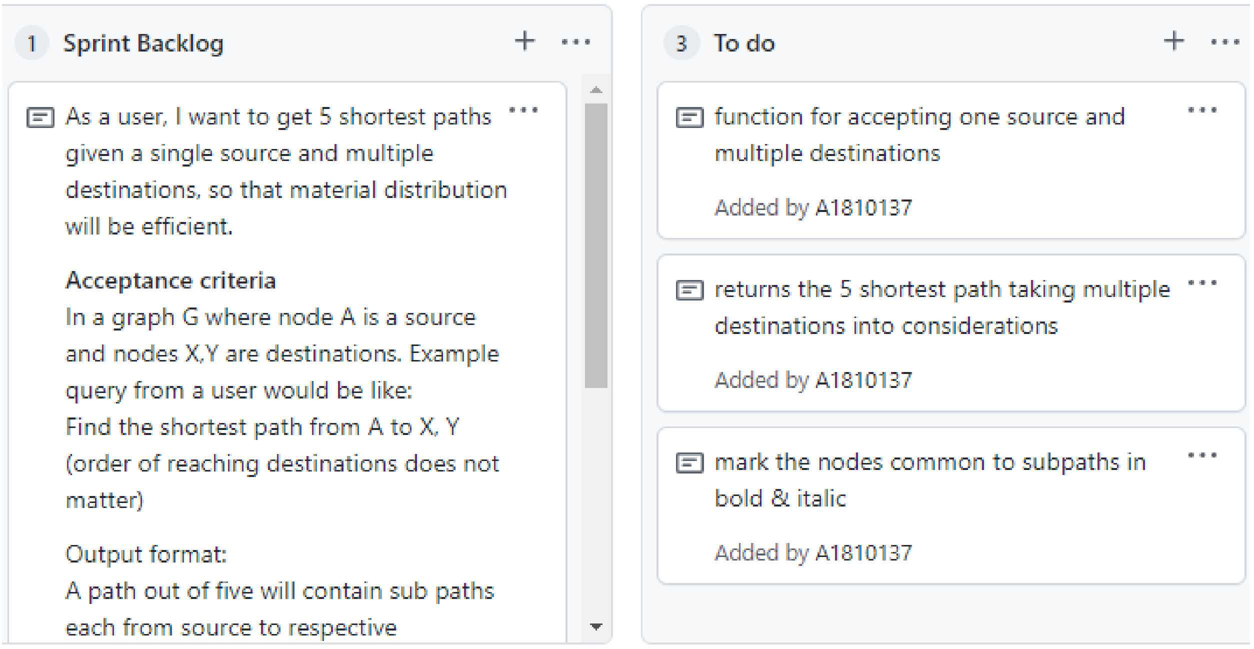
Added by A1810137

Create a few test cases to ensure the algorithm returns the shortest paths correctly

Added by A1810137

Sprint Backlog and User Stories:

Sprint backlog


1 Sprint Backlog

As a user, I want to get 5 shortest paths given a single source and multiple destinations, so that material distribution will be efficient.

Acceptance criteria

In a graph G where node A is a source and nodes X,Y are destinations. Example query from a user would be like:

Find the shortest path from A to X, Y (order of reaching destinations does not matter)

Output format:

A path out of five will contain sub paths each from source to respective

3 To do

function for accepting one source and multiple destinations

Added by A1810137

returns the 5 shortest path taking multiple destinations into considerations

Added by A1810137

mark the nodes common to subpaths in bold & italic

Added by A1810137

Definition of Done

The project must fit the following criteria for the current user story:

- The recursive search algorithm is able to take in one source and multiple destinations
- Returns five shortest path for multiple considerations
- Each returned shortest paths will have subpaths corresponding to each destination
- Mark the nodes common to subpaths in bold and italic

