THE UNIVERSITY of ADELAIDE

Eng 3006 Software Engineering & Project

Shortest Path Algorithm for Material Transportation

Final Report of Group Path 12

Author: Adam Rebes (a1793912)

November 2023

Project Vision Reflection

Looking back on the original project vision, the final result of the project achieved the core essence of what we sought out to do. The main goal set forth by the project vision was to determine, for a given source and destination, the five shortest paths given a graph of devices and conveyors representing a factory layout. We achieved this goal in both our MSSQL implementation as well as our python implementation, with both allowing the user to construct the factory layout using a given excel or csv file and decide on a source and destination which the five shortest paths are then returned for. Furthermore, another goal in the vision was to have the ability to avoid devices deemed in use or unavailable. This was also among what was achieved in the final project, with the algorithm in our MSSQL implementation ignoring paths that would take it through a node marked as inUse. Adding, removing, and updating devices is also something set forth in the project vision which was addressed in our MSSQL implementation using stored procedures which allow devices and conveyors to be created, removed, or alter attributes such as cost and inUse. Though there were user stories we ultimately didn't achieve such as having shortest paths calculated for x destinations these were enhancements that were separate from the original project vision, so ultimately the project vision was adequately fulfilled.

Customer Q&A Reflection

One of the most important questions asked during a client meeting was regarding the way our program was to take input. We were unsure whether the user expected to manually build the database from scratch and then connect to it to run our scripts, whether they wanted us to create the input interface in a different programming language and then pipe the input into an MSSQL database, or something else entirely. When asked how we needed to handle input we were told that we would be given an excel file with the data. It wasn't initially clear to us whether we were to import the excel file as is or if it was just a reference to how the data would look like. We asked many follow-up questions about this, some in subsequent meetings and were told that importing an excel file is fine for the user input, and that what was most important is that the program can work with a database on a SQL Server.

Another important question that we asked was about traversal costs. Initially when we asked about traversal costs we were told that everything had a cost of 1. We inquired further, asking if it was both nodes and edges that had a cost of 1 to which we were told that only edges have a cost. In later meetings this changed, and we were told that nodes and edges would both have a cost and later on were also told that the costs would eventually be a value instead of just 1. When we noticed there was no cost in the excel data file given, we were told we can add that ourselves and assume there would be a cost given in the final scenario.

Later on in the project, once we were clearer on input and had started work on the shortest paths algorithm and had shown the first demo, we asked some questions about what was expected of our algorithm. We had initially been using the shortest path function included in MSSQL and asked if that was satisfactory and what we should be working towards with each

demo. The response to this was that it's fine to use any of the tools available to us in MSSQL and that the most important thing to work towards was to have the program return 5 shortest paths between one source and destination, and that accommodating single source and multiple destinations, as well as optimizing runtime, is of lower priority and should be done after.

The client meetings often went very well and were good for getting direct answers to questions we had. We were always able to show all of our demo and we got really useful feedback on it which we were able to use to decide what to work on in subsequent sprints, and each time we came out of a meeting we had a better grasp of what the project should look like.

However, I found that during the client meetings it could be difficult to think of all the questions that we wanted to ask and that sometimes we wouldn't remember them until after the meeting was over and we're discussing our sprint with the group. Something we could have done better in this regard would have been to start the zoom meeting earlier before it was time to meet with the product owner. In this time, we could have brainstormed and wrote down the questions we wanted to ask instead of just trying to remember them in the meeting. We also should have taken notes during the meeting, of the answers to questions, and uploaded them to a communication channel.

In addition to this, I think that only having the meetings every two weeks could sometimes be too long of a gap for questions which come up afterwards as a result of the meeting and then require waiting until the next meeting to address. It would have been better, especially during some of the busier sprints, to get the chance to talk to the product owner every week, even if it would have been a shorter meeting we'd be able to still ask many questions by having a shorter demo or skipping it for those meetings.

Users and User Stories

It is expected that there will be three key users of this product.

1. The Administrator

   The Administrator is the user first involved with the product. They are responsible for having accurate information regarding the devices within the factory and how they connect, and their first action is building the database containing this data and ensuring that it is integrated with the product so that it can be accessed by later users. During the regular operation of the system, it is The Administrator's responsibility to ensure the database reflects the devices in the factory, and so their actions also include adding, removing, or updating devices, or marking them for exclusion as required.

   Two user stories fundamental to The Administrator are "As a user, I want to store devices in the system so that I can add/remove/update them," and "As a user I want to mark devices to exclude, so that shortest paths can be identified avoiding them." The first is essential, as there must be a way for The Administrator, with a reasonable degree of ease, to integrate the initial database with the program such that it can later be altered

by adding, removing, or updating devices, as reflects any updates to the factory's layout. Furthermore, the second story is crucial to The Administrator as there must be a way to mark devices for exclusion in events such as one temporarily breaking down, and thus not being able to be used, though not being removed from the system permanently. Without the functionality of these user stories, The Administrator would not be able to provide later users with an accurate representation of the devices in the factory, and thus it would be significantly more difficult to calculate accurate shortest paths.

2. The Manager

   The Manager is the person responsible for interacting with the product in the event that shortest paths must be determined. This may be when new material is brought to the factory, or when the factory has recently updated its layout. The Manager's role includes actions such as inputting the correct source and destinations into the program, and running the program to determine the five shortest paths. From these, The Manager is then responsible for deciding what the one best path is, and communicating this to the factory workers.

   User stories fundamental to The Manager are "As a user, I want to get 5 shortest paths given a single source and multiple destinations, so that material distribution will be efficient," and "As a user, I want the execution time of each operation to be as 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 the user experience aspect will be improved." The first story is imperative to this user, as they must be able to give the program the desired source and destinations, and the program must then output the five shortest paths. Without this functionality, the product is useless to The Manager, as this story reflects almost their entire role. The latter user story is also essential, as an inefficient system would not be able to be used repeatedly in quick succession, as may be needed in an environment such as a factory. Furthermore, The Manager would not be able to analyse the output and communicate the best path to the factory workers if the output was not intuitive, and so this is also fundamental to their user actions.

3. The Factory Worker

   The Factory Worker is a person responsible for physical moving materials around the factory following the most efficient route. They are in communication with The Manager, who informs them of the calculated best path, and they enact this on the factory floor. They also may communicate with The Administrator or The Manager when there are changes to the devices in the factory – either additions, subtractions or alterations.

   The user story fundamental to The Factory Worker is "As a user, I want to get the shortest path between 2 given devices so that material transportation will be efficient." Though finding the five shortest paths is a step towards this, The Factory Worker is only responsible for enacting the one best path for transporting material from a source device

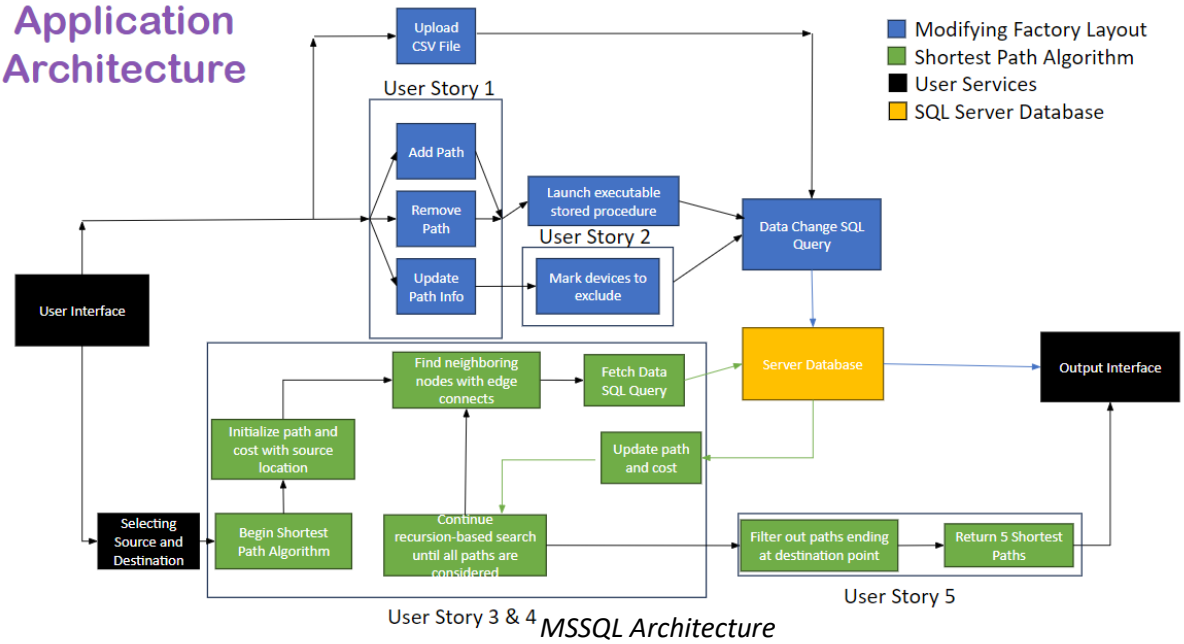
to a destination, and therefore this story is that which is most important, in reflecting the simplified data that is required by The Factory Worker.

As some user stories have become apparent since writing the initial report, there are some differences in the users and their responsibilities. Though there remain roles including the managing of the database and the running of the shortest paths algorithm, these have been better defined as The Administrator and The Manager, and their responsibilities and actions slightly better defined. The most significant difference, however, has been the identification of The Factory Worker as a potential user of the product. Though they may not often interact directly with the program, they are in direct communication with those who do, and are also directly affected by the program's outcomes. Therefore, they are a key user, with a key functionality involved with the single shortest path user story, which provides them with the information needed to physically transport materials efficiently around the factory.

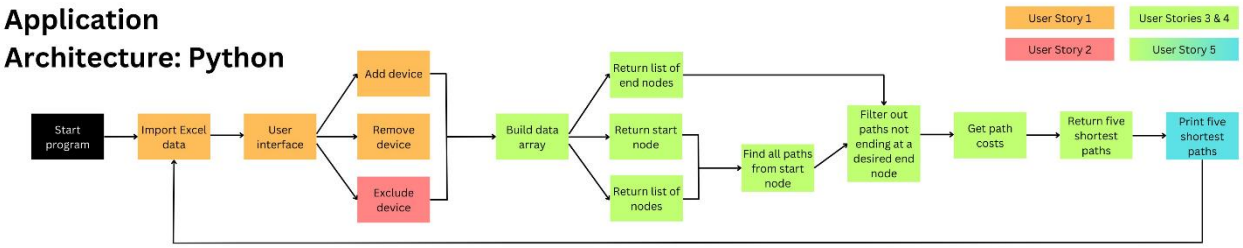
Software Architecture

Below are the software architectures for both implementations of the project. The diagrams use the following numbering of user stories:

1. As a user, I want to store devices in the system so that I can add/remove/update them.
2. As a user I want to mark devices to exclude, so that shortest paths can be identified avoiding them.
3. As a user, I want to get the shortest path between 2 given devices so that material transportation will be efficient.
4. As a user, I want to get 5 shortest paths given a single source and multiple destinations, so that material distribution will be efficient.
5. As a user, I want the execution time of each operation to be as 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 the user experience aspect will be improved.



