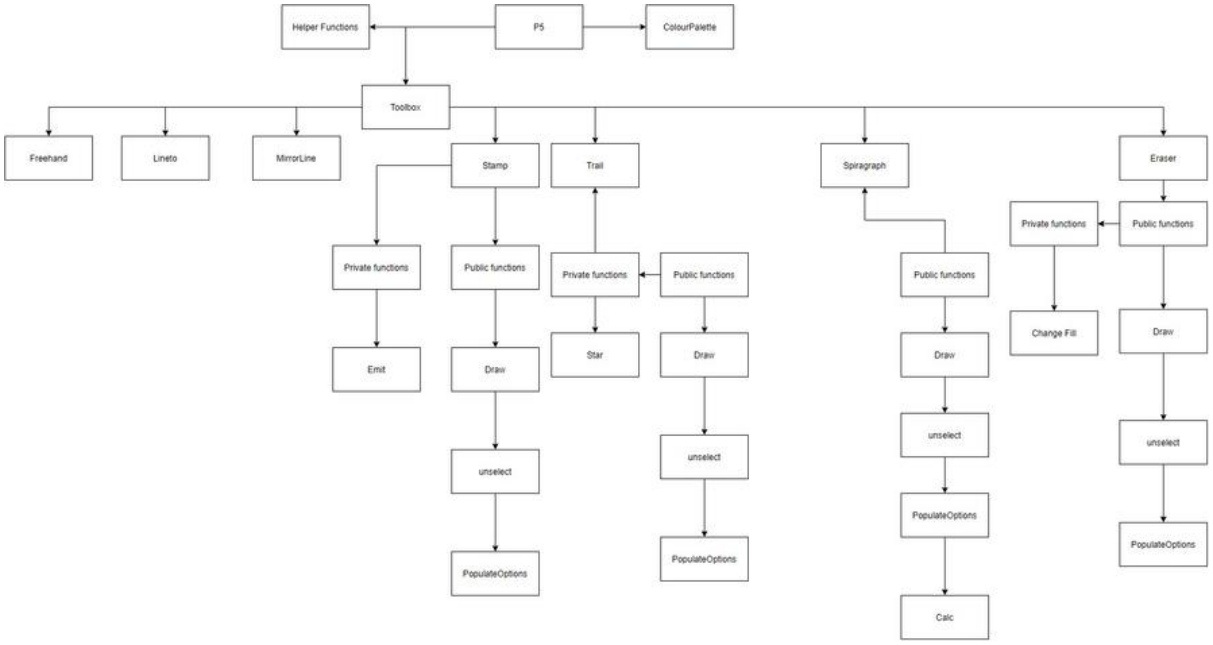
Drawing report by Mohamed Abdalla:

Implementation diagram:



Features implemented

- **Stamp**
- **Spirograph**
- **Colour palette extension**
- **Eraser**
- **Trail**

Planning:

My primary audience is artists. My key concern when developing the extensions was ensuring that the tools were universally understandable and avoiding using the English language on tools rather, I used icons or shapes to communicate what the current tool was.

To plan for the features in my drawing I used numerous YouTube videos and p5 references to learn how to implement the different tools which I will reference below.

References:

https://www.youtube.com/watch?v=0dwJ-bkJwDI

https://www.youtube.com/watch?v=qm1k4qOwlg8&t=7473s

https://www.youtube.com/watch?v=aEptSB3fbqM

https://p5js.org/examples/form-star.html

https://editor.p5js.org/maxremfort/sketches/hqxwqoF-w

https://editor.p5js.org/Pole/sketches/QdQuBgD9K

https://editor.p5js.org/cassie/sketches/HJC08Is67

Code structure:

I structed my code by separating each tool into different files using a modularised approach. Each tool was constructed using a constructor function and I only made the essential details of the tools available thus minimising the effect each tool would have on other tools inside the drawing app.

The drawing app is not aware of the tools called with the toolbox constructor function since I took an object-oriented approach. The toolbox is only aware of the essential functions inside each drawing tool to be used in the drawing app.

I limited the number of parameters used by either the public or private functions inside the drawing tools to minimize the errors which would be caused if every function either had too many parameters or was being called in other functions. This would result in code that would become quite unreadable, so I decided to avoid this.