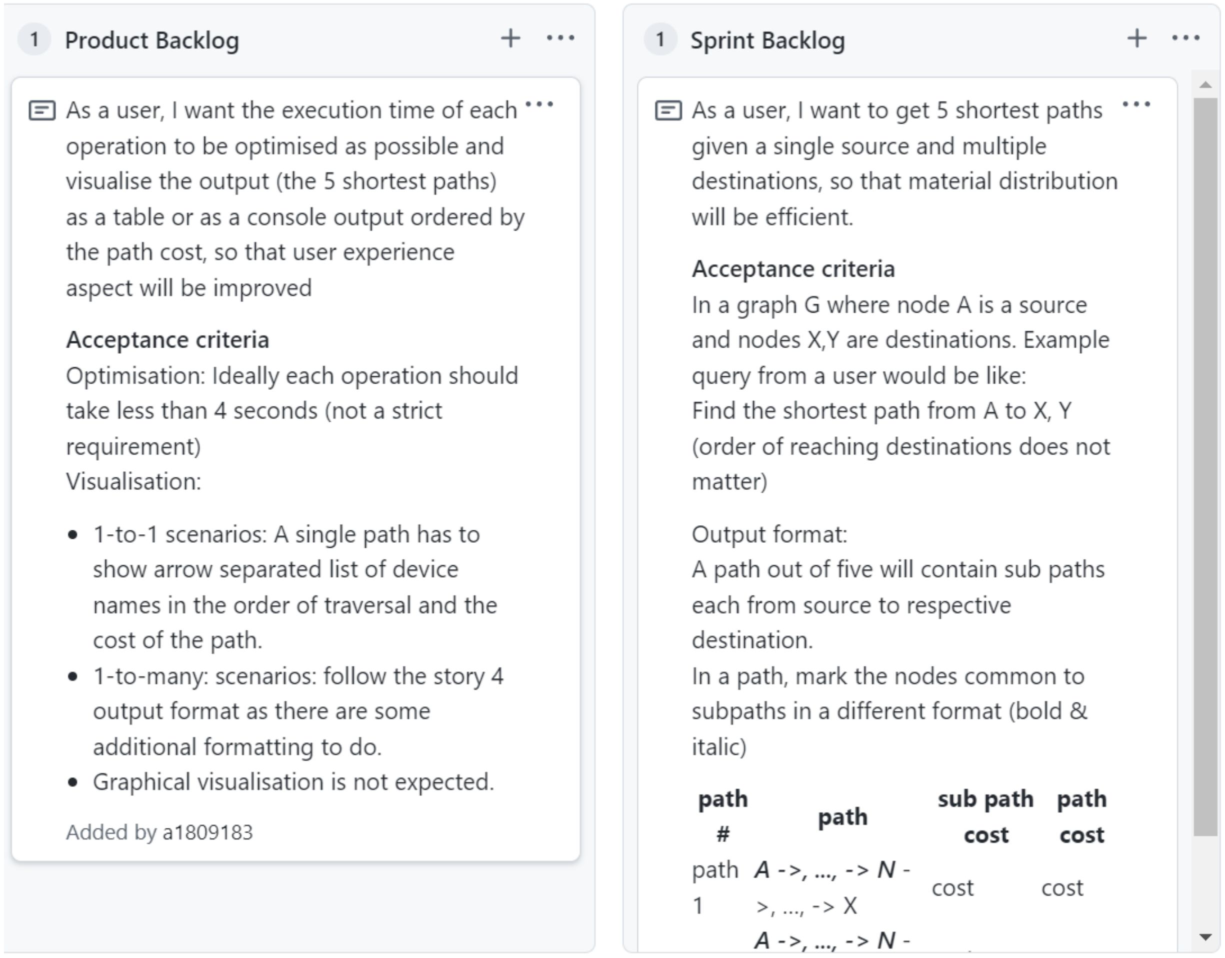
- Functionality has been tested under various scenarios and verified it works
- Necessary documentation and comments in the code have been made

Summary of Changes:

User story 4 has been provided which is to get the 5 shortest path and subpaths for a single source and multiple destinations. We have managed to modify our database schema through alteration query to our existing database to add cost and is_source, is_destination Boolean attributes. This is an important addition due to requirement changes. Besides that, we have also started testing the recursive search function with the graph given by the product owner. A few problems arise when trying to test it with our recursive search functions. The main problem is that the search function will crash when a loop in the graph/map has been encountered. Therefore, Dijistra's algorithm has been implemented but was only able to return one shortest path. More work has to be done to fully implement the algorithm to our user story.

Snapshot 4.2:

Product Backlog and Task Board:



1 Product Backlog

As a user, I want the execution time of each operation to be optimised as possible and visualise the output (the 5 shortest paths) as a table or as a console output ordered by the path cost, so that user experience aspect will be improved

Acceptance criteria
Optimisation: Ideally each operation should take less than 4 seconds (not a strict requirement)
Visualisation:

- 1-to-1 scenarios: A single path has to show arrow separated list of device names in the order of traversal and the cost of the path.
- 1-to-many: scenarios: follow the story 4 output format as there are some additional formatting to do.
- Graphical visualisation is not expected.

Added by a1809183

1 Sprint Backlog

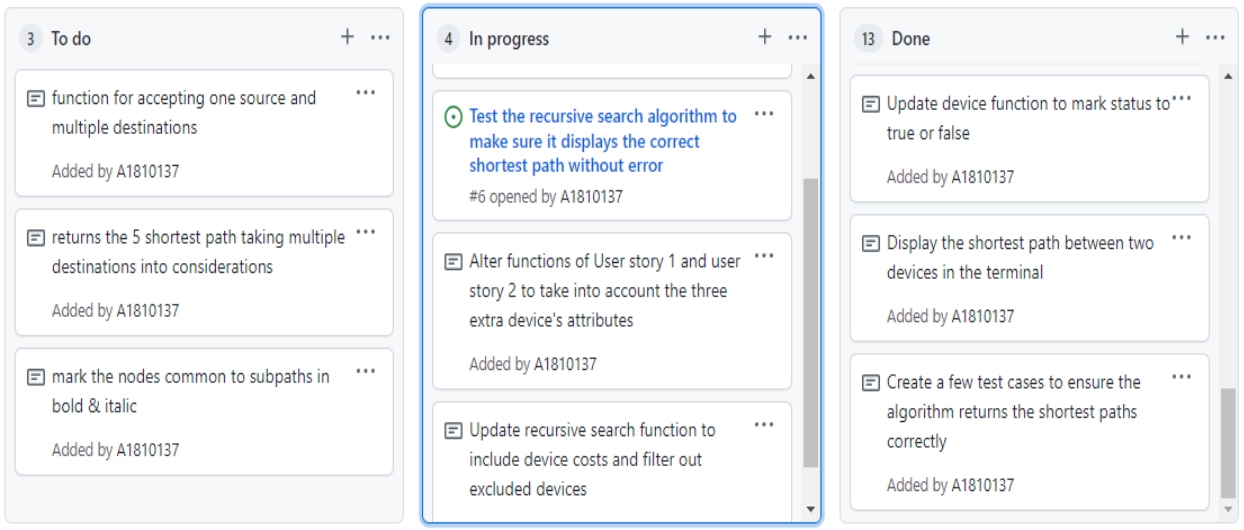
As a user, I want to get 5 shortest paths given a single source and multiple destinations, so that material distribution will be efficient.

