Final Project – Report: Sumaiya Chowdhury and Aliyah Sayed

Planning stage:

Week 1:

We took this week to plan out and decide on what application to work from and what features to implement. Our final decision was to build off the drawing application.

We then took the time to consider what new features and tools could be added to either expand on or enhance user experience, as seen in the list below.

Potential features to implement:

- Stamp tool
- Shape tool (e.g., rectangle, ellipses)
- Fill tool
- Eraser
- Blur tool
- Different brush sizes
- Eyedropper tool
- Import image
- Gradient tool
- Texture brushes
- Background colours
- Spirograph

By adding these features, we wanted to either make the application easier to use, or expand what can be done within the app.

Week 2:

We started working on expanding the drawing application and looking at how to implement new features we had considered in the previous week. We began by testing the already existing code by slightly modifying it and seeing how it changes the output, this allowed us to gather a better understanding of how the application works and how and what features we could add to improve the application.

We also started to work on features such as an eraser tool.

Week 3:

We continued implementing new tools to the drawing application. First of all, we began implementing the stamp tools and brush thickness slider.

We considered making changes to the apps design and user interface to improve utility and user experience. Our aim is to make all available features look cohesive and easy to find and

select. To start with we began looking at other already existing drawing software, we took note of how each software's UI improves their usability and how this could be translated to our application. As we added features such as the brush thickness slider, we began to edit our applications user interface so that these features are easily available at all times for users to interact with.

Implementation stage:

This part of development spanned over week 3 to the 20th of April. We spent this time implementing new tools into our drawing app. Below we have documented our reasoning for making each feature and the process of adding it to the app.

Stamp tools:

We made two stamp tools, each with different shapes. By using an if statement, the function will only draw the stamp onto canvas if the user pressed the mouse. The star shape was made through the use of beginShape(), while the heart stamp consists of ellipses and triangles. Both stamps have coordinates adjusted by the mouseX and mouseY, meaning that the stamps position will the accurate to where user clicks their mouse. Both functions also allow the user to print the stamp multiple times when dragging their mouse.

Code below:

starStampTool.js

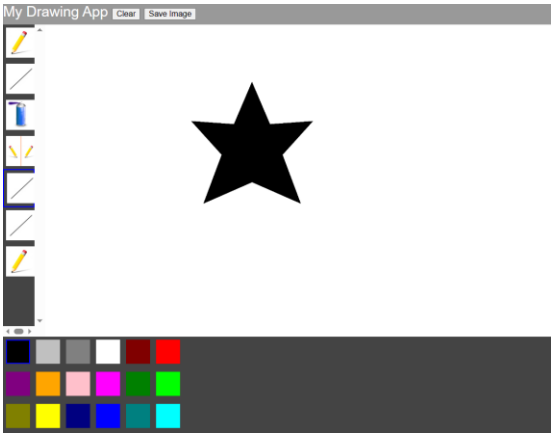
```
// stamp tool function
function starStampTool(){
    this.icon = "assets/lineTo.jpg"; //add new icon image later
    this.name = "starStamp";

    this.draw = function(x,y){
        if(mouseIsPressed){
            beginShape();
            vertex(mouseX + 20, mouseY - 50);
            vertex(mouseX - 10, mouseY + 20);
            vertex(mouseX - 80, mouseY + 15);
            vertex(mouseX - 30, mouseY + 70);
            vertex(mouseX - 60, mouseY + 150);
            vertex(mouseX + 20, mouseY + 115);
            vertex(mouseX + 100, mouseY + 150);
            vertex(mouseX + 70, mouseY + 70);
            vertex(mouseX + 120, mouseY + 15);
            vertex(mouseX + 50, mouseY + 20);
            endShape();
        }
    };
}
```

