**School of Computing**
FACULTY OF ENGINEERING AND
PHYSICAL SCIENCES

UNIVERSITY OF LEEDS

# Final Report

## A Web Application To Visualize Mathematical Morphology on Grid Graphs

**Arjun Krishnan**

**Submitted in accordance with the requirements for the degree of**
**BSc Computer Science with Artificial Intelligence**

**2023/24**

**COMP3931 Individual Project**

The candidate confirms that the following have been submitted*:*

| **Items** | **Format** | **Recipient(s) and Date** |
|---|---|---|
| *Final Report* | *PDF file* | *Uploaded to Minerva (30/04/24)* |
| *Scanned participant consent forms* | *PDF file* | *Uploaded to Minerva (30/04/24)* |
| *Link to online code repository* | *URL* | *Sent to supervisor and assessor (30/04/24)* |

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student) Arjun

©2024The University of Leeds and Arjun Krishnan

## Summary

The project aims to provide a tool to visualize Mathematical Morphology on grid graphs. This report contains information on how the application was developed along with user evaluation.

A thorough evaluation was conducted on the existing tools that perform morphological operations. This revealed that all the tools perform this only on images, which provides an opportunity to extend this to graphs. The project focuses on a particular type of graph called squared grid graph. The concept of Mathematical Morphology is also explained in detail to inform the reader about the operations that are to be implemented in the software solution.

The project outlines the requirements the application needs to fulfil along with a project plan that evenly divides the development, justifies tools that were used in development and explains the implementation in detail, explaining each part of the user interaction and the technical details behind each interaction.

After development, evaluation of the tool was conducted in the form of user evaluation along with unit testing. This is used to validate that the project deliverables are met and to gain additional feedback on user experience and to identify potential areas of improvement.

Ideas for future work have been mentioned at the end of the report.

## Acknowledgements

I would like to thank my assessor Kristina Vuskovic in providing guidance and feedback on during the assessor meeting.

I would like to thank My supervisor Dr John Stell, whose invaluable feedback and explanation of the topic allowed me to progress this far in my project. The almost weekly meetings during the second semester really helped me understand the topic and helped with the development of this project.

I would also like to thank my family for their support in my academic journey.

# Table of Contents

## Nomenclature

MM: Mathematical Morphology

SE: Structuring Element

API : Application Programming Interface

# Chapter 1
# Introduction and Background Research

## 1.1 Introduction

Mathematical Morphology (MM) is a theory and technique initially used in analysing geometrical structures. It is based on set theory, lattice theory and topology.

Mathematical morphology introduced the 1960's by J Serra and Mathreon [1] introduced the topic to analyse geological structures and material structures. It has since expanded to image processing [2] , since principles of MM can be applied to any structure it is used in image processing tasks such as noise reduction, texture analysis, shape recognition and feature extraction. Other applications include Licence Plate Detection [3], Microaneurysm detection [4].

Morphological Filtering on graphs [5] has been a more recent development, the idea being taking an image as a graph structure and performing morphological operations on the underlying graph.

While visualizing MM operations is common on images, visualizing these operations on graphs are uncommon with no available interactive solutions. The aim of this project is to address this gap by developing a visual tool that allows users to input grids and graphs and choose a structuring element which will allow them to perform morphological operations on the graph, resulting in an output graph that visualizes the operation. This tool could be beneficial to students who are keen to explore the topic and researchers who are keen to know more about MM.

## 1.2 Researching Previous Work in Visualizing MM

To understand the gap in tools that visualize MM, a thorough analysis was conducted on existing plugins and libraries. To develop the tool possible platforms are a software or a web application. Based on the research conducted there are solutions available to visualize MM in the form of libraries and plugins. The most common programming language used to develop these libraries is Python. There are no publicly available solutions that visualize MM on graphs, with all existing libraries providing support on images only.

### 1.2.1 MorphoLibJ

MorphoLibJ [6] was developed by  Legland et al in 2016 as a plugin for ImageJ, a java-based image processing program developed by NIH Image. Its seamless integration enhances the capabilities of ImageJ, allowing it to perform MM tasks on images, expanding previous plugin capabilities from 2D to also include 3D images. It supports all major operating systems and available on a web application, allowing users to access all features without any platform constraints. The library is extensive with nearly 200 classes and interfaces and was designed with modularity and reusability in mind, catering to final users, plugin developers and core developers.

MorphoLibJ performs basic morphological operations such as opening, closing, erosion, dilation, advanced operations such as Morphological reconstruction, Watershed segmentation, 2D/3D measurements and Binary/label Image utilities. [add image]

While the application does perform MM operations flawlessly on images, it is limited to certain image types. It is unable to input graphs of any kind and therefore cannot perform MM on graphs. Its User Interface is outdated with the web application being slow and unresponsive and unable to take image inputs at times. The standalone version of ImageJ does not include morpholibJ and does require additional setup to be able to access these operations.

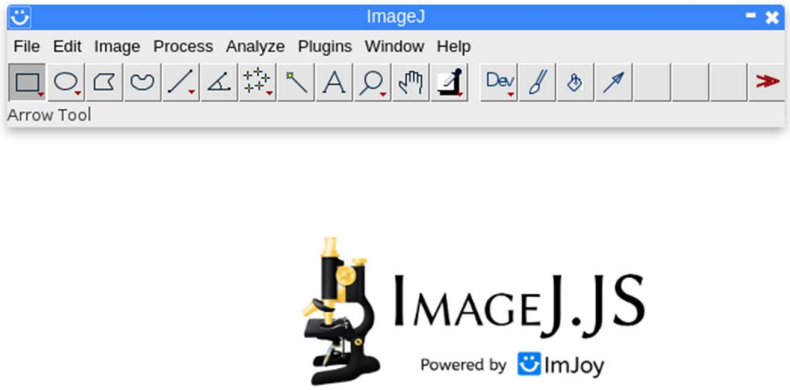



Fig 1 [6] – MorphoLibJ Interface

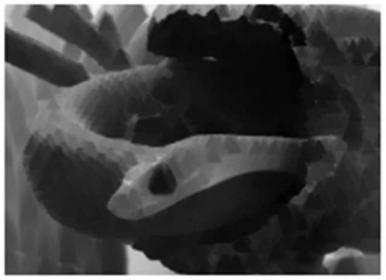
### 1.2.2 Mamba Image

MAMBA (MAthematical Morphology liBrAry Image) [7] is an open-source MM library written in Python and C to handle MM operations. It is also available across all major OS platforms.

It even provides a GUI interface if the user is on windows. It allows users to apply MM operations and other operations to an input 2D or 3D image. The advantage of using MAMBA is the performance, as MAMBA provides multiple variants of the same operation which allows users to select the one that is the most effective in each scenario.

MAMBA also suffers from some limitations. It has very specific system requirements, preventing users who do not have python 2.7 or 3.4 access to it. It also has some limitations in functionality when using 3D images compared to 2D. While it provides a grid data structure, it cannot be used to visualize MM on grids/graphs. The accuracy of this library is also in question as mentioned on their homepage.

snake.png



snake_ero10rt.png

Fig 2 - Example of Erosion of an image using Mamba

### 1.2.3 OpenCV

OpenCV (Open-Source Computer Vision Library) [8] is a C++/Python collection of hundreds of computer vision algorithms. It provides basic MM operations, requiring a structuring element in the form of a matrix (achieved using NumPy) with a few in-built structuring elements of different shapes. It has more flexibility in data types compared to the above-mentioned libraries as it allows users to use simple grid structures. With performance optimization in mind along with Machine Learning and Deep Learning capabilities, it can be used in more sophisticated image processing tasks such as object detection, face

recognition and video analysis. However, OpenCV is limited to basic MM operations and does not extend to graphs.

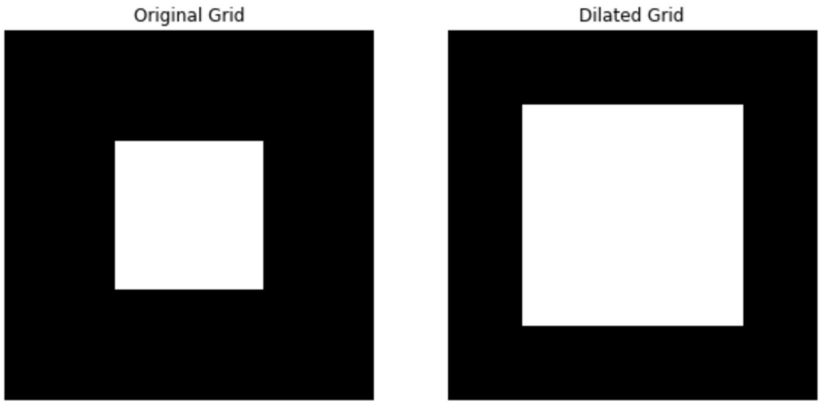



Fig 3 - Dilation on an image using openCV

### 1.2.4 Scikit-Image

Scikit image [9] is a python-based library for scientific image analysis. It provides an extensive list of MM operations from basic to advanced operations, with variations for several use cases. It can support multiple image types, which are required to be formatted into NumPy arrays before processing. It was designed with ease of use in mind and has a very high accuracy. Like OpenCV, it can be used on images and graphs. While it does have a graph data structure, MM operations on it are not supported.

### 1.2.5 MATLAB

MATLAB (Matrix Laboratory) [10] is a high-performance programming language and interactive environment developed my MathWorks. It has a variety of applications which includes Data Science, Artificial Intelligence, Image Processing and Computer Vision and more. It provides APIs for multiple programming languages such as C/C++, Python, Fortran and Java. It is a popular tool among academics and in industry due to its vast capabilities. It provides functionality to perform basic and advanced MM operations. The main limitation of MATLAB would be that it is not open source and requires a licence to access its features. It also does not have any inbuilt applications that handle MM and cannot be used with graphs.