MSSQL Architecture



Python Architecture

The MSSQL architecture was chosen after considering the needs of the project. Each user story is accounted for and grouped together with related user stories to improve usability. The user has the option to create their factory layout using a csv file which is a crucial part of the first user story. The user story is further addressed by allowing users to add, remove, and update devices and conveyors, which additionally addresses the second user story with the ability to mark devices to exclude. The user also has the option to select a source and destination and then run the shortest paths algorithm, which gives them back 5 shortest paths as specified by user stories 3 & 4, also using additional techniques to address user story 5.

As the python architecture is more of a proof of concept of what the MSSQL managed to achieve, there is a large overlap in the architecture. One of the key differences is the lack of a server database, with the data needing to be imported from an excel file each time.

Tech Stack and Standards

As the requirements for a user interface for this product were minimal, the front-end of the tech stack for the main (SQL) aspect of the product included the Microsoft Server Management Studios environment. For the Python aspect of the product, this front-end was Visual Studio Code. The back-end of the main tech stack included the MSSQL framework for database storage, which was accessed and interacted with by the T-SQL programming language. The python programming language was used with the back-end of the Python aspect of the product, interacting with a database stored in Microsoft Excel.

Tools for communication and development:

- Slack
- Zoom
- Facebook Messenger
- Discord
- GitHub
- Overleaf

Coding Standards:

- Code will be commented such that the functionality is described adequately to ensure an observer might understand code functions.
- Select functions will be split from the main code base to ensure a modular programming design and to reduce code clutter.
- A naming standard or convention will be utilised for variables and functions to ensure the code is understandable.
- Code length will be optimized to ensure program run-time is minimised.
- Functions will have singular purposes to reduce code clutter and will call on other functions to avoid having extra functionality.
- Testing scenarios will be developed prior to code development to ensure that developers will know what scenarios their code will aim to work towards.
- Before pushing code onto the main server, code review will be done to ensure there are no bugs.
- If a bug reaches the main code base, there will be exit conditions on code to avoid stack overflow and other issues.

Microsoft server management studios was used for the front-end in order to take advantage of Microsoft SQL Server (MSSQL) as recommended by the client, and as was outlined in the initial report. One of the benefits of this is the ease it provides in connecting to a server database as well as providing useful tools such as node and edge tables as well as functions such as a shortest path function. The downside of this was that in this environment any programming would need to be done using SQL which is better used as a language for querying a database rather than programming. This also caused confusion in how user input would be given, as the standard way of getting input in programming language doesn't really work either.

In an attempt to try to counteract some of these shortcomings, we also worked on a python implementation which would allow for a familiar way of getting user input, calling functions and conditionals and loops, among other python functionality we were experienced with. However, as we improved at working in the MSSQL environment, we were able to take advantage of functionality such as importing data, the shortest path function provided within and the ability to create stored procedures to make good progress in implementing the program. Along with this, ultimately one of the most important aspects which made working in MSSQL more viable in the end over python was that everything was already integrated with a database while for python it would need additional work to connect to a database and require the use of MySQL queries, defeating a lot of the purpose of using python in the first place. Therefore, our plan from the initial report of using Microsoft Server Management Studios was sound and what ended up working, with the python implementation being unnecessary.

The coding standards we used stayed unchanged from the initial report. Not every bit of code written managed to meet these standards, but whenever it did we found that it largely improved the quality of the code and were worthwhile to keep the same. For example, the detailed commenting of the SQL stored procedures as well as python code was crucial in

these pieces of code being able to be read and utilized by the whole team. This also goes for the use of unambiguous variable names, the use of functions where possible and having shorter code lengths where possible all going a long way in helping the whole team understand as much written code as possible and thus being able to contribute towards code written by others. We weren't able to test prior to developing code as much as we'd hoped but did utilize a large amount of regression testing to find bugs.

Our communication & development tools stayed largely the same as the initial report. Initially we used Facebook messenger for most communication while uploading code changes to GitHub, however, we soon started using discord for most communication including the distribution of code, while still using GitHub to manage the task board and user stories. This worked well as it allowed us to communicate and exchange code in the same place.

Group Meetings and Team Member Roles

With a combination of some group members being able to meet in person, and others joining virtually, the group was able to meet for one hour every week, as well as for the fortnightly twenty-minute sprint retrospective meetings and a half-hour virtual group meeting following each of them. We were not able to successfully timebox these meetings, as we discovered most often the issues that needed the most discussion time could not be determined until all group members were present in the meeting. This meant that had timeboxing been strictly enforced, some issues would not have been resolved, whilst others would have taken more time than necessary, reducing the overall productivity of the meeting. The group could have made more of an effort to timebox as the project progressed and each person had a better understanding of how long an item would need to be discussed for, however, timing was never an issue for our group meetings. The group was able to communicate effectively and efficiently, covering any issue raised by any group member within the timeframe of the meeting. Therefore, time spent organising timeboxing, even if the timeboxing was successful, would have added minimal value to the group meetings, and thus would have been time better spent elsewhere.

Sprint retrospective meetings were scheduled fortnightly on Thursdays from 4:40pm to 5:05pm. Further, Slack was used as an additional feedback channel between the team and the customer. This was useful for when we needed to speak to the customer outside of retrospective meetings, but usually most information was exchanged within the meetings on zoom.

The Scrum Masters for each sprint were as follows:

Week	Sprints	Scrum Master
4	1	Ayman Shaawi
5	2	William Robinson
6	2	Stephanie Turner
7	3	Nelson Peterson
8	3	Addy Dhingra
9	4	Gurvir Singh
10	4	Rukudzo Ndudzo
11	5	Adam Rebes
12	5	Murray Chahl

As early as the first sprint the group had already decided on a time that everyone could attend: 12pm on Wednesdays. The meetings were often productive and usually every member was present with some exceptions where 1 or 2 members have been unable to attend. We didn't always have a meeting agenda ahead of time, the goal being more so to decide what each team member is working on for the current week of the sprint, check in on how members are doing with previously allocated tasks and address any questions anyone has. Even if we couldn't address anything we'd still take the time to at least hear what it's about so that we'd know about it. However, the main priority was that people had things that they were working on until the next meeting.

A downside of this approach was that even though we knew the general tasks that everyone was working on each week, we didn't have time to go in-depth with each of the tasks and so sometimes we may find there was overlap in what we were working on, or someone would run into some trouble and we'd be unable to help until the next meeting and then the next meeting would be spent largely talking about those things.

A solution was that oftentimes we'd split tasks among several people, giving tasks to the people with best suited for the task so that they could communicate and answer each other's questions while working between meetings. This allowed us to continue to be able to spend meeting times getting a general understanding on what everyone is working on rather than in-depth one focusing on less parts of the project. However, this meant that a lot of times people only had a detailed understanding of what they were working on and not much else, making it harder to integrate the work of the entire team.

As one of two people who had prior experience with SQL, in a lot of the meetings in earlier sprints my role was explaining how SQL differed from conventional programming languages and potential ways we may deal with user input, programming aspects such as conditionals, and the way we'd have to interact with a database to determine the shortest paths. When team members presented their ideas, I'd try to explain why that may or may not work in SQL and suggest possible alternatives. In later sprints, I helped in creating and integrating the

SQL scripts we'd created to handle imported data and make node and edge tables with the SQL implementation of shortest paths. I attended every meeting in person and would often help find an available room for the meeting as well as staying after the meeting with other team members to talk more in-detail about what was discussed in the meeting and do further work on the project.

Snapshots

THE UNIVERSITY of ADELAIDE

Eng 3006 Software Engineering & Project

Shortest Path Algorithm for Material Transportation

Snapshot 1.1 of Group Path 12

Authors: William Robinson (a1724943), Ayman Shaawi(a1799924), Gurvir Singh (a1796192), Stephanie Turner(a1833535), Nelson Peterson (a1801266), Adam Rebes (a1793912), Murray Chahl (a1764549), Rukudzo Ndudzo (a1802422), Addy Dhingra (a1803893)

August 2023

Page count: 5

Contents

1 Product Backlog, Task Board Sprint, Backlog and 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.

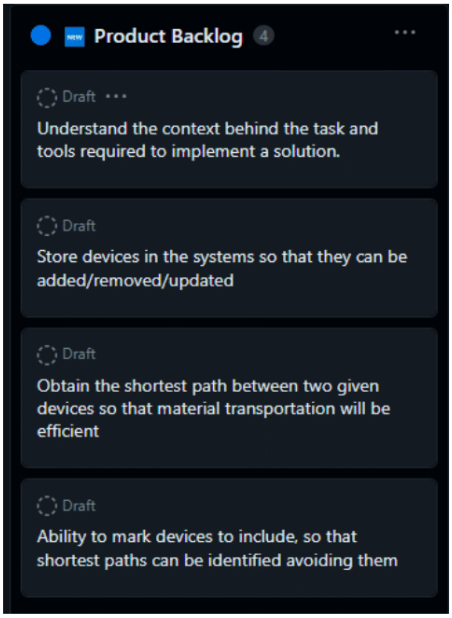


Figure 1: Product Backlog

1.1 Sprint Backlog and User Stories

For the first initial sprint, the first user story to solve is "I want to store devices in the system so that I can add/remove/update them". This story was chosen first as it constructs the initial storage framework of the design which future code will need to reference and be built around. The design of adding and removing nodes will be important for future user stories, as these functions will underpin the rest of the software.