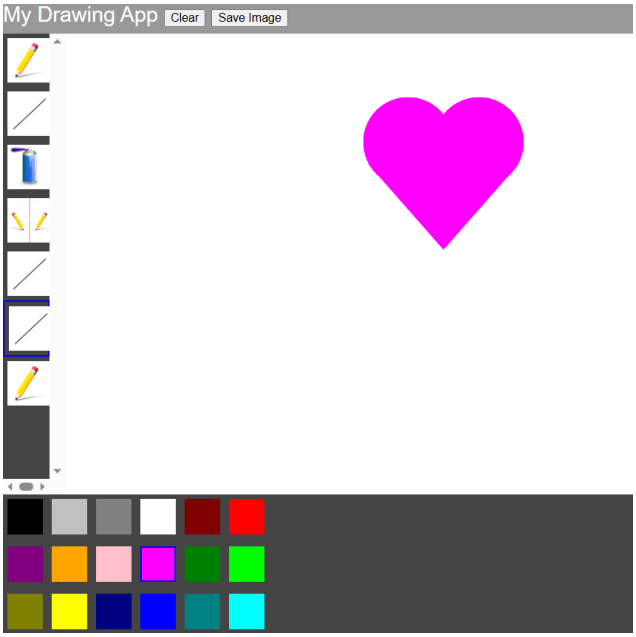
heartStampTool.js

```
// stamp tool function
function heartStampTool(){
    this.icon = "assets/lineTo.jpg"; //add new icon image later
    this.name = "heartStamp";

    this.draw = function(x,y){
        if(mouseIsPressed){
            ellipse(mouseX - 40, mouseY, 100, 100);
            ellipse(mouseX + 40, mouseY, 100, 100);
            triangle(mouseX - 83.8, mouseY + 25, mouseX + 83.8, mouseY + 25, mouseX, mouseY + 120);
        }
    };
}
```

Output:

When mouse is clicked.





When mouse is dragged.

Spirograph tool:

We also implemented a tool that produces spiral-like patterns that the users can adjust through changing the radius values using sliders. The value of the radius variables is updated when the value of the corresponding sliders is changed.

Code below:

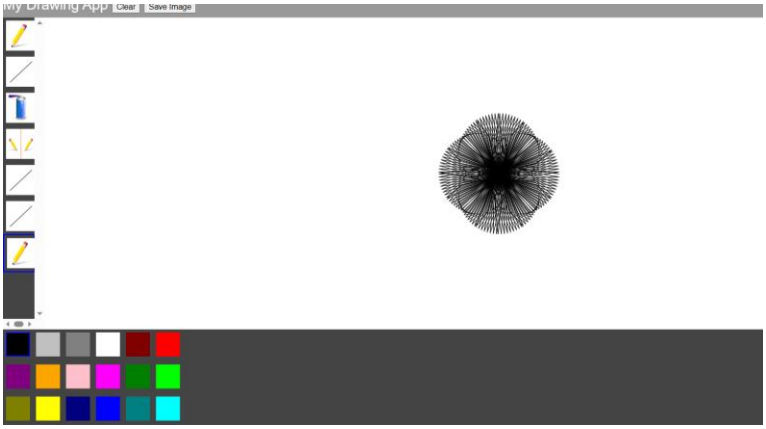
spirographTool.js

This draws the spirograph:

```
    this.draw = function(){

        translate(width / 2, height / 2);

        var x1 = radius1 * cos(angle1);
        var y1 = radius1 * sin(angle1);

        var x2 = x1 + radius2 * cos(angle2);
        var y2 = y1 + radius2 * sin(angle2);

        line(previousX, previousY, x2, y2);

        previousX = x2;
        previousY = y2;

        angle1 += 1;
        angle2 += 5;

        fill(0);
        raduis1_text = text("radius 1 = " + radius1, 20, offset - 20);
        raduis2_text = text("radius 2 = " + radius2, 40, (offset * 2) - 20);

    };
```

Output:

When radius1 = 50:



Eraser:

An eraser tool was also added to allow users to erase what they've drawn. Like the other brush tools, the eraser's size can be changed through the brush width slider. After implementing the