## 1.3 MM: An Overview

This section aims to explain the prerequisite information required to fully understand the effect of MM operations on images. This section uses the example of binary images for simplicity, but keep in mind that the method remains same for other image types. Before reviewing this, here are 2 set operations that are important to understand before covering the concept of dilation and erosion.

On a Set B:

$B_z$ : Translation of B by a vector z

= {(x + z, y + z) | (x , y) ∈ B}

= {c | c = b + z , b ∈ B}

B' : reflection of B

B' = {c | c = -b, b ∈ B }

B' = {(-x, -y) | (x, y) ∈ B}

### 1.3.1 Image Representation

Binary images are the simplest form of images, where each pixel can have two possible values 0 or 1. Binary images make it easier to identify pixels of interest (usually pixels with value 1) and background pixels (pixels with value 0). The image can be visualized as a grid of pixels, which can be represented in matrix form. In the binary case consider grid to be represented as a matrix with 0's and 1's. For other image types, the grid representation remains the same, but the matrix representation will change (0 to 255 for 8-bit greyscale images and RGB channels for coloured images)

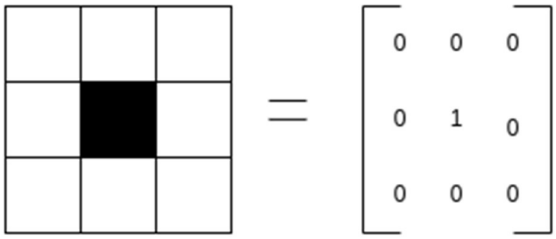



Fig 4 - Binary Image Representation Left: Grid representation, right: Matrix representation

### 1.3.2 Structuring Element (SE)

A structuring element (SE) is a predefined set that is used to transform an input image based on its geometric structure. The main purpose of SE is to interact with the image by placing it in all parts of the image and depending on the operations such as dilation, erosion opening, closing and more determine which pixels intersect with it and which pixels fit in it [ref]. It is represented in matrix form (usually odd dimensions) and has 3 main characteristics:

1. Shape: SE can have various shapes, such as squares, circles, lines, or any custom shape, based on the image processing task. The shape determines which pixels in the neighbourhood of a target pixel are considered when applying a morphological operation.
2. Size: The size of SE can change how many pixels are manipulated during a MM operation. Larger the SE, more pixels being operated on resulting in a larger effect on the image.
3. Origin: SE must have a point of origin (most commonly its centre), which will determine which pixels will be operated on.

### 1.3.3 Dilation

“The dilation of X by B (which is the SE) is the set of all the points such that B hits X.”[1]

Dilation: $X \oplus B = \{\mathrm{z} \mid \hat{B} \cap X \subseteq X\}$

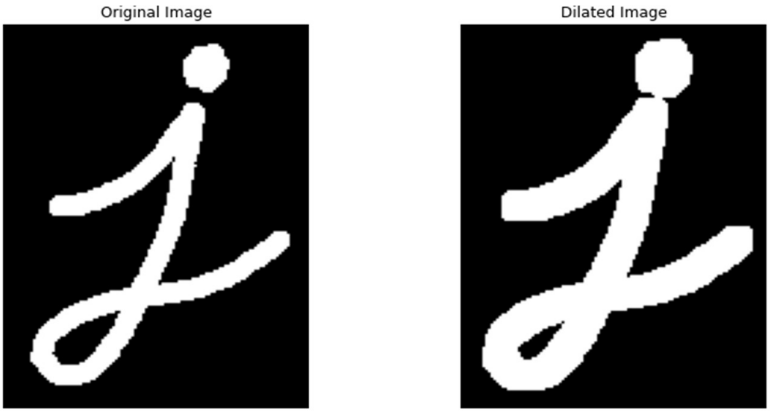



Fig 5 [22] - Dilation of an image

When applied on images, depending on the SE, features of the input image, such as lines and shapes are enlarged. This is done by finding pixels such that shifted SE has any overlap with the original set resulting in the SE pixels being added to it giving features thicker

boundaries. The resulting image will always be part (subset) of the input image. It is important to note that the SE must be chosen carefully, as in some cases dilation can alter the image completely. Considering the image above, if the SE was larger, the dot on the J will merge resulting in an image with a completely different meaning.

### 1.3.4 Erosion

"The erosion of X by B (SE) is the set of all the points such that B is included in X" [1].

Erosion: $X \ominus B = \{z \mid B_z \subseteq X\}$

The SE is shifted through the entire image and any pixels part of the image that does not completely align with the SE is removed. When images are eroded, its features are reduced due to pixels being removed from its boundaries. Erosion is also dependent on the SE like dilation and the resulting image will be part (subset) of the input image.

Erosion and Dilation when performed consecutively in any order using the same SE visually look like they cancel each other out. This is not the case. If erosion removes significant parts of an object, dilation might not fully restore the original object because some information about the object's original shape may have been lost during erosion. Similarly, if dilation merges two objects, erosion might not be able to separate them again completely.

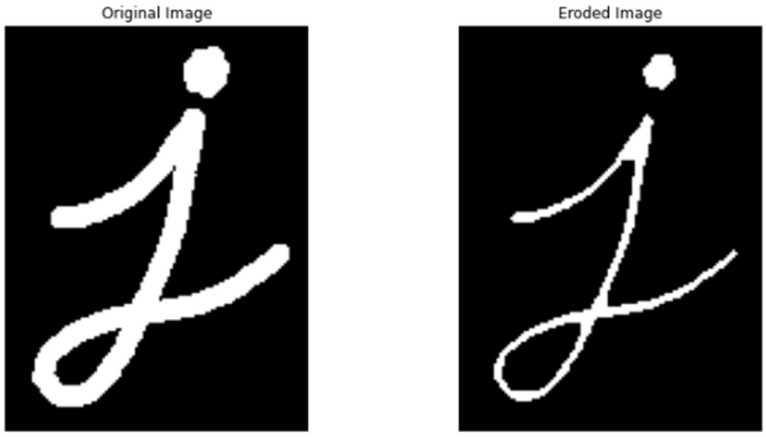



Fig 6 [22] – Erosion of an image

### 1.3.5 Opening and Closing

Opening an image by a structuring element is defined as the erosion of the image by the structuring element, followed by dilation of the result by the same structuring element.

$X \circ B = (X \ominus B) \oplus B$

When used with images, opening tends to remove small objects or artifacts from an image while leaving the size and shape of larger objects mostly unchanged [ref]. The initial erosion step removes small features and detaches objects that are close together. The dilation step restores the size of the larger objects but does not bring back the small objects that were removed by erosion. This makes opening useful for noise reduction and for separating objects that are close to one another.

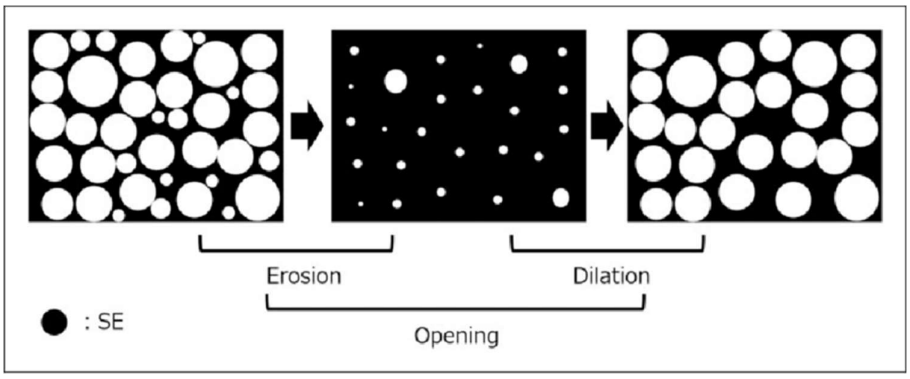



Fig 7 [13] – Opening operation on an image

Closing an image by a structuring element is defined as the dilation of the image by the structuring element, followed by erosion of the result by the same structuring element.

$X \cdot B = (X \oplus B) \ominus B$

Closing is useful for filling in small holes and gaps within objects in an image, as well as connecting objects that are close together [10]. The initial dilation step bridges narrow gaps and connects objects that are near each other, while the erosion step restores the objects to their original size but keeps the small gaps filled. This can be used to smooth out the contours of objects or fill in small imperfections and breaks in the objects.

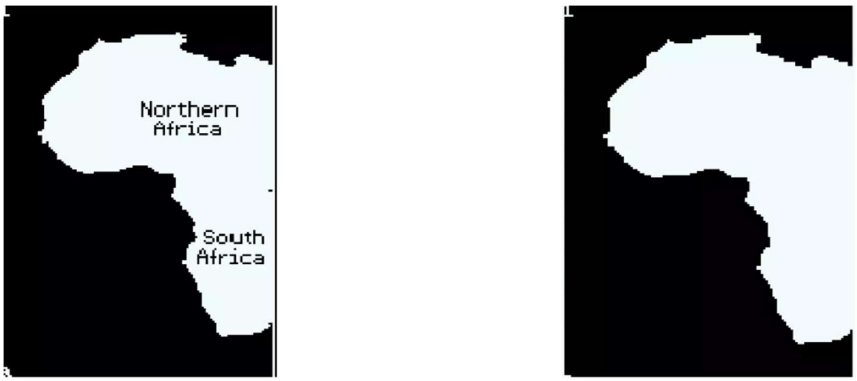



Fig 8 [23] – Closing operation on an image (Left: Before Closing, Right: After Closing)

# Chapter 2
# Methods

The development of this application has been split into 2 parts. Part 1 is the graph generation, subgraph selection and user input of SE and MM operation. Part 2 is the implementation of MM operations on the subgraph and displaying the resulting graph to the user.