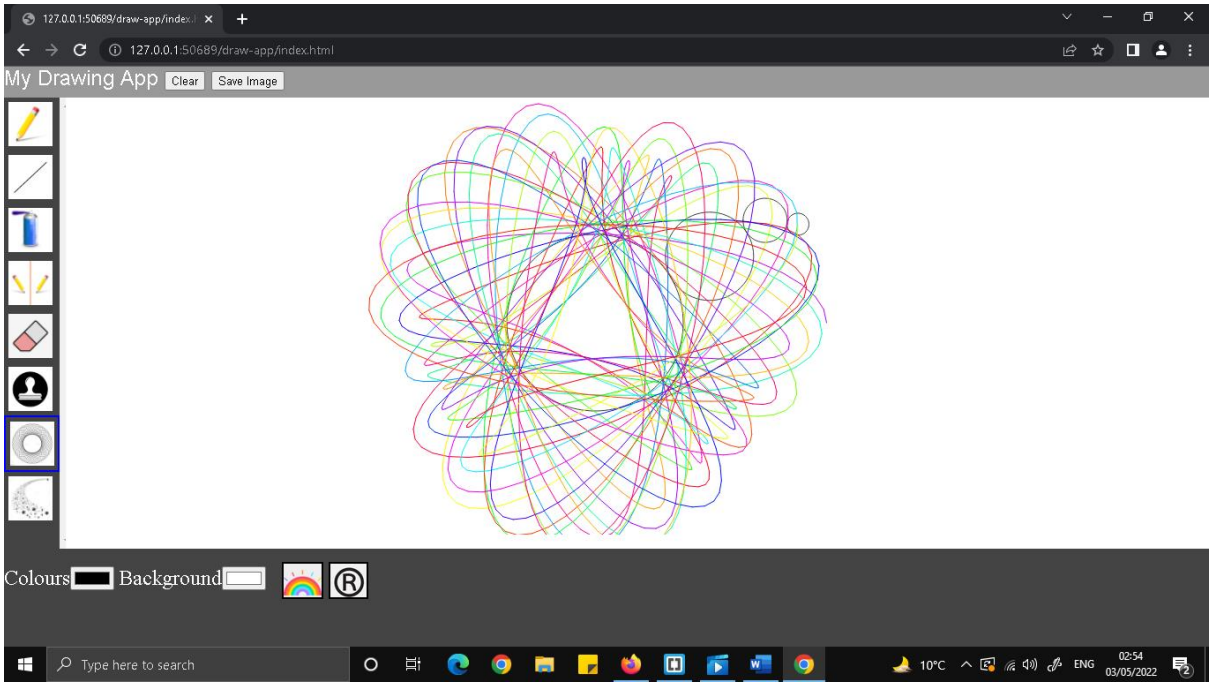
I made use of ES6 variables such as let and const to avoid hoisting which is an issue with the var keyword where the variable is hoisted to the top of its scope. This can make debugging tools even more complicated because of hoisting it's hard to tell where the variable might be defined. Const and let avoid this by not hoisting furthermore const stops the value of the variable being changed throughout the program making debugging a lot easier.

Coding Techniques:

- ES6 variables (let, const)
- Arrays, 2D arrays, arrays of objects and arrays of arrays
- Private Objects and constructors
- Public and private functions
- Iterations using for of or standard iteration
- Ternary operator, switch operator and if statements
- Shape functions
- jQuery