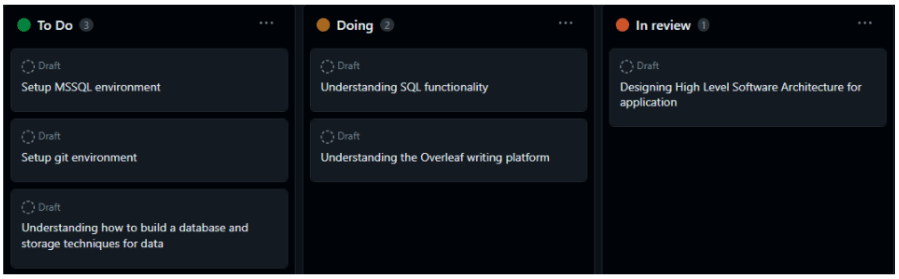


Figure 2: Sprint backlog

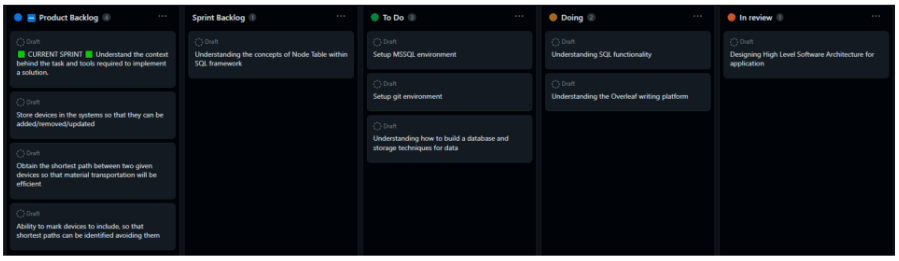


Figure 3: Task Board

The initial sub task involves the current product backlog of understanding the concepts behind the task and researching the tools required to implement the solution. This involves understanding the MSSQL programming environment and how to interact with the database to record and analyse data through the concepts of node tables. Once understood, it will give us the opportunity to store a digital map onto the database which can be used to add, remove, and update paths.

1.2 Definition of Done:

- If code has been written, it must follow the coding standards defined in the initial report.
- If code has been written, it must be reviewed by members to ensure proper functionality and appropriate software design, as well as its relation to the product backlog.
- If code has been written, it must pass localization testing defined by the programmer, automated testing written by the team, and automated regression testing for any future alterations to ensure correct behaviour.
- If code has been written, it must ensure functionality continues to work on build server, and local device.

- The created sprint task must be linked towards the product backlog which in turn must be linked towards the user story.
- Any feedback provided by the client, product owner, and scrum team should be analysed for practicality/feasibility and implemented once deemed beneficial to the user.
- For any implementation of code or system architecture, sufficient documentation is required to justify its relation with the user story, and configuration changes must be well-documented.
- The documentation and code must be deemed acceptable by the criteria given by the product owner, as well as basic software/engineering standards, and be signed off by the product owner.
- Verification and validation of key scenarios are considered to ensure basic functionality before in depth processes are considered for implementation.
- The final product must address the provided user stories to a reasonable standard, being able to at the very least produce a shortest path between two specified devices.

1.3 Summary of changes:

As this is the first snapshot, there are no changes to report since the previous one.

THE UNIVERSITY of ADELAIDE

Eng 3006 Software Engineering & Project

Shortest Path Algorithm for Material Transportation

Snapshot 2.1 of Group Path 12

Authors: William Robinson (a1724943), Ayman Shaawi(a1799924), Gurvir Singh (a1796192), Stephanie Turner(a1833535), Nelson Peterson (a1801266), Adam Rebes (a1793912), Murray Chahl (a1764549), Rukudzo Ndudzo (a1802422), Addy Dhingra (a1803893)

August 2023

Page count: 5

Contents

1 Product Backlog, Task Board Sprint, Backlog and 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.

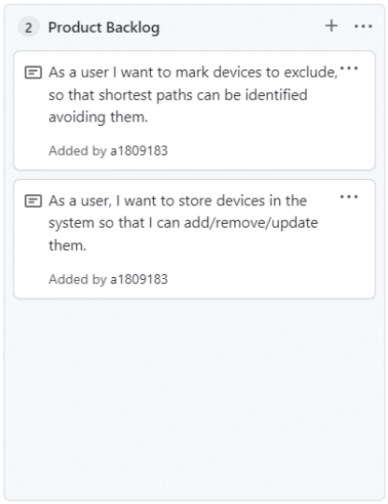


Figure 1: Product Backlog

1.1 Sprint Backlog and User Stories

After much deliberation, the user story had been changed and instead, the story to solve will be “I want to get the shortest path between 2 given devices so that material transportation will be efficient”. We concluded that the previous user story would be much more difficult to implement if we did not have a grasp on the design of the node tables themselves hence the adding and removing would be irrelevant.

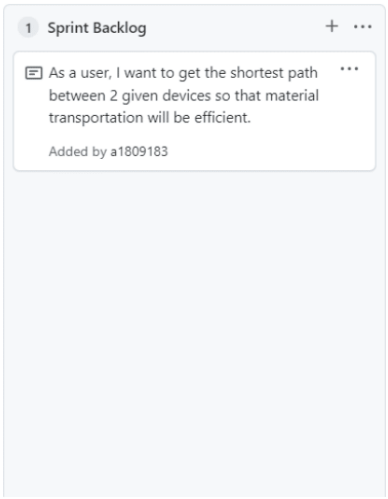


Figure 2: Sprint backlog

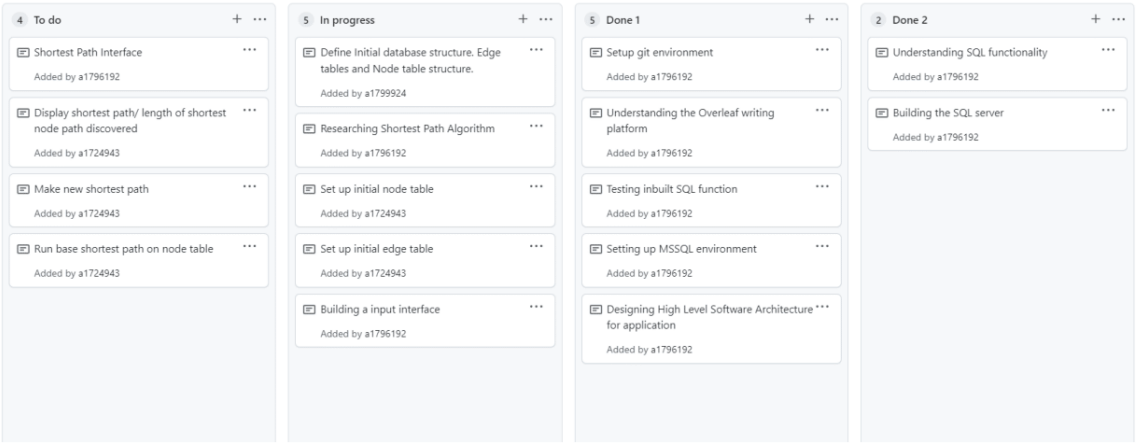


Figure 3: Task Board

The sub tasks for the new user story that we have selected will direct the focus on building an early model of the problem. The user provided a set of data, hence, the team will focus on developing a method to integrate input data into MSSQL. In order to manipulate the given data to achieve the shortest path, the team will develop a method to implement node and edge tables in MSSQL. Finally, the team will delve into researching about the optimal shortest path algorithm to be used for the given problem.

1.2 Definition of Done:

- If code has been written, it must follow the coding standards defined in the initial report.
- If code has been written, it must be reviewed by members to ensure proper functionality and appropriate software design, as well as its relation to the product backlog.
- If code has been written, it must pass localization testing defined by the

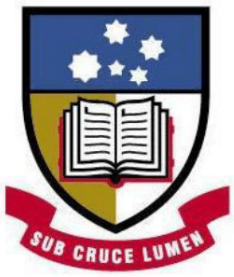
programmer, automated testing written by the team, and automated regression testing for any future alterations to ensure correct behaviour.

- If code has been written, it must ensure functionality continues to work on build server, and local device.

- The created sprint task must be linked towards the product backlog which in turn must be linked towards the user story.
- Any feedback provided by the client, product owner, and scrum team should be analysed for practicality/feasibility and implemented once deemed beneficial to the user.
- For any implementation of code or system architecture, sufficient documentation is required to justify its relation with the user story, and configuration changes must be well-documented.
- The documentation and code must be deemed acceptable by the criteria given by the product owner, as well as basic software/engineering standards, and be signed off by the product owner.
- Verification and validation of key scenarios are considered to ensure basic functionality before in depth processes are considered for implementation.
- The final product must address the provided user stories to a reasonable standard, being able to at the very least produce a shortest path between two specified devices.

1.3 Summary of Changes:

Since the last snapshot, the user story to solve has been changed from “I want to store devices in the system so that I can add/remove/update them” to “I want to get the shortest path between 2 given devices so that material transportation will be efficient”. The group has divided itself into smaller groups handling nodes and edge tables, integrating data into MSSQL, and starting research into the shortest path algorithm.



THE UNIVERSITY of ADELAIDE

Eng 3006 Software Engineering & Project

Shortest Path Algorithm for Material Transportation

Snapshot 2.2 of Group Path 12

Authors: William Robinson (a1724943), Ayman Shaawi(a1799924), Gurvir Singh (a1796192), Stephanie Turner(a1833535), Nelson Peterson (a1801266), Adam Rebes (a1793912), Murray Chahl (a1764549), Rukudzo Ndudzo (a1802422), Addy Dhingra (a1803893)

September 2023

Page count: 5

Contents

1 Product Backlog, Task Board Sprint, Backlog and 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.

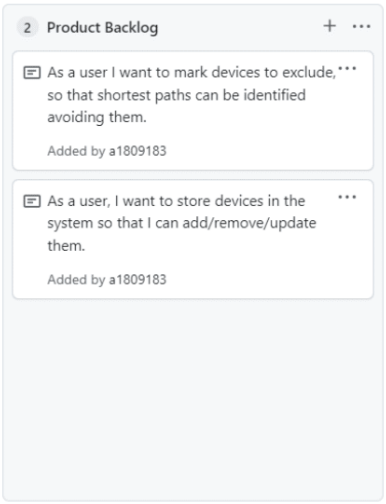


Figure 1: Product Backlog

1.1 Sprint Backlog and User Stories

After much deliberation, the user story had been changed and instead, the story to solve will be “I want to get the shortest path between 2 given devices so that material transportation will be efficient”. We concluded that the previous user story would be much more difficult to implement if we did not have a grasp on the design of the node tables themselves hence the adding and removing would be irrelevant.