Acceptance criteria
In a graph G where node A is a source and nodes X,Y are destinations. Example query from a user would be like:
Find the shortest path from A to X, Y (order of reaching destinations does not matter)

Output format:
A path out of five will contain sub paths each from source to respective destination.
In a path, mark the nodes common to subpaths in a different format (bold & italic)

path #	path	sub path cost	path cost
path 1	*A* ->, ..., -> *N* ->, ..., -> X	cost	cost
	A ->, ..., -> *N* -		

The Task Board


3 To do

function for accepting one source and multiple destinations
Added by A1810137

returns the 5 shortest path taking multiple destinations into considerations
Added by A1810137

mark the nodes common to subpaths in bold & italic
Added by A1810137

4 In progress

Test the recursive search algorithm to make sure it displays the correct shortest path without error
#6 opened by A1810137

Alter functions of User story 1 and user story 2 to take into account the three extra device's attributes
Added by A1810137

Update recursive search function to include device costs and filter out excluded devices

13 Done

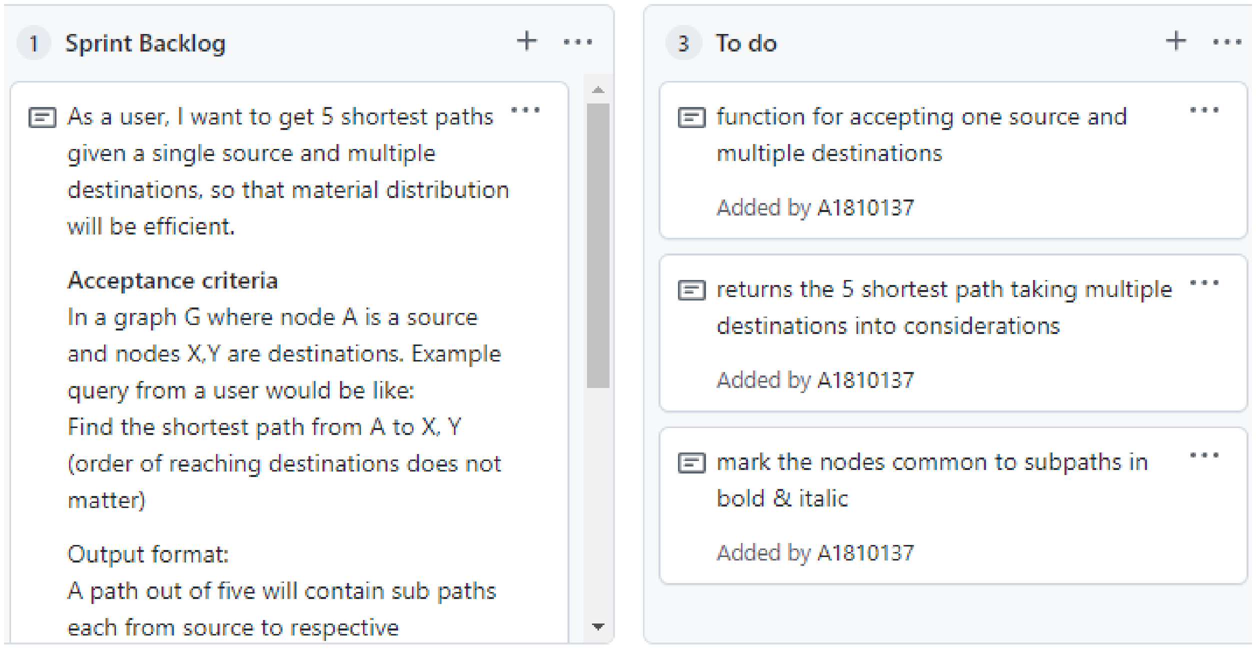
Update device function to mark status to true or false
Added by A1810137

Display the shortest path between two devices in the terminal
Added by A1810137

Create a few test cases to ensure the algorithm returns the shortest paths correctly
Added by A1810137

Sprint Backlog and User Stories:

Sprint backlog


1 Sprint Backlog

As a user, I want to get 5 shortest paths given a single source and multiple destinations, so that material distribution will be efficient.

