THE UNIVERSITY of ADELAIDE

Eng 3006 Software Engineering & Project

Shortest Path Algorithm for Material Transportation

Sprint Retrospective 3 of Group Path 12

SEP Group Path 12 Members: Adam Rebes (a1793912), William Robinson (a1724943), Ayman Shaawi(a1799924), Gurvir Singh (a1796192), Stephanie Turner(a1833535), Nelson Peterson (a1801266), Murray Chahl (a1764549), Rukudzo Ndudzo (a1802422), Addy Dhingra (a1803893)

October 2023

Snapshots

Snapshot 3.1

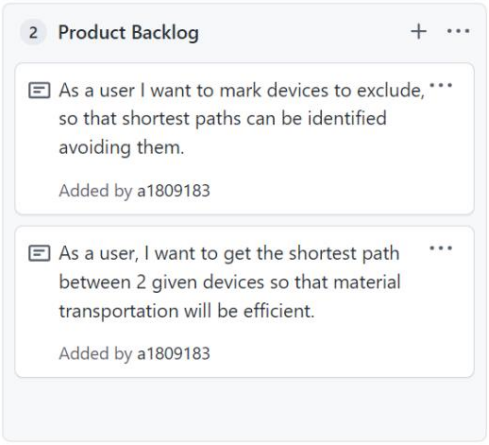


Figure 1: Product Backlog

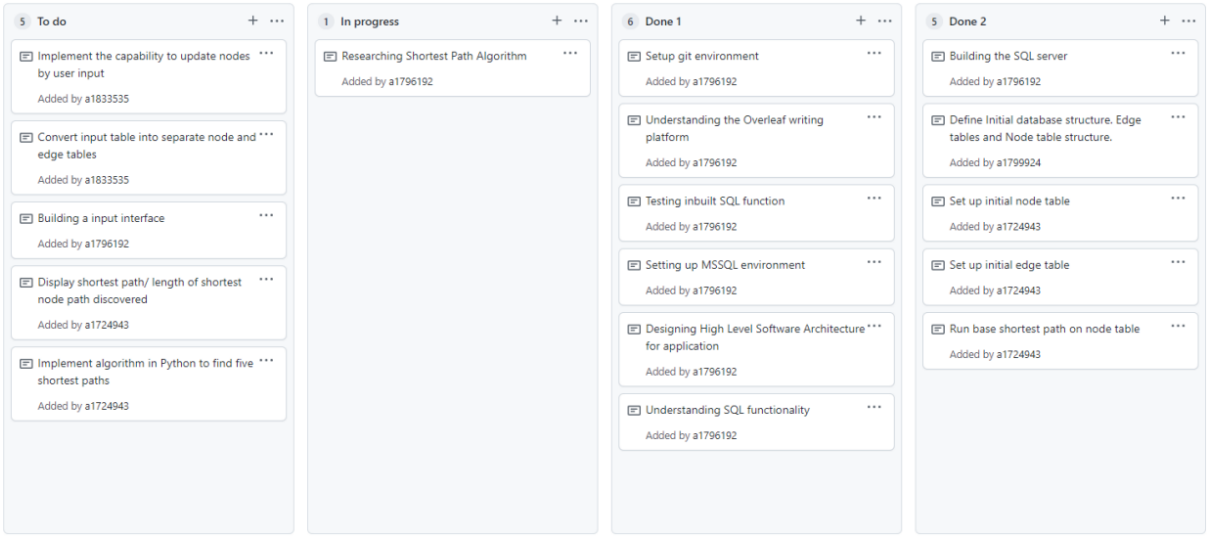


Figure 2: Task Board

Sprint Backlog and User Stories

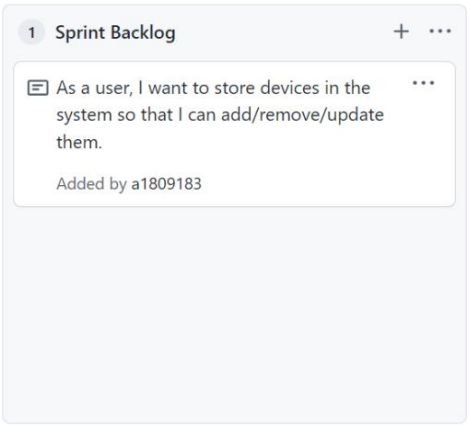


Figure 3: Sprint Backlog

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.