The requirements of this project are non – technical and are as follows:

1. The user must be able to input a grid and select a subgraph from this grid
2. The user must be able to input their own SE
3. The user must be able to perform dilation and erosion on the subgraph
4. The user must be able to perform opening and closing on the subgraph

## 2.1 Project Management

The project used a combination of Agile and Waterfall methods, resulting in the best of both methodologies being implemented in this project. The Agile method focuses on continuous improvement, taking an iterative approach in development where each feature goes through a cycle of development, testing and feedback[12]. The requirements of the project were complicated hence splitting the work into smaller more manageable tasks along with the flexibility of this approach allowed multiple features to be developed simultaneously with the main priority being developing the different MM operations during the later sprints. My supervisor acted as the stakeholder, from whom I got feedback on the functionality and accuracy of the MM operations.

The Waterfall method works best when there are predefined requirements that rarely change, following a fixed plan where one phase starts only after the previous phase is complete [12]. This method was partly used since features like MM operations depended on the completeness and accuracy of previous features like graph generation, subgraph selection and SE selection, resulting in a linear development of some features. The project comprised of 4 sprints:

### 2.1.1 Sprint 1

The first sprint aimed to create wireframe prototypes of the tool and selecting an appropriate framework for development. It also involved identifying core features required to create a

Minimum Viable Product and outlining a sequence of actions that the user will perform while interacting with the tool. Understanding the basics of d3.js was a crucial step during this sprint as it was integral to rendering the graphs. The outcome of sprint 1 is the Wireframe and process flow diagram shown below.

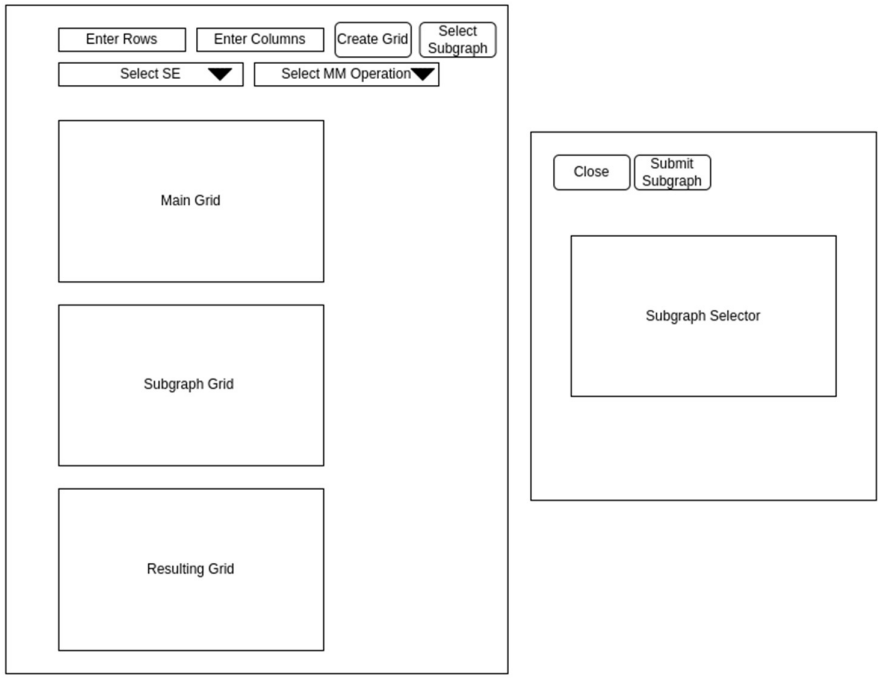



Fig 9 – Wireframe prototype of the proposed system

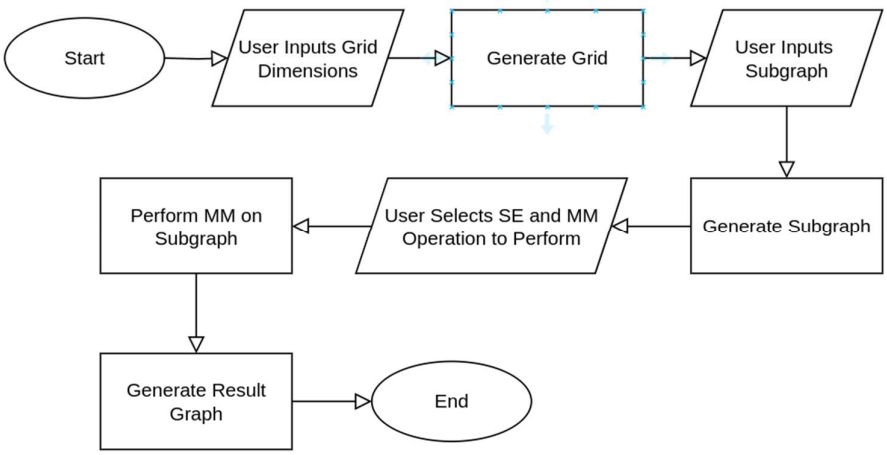



Fig 10 – User process flow diagram of the proposed system

### 2.1.2 Sprint 2

Sprint 2 focuses on developing the key features that are required before implementing the actual MM operations. This includes identifying appropriate data structures to hold information about the grids, algorithms to generate dynamic grids, identifying ways to creating a conditional popup to select the subgraph and to make the subgraph selection interactive.

### 2.1.3 Sprint 3

Sprint 3 focuses on implementing the basic MM operations such as dilation and erosion on the grid. This involves identifying some basic SE's and visualizing the operations on subgraphs and creating algorithms to identify a node's neighbours and neighbouring edges. The main challenge in this sprint is to create the algorithm that performs dilation on the subgraph given the SE, which will be discussed later. The outcome of this sprint would be the Minimum Viable Product (MVP) for this project.

### 2.1.3 Sprint 4

The final sprint focuses on implementing the advanced MM operations opening and closing on the grid. This becomes much easier as these are combinations of dilation and erosion. Another objective would be to allow users to enter a custom SE to which MM operations are applied. Code cleanup and comprehensive documentation is also completed during this sprint.

## 2.2 Technical Justifications and Version Control

The project aimed to create a web application when developing MM operations, which presented a challenge when selecting the appropriate codebase. The server-side code was the main concern when it came to selecting a framework. This is due to the wide variety in terms of programming languages and the different capabilities offered by each one. Ultimately the choice was between python and node.js, with python being the popular choice when performing MM on images.

Another deciding factor was the need to create an original solution due to the lack of available third-party libraries. This requirement highlighted the importance of selecting a framework that could accommodate the custom development of these complex functionalities.

A React.js frontend and Node.js backend was selected, with the use of d3.js for rendering the graphs. Due to the requirement of having users react with parts of the graph, node.js being one of the most popular back-end JavaScript frameworks was the better choice as it is better suited to interact with other JavaScript Frameworks compared to python-based solutions. The reasons behind selecting React.js was due to its popularity, ease of use, performance and extensive third-party support. ViteJs was used as the build tool for this project due its quick and easy setup for web–based projects.

The choice of selecting a web application over a standalone application has 2 main advantages. Web applications offer easy to implement cross-platform compatibility, simplifying development and ensuring accessibility across a wide range of operating systems. This flexibility makes it easier to target multiple platforms without the need for platform-specific adaptations. Web applications are comparatively lightweight, enabling them to run efficiently on lower-end systems. This is particularly beneficial as it broadens the application's potential user base, ensuring that individuals with less powerful hardware can still access the application.

GitHub was used to track and manage progress throughout the development cycle of the project. Each push to the branch was after thorough testing and some commits were needed to address bugs in the original code. GitHub is the number one version control tool in the market, even though similar solutions such as GitLab and BitBucket exist, Git is the most popular among students, researchers and enterprises and using it prevented the risk of data loss which would have severely impacted the project.

Visual studio code was the Integrated Development Environment (IDE) selected for this project. Its ability to recognize syntaxes of multiple programming languages and its easy integration with GitHub makes it the best choice as a development environment for this project

## 2.3 Part 1: Graph Generation and Interactivity

### 2.3.1 Graph Generation

Before the user can apply any MM operations, we first need to determine how the user would input their graph and how the graphs would be represented to the user and to the system.

The user is asked to input graph dimensions, after which they will click on the “Create Grid ” button, resulting in a POST request being sent to the server to generate the data required to

render the graph to the client. The server takes in the dimensions provided by the user generates an adjacency matrix to represent the graph. As the project focuses on applying MM to undirected graphs of a grid structure, Adjacency matrices were best suited in cases where the graphs are dense [14] as it stores all possible edge relations and allows for quick insertion and deletion of edges [15].