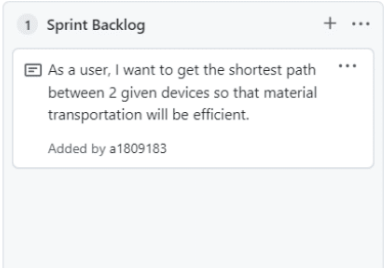


Figure 2: Sprint backlog

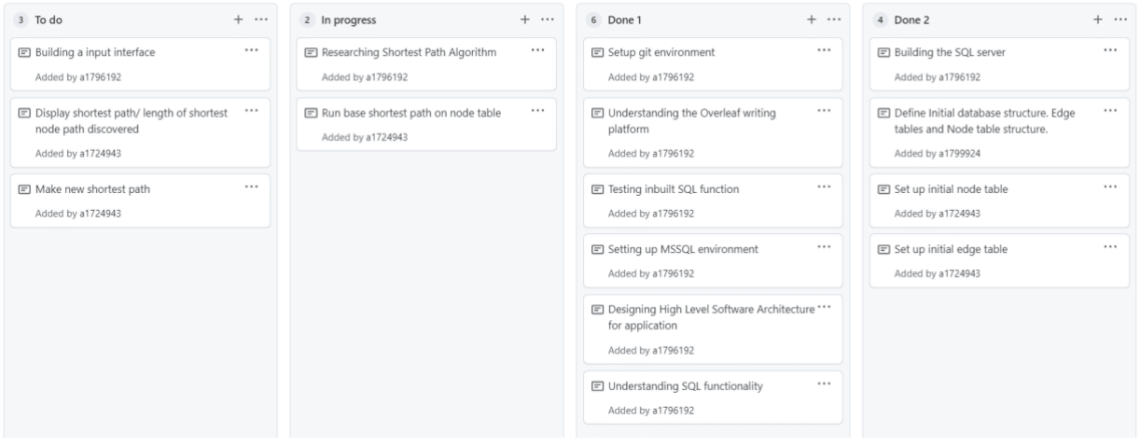


Figure 3: Task Board

The sub tasks for the new user story that we have selected will direct the focus on building an early model of the problem. The user provided a set of data, hence, the team will focus on developing a method to integrate input data into MSSQL. In order to manipulate the given data to achieve the shortest path, the team will develop a method to implement node and edge tables in MSSQL. Currently the selected user story is the shortest path algorithm. The current task is to conduct research on shortest path techniques in SQL and what mathematical equation to implement. Just a simple model needs to be made to read the minimum information required to find the shortest path. The shortest path algorithm must get data from the node and edge table and hence those data platforms must be built.

1.2 Definition of Done:

- If code has been written, it must follow the coding standards defined in the initial report.
- If code has been written, it must be reviewed by members to ensure proper functionality and appropriate software design, as well as its relation to the product backlog.
- If code has been written, it must pass localization testing defined by the programmer, automated testing written by the team, and automated regression testing for any future alterations to ensure correct behavior.
- If code has been written, it must ensure functionality continues to work on build server, and local device.

- The created sprint task must be linked towards the product backlog which in turn must be linked towards the user story.
- Any feedback provided by the client, product owner, and scrum team should be analysed for practicality/feasibility and implemented once deemed beneficial to the user.
- For any implementation of code or system architecture, sufficient documentation is required to justify its relation with the user story, and configuration changes must be well-documented.
- The documentation and code must be deemed acceptable by the criteria given by the product owner, as well as basic software/engineering stan- dards, and be signed off by the product owner.
- Verification and validation of key scenarios are considered to ensure basic functionality before in depth processes are considered for implementation.
- The final product must address the provided user stories to a reasonable standard, being able to at the very least produce a shortest path between two specified devices.

1.3 Summary of Changes:

Since the last snapshot on 28/08/2023, members have been divided into teams of 2 and 3 and are assigned to work on the sprint tasks. Current focus in previous snapshot 2.1 was to focus on research on shortest path algorithms and SQL environments. Currently we have implemented a basic form of shortest path algorithm implemented by the SQL software. Given basic information such as node names and data, the system can find the shortest path. Furthermore, given the example inputs given, we can manipulate the data and filter the data for the desired outputs. However, data could only be changed manually through the server entries and not the SQL framework. We constructed edge and node tables by creating specific csv files with the appropriate headings. This was done to test the initial shortest path algorithm.

THE UNIVERSITY of ADELAIDE

Eng 3006 Software Engineering & Project

Shortest Path Algorithm for Material Transportation

Snapshot 3.1 of Group Path 12

Authors: William Robinson (a1724943), Ayman Shaawi(a1799924), Gurvir Singh (a1796192), Stephanie Turner(a1833535), Nelson Peterson (a1801266), Adam Rebes (a1793912), Murray Chahl (a1764549), Rukudzo Ndudzo (a1802422), Addy Dhingra (a1803893)

September 2023

Page count: 5

Contents

1 Product Backlog, Task Board Sprint, Backlog and 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.

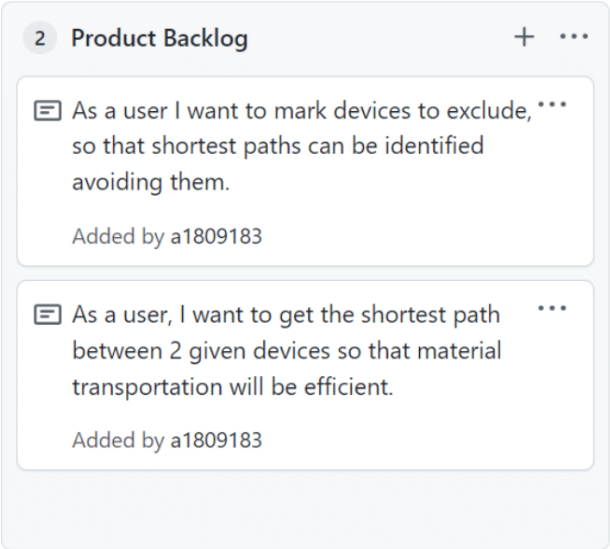


Figure 1: Product Backlog

1.1 Sprint Backlog and User Stories

For the third sprint, the user story to work on will be "As a user, I want to store devices in the system so that I can add/remove/update them." We believe the most effective way to implement this user story will be to build some form of user interface, which is necessary at this point in the project for ease of testing and product demonstration. Thus, this is the logical user story to attempt at this point in time.

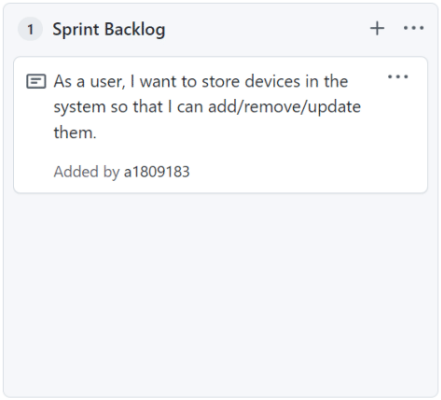


Figure 2: Sprint backlog

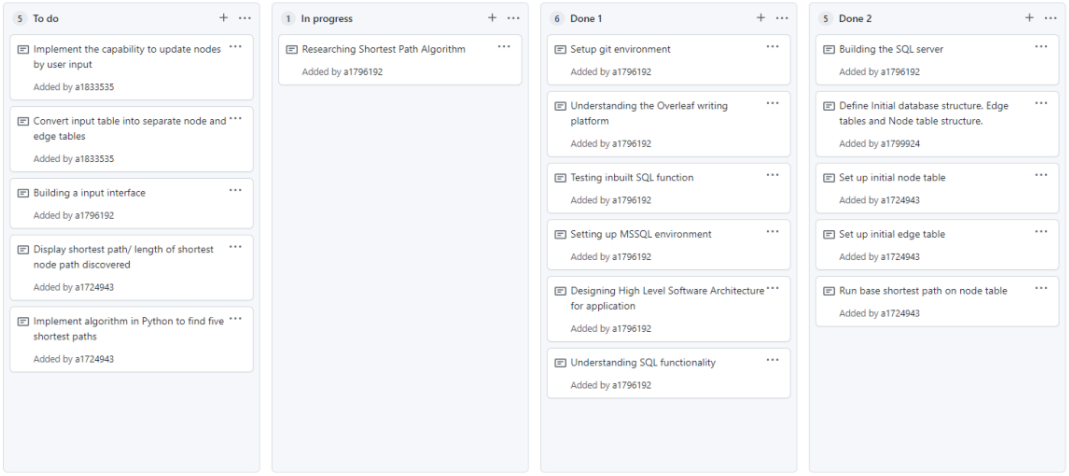


Figure 3: Task Board

The subtasks for this sprint reflect the two-fold approach the team is taking to implement the sprint. Most tasks focus on taking user inputs in SQL and converting these inputs into the formats necessary for operations relevant to shortest path algorithms. Some team members will meanwhile be working in Python to implement an algorithm to return the top five shortest paths, as per the project description. Between these two approaches, there project will have the functionality to take and convert inputs, and return five shortest paths, as required. The team will then, in future sprints, connect these two parts of the project, that they may work as one system.

1.2 Definition of Done:

- If code has been written, it must follow the coding standards defined in the initial report.
- If code has been written, it must be reviewed by members to ensure proper functionality and appropriate software design, as well as its relation to the product backlog.
- If code has been written, it must pass localization testing defined by the programmer, automated testing written by the team, and automated regression testing for any future alterations to ensure correct behaviour.
- If code has been written, it must ensure functionality continues to work on build server, and local device.
- The created sprint task must be linked towards the sprint backlog which in turn must be linked towards the user story.
- Any feedback provided by the client, product owner, and scrum team should be analysed for practicality/feasibility and implemented once deemed beneficial to the user.
- For any implementation of code or system architecture, sufficient documentation is required to justify its relation with the user story, and configuration changes must be well-documented.
- The documentation and code must be deemed acceptable by the criteria given by the product owner, as well as basic software/engineering standards, and be signed off by the product owner.

1.3 Summary of changes:

So far we have successfully created a test for our user story and discovered the shortest path using a MSSQL written program. We have also researched the multiple ways of putting together a shortest path algorithm using the SQL format. This was done for both breadth first search and depth first search algorithms. We expect to be able to put together some tests together for these algorithms in the coming weeks to test using a different language such as python to show proof of concept.