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, the 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.

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.

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.

Snapshot 3.2

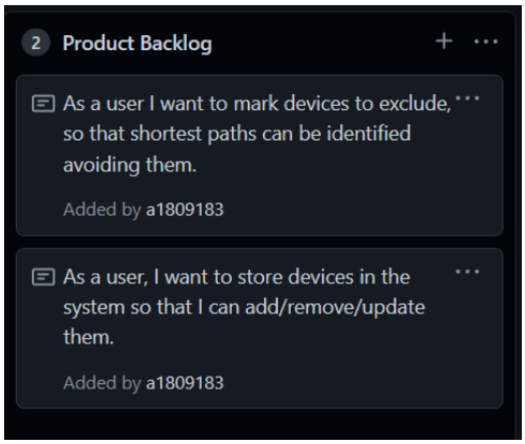


Figure 1: Product Backlog

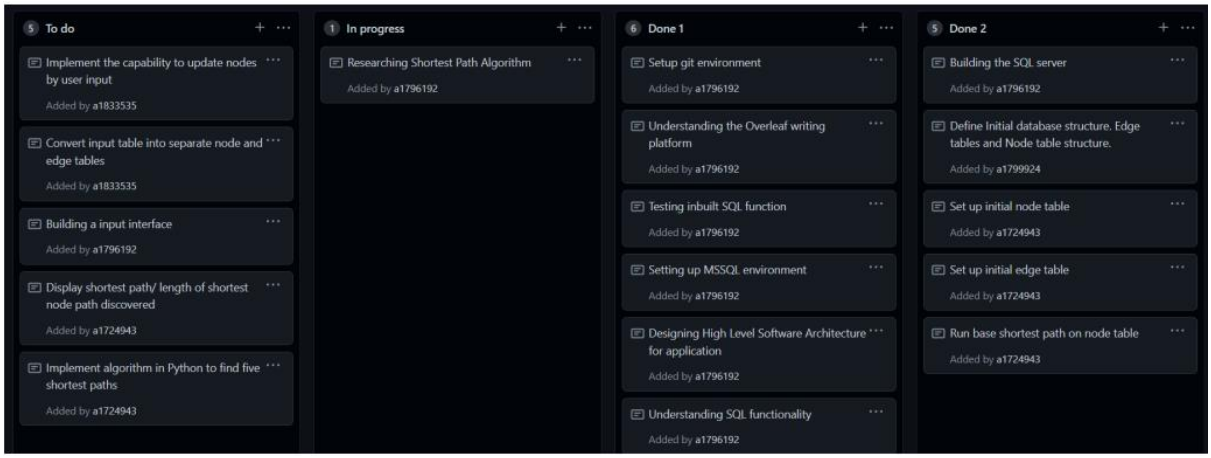


Figure 2: Task Board

Sprint Backlog and User Stories

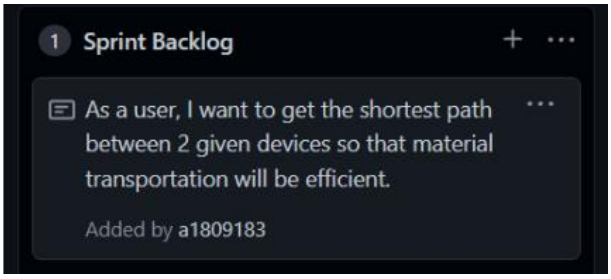


Figure 3: Sprint Backlog

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.

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, the 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.

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.

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.