After the matrix is created, the server also creates two separate data structures, an array that holds all the nodes stored as objects with id as a key and integers from 0 to m*n-1 (where mxn are the dimensions of the grid) being the value. Another array is used to store all edges, where each edge is represented as an object that contains the link id, source and target nodes and an edge type for either horizontal or vertical edges, as shown in Fig 11

```
Nodes: [ { id: 0 }, { id: 1 }, { id: 2 }, { id: 3 } ]
Edges: [
  { id: 'link-0-1', source: 0, target: 1, edgetype: 'Horizontal' },
  { id: 'link-0-2', source: 0, target: 2, edgetype: 'Vertical' },
  { id: 'link-1-0', source: 1, target: 0, edgetype: 'Horizontal' },
  { id: 'link-1-3', source: 1, target: 3, edgetype: 'Vertical' },
  { id: 'link-2-0', source: 2, target: 0, edgetype: 'Vertical' },
  { id: 'link-2-3', source: 2, target: 3, edgetype: 'Horizontal' },
  { id: 'link-3-1', source: 3, target: 1, edgetype: 'Vertical' },
  { id: 'link-3-2', source: 3, target: 2, edgetype: 'Horizontal' }
]
```

Fig 11 – Data Structure of Nodes and Edges

This data is then sent to the client where it is stored and used to display the graph as a SVG.SVG images does not lose quality with change in resolution, scalable [16] and can allow user interaction[17] making it the best image format for this project. The grid component takes all the data received from the server , calculates the dimensions required to draw the grid, generates a SVG image that has the correct dimensions and then uses d3.js to draw the nodes and edges. Fig 12 shows the result when the user requests for a 2x2 grid.

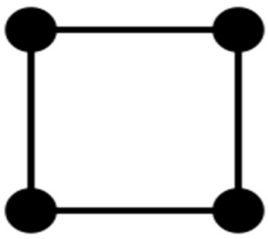


Fig 12 – Grid Displayed in the application

All the graph types are similar to the original method as shown above. The rendering of nodes and edges are identical with slight differences in appearance. The graphs used to display the SE , subgraph and output graph are coloured differently based on each case.

### 2.3.2 Graph Interactivity

There are 2 scenarios where the user will be interacting with the graphs. In both scenarios this interaction is handled through a popup , with selected nodes and edges being added to different arrays which are used to check weather a node or edge has been selected. This would result in a graph where the selections are highlighted to help the user visualize their graph interaction.

Scenario 1: Subgraph Selection

When the user selects the “Select Subgraph” button, a popup window shows the user a grid from which the subgraph is to be selected. The subgraph component has event listeners that register when a user clicks on any part of the graph.  It also maintains a list of selected nodes and edges which would be passed on to a subgraph component that highlights the selected subgraph. When the user initially clicks on an element, the element is added to either the list of nodes or edged depending on what the user selects. This will result in the colour changing from black to red. If the user selects an element that was already selected, it would be removed from the list and colour will change to black.

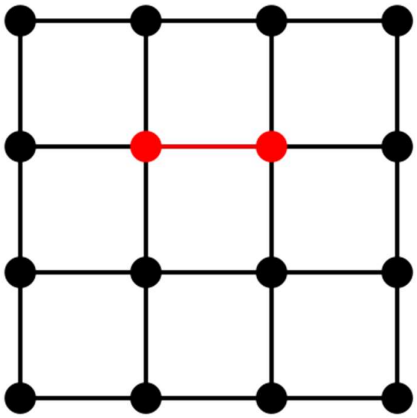


Fig 13 – selection of nodes and edges in the subgraph selection popup

In the case of when the user selects an edge, the edge and the 2  nodes that the edge connects are selected, provided that the 2 nodes were not previously selected. After the user

Selects their preferred subgraph and clicks on the “Submit Subgraph” button, the popup closes and the list of selected elements are passed on to the subgraph component. This component will make sure that the selected component are displayed in blue as shown in fig

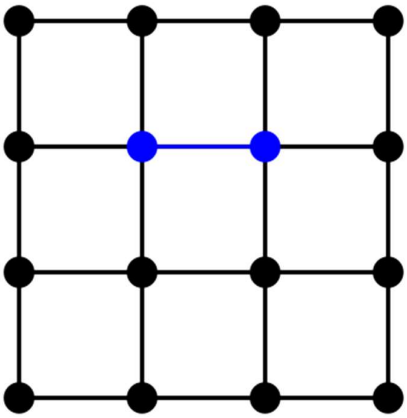


Fig 14 – Rendering of subgraph in the application

Scenario 2: SE Selection

When the user selects the “Select SE” button, a popup window shows the user a grid from which the SE is to be selected. It also keeps track of the selected nodes and edges in 2 separate lists A third list is maintained to store the current user selected origin.

The SE component has two types of event listeners that register when a user clicks on any part of the graph. The first event listener is the same as in subgraph with the colour changing from black to red (element added to list) when selected and red to black when deselected (element removed from list). The second event listener detects whether the user has right clicked on an element. When a right click occurs, the element is set as the origin. If the origin is an edge its colour changes to green.

Similar to scenario 1 , when an edge is selected, the nodes that are linked to the edge are also selected.

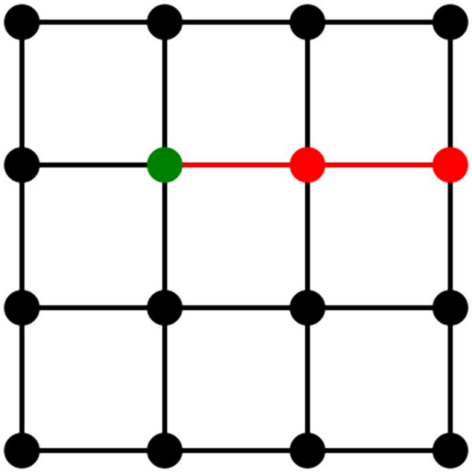


Fig 15 – Selection of a SE within the Select Node Function popup

After the user submits the SE by clicking on the “Submit SE” button, the popup closes and only the selected nodes and edges are displayed , with the origin being green as shown in fig 16.

Fig 16 – Rendering of SE in the application

When the node is the origin, the node does not have to be part of the SE, and can be used a just a reference point. The application accommodates this by changing the colour of an unselected node from black to yellow when the user right clicks on a black coloured node as shown in fig 17.

Fig 17– Rendering of Node case SE where node is not part of the solution

## 2.4 Part 2: Implementation of MM Operations

### 2.4.1 Structuring Element

The user has 2 choices when it comes to selecting a structuring element. If the user would like to view some base cases, they have the option to select a SE from the “Select SE”

dropdown. Else if the user wishes to enter their own SE, they must select “custom SE” from the dropdown and then interact with the SE selector as described in scenario 2 of 2.3.2.

When applying the SE to any subgraph, the predefined SE cases are handled by separate functions as the underlying pattern that is to be applied

The application provides the user with a few predefined cases of SE’s that the user can select in the form of a dropdown. These can be directly applied with MM operations without the user having to come up with their own custom structuring element. They are:

1. Node: This SE has only one node to it, which is also the origin. It is used to check if the subgraph has nodes. If the subgraph contains both nodes and edges the edges will be removed resulting in only the nodes. This occurs regardless of what MM operation is applied to it.
2. Cross Shaped (No Edges): This SE has a total of 5 nodes. The origin would be the node in the center, with the immediate neighbour in all directions also being part of the SE.
3. Node + Edge + RNode: This SE has 2 nodes and 1 edge. The origin would be the node, with its right neighbour and horizontal edge between them being part of the SE.
4. Horizontal Edge: This SE has 8 nodes and 7 edges. The origin will be the horizontal edge. For each node connected to the horizontal edge, its neighbouring nodes and edges are part of the SE.
5. Vertical Edge: This SE has 8 nodes and 7 edges. The origin will be the vertical edge. For each node connected to the vertical edge, its neighbouring nodes and edges are part of the SE

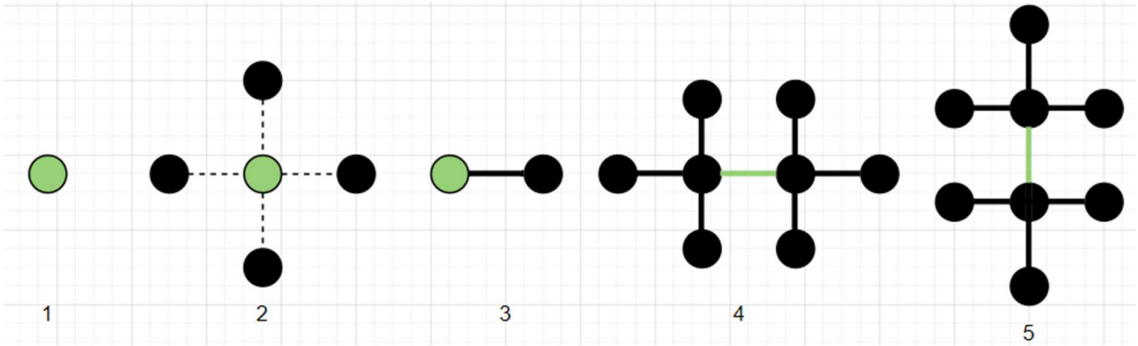



Fig 18 – Visual of each SE in order. Dashed edges to represent lack of edge and origin green.

The user can create their own 3 part custom SE when selecting the “Custom SE” option from the SE dropdown. When selected it will then display 3 buttons:

1. Select Node Function: This button allows the user to select a SE that is to be applied when a node is detected in the subgraph.
2. H Edge Function: This button allows the user to select a SE that is to be applied when a horizontal edge is detected in the subgraph
3. V Edge Function: This button allows the user to select a SE that is to be applied when a vertical edge is detected in the subgraph.

Fig 19 shows an example of a 3 part SE. This allows the user complete customization when it comes to selecting a SE.