Acceptance criteria
In a graph G where node A is a source and nodes X,Y are destinations. Example query from a user would be like:
Find the shortest path from A to X, Y (order of reaching destinations does not matter)

Output format:
A path out of five will contain sub paths each from source to respective

3 To do

function for accepting one source and multiple destinations
Added by A1810137

returns the 5 shortest path taking multiple destinations into considerations
Added by A1810137

mark the nodes common to subpaths in bold & italic
Added by A1810137

Definition of Done

The project must fit the following criteria for the current user story:

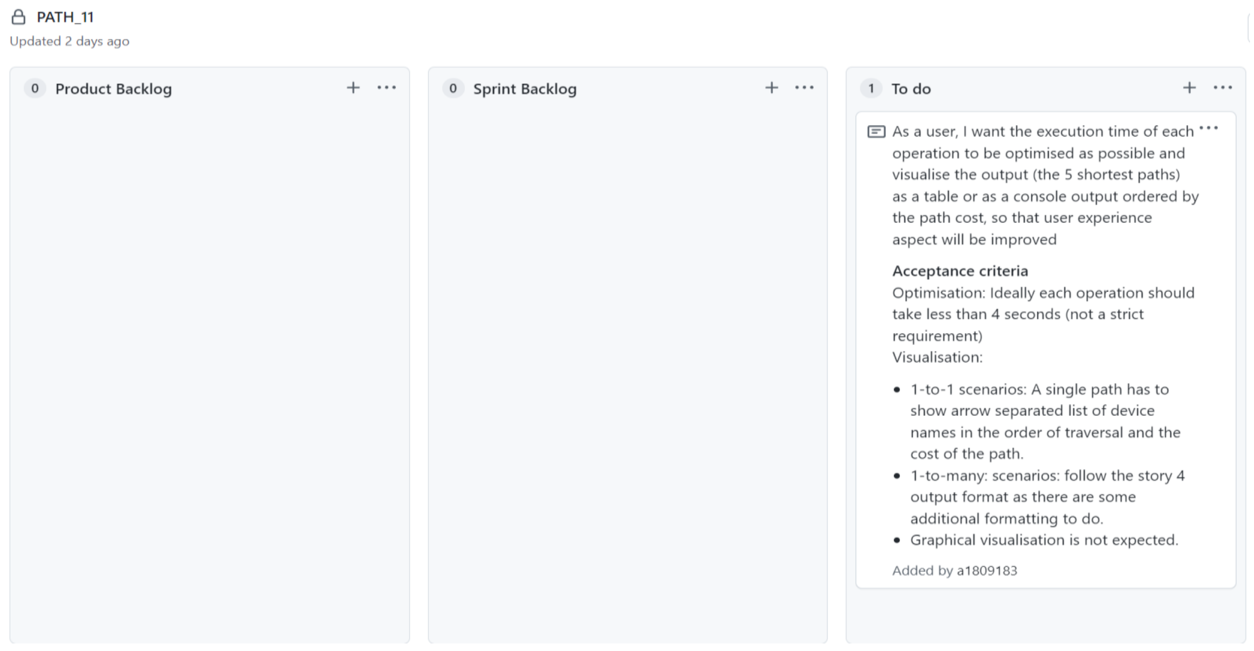
- The recursive search algorithm is able to take in one source and multiple destinations
- Returns five shortest path for multiple considerations
- Each returned shortest paths will have subpaths corresponding to each destination
- Mark the nodes common to subpaths in bold and italic
- Functionality has been tested under various scenarios and verified it works
- Necessary documentation and comments in the code have been made

Summary of Changes:

The problem with the recursive search function to tackle loops in the graphs has been solved. It is now able to run on small test cases. However, another issue arises when implementing the algorithm in the full graph nodes given by the product owner. There are over 100 nodes and since it is an exhaustive search function, the algorithm quickly hits the limit of available resources in MSSQL. This problem needs to be solved by ensuring a more optimal way of storing data. The Dijkstra's algorithm can solve this problem but there are still several limitations. We still have not made it return more than one shortest path due to limited technical knowledge. The Dijkstra algorithm also only accounts for fixed device and path cost. Due to these issues, we have not managed to start on User Story 4. However, we have worked our way to increase the user experience by improving the functions so that they take in device names as inputs instead of device ids.

Snapshot 5.1:

Product Backlog and Task Board:


PATH_11
Updated 2 days ago

0 Product Backlog

0 Sprint Backlog

1 To do

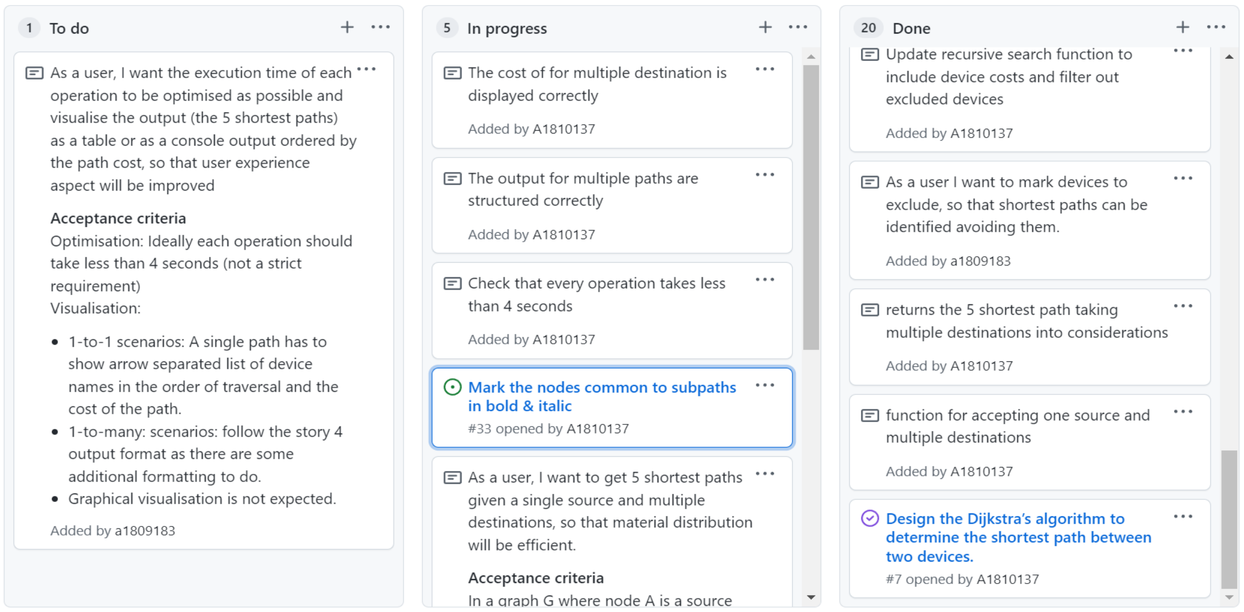
As a user, I want the execution time of each operation to be optimised as possible and visualise the output (the 5 shortest paths) as a table or as a console output ordered by the path cost, so that user experience aspect will be improved

Acceptance criteria
Optimisation: Ideally each operation should take less than 4 seconds (not a strict requirement)
Visualisation:

- 1-to-1 scenarios: A single path has to show arrow separated list of device names in the order of traversal and the cost of the path.
- 1-to-many: scenarios: follow the story 4 output format as there are some additional formatting to do.
- Graphical visualisation is not expected.

Added by a1809183

The Task Board



1 To do

As a user, I want the execution time of each operation to be optimised as possible and visualise the output (the 5 shortest paths) as a table or as a console output ordered by the path cost, so that user experience aspect will be improved

Acceptance criteria
Optimisation: Ideally each operation should take less than 4 seconds (not a strict requirement)
Visualisation:

- 1-to-1 scenarios: A single path has to show arrow separated list of device names in the order of traversal and the cost of the path.
- 1-to-many: scenarios: follow the story 4 output format as there are some additional formatting to do.
- Graphical visualisation is not expected.

Added by a1809183

5 In progress

The cost of for multiple destination is displayed correctly
Added by A1810137

The output for multiple paths are structured correctly
Added by A1810137

Check that every operation takes less than 4 seconds
Added by A1810137

Mark the nodes common to subpaths in bold & italic
#33 opened by A1810137

As a user, I want to get 5 shortest paths given a single source and multiple destinations, so that material distribution will be efficient.

Acceptance criteria
In a graph G where node A is a source

20 Done

Update recursive search function to include device costs and filter out excluded devices
Added by A1810137

As a user I want to mark devices to exclude, so that shortest paths can be identified avoiding them.
Added by a1809183

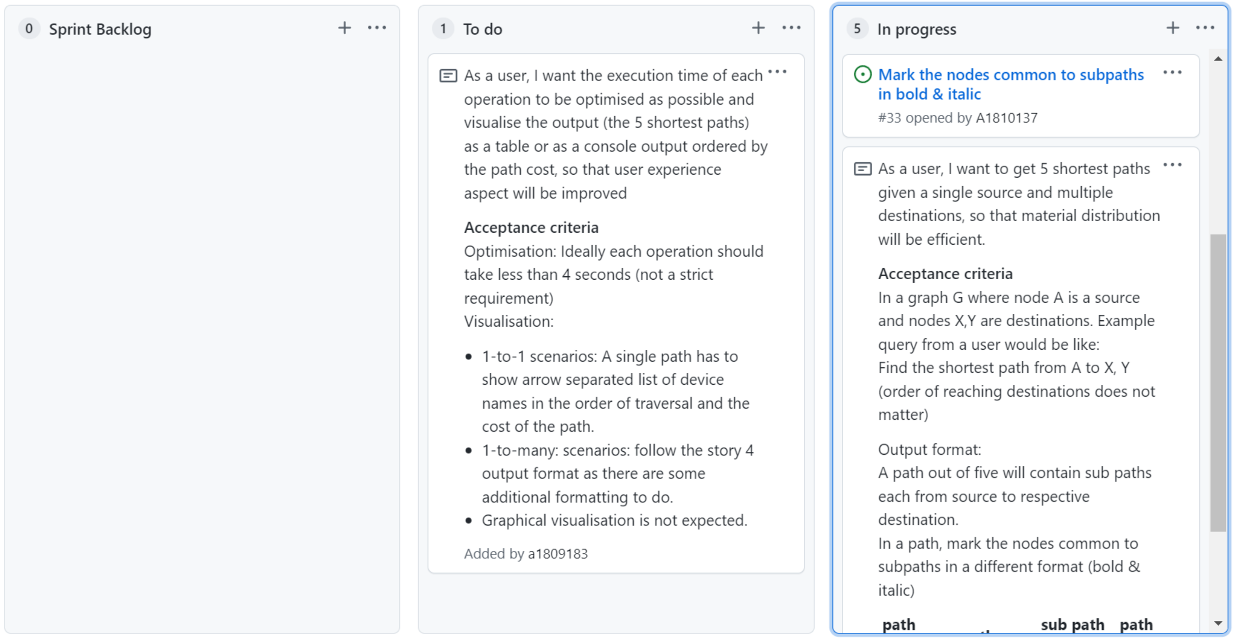
returns the 5 shortest path taking multiple destinations into considerations
Added by A1810137