THE UNIVERSITY of ADELAIDE

Eng 3006 Software Engineering & Project

Shortest Path Algorithm for Material Transportation

Snapshot 3.2 of Group Path 12

Authors: William Robinson (a1724943), Ayman Shaawi(a1799924), Gurvir Singh (a1796192), Stephanie Turner(a1833535), Nelson Peterson (a1801266), Adam Rebes (a1793912), Murray Chahl (a1764549), Rukudzo Ndudzo (a1802422), Addy Dhingra (a1803893)

September 2023

Page count: 5

Contents

1 Product Backlog, Task Board Sprint, Backlog and 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.

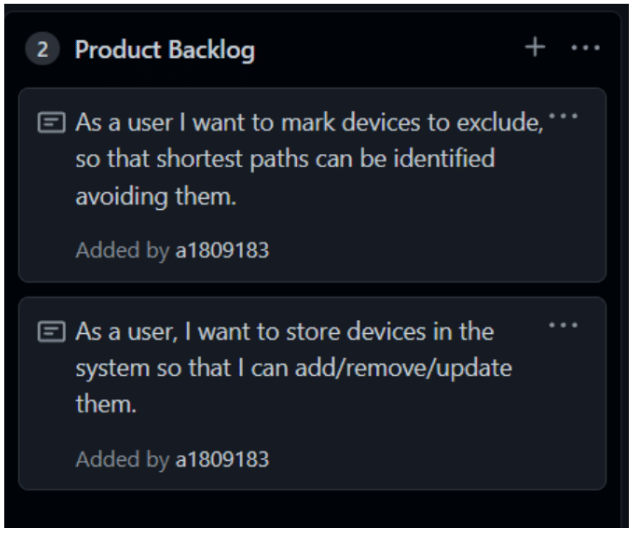


Figure 1: Product Backlog

1.1 Sprint Backlog and User Stories

For the third sprint, the user story to work on will be "As a user, I want to store devices in the system so that I can add/remove/update them." We believe the most effective way to implement this user story will be to build some form of user interface, which is necessary at this point in the project for ease of testing and product demonstration. Thus, this is the logical user story to attempt at this point in time.

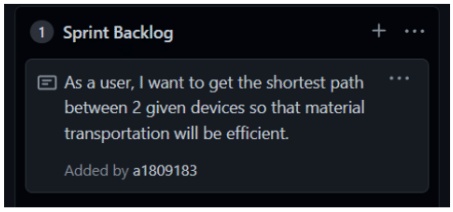


Figure 2: Sprint backlog

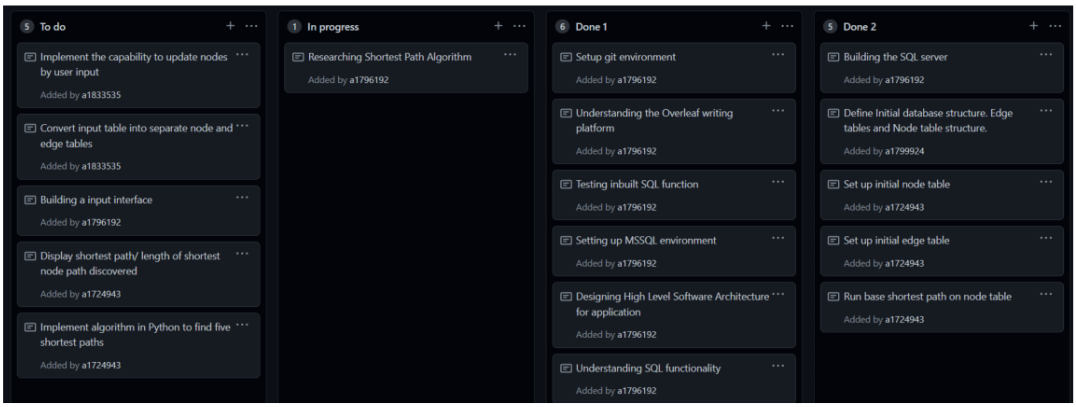


Figure 3: Task Board

The subtasks for this sprint reflect the two-fold approach the team is taking to implement the sprint. Most tasks focus on taking user inputs in SQL and converting these inputs into the formats necessary for operations relevant to shortest path algorithms. Some team members will meanwhile be working in Python to implement an algorithm to return the top five shortest paths, as per the project description. Between these two approaches, there project will have the functionality to take and convert inputs, and return five shortest paths, as required. The team will then, in future sprints, connect these two parts of the project, that they may work as one system.

1.2 Definition of Done:

- If code has been written, it must follow the coding standards defined in the initial report.
- If code has been written, it must be reviewed by members to ensure proper functionality and appropriate software design, as well as its relation to the product backlog.

- If code has been written, it must pass localization testing defined by the programmer, automated testing written by the team, and automated regression testing for any future alterations to ensure correct behaviour.
- If code has been written, it must ensure functionality continues to work on build server, and local device.
- The created sprint task must be linked towards the sprint backlog which in turn must be linked towards the user story.
- Any feedback provided by the client, product owner, and scrum team should be analysed for practicality/feasibility and implemented once deemed beneficial to the user.
- For any implementation of code or system architecture, sufficient documentation is required to justify its relation with the user story, and configuration changes must be well-documented.
- The documentation and code must be deemed acceptable by the criteria given by the product owner, as well as basic software/engineering standards, and be signed off by the product owner.

1.3 Summary of changes:

The group has continued research into potential shortest path algorithms to use, finding one that may be of particular promise with existing implementation examples potentially already present in the SQL language (Yen's algorithm for K-shortest paths). This algorithm will be tested in the coming weeks to determine viability. Work continues on developing and testing an implementation that would allow for a user to add, remove, and update entries in the database, which is a requirement for the project. While the test instance for the shortest path algorithm was indeed able to produce the shortest path between two specified nodes, it lacks capability to find multiple shortest paths, and thus is not suitable for the project. However, it served as a useful proof of concept and exercise in the usage of node/edge tables in MSSQL.

THE UNIVERSITY of ADELAIDE

Eng 3006 Software Engineering & Project

Shortest Path Algorithm for Material Transportation

Snapshot 4.1 of Group Path 12

Authors: William Robinson (a1724943), Ayman Shaawi(a1799924), Gurvir Singh (a1796192), Stephanie Turner(a1833535), Nelson Peterson (a1801266), Adam Rebes (a1793912), Murray Chahl (a1764549), Rukudzo Ndudzo (a1802422), Addy Dhingra (a1803893)

October 2023

Page count: 6

Contents

1. Product Backlog, Task Board Sprint, Backlog and 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.
- As a user, I want to get 5 shortest paths given a single source, and multiple destinations.
- As a user, I want to optimize the performance of the algorithm to produce the shortest paths.

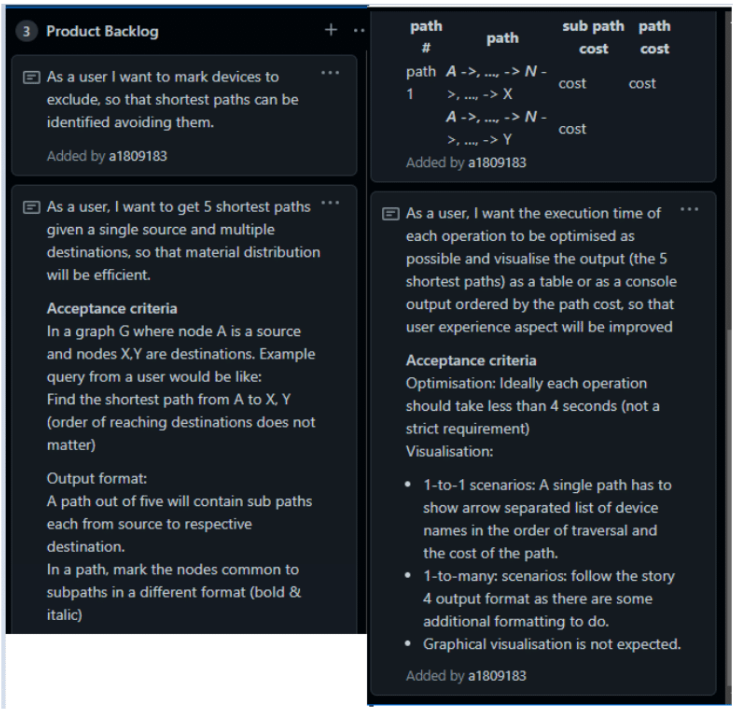


Figure 1: Product Backlog

1.1 Sprint Backlog and User Stories

The user story that the group will dedicate work resources is “As a user, I want to get 5 shortest paths given a single source and multiple destinations so that material distribution will be efficient” as well as fixing the deliverables for the previous user story. The group will dedicate some members to fix the calculation of the cost of the shortest paths to include the cost of the nodes.