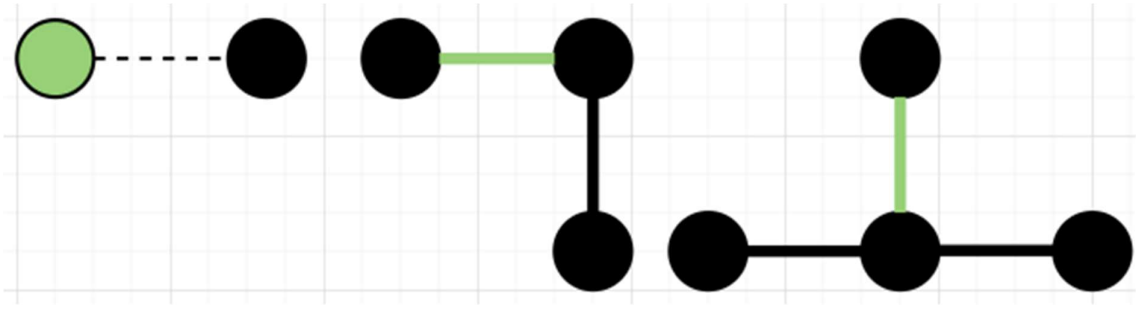


Fig 19 - 3 part SE. Each origin is marked in green. Dashed line represents that edge is not part of the solution

### 2.4.2 Helper Functions

In order to carry out the MM operations, a few helper functions were created to ensure that the code remains modular and making it easier to handle the multiple base cases and custom SE’s.

`function getNeighbors(nodeId, rows, cols):` Takes a nodes id and the dimensions of the grid as integers and returns an array of the nodes neighbours as integers. It does so by calculating what row and column the node is on, checking if its all its neighbours are within the graph and adding it to the array.

`function getConnectedLinks(nodeId, links):` Takes a nodes id and the array of all edges, and returns all the edges that are connected to the node.

`const calculatePositionsNC = (originNode, nodes, links, rows, cols) => {}:` Takes the node that would be the point of origin, the dimensions and the list of rows and

edge objects, and calculate the relative position of the nodes and links from the origin. This function is extended to cases where SE can also be an edge as we can take any of the nodes that connect the edge as a point of origin for the sole purpose of calculating relative positions .

### 2.4.3 Dilation

When the user selects "Dilation" from the MM operations dropdown and after selecting a SE and clicking on the "Perform Operation" button, dilation occurs. The grid dimensions, SE, and subgraph are then sent to the server. The server then tries to "match" the SE with the entire subgraph by using the origin as a reference point. For every successful match , The nodes and edges of the SE gets added to the subgraph, resulting in any elements that are part of the SE but not the subgraph being added to the subgraph. The server then sends this data back to the client which is rendered in a similar way as the subgraph with the dilated nodes being a different colour.

Dilation does not ensure that all the nodes in the original subgraph will be present in the result. For example a SE that has nodes only when used for dilation would result in a subgraph having nodes only, even if edges were originally present.

If the user selects a predefined SE, as these are preset patterns, for every successful match, the server adds the nodes first , followed by the edges in the same pattern that the user has selected.

In the case where the user inputs a 3 part structuring element, the server starts by handling the node case. For each node in the subgraph , nodes and edges that are present in the node case are added. The origin node is only added if the user has selected a highlighted node as its origin . The server then follows the same process for each horizontal edge and vertical edge in the subgraph, adding the nodes ,edges and origin of the horizontal case and vertical case respectively to the resulting subgraph. The nodes and edges for each case are added by calculating relative positions of the nodes and edges from the origin , which is then applied to every match in order to successfully capture the pattern of the SE on to the resulting subgraph.

Edge Cases in Dilation: There are 2 edge cases in dilation.

1. No matches: When there are no matches found when performing dilation, it results in an empty subgraph. This could be because the original subgraph did not have any elements or there were no successful matches.

2. Out of Bounds: There are certain cases where a match would occurs but the elements that are to be added would be outside the original grid. In this case these nodes or edges are not added and hence not displayed. For example if an SE that is of a line shape with a node as its origin matches with the outermost node in a subgraph, it will try to add nodes that are out of bounds.

Fig 20 provides a visual example of how dilation is performed on graphs.

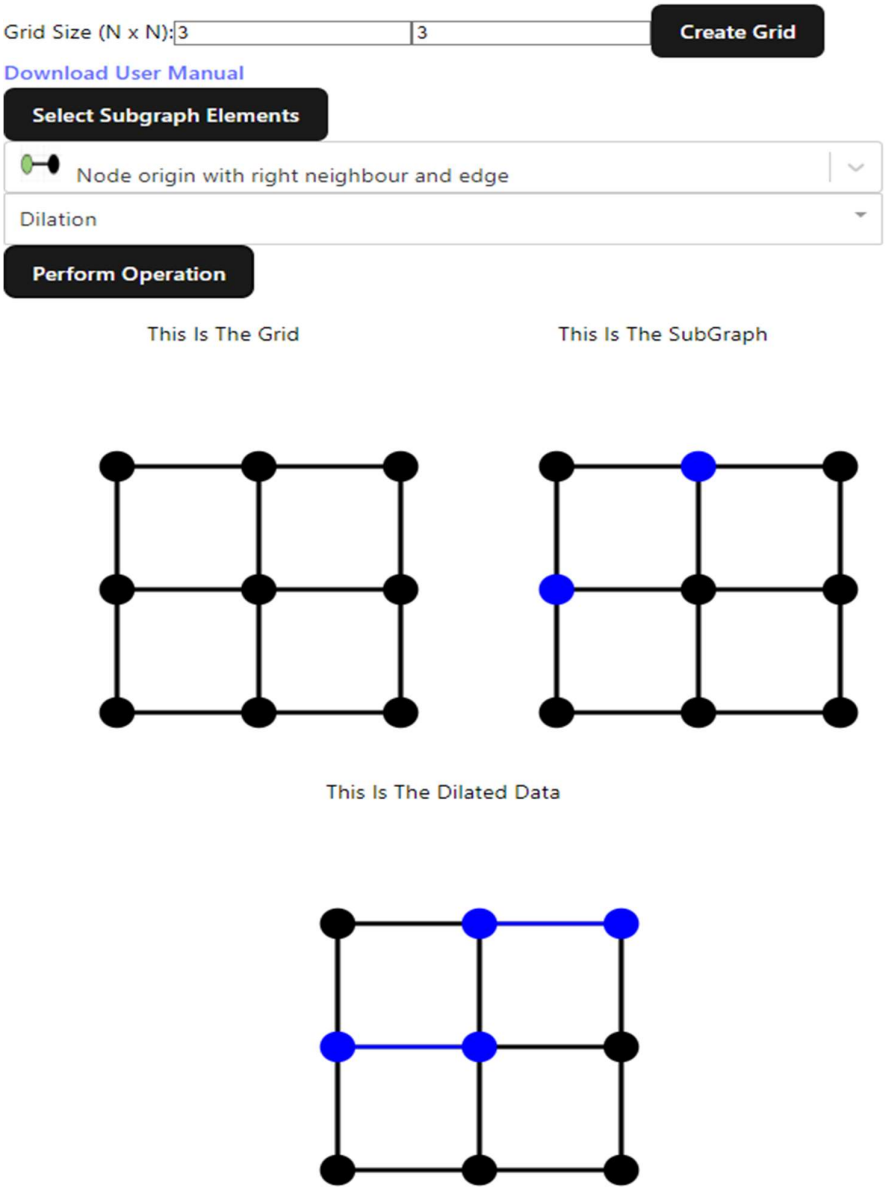



Fig 20 - Dilation of subgraph using a preset SE

### 2.4.4 Erosion

When the user selects “Erosion” from the MM operations dropdown and after selecting a SE and clicking on the “Perform Operation” button, erosion occurs. The grid dimensions, SE, and subgraph are then sent to the server. The server then tries to “match” the SE with the entire subgraph. For every successful match, the origin is added to the subgraph. The server then sends this data back to the client which is rendered in a similar way as the subgraph with the eroded nodes being a different colour.

Erosion, also like dilation does not ensure that all the nodes in the original subgraph will be present in the result. For example a SE that has a node as its origin with , its right neighbour and the horizontal edge between these nodes would only have the origin node in the resulting subgraph after erosion.

Erosion can be considered as the largest part of the input subgraph, when dilated, would not add nodes or edges outside the subgraph.

When the user performs erosion using a predefined SE, the server logic is similar to dilation but slightly altered. Flags are used to determine whether the pattern is within the subgraph, if any part of the pattern is outside or does not match fully with the SE, the flag is set to false. Only when the flag is true, that is when the pattern matches and is within the subgraph the origin is added to the resulting subgraph at that position.

When the users perform erosion with their own 3 part SE. The server starts by calculating the relative position of nodes and edges from the origin for each the Node, Horizontal edge and Vertical edge cases. Within each case, flags are used to determine whether the relative positions of nodes are within the subgraph and are part of the subgraph. The same is done for the edges . If the flag remains true, the origin is added at that particular position.

Edge Cases in Erosion: There are 2 edge cases in erosion.

1. No matches: When there are no matches found when performing dilation, it results in all the nodes and edges in the original subgraph to be present in the result.
2. Out of Bounds: There are certain cases where a match would occur but the elements that are to be added would be outside the original grid. In this case these nodes or edges are not added and hence not displayed. For example, if an SE that is of a line shape with a node as its origin matches with the outermost node in a subgraph, it will try to add nodes that are out of bounds

Fig 21 provides a visual example of how dilation is performed on graphs.