function for accepting one source and multiple destinations
Added by A1810137

Design the Dijkstra's algorithm to determine the shortest path between two devices.
#7 opened by A1810137

Sprint Backlog and User Stories:

Sprint backlog



0 Sprint Backlog

1 To do

As a user, I want the execution time of each operation to be optimised as possible and visualise the output (the 5 shortest paths) as a table or as a console output ordered by the path cost, so that user experience aspect will be improved

Acceptance criteria
Optimisation: Ideally each operation should take less than 4 seconds (not a strict requirement)
Visualisation:

- 1-to-1 scenarios: A single path has to show arrow separated list of device names in the order of traversal and the cost of the path.
- 1-to-many: scenarios: follow the story 4 output format as there are some additional formatting to do.
- Graphical visualisation is not expected.

Added by a1809183

5 In progress

Mark the nodes common to subpaths in bold & italic
#33 opened by A1810137

As a user, I want to get 5 shortest paths given a single source and multiple destinations, so that material distribution will be efficient.

Acceptance criteria
In a graph G where node A is a source and nodes X,Y are destinations. Example query from a user would be like:
Find the shortest path from A to X, Y (order of reaching destinations does not matter)

Output format:
A path out of five will contain sub paths each from source to respective destination.
In a path, mark the nodes common to subpaths in a different format (bold & italic)

path **sub path** **path**

Definition of Done

The project must fit the following criteria for the current user story:

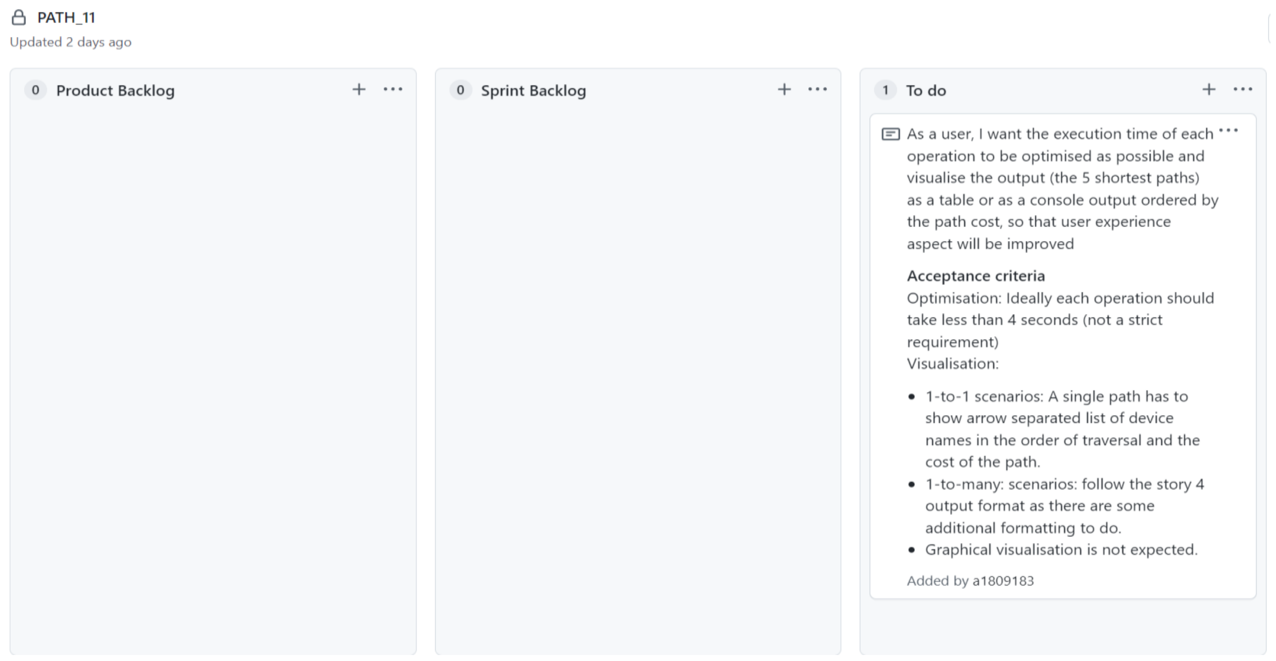
- The search algorithm for all scenarios runs under 4 seconds
- Functions takes device names as inputs instead of id
- The shortest paths displayed in a table
- The path has to show arrow separated list of device names in the order of traversal
- Cost displayed beside the paths
- Functionality has been tested under various scenarios and verified it works
- Necessary documentation and comments in the code have been made

Summary of Changes:

Dijkstra's algorithm has been successfully implemented to replace the limited recursive search algorithm. The Dijkstra algorithm was able to return the five shortest paths for the full graphs given by the product owner in a short period of time. The algorithm also considers the status of the devices when returning the shortest paths. Most of the functions from user story 1 to 3 has been modified to take in device's names instead of id to run. These modifications have improved the user experience, making it easier to run the functions. We have also started implementing user story 4 to take in multiple destinations. We managed to make it work for certain paths. The costs of the paths still have to be considered.