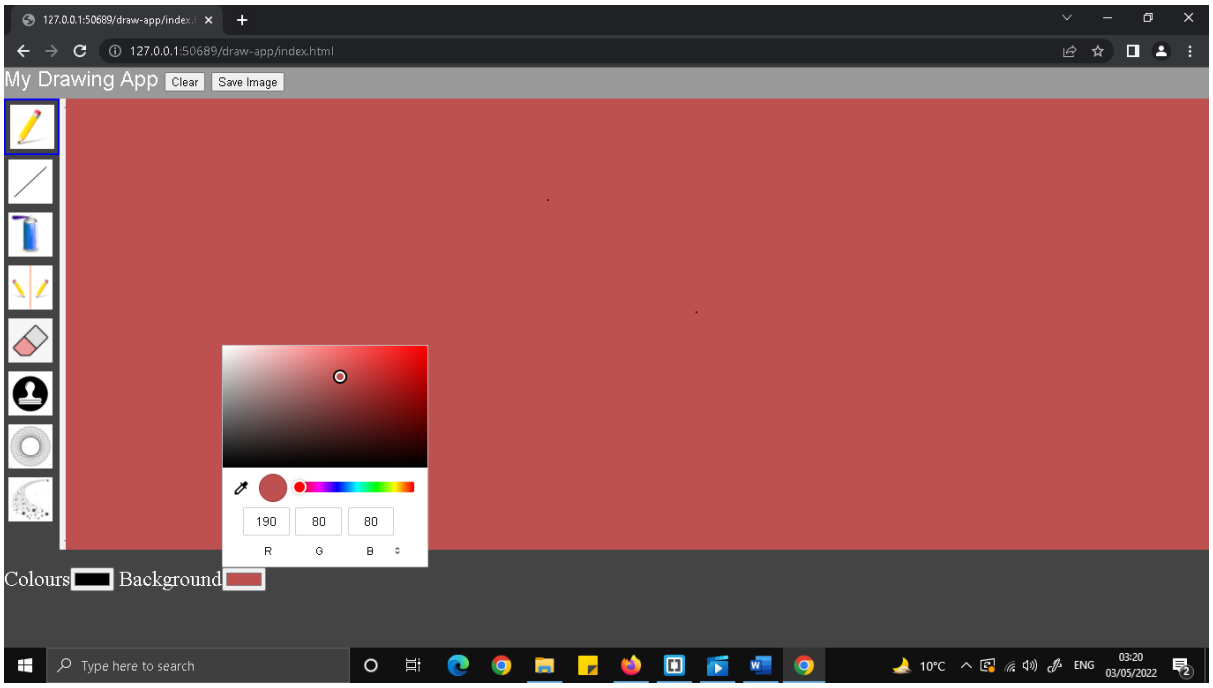
Spirograph Tool:

One of the changes I made was to the spirograph tool. My goal was to make the lines being drawn appear to be a rainbow of lines which was technically challenging because it would require picking a certain point in the lines being drawn then change the stroke of that line while it was being drawn to the screen. To complete this task, I would need a 2D array. First, I created a function that would calculate the x position, y position and radius of the lines. Using the newly created function I inserted x position, y position and radius arrays into the function as parameters. Once I had the positions, I simply pushed the positions into a different array called paths then I iterated over the paths array to draw the vertices while setting the stroke using the path colour variable which I Incremented by a value every loop of the iteration.



Background changer:

To change the background of the canvas was very challenging to do with JavaScript alone because simply changing the background would delete all the drawings currently on the canvas so instead, I decided to change the container background that the canvas was on. I used JQeury to select the container ID then update the background colour using the values from another colour picker html element I created that was on the page.



Evaluation:

Improvements that could be made to stamps:

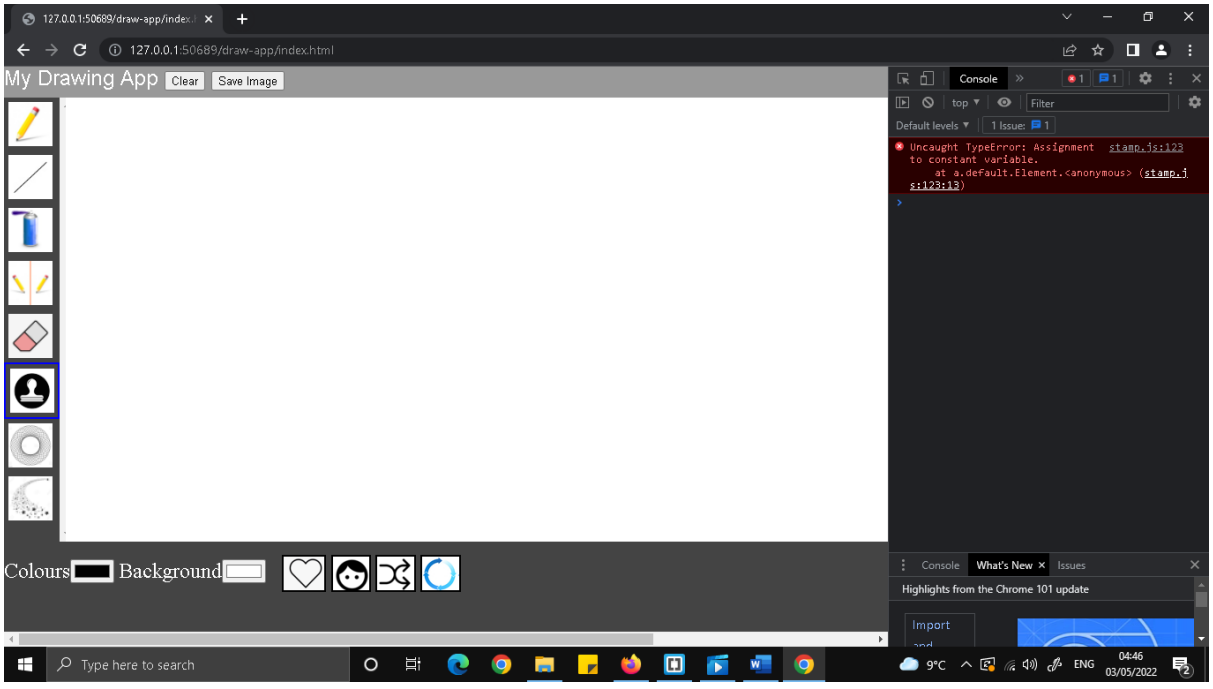
- Resizable stamps
- A load button where the user can upload images to stamp on the canvas
- Stamps that can be filled in