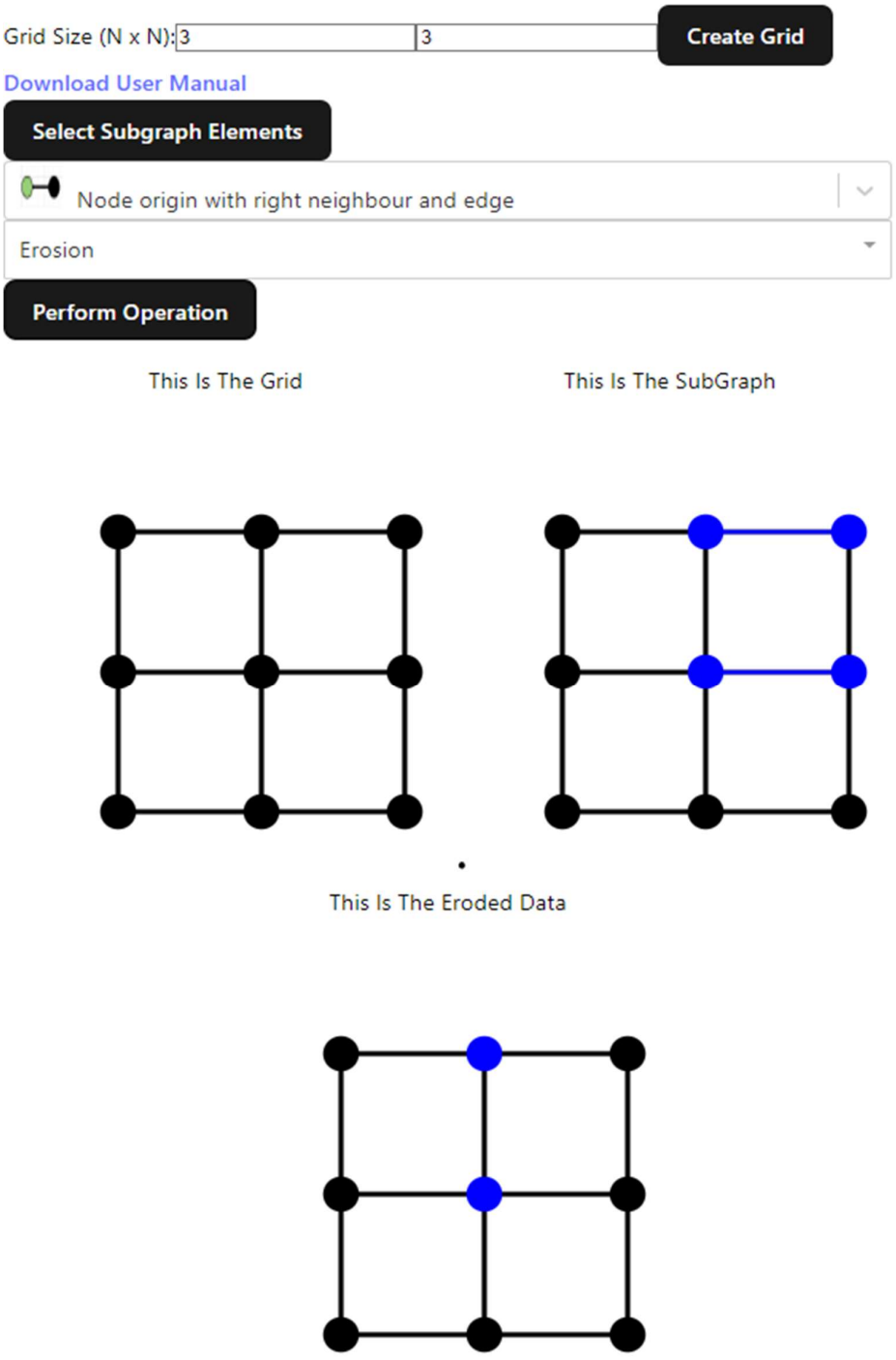



Fig 21 -Erosion of subgraph using preset SE

### 2.4.5 Opening and Closing

The implementation of opening and closing becomes a lot simpler as these are directly based on the dilation and erosion functions that were implemented earlier. The modularity of the dilation and erosion functions allow them to be called

The user can select “opening” in the dropdown menu, and after selecting a SE , opening occurs. The opening function is implemented by calling the erosion function and then passing the eroded nodes and edges to the dilation function (erosion followed by dilation). The resulting nodes and edges are then passed back to the client. This process is the same regardless of the type of SE the user provides (predefined or custom).

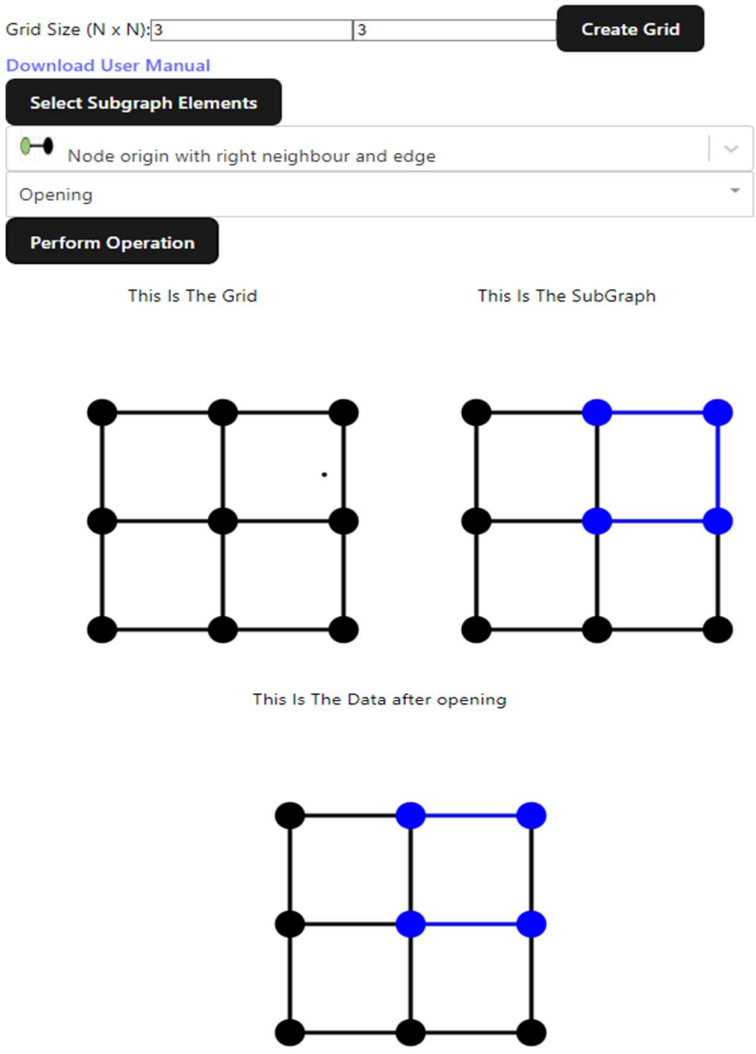



Fig 22 – Opening operation on a subgraph using a preset SE

The user can select “closing” from the dropdown menu, , and after selecting a SE , closing occurs. The closing function is implemented by calling the dilation function and then passing the dilated nodes and edges to the erosion function (dilation followed by erosion). The resulting nodes and edges are then passed back to the client. This process is the same regardless of the type of SE the user provides (predefined or custom) .

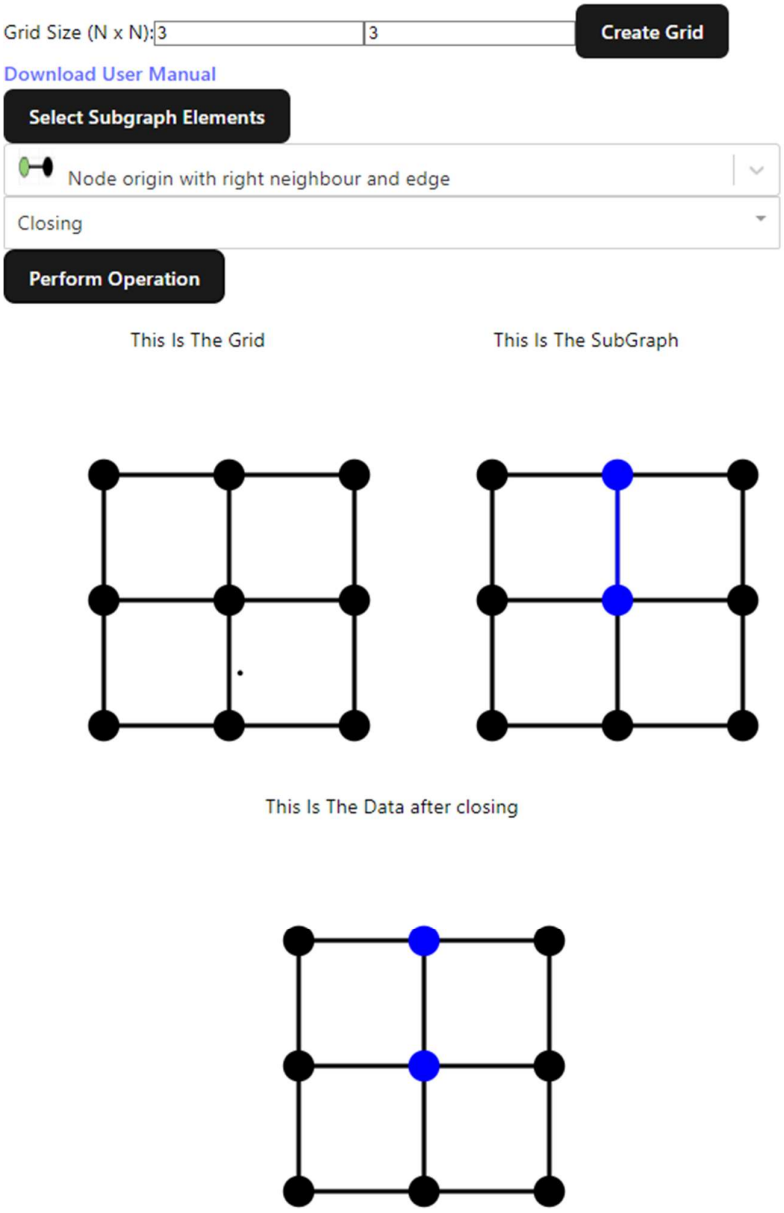



Fig 23 – Closing operation on a subgraph using a preset SE

# Chapter 3
# Results

## 3.1 Testing the Application

At the end of each sprint and before each supervisor meeting, the code was tested manually with a few known examples.

Unit testing was conducted on the server-side code during the final sprint to ensure that all functions required for the operations and the operations themselves provided the desired output. Unit testing means testing the code in a separate environment, usually testing the smallest parts such as functions, one function at a time[18]. This becomes particularly useful when it comes to functions that are used extensively, such as the helper functions for the MM operations as it ensures that these methods are accurate and able to handle all possible scenarios. This testing was crucial as it made resolving issues with the code was quicker and easier ,as the automated test cases test for basic functionality, edge cases and error cases, which has reduced the time taken to complete the code quality checks in the final sprint.

Jest JS was the unit testing framework that was selected for testing [19]. The extensive documentation , usability across multiple JavaScript frameworks and easy setup made it the best choice for this project. Fig shows an overview of each unit that was tested in this project. Functions that are used to create the data structure, helper functions ,basic cases and edge cases for dilation and erosion were tested.

```
> test
> jest

 PASS  ./CreateDS.test.js
 PASS  ./simpleMM.test.js
 PASS  ./helperFunctions.test.js

Test Suites: 3 passed, 3 total
Tests:       34 passed, 34 total
Snapshots:   0 total
Time:        1.329 s, estimated 2 s
Ran all test suites.
```

Fig 24 – Unit tests using Jest

## 3.2 User Evaluation

Along with automated tests. A thorough user evaluation of the system was conducted. This was done with the help of google forms. All participants who took part in the user evaluation had no prior knowledge of MM. This is mainly because MM is a niche topic and is not known to many users.

The form initially asks the users whether they are familiar with the concept of MM. If the user is not familiar, A brief explanation is provided to the user on the concept of MM, highlighting the purpose of the SE, what a 3 part SE is and brief explanation of each MM operation that is performed by the tool. If the user is already familiar with the concept , they are redirected to the evaluation directly.

The main purpose of this user evaluation is to evaluate the usability of the application. It also aims to evaluate how the requirements of this project ( mentioned in chapter 2) are met. 5 users were selected for this evaluation.

### 3.2.1 User Evaluation Questionnaire

The questionnaire is divided into 4 sub sections:

1. Part 1: Selecting a Grid and Subgraph: The user is asked to follow a series of tasks. First task is to input a grid and after which they are tasked to create a subgraph by following examples provided to them. They are then asked whether their output matched the expected result and then asked whether the instructions provided by the subgraph selector popup was adequate, how easy the process was (on a scale of 1 to 5). and  any feedback on this section.
2. Part 2: Performing Dilation and Erosion: For Dilation and Erosion , the user is first tasked to replicate the grid and subgraph provided by the form, and were then asked to select the SE and MM operation that was provided by the form. This section used known examples of Dilation and Erosion. The users are then asked if their output matched the example , how easy it was to perform these operations (on a scale of 1 to 5).
3. Part 3: Performing Opening and Closing: This section is similar to the previous section, where examples were provided for the user to follow along with. The users are then asked if their output matched the example , how easy it was to perform these operations (on a scale of 1 to 5) . The user is then asked to comment on how all the process of performing MM with predefined SE’s can be improved.

4. Part 4: Creating a 3 part SE: The user is directed on how to create a 3 part SE. The user is then asked how easy it was to create the 3 part SE and if there are any improvements on this process.

After the user completes all 4 sections, they are then encouraged to explore the application and are then asked to provide feedback on the application. Note the scale of 1 to 5 for ease of use refers 1 as difficult to use and 5 being easy to use.

### 3.2.2 Discussion of Results

**Task 1**: All the users have found the process of selecting a grid and subgraph easy, with 60% giving a score of 5 and 40% giving a score of 4. They have found that the instructions provided on the popup to be adequate and the process of selecting elements to be intuitive. Some users have suggested the idea of a “Clear Selections” button instead of closing the popup and re-opening the popup to make it easier to clear all selected nodes and edges in case of the user making a mistake in their selection. This feature must be extended to all other cases where popups are used such as the custom SE popups.

**Task 2 and 3**: All the users have found the process of applying MM operations with predefined structuring elements a simple process, with an average score of 4. They have found the predefined SE dropdown to be quite informative due to the image of each origin being displayed along with the text of each SE. Some users have mentioned that adding some basic examples would make it easier for new users to try the operations and compare with existing results.

One user had mentioned that the blue colour does not pair well with the black colour of the nodes and edges, and they have mentioned it being quite difficult to differentiate between selected elements and non-selected elements. This highlights the need for improvements in accessibility.

Users have also mentioned that a basic user manual explaining the concept and how to use the interface would be necessary.

One user has mentioned that after exploring the application, visualising the MM operations on larger grids was more challenging as the user had to now scroll to view the results.

**Task 4**: The users have given an average score of 3 on ease of use for selecting a 3 part SE. The users have found that the instructions on how to select the elements and origin to be adequate. One user has identified an error in the SE selection code, where the screen goes blank if the user has not selected an origin for the SE.

# Chapter 4
# Discussion

## 4.1 Conclusions

The user evaluations and unit testing have shown that users can perform MM operations on grid graphs of any dimension. Users were successfully able to select a subgraph , select either a predefined SE or input their own 3 part SE, select 1 of 4 provided MM operations and perform it on the subgraph , with the result being displayed. Users have also mentioned that the tasks were easy and intuitive, highlighting the ease of use of the application.

Some of the issues highlighted in the user evaluation have been rectified. A URL is provided ,which is the download link to a user manual that explains the concept of MM and provides a brief explanation of MM, SE and a quick guide on how to use the application.

A key issue that has been identified in the user evaluation is the lack of accessibility. Accessibility was not considered during development. Providing users the option to change the colours would have improved the experience for users who are sensitive to certain colours. The application does suffer from issues when the user tries to input graphs of larger dimensions. These graphs due to the rendering algorithm end up taking too much screen space , preventing all the visuals from being within the screen space, which forces the user to scroll to see the resulting subgraph at times, making it difficult to interpret the result. Ideally the display of all graphs and SE should fit within the screen.

Another issue that has been identified is the lack of examples provided. A few examples of known cases could have highlighted the functionality of the application and could have been used as a reference for new users to grasp the concept better.

In summary the application has met the deliverables as it is able to create subgraphs and perform MM on the subgraphs but requires some improvements to make it easier for all users to effectively utilise the features of the application.

## 4.2 Ideas for future work

Immediate improvements to the application would be to address the accessibility issue mentioned in section 4.1, which would mean identifying a different method to render the larger sized grids. Popups also need improvement to handle larger grids.

A few examples on how to perform the operations could be added in the form of images and video demonstrations.

With additional time to work on the project, a new aim would be to expand on the existing functionality of the application. In addition to grid graphs, the user should be able to perform MM on other types of graphs, such as 8 adjacent graphs and directed graphs. This would require more research on suitable data types to store the graphs and the redesign of existing methods to handle the new graph types, new methods to input the graphs a

Another objective would be to take the current functionality of the application and design several Application Program Interfaces (API's) for different programming languages. This could allow other developers to integrate MM operations into their projects and could reduce their development time, allowing the concept of MM to spread into more projects.

## List of References


1. *Serra, J. 1986. Introduction to mathematical morphology, Computer Vision, Graphics, and Image Processing Volume 35, Issue 3 pp 283-305. [Accessed 07/01/2024]*
2. Haralick, R.M., Sternberg, S.R. & Zhuang, X., 1987. Image Analysis Using Mathematical Morphology. IEEE Transactions on Pattern Analysis and Machine Intelligence, 9(4), pp.532-550. [Accessed 08/01/2024]
3. Lin, H., Zhao, J., Li, S. and Qiu, G., 2020. License plate location method based on edge detection and mathematical morphology. In: 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chongqing, China, 12-14 June 2020. IEEE, pp. 853-857. Available at: https://doi.org/10.1109/ITNEC48623.2020.9085121 [Accessed 08/01/2024].
4. Purwita, A. A., Adityowibowo, K., Dameitry, A. and Atman, M. W. S., 2011. Automated microaneurysm detection using mathematical morphology. In: 2011 2nd International Conference on Instrumentation, Communications, Information Technology, and Biomedical Engineering, Bandung, Indonesia, 8-9 November 2011. IEEE, pp. 117-120. [Online] Available at: https://doi.org/10.1109/ICICI-BME.2011.6108606 [Accessed 09/01/2024].
5. Cousty, J., Najman, L., Dias, F. and Serra, J., 2013. Morphological filtering on graphs. Computer Vision and Image Understanding, 117(4), pp.370-385. Available at: https://doi.org/10.1016/j.cviu.2012.08.016 [Accessed 09/01/2024].
6. Legland, D., Arganda-Carreras, I., & Andrey, P. (2016). MorphoLibJ: integrated library and plugins for mathematical morphology with ImageJ. Bioinformatics, 32(22), 3532–3534 [Online]. doi:10.1093/bioinformatics/btw413 [Accessed 09/01/2024]
7. Anon n.d. Mamba Image. Mamba-image.org. [Online]. [Accessed 09/01/2024]. Available from: https://www.mamba-image.org/.
8. OpenCV. OpenCV: Morphological Transformations. Opencv.org. [Online]. [Accessed 09/01/2024]. Available from: https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html.
9. Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu and the scikit-image contributors. scikit-image: Image processing in Python. PeerJ 2:e453 (2014) https://doi.org/10.7717/peerj.453
10. The MathWorks Inc. (2022). Morphological Operations Documentation, Natick, Massachusetts: The MathWorks Inc. https://www.mathworks.com. [Accessed 09/01/2024]