Snapshot 5.2:

Product Backlog and Task Board:



As a user, I want the execution time of each operation to be optimised as possible and visualise the output (the 5 shortest paths) as a table or as a console output ordered by the path cost, so that user experience aspect will be improved

Acceptance criteria

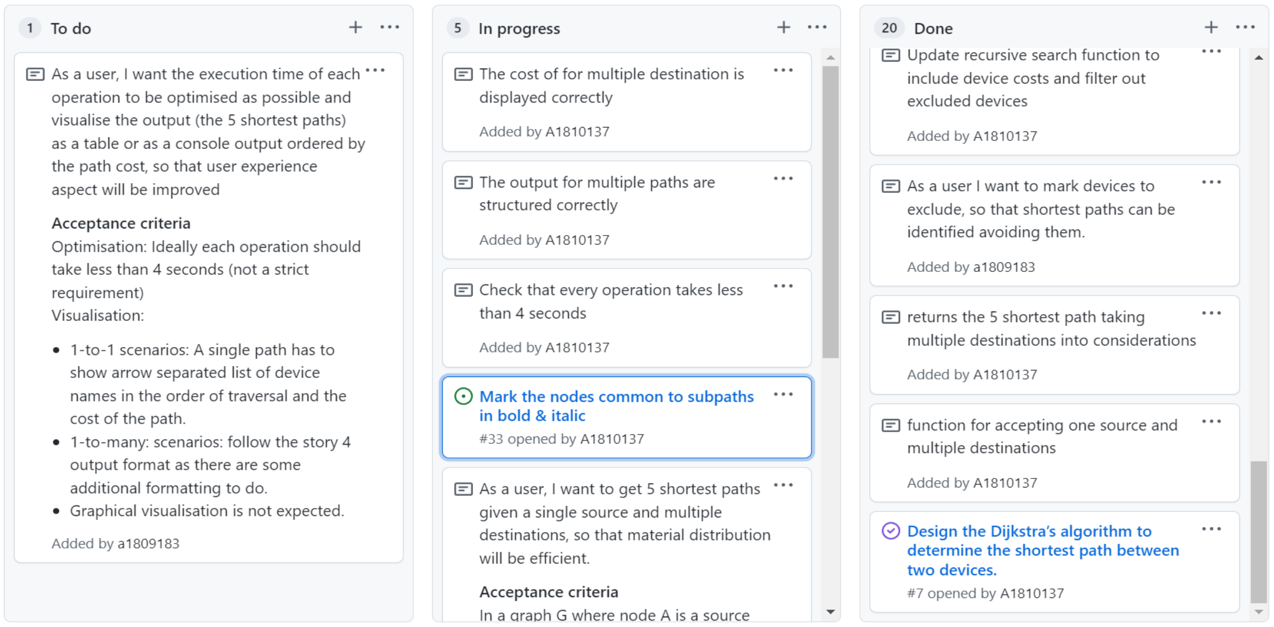
Optimisation: Ideally each operation should take less than 4 seconds (not a strict requirement)

Visualisation:

- 1-to-1 scenarios: A single path has to show arrow separated list of device names in the order of traversal and the cost of the path.
- 1-to-many: scenarios: follow the story 4 output format as there are some additional formatting to do.
- Graphical visualisation is not expected.

Added by a1809183

The Task Board



1 To do

As a user, I want the execution time of each operation to be optimised as possible and visualise the output (the 5 shortest paths) as a table or as a console output ordered by the path cost, so that user experience aspect will be improved

Acceptance criteria
Optimisation: Ideally each operation should take less than 4 seconds (not a strict requirement)
Visualisation:

- 1-to-1 scenarios: A single path has to show arrow separated list of device names in the order of traversal and the cost of the path.
- 1-to-many: scenarios: follow the story 4 output format as there are some additional formatting to do.
- Graphical visualisation is not expected.

Added by a1809183

5 In progress

The cost of for multiple destination is displayed correctly

Added by A1810137

The output for multiple paths are structured correctly

Added by A1810137

Check that every operation takes less than 4 seconds

Added by A1810137

Mark the nodes common to subpaths in bold & italic
#33 opened by A1810137

As a user, I want to get 5 shortest paths given a single source and multiple destinations, so that material distribution will be efficient.

Acceptance criteria
In a graph G where node A is a source

20 Done

Update recursive search function to include device costs and filter out excluded devices

Added by A1810137

As a user I want to mark devices to exclude, so that shortest paths can be identified avoiding them.