Disadvantages with stamp tool:

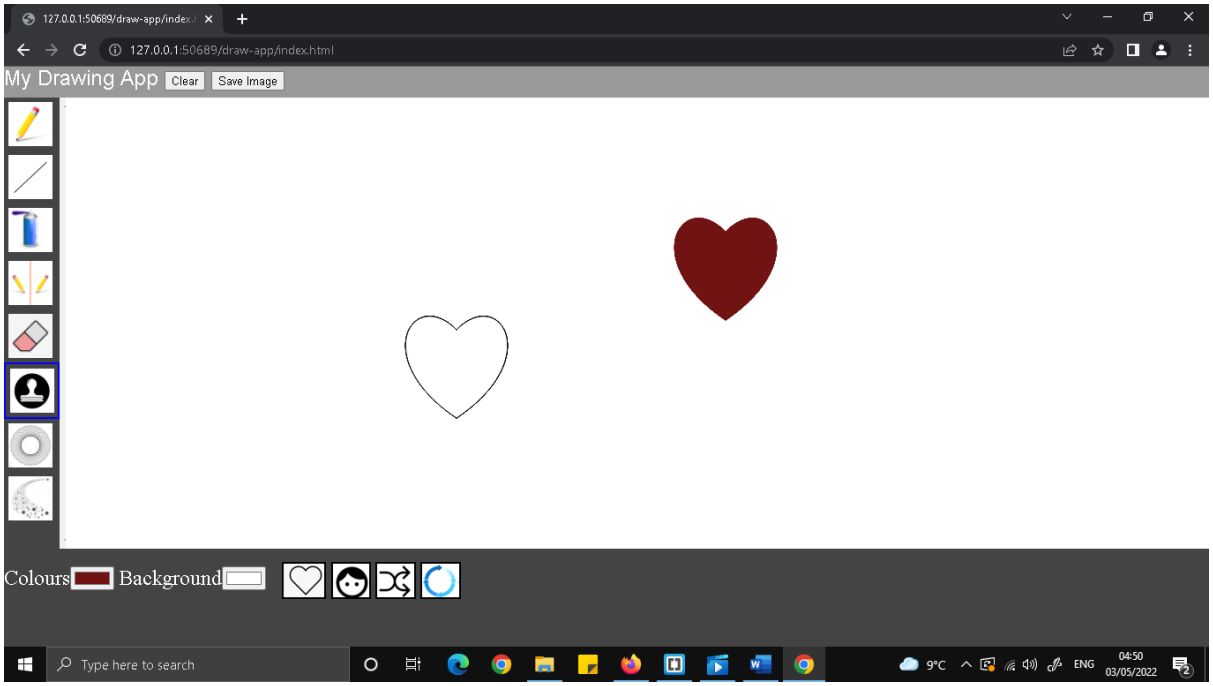
- Leaves a trail when the mouse is held down and moved
- The fill of the shapes can rarely be interfered with by other tools in the drawing app
- There is a large amount of code written for one constructor and could possibly be shortened

Bugs:

One of the bugs I encountered with the stamps was not being to draw the stamps when I clicked the option buttons.



Check was a variable that was being changed throughout the program so const would not work so I changed the line into let instead of const and that fixed the issue.



Improvement that could be made to trail tool:

- 3D shape trails
- Changing the fill of the trail while the mouse is moving
- Using a load button to add custom trails
- Creating trails that can rotate while the cursor is moving

Improvements that could be made to spirograph tool:

- More spirograph options to draw
- 3D spirograph
- Rotating spirographs
- Small spirographs drawn using the mouseX and mouseY positions

Progress log:

https://trello.com/invite/b/AKmY1tEr/a9145eb3e1728b21c1c9408d5b5a1ba9/work