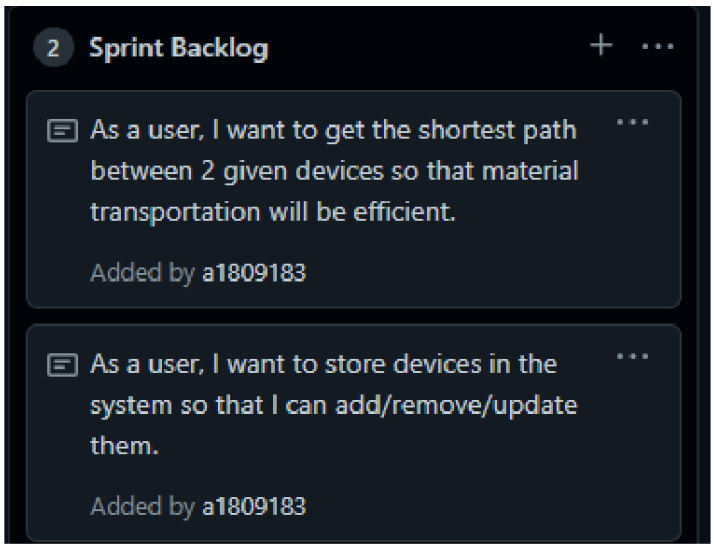


Figure 2: Sprint backlog

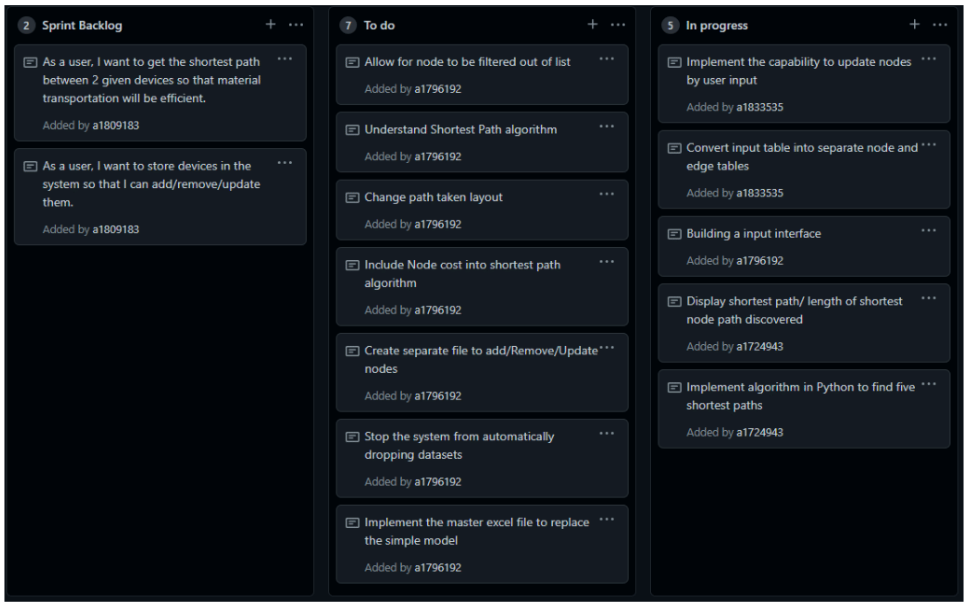


Figure 3: Task Board

The current subtasks include issues that were raised with older deliverables with the purpose of the subtasks to rectify the issues. The subtasks focus on the correct calculation of costs in the **shortest path algorithm**, creating procedures to add and delete **edge/nodes** and decoupling code into modular queries. As shown in Figure 2, our current sprint backlog task revolves around updating, and optimizing existing implementations to fix bugs mentioned by the advisor. The bugs as mentioned include the dataset automatically deletes itself after every execution, node cost is not included in the path algorithm, including the master excel file into the SQL server instead of the existing basic implementation. Future goals are to understand the shortest path algorithm in SQL in detail to be able to start the deliverable mark devices for exclusion.

1.2 Definition of Done:

- If code has been written, it must follow the coding standards defined in the initial report.
- If code has been written, it must be reviewed by members to ensure proper functionality and appropriate software design, as well as its relation to the product backlog.
- If code has been written, it must pass localization testing defined by the programmer, automated testing written by the team, and automated regression testing for any future alterations to ensure correct behavior.
- If code has been written, it must ensure functionality continues to work on the build server, and local device

.

- The created sprint task must be linked towards the product backlog which in turn must be linked towards the user story.
- Any feedback provided by the client, product owner, and scrum team should be analysed for practicality/feasibility and implemented once deemed beneficial to the user.
- For any implementation of code or system architecture, sufficient documentation is required to justify its relation with the user story, and configuration changes must be well-documented.
- The documentation and code must be deemed acceptable by the criteria given by the product owner, as well as basic software/engineering stan- dards, and be signed off by the product owner.
- Verification and validation of key scenarios are considered to ensure basic functionality before in depth processes are considered for implementation.
- The final product must address the provided user stories to a reasonable standard, being able to at the very least produce a shortest path between two specified devices.

1.3 Summary of Changes:

Since the last snapshot in week 8, we have split members to work on deliverables “Add/Remove Path” and “Shortest Path Algorithm”. From a given master list which is made by the user, the SQL code can automatically construct node and edge tables and implement an ID number to such a path and node point. Filtering methods have been used to prevent duplicate edge and node ID to be presented to avoid complications when referencing back to the dataset. The user can now determine which source and destination point the shortest path algorithm uses by naming the points in the respective variable slots. A model of the shortest path algorithm, in both default SQL functionality and custom functionality has been implemented. This current method can find the total cost of the edge path and represent which path was taken. Edge points can be changed in cost where the shortest path algorithm can detect the change and recalculate path.

THE UNIVERSITY of ADELAIDE

Eng 3006 Software Engineering & Project

Shortest Path Algorithm for Material Transportation

Snapshot 4.2 of Group Path 12

Authors: William Robinson (a1724943), Ayman Shaawi(a1799924), Gurvir Singh (a1796192), Stephanie Turner(a1833535), Nelson Peterson (a1801266), Adam Rebes (a1793912), Murray Chahl (a1764549), Rukudzo Ndudzo (a1802422), Addy Dhingra (a1803893)

October 2023

Page count: 5

Contents

1 Product Backlog, Task Board Sprint, Backlog and 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.
- As a user, I want to get 5 shortest paths given a single source, and multiple destinations.
- As a user, I want to optimize the performance of the algorithm to produce the shortest paths

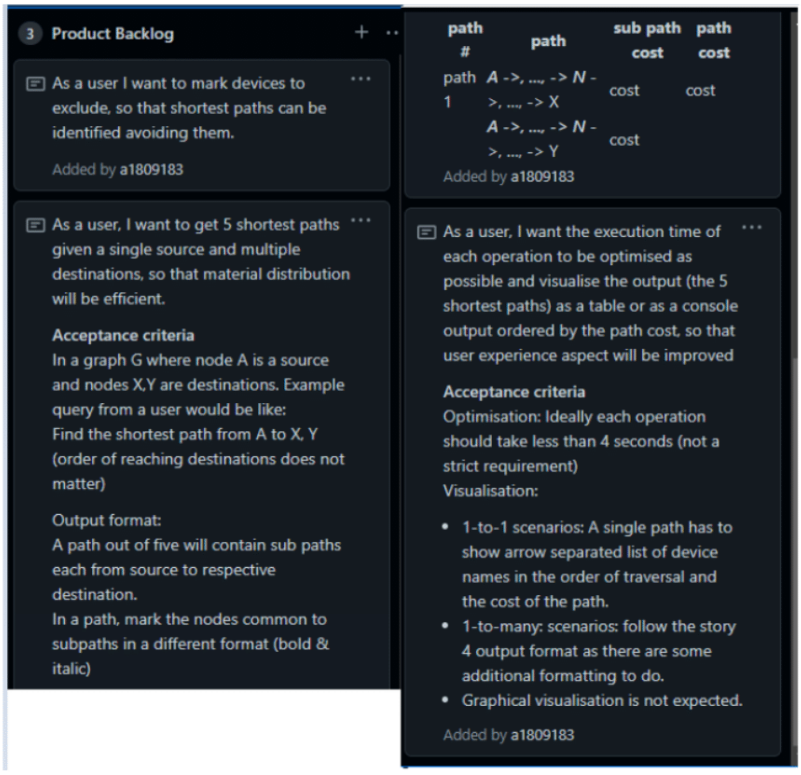


Figure 1: Product Backlog

1.1 Sprint Backlog and User Stories

The user story that the group will dedicate work resources is "As a user, I want to get 5 shortest paths given a single source and multiple destinations so that material distribution will be efficient" as well as fixing the deliverables for the previous user story. The group continues to dedicate some members to fix the calculation of the cost of the shortest paths to include the cost of the nodes. Additionally, some group members are assigned to develop testing strategies and documentation.

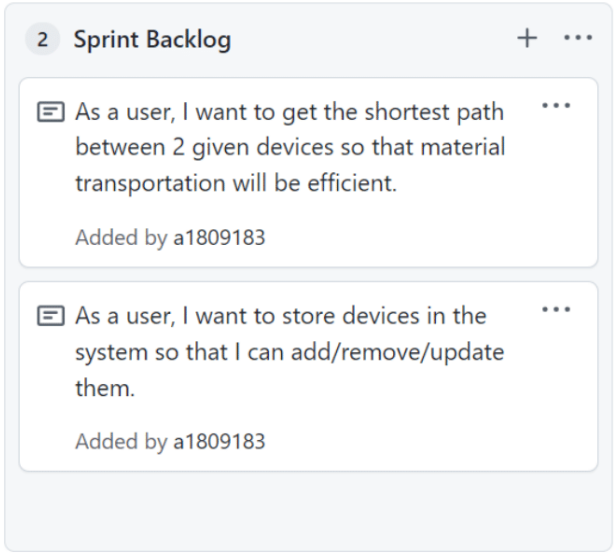


Figure 2: Sprint backlog

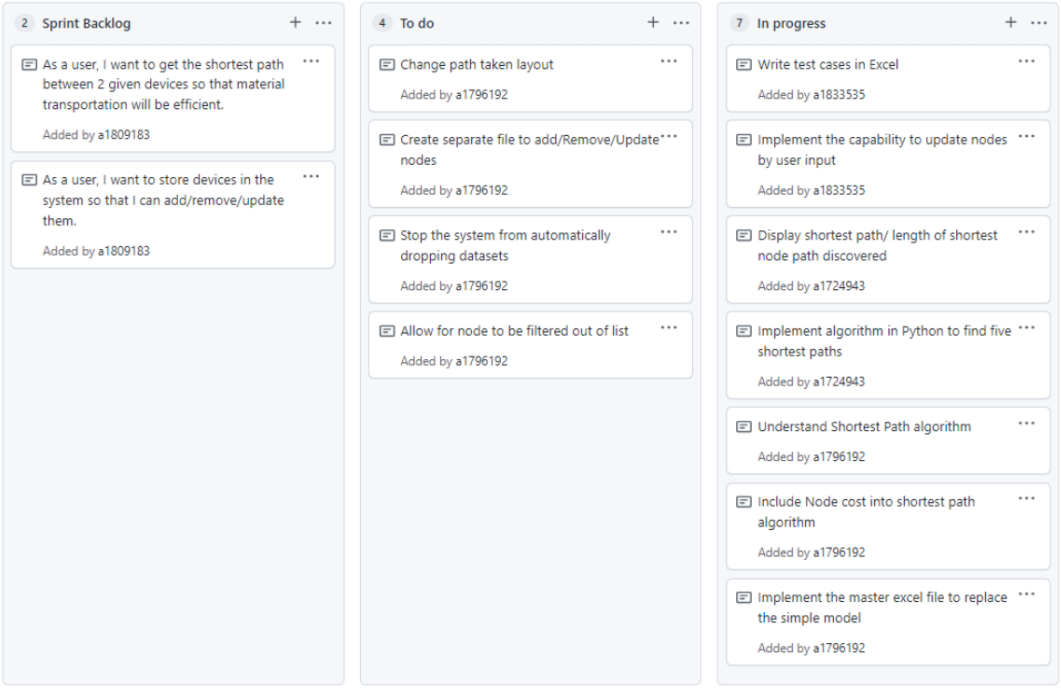


Figure 3: Task Board

The current subtasks include issues that were raised with older deliverables with the purpose of the subtasks to rectify the issues. The subtasks focus on the correct calculation of costs in the shortest path algorithm, creating procedures to add and delete edge/nodes and decoupling code into modular queries. As shown in Figure 2, our current sprint backlog task revolves around updating, and optimizing existing implementations to fix bugs mentioned by the advisor. The bugs as mentioned include the dataset automatically deletes itself after every execution, node cost is not included in the path algorithm, including the master excel file into the SQL server instead of the existing basic implementation.