Added by a1809183

returns the 5 shortest path taking multiple destinations into considerations

Added by A1810137

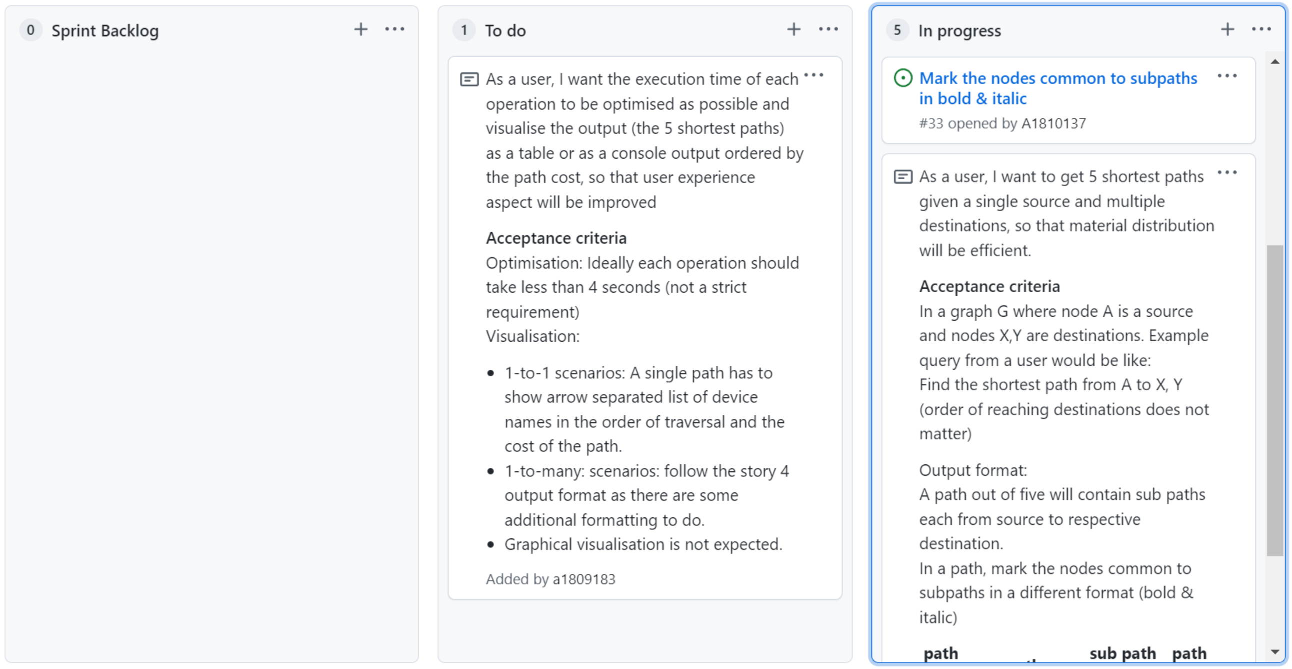
function for accepting one source and multiple destinations

Added by A1810137

Design the Dijkstra's algorithm to determine the shortest path between two devices.
#7 opened by A1810137

Sprint Backlog and User Stories:

Sprint backlog



0 Sprint Backlog

1 To do

As a user, I want the execution time of each operation to be optimised as possible and visualise the output (the 5 shortest paths) as a table or as a console output ordered by the path cost, so that user experience aspect will be improved

Acceptance criteria
Optimisation: Ideally each operation should take less than 4 seconds (not a strict requirement)
Visualisation:

- 1-to-1 scenarios: A single path has to show arrow separated list of device names in the order of traversal and the cost of the path.
- 1-to-many: scenarios: follow the story 4 output format as there are some additional formatting to do.
- Graphical visualisation is not expected.

Added by a1809183

5 In progress

Mark the nodes common to subpaths in bold & italic
#33 opened by A1810137

As a user, I want to get 5 shortest paths given a single source and multiple destinations, so that material distribution will be efficient.

Acceptance criteria
In a graph G where node A is a source and nodes X,Y are destinations. Example query from a user would be like:
Find the shortest path from A to X, Y (order of reaching destinations does not matter)

Output format:
A path out of five will contain sub paths each from source to respective destination.
In a path, mark the nodes common to subpaths in a different format (bold & italic)

path **sub path** **path**

Definition of Done

The project must fit the following criteria for the current user story:

- The search algorithm for all scenarios runs under 4 seconds
- Functions takes device names as inputs instead of id
- The shortest paths displayed in a table
- The path has to show arrow separated list of device names in the order of traversal
- Cost displayed beside the paths
- Functionality has been tested under various scenarios and verified it works
- Necessary documentation and comments in the code have been made

Summary of Changes:

The tasks from user story 4 are almost done. We have successfully returned 5 shortest paths from multiple destinations. The only task left from user story 4 was to bold common paths of the destinations, but this proved to be challenging to implement in MSSQL. User story 5 is all about optimization. Since switching to the Dijkstra's algorithm, the search ran smoothly and fast even with a larger dataset. So, it fulfilled the criteria to run under 4 seconds. Some stored procedures were slightly modified to be more user friendly especially when inputting commands. We made sure that commands take in device names as inputs instead of device ids. This change made running the commands much easier which significantly enhanced the user experience. Some tasks from user story 5 have been completed such as the visual output since it has been implemented correctly from the beginning stages of our sprint.