11. APM n.d. What is agile project management? Apm.co.uk. [Online]. [Accessed 17/01/2024]. Available from: https://www.apm.org.uk/resources/find-a-resource/agile-project-management/.
12. Adobe n.d. Waterfall Methodology: A Complete Guide. Adobe.com. [Online]. [Accessed 17/01/2024]. Available from: https://business.adobe.com/blog/basics/waterfall.
13. Tree Crown Size Estimated Using Image Processing: A Biodiversity Index for Sloping Subtropical Broad-Leaved Forests - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Operations-of-mathematical-morphology-for-a-specific-structuring-element-Binary-images_fig4_318790302 [Accessed 17/01/2024] [CC by 4.0]
14. *Programiz*. Adjacency matrix. *Programiz.com*. [Accessed 7/02/2024]. Available from: https://www.programiz.com/dsa/graph-adjacency-matrix.
15. Khan Academy n.d. Representing graphs (article). *Khan Academy*. [Online]. [Accessed 07/02/2024 2024]. Available from: https://www.khanacademy.org/computing/computer-science/algorithms/graph-representation/a/representing-graphs.
16. W3. SVG tutorial. *W3schools.com*. [Accessed 7 March 2024]. Available from: https://www.w3schools.com/graphics/svg_intro.asp.
17. Mustapha, R. 2023. How to make a clickable SVG map with HTML and CSS. *freecodecamp.org*. [Online]. [Accessed 7 March 2024]. Available from: https://www.freecodecamp.org/news/how-to-make-clickable-svg-map-html-css/.
18. pp_pankaj Follow, P. 2019. Unit testing - software testing. GeeksforGeeks. [ [Accessed 15 April 2024]. Available from: https://www.geeksforgeeks.org/unit-testing-software-testing/.
19. Anon n.d. Jest. Jestjs.io. [Online]. [Accessed 15 April 2024]. Available from: https://jestjs.io/.
20. W3C. Web Content Accessibility Guidelines (WCAG) 2.1. Www.w3.org. [Accessed 20 April 2024b]. Available from: https://www.w3.org/TR/WCAG21/.
21. Heusser, M. 2017. test-driven development (TDD). Software Quality. [Accessed 21 April 2024]. Available from: https://www.techtarget.com/searchsoftwarequality/definition/test-driven-development
22. A reconfigurable real-time morphological system for augmented vision - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Morphological-operations-a-Original-image-b-

dilated-image-and-c-eroded-image_fig1_260938543 [accessed 28 Apr, 2024] [CC by 2.0]

23. Saravana, C. n.d. Morphological operations. SlideShare. [Online]. [Accessed 28 April 2024]. Available from: https://www.slideshare.net/cs1973/morphological-operations.
24. Owens, R. 1997. Mathematical Morphology. Ed.ac.uk. [Accessed 29 April 2024]. Available from: https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT3/node3.html.

# Appendix A
# Self-appraisal

## A.1 Critical self-evaluation

Overall, I am satisfied with the outcome of this project as the feedback from user evaluation has given me a lot to think about in regards to future work along with the fact that all deliverables listed in this project were achieved in the given timeframe.

I was most excited for the software development part of this project. My experience with several frameworks was crucial in my selection of frameworks as for my project, JavaScript was the most powerful tool to help with visualization. The project planning methodologies and wireframe prototypes of the interface were industry standard and in line with my previous professional experiences. The initial coding process was the most challenging as selecting appropriate datatypes and facilitating user interaction for subgraph and SE selection were coded from scratch but after the initial hurdle , development of the operations and custom SE functionality became much easier.

I did underestimate the complexity of this project during the initial planning stage. The initial prototypes were incorrect, and the first MVP took much longer than expected to deliver. This was due to the lack of initial understanding on how these operations work on graphs as the implementation is different when compared to how the operations work on images.

A better approach to projects of this size in the future would be to introduce Test Driven Development (TDD) as part of the development process. It involves designing test cases before writing any code, which can result in higher quality code being produces initially when compared to regular development which involves coding first followed by testing.[21]

Another factor that slowed down development was the time it took for me to completely understand how the SE works when it is used to perform MM operations. In the initial stages, I was under the impression that hard coding basic cases would be enough but then understood that implementation of a 3 part SE would complete the project as it provides users more control as to what SE they choose when performing operations.

Feedback from user evaluation is also in line with what I want to improve on in this project. The interface needs improvement and a more detailed documentation explaining the concept of MM is needed. A more efficient data structure must be considered for any future development as the adjacency matrix performs well when the graphs are dense but in the case of larger graphs and sparse graphs would be inefficient. The program also needs to be

able to handle larger grid sizes, as currently they take up too much screen space, making it impossible to view the operations properly on the screen.

## A.2 Personal reflection and lessons learned

Before working on this project, I have had previous experience working on coding projects which at most have taken 2 months to complete. While previous projects have allowed me to understand how to translate technical requirements into code based solutions, I have never taken up a project that required extensive research and development across several months.

Taking up this project meant that I had no reliance on any external sources as no solutions are publicly available. This made me actively seek out advice from my supervisor, which resulted in almost weekly meetings during the development phase in the second semester. I have learnt that asking for advice and help is not a bad thing, and that the feedback from my supervisor, who is an expert in the field of MM , was invaluable when developing this project. His explanation on the topic is what helped me more than any online resources when it came to understanding the fundamental concepts.

This project also tested my ability to adapt to unexpected changes. My initial belief was that existing libraries will be enough but after understanding the concept realized that this would not be applicable to graphs. Initial prototypes took a lot longer than expected which caused additional delays in subsequent sprints of the project.

This project overall has helped me understand how to conduct proper research and reinforce my understanding of industry standards when it comes to software development.

## A.3 Legal, social, ethical and professional issues

### A.3.1 Legal issues

The Project does not face any legal issues. The project uses several open-source frameworks that permit the use and distribution of their software. D3.js while being an open-source framework required a copyright notice to be attached to the project, which is done so in a file called NOTICE.txt in the root directory of this project. In order to prevent any legal issues that occur from this project, the project uses an MIT open-source licence, clearly stating that reuse of the project is allowed but warranty not being provided. During user testing, users were given the sheet and consent form which highlights their role in the evaluation while keeping them anonymous.

The application does not collect any user information as the user is interacting with the application. This is due to the fact a user account feature was not considered during the development of the project. All images that have been used in the report have been referenced and attribution has been provided in cases where images are licenced.

### A.3.2 Social issues

Accessibility was not considered during the development of this project. The project did not consider users that have visual impairments such as colour blind individuals. The project could have followed the Web Content Accessibility Guidelines (WCAG) [20] to ensure that industry standard accessibility criteria were met.

The project is also only accessible on desktop machines only and mobile interfaces were not considered during development due to the number of grids that would need to be shrunk into a small mobile device.  There are some issues with visualization as the entire MM process could not be contained inside 1 single row, which would have been ideal.

### A.3.3 Ethical issues

Part of the evaluation of this project was done with the help of user feedback. Each user w given the project information sheet that states project aim , what data is collected from the user and ensuring that personal details of the user will not be collected. The user has the option to withdraw for the evaluation at any time. Consent forms were also handed out to each user which they have signed.

The accuracy of the methods described in this project were the main concern and the number 1 priority during development in the case that this tool is used to teach the concept of MM to new students.

### A.3.4 Professional issues

Professional standards were met throughout this project, from using industry standards in project management, version control tools such as GitHub and even in selection of appropriate frameworks for the development of this project. Code quality was maintained along with proper unit testing of code to test for basic functionality.

# Appendix B
# External Materials

The external materials used in this project are mentioned below:

- Project Frontend and Backend Configurator: ViteJS. https://vitejs.dev/
- Graph Drawing Framework: D3.js. https://d3js.org/
- Popups component for Subgraph and custom SE selection: reactjs-popup. https://www.npmjs.com/package/reactjs-popup
- Dropdown component for Preset MM operation: react-dropdown. https://www.npmjs.com/package/react-dropdown
- Facilitate requests to Server: axios. https://axios-http.com/
- Server Code Unit Testing: JestJS. https://jestjs.io/
- For Preset SE selection: React select: https://www.npmjs.com/package/react-select

# Appendix C: Additional Materials

## User Evaluation Questionnaire Questions

1. How Simple is Subgraph Selection on a scale of 1 to 5? (1: Difficult, 5: Easy)
2. Any feedback on how to improve the process of inputting and selecting a subgraph?
3. How Simple is it to perform Dilation using a predefined structuring element on a scale of 1 to 5? (1: Difficult, 5: Easy)
4. How Simple is it to perform Erosion using a predefined structuring element on a scale of 1 to 5? (1: Difficult, 5: Easy)
5. How Simple is it to perform Opening using a predefined structuring element on a scale of 1 to 5? (1: Difficult, 5: Easy)
6. How Simple is it to perform Closing using a predefined structuring element on a scale of 1 to 5? (1: Difficult, 5: Easy)
7. Any improvements in the tasks completed so far?
8. How Simple is it to create your own 3 part structuring element on a scale of 1 to 5? (1: Difficult, 5: Easy)
9. Any other feedback you can provide after exploring the functionality of the application?