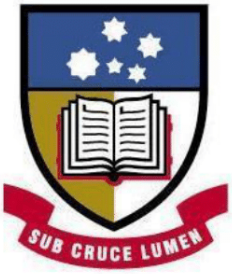
The shortest path algorithm in SQL is beginning to be better understood, and the capability to import data from Excel in to relevant programs and formats is being implemented.

1.2 Definition of Done:

- If code has been written, it must follow the coding standards defined in the initial report.
- If code has been written, it must be reviewed by members to ensure proper functionality and appropriate software design, as well as its relation to the product backlog.
- If code has been written, it must pass localization testing defined by the programmer, automated testing written by the team, and automated regression testing for any future alterations to ensure correct behaviour.
- If code has been written, it must ensure functionality continues to work on build server, and local device.
- The created sprint task must be linked towards the sprint backlog which in turn must be linked towards the user story.
- Any feedback provided by the client, product owner, and scrum team should be analysed for practicality/feasibility and implemented once deemed beneficial to the user.
- For any implementation of code or system architecture, sufficient documentation is required to justify its relation with the user story, and configuration changes must be well-documented.
- The documentation and code must be deemed acceptable by the criteria given by the product owner, as well as basic software/engineering standards, and be signed off by the product owner.
- Verification and validation of key scenarios are considered to ensure basic functionality before in-depth processes are considered for implementation.
- The final product must address the provided user stories to a reasonable standard, being able to at the very least produce a shortest path between two specified devices.

1.3 Summary of changes:

So far in this sprint, the team has successfully implemented functions to import the factory data from the Excel document in both Python and SQL. The team has refined its Python program, focusing primarily on using lists and a depth first search algorithm. Additionally, the team has furthered its progress with implementing functions for developing node and edge tables, and finding five shortest paths in SQL. Though this has not been completed, it has developed significantly throughout the sprint so far, and some group members have written several test cases in hopes of having some working code before the end of this sprint.



THE UNIVERSITY of ADELAIDE

Eng 3006 Software Engineering & Project

Shortest Path Algorithm for Material Transportation

Snapshot 5.1 of Group Path 12

Authors: William Robinson (a1724943), Ayman Shaawi(a1799924), Gurvir Singh (a1796192), Stephanie Turner(a1833535), Nelson Peterson (a1801266), Adam Rebes (a1793912), Murray Chahl (a1764549), Rukudzo Ndudzo (a1802422), Addy Dhingra (a1803893)

October 2023

Page count: 6

Contents

1. Product Backlog, Task Board Sprint, Backlog and 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 mark devices to exclude, so that shortest paths can be identified avoiding them.
- As a user, I want to get 5 shortest paths given a single source, and multiple destinations.
- As a user, I want to optimize the performance of the algorithm to produce the shortest paths.

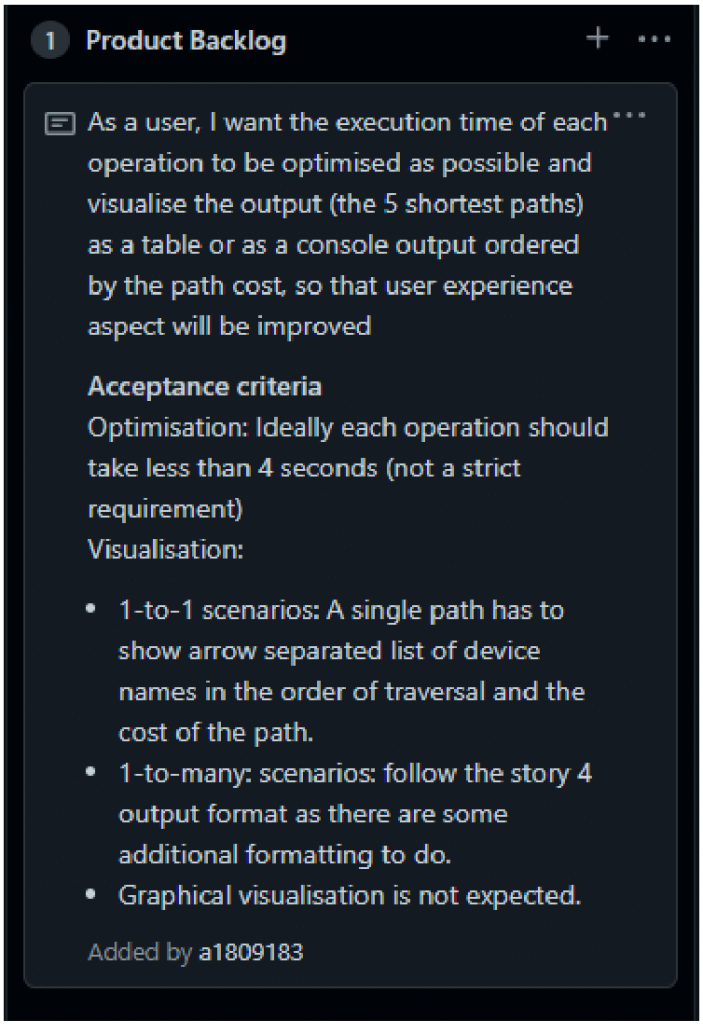


Figure 1: Product Backlog

1.1 Sprint Backlog and User Stories

The group will focus on ""As a user, I want to get 5 shortest paths given a single source and multiple destinations", and "As a user I want to mark devices to exclude, so that shortest paths can be identified avoiding them". Furthermore the system should be optimized to prevent the SQL server from storing junk data.

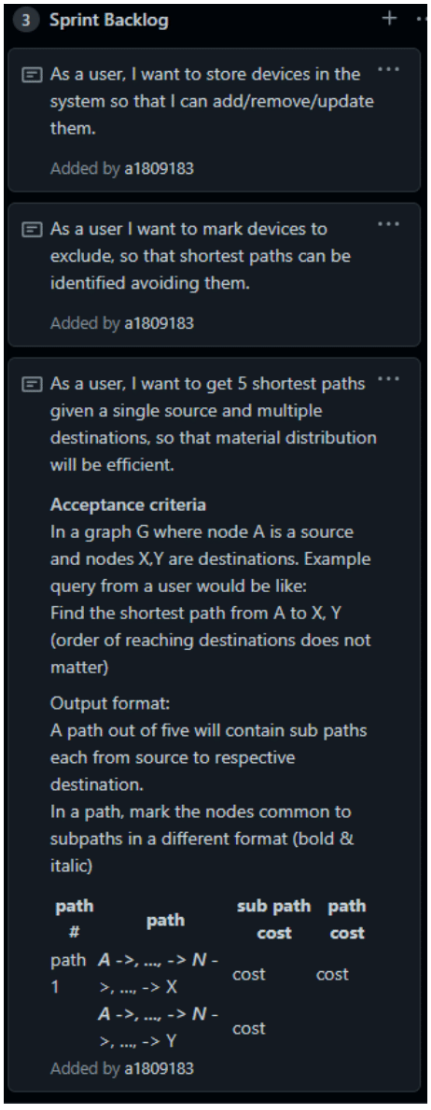


Figure 2: Sprint backlog

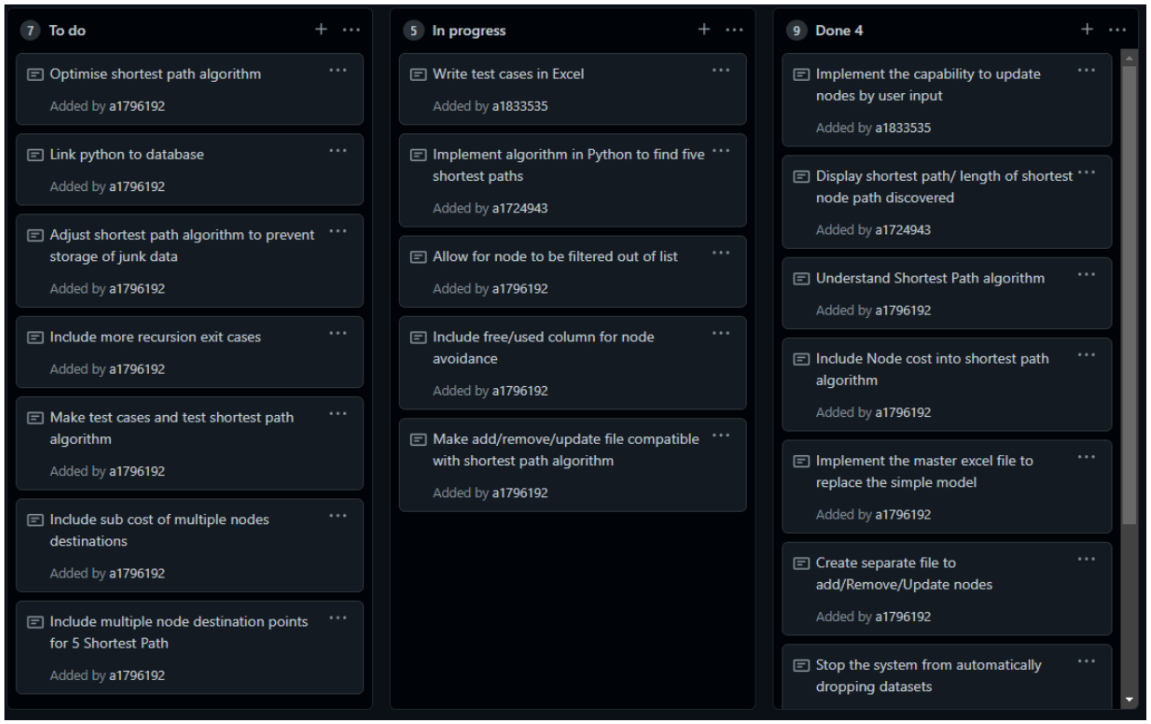


Figure 3: Task Board

Currently, the user story focuses on **5 Shortest Path algorithms,** and **avoids nodes**. The subtasks for **5 shortest paths** include modifying the shortest path algorithm to accept multiple destination points by including an OR condition within the recursion exit conditions. Furthermore, extra code must be included to calculate the sub cost of paths by examining the split off node point of each destination source. For user story **avoid nodes**, the current system indicates used nodes by name, and will be replaced with a column of bit data representing free/used state. For early consideration of user story **Process Time**, extra exit conditions would be included into the shortest path recursion algorithm to prevent calculations and storage of junk data. With all these modifications of the original shortest path algorithm, test cases will allow verifications that the user stories are successfully achieved.