"I attended the sprint review/planning meeting on Thursday the 5th of October with the tutor Dileepa Pitawela"

Sprint

What went well?

While continuing to work in MSSQL we have also split part of the team into working in python for the ease it provides in creating user-friendly ui and input, as well as for writing and experimenting with shortest path algorithms. The python code takes an excel file as the initial input of the plant layout and then allows the user to specify the changes they want to make to the structure. Meanwhile in mssql, input is currently taken by creating the table with a sql script which can be modified manually

to change existing rows or add new rows, node and edge tables are then made automatically using additional sql queries. We also implemented the ability to run shortest path on this data, with it currently returning a single shortest path factoring in edge cost. We also continued further research into other shortest path algorithms which would enable us to return five shortest paths and discovered Yen's algorithm.

What could be improved?

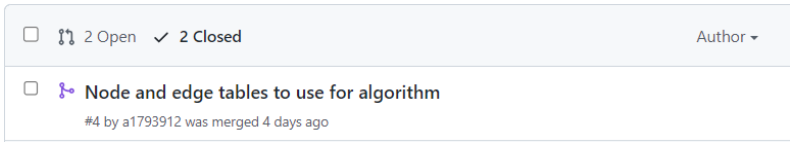
There's much which could have been improved in the sprint. We hadn't really considered the database in our implementations, e.g. currently our mssql implementation of shortest paths treats each run as separate by dropping the database each time and reconstructing the database and tables again each time. The python implementation currently doesn't connect to any database and all, with all the data stored locally. It will be trickier to incorporate the database in the python as it will require importing external modules, and we haven't looked much into that at this stage. Also, the shortest path algorithm in python isn't complete yet and currently doesn't return any path or cost. Further, in the MSSQL implementation, while it does incorporate edge cost, it has yet to incorporate node cost.

What will the group commit to improve in the next sprint?

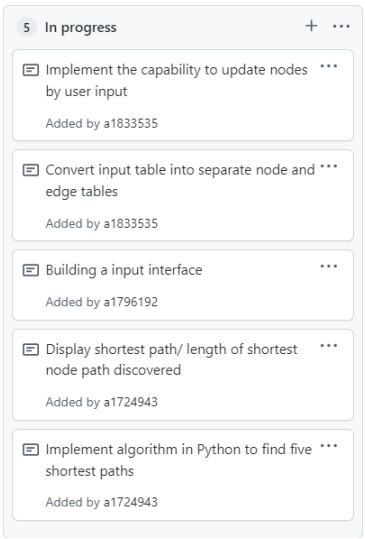
We need to add node costs to the initial data we use to construct the node and edge tables, and instead of building the table using a sql script each time, we should modify the excel file to have all the data and use the MSSQL's in-built importing functionality to build the initial data table from that which can persist in the database. For making data modifications our current method will no longer work so we'll have to look into whether we can modify the database data manually using MSSQL. We will continue to explore the python route and will need to import a module to connect and query the database. Interfacing with the database will be more difficult via the python route but we believe it's still a worthwhile option to explore due to having more control over the user interface and more-intuitive programming. We will continue writing the shortest path algorithm in both environments with the goal of implementing Yen's algorithm and returning 5 shortest paths.

Sprint Progress

This sprint has revolved around a variety of interconnected tasks which we've started and are still currently in-progress. I've been working on the MSSQL side spending a large amount of time trying to get data in a good form for running the shortest path algorithm on it. I made a pull request of an earlier instance of this:



This version did include a node cost along with all the other parameters but was ultimately more helpful as a visualisation of what the final node and edge tables should look like with their being too much manual inputting needed before running the shortest path algorithm e.g. the edge table connections needed to be put in manually into the edge table.



Working alongside other group members we were able to get an initial table that more-closely resembled that of the given excel file data, however it lacked an incorporation of node cost. We were able to adapt some sql code to run shortest path successfully on this data and return a single shortest path incorporating edge cost, which we presented as a demo to the product owner.