1.2 Definition of Done:

- If code has been written, it must follow the coding standards defined in the initial report.
- If code has been written, it must be reviewed by members to ensure proper functionality and appropriate software design, as well as its relation to the product backlog.
- If code has been written, it must pass localization testing defined by the programmer, automated testing written by the team, and automated re-gression testing for any future alterations to ensure correct behavior.
- If code has been written, it must ensure functionality continues to work on

the build server, and local device.

- The created sprint task must be linked towards the product backlog which in turn must be linked towards the user story.
- Any feedback provided by the client, product owner, and scrum team should be analysed for practicality/feasibility and implemented once deemed beneficial to the user.
- For any implementation of code or system architecture, sufficient documentation is required to justify its relation to the user story, and configuration changes must be well-documented.
- The documentation and code must be deemed acceptable by the criteria given by the product owner, as well as basic software/engineering stan- dards, and be signed off by the product owner.
- Verification and validation of key scenarios are considered to ensure basic functionality before in depth processes are considered for implementation.
- The final product must address the provided user stories to a reasonable standard, being able to at the very least produce a shortest path between two specified devices.

1.3 Summary of Changes:

In the week following the last snapshot we've done further work on the shortest path algorithm so it now returns the five shortest paths factoring in node and edge cost, with the path taken being displayed as the device names with arrows between them. More recursion exit conditions have been added to increase the speed of the calculation of the paths by exiting a recursive instance which won't lead to calculating a shortest path. In addition, further work was done on the script produced to take imported csv data and turn it into node and edge tables, so that they now have all the necessary attributes needed for the final shortest path algorithm and accounts for NULL values in the imported csv data. From the node and edge tables a temporary table was created to match the form the shortest path algorithm currently uses so that imported data can now easily be used in the shortest path algorithm. Finally, the running of the shortest paths is no longer independent each time, in that the database is no longer dropped so the database can be altered with procedures and the new layout will be reflected in the results produced when running the shortest path algorithm.

THE UNIVERSITY of ADELAIDE

Eng 3006 Software Engineering & Project

Shortest Path Algorithm for Material Transportation

Snapshot 5.2 of Group Path 12

Authors: William Robinson (a1724943), Ayman Shaawi(a1799924), Gurvir Singh (a1796192), Stephanie Turner(a1833535), Nelson Peterson (a1801266), Adam Rebes (a1793912), Murray Chahl (a1764549), Rukudzo Ndudzo (a1802422), Addy Dhingra (a1803893)

October 2023

Page count: 6

Contents

1. Product Backlog, Task Board Sprint, Backlog and 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 mark devices to exclude, so that shortest paths can be identified avoiding them.
- As a user, I want to get 5 shortest paths given a single source, and multiple destinations.
- As a user, I want to optimize the performance of the algorithm to produce the shortest paths.
- As a user, I want to get the shortest path between 2 given devices so that material transportation will be efficient.

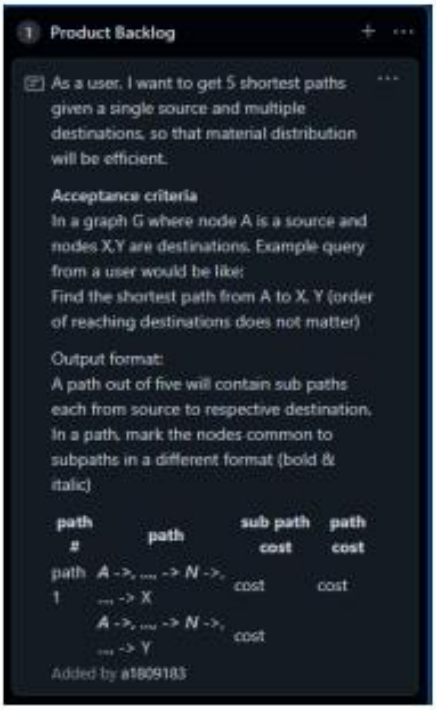


Figure 1: Product Backlog

1.1 Sprint Backlog and User Stories

The group will focus on "As a user, I want to optimise the performance of the algorithm to produce the shortest paths. "

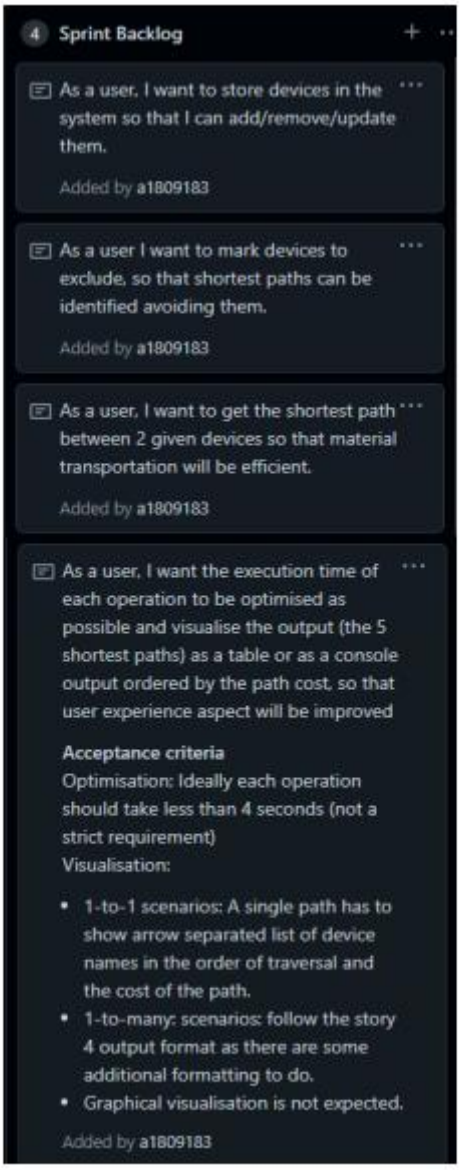


Figure 2: Sprint backlog

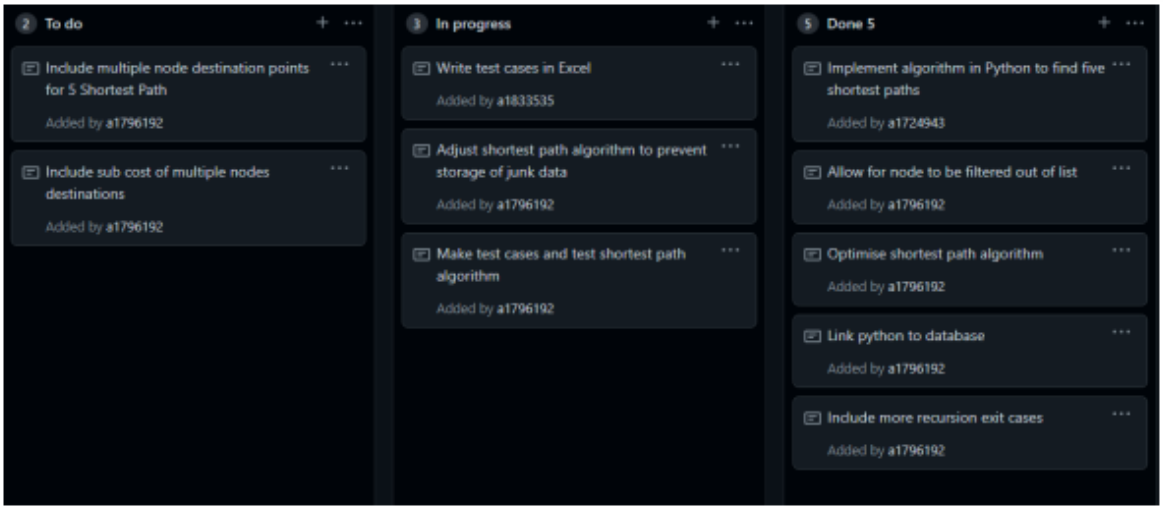


Figure 3: Task Board

Currently, the user story being implemented is **5 Shortest Paths**, and **Optimising Execution Time**. The subtasks for **Optimising execution time** included adding an extra dataset which remembers if a certain node is linked to source/destination. Currently, the system keeps recalculating all possible combinations where the same cycle will be recalculated multiple times which rapidly increases process time. Exiting the avoid node recursion cycle further allowed for less recursion processes. Storage of junk data by the recursion algorithm must be fixed to prevent decreases in online server capacity. For user story **5 Shortest Paths,** we are still expanding the shortest path algorithm to consider multiple destinations but with limited success. The sub cost calculations have been developed but need to examine the node split off point to determine the difference in cost.

1.2 Definition of Done:

- If code has been written, it must follow the coding standards defined in the initial report.
- If code has been written, it must be reviewed by members to ensure proper functionality and appropriate software design, as well as its relation to the product backlog.
- If code has been written, it must pass localization testing defined by the programmer, automated testing written by the team, and automated regression testing for any future alterations to ensure correct behaviour.
- If code has been written, it must ensure functionality continues to work on the build server, and local device.
- The created sprint task must be linked towards the product backlog which in turn must be linked towards the user story.
- Any feedback provided by the client, product owner, and scrum team should be analysed for practicality/feasibility and implemented once deemed beneficial to

the user.

- For any implementation of code or system architecture, sufficient documentation is required to justify its relation to the user story, and configuration changes must be well-documented.
- The documentation and code must be deemed acceptable by the criteria given by the product owner, as well as basic software/engineering stan- dards, and be signed off by the product owner.
- Verification and validation of key scenarios are considered to ensure basic functionality before in depth processes are considered for implementation.
- The final product must address the provided user stories to a reasonable standard, being able to at the very least produce a shortest path between two specified devices.

1.3 Summary of Changes:

In the week following the last snapshot we've done further work on the optimisation of the shortest paths algorithm as well as bug fixing in the stored procedures for updating nodes and edges. The group also implemented code to ignore edges and nodes that are in use. Alterations to one of the where statement conditions were added to ignore edges with an inUse column set to 1. The bugs present in the stored procedures were due to incorrect assumptions that were made, and were rectified to encompass the desired functionality. Mainly the stored procedure to update the edge was changed to allow for multiple edges between the same two nodes. The code was optimised to run under the desired 4 seconds by calculating paths that can reach the source from the destinations. Nodes that cannot reach the destination would not be considered when running the shortest paths algorithm, saving many iterations on all possible paths that can not make it to the destination. This requires an extra column to signify if the current node is linked to the destination, however this is stored in a temporary table created for the destination when the shortest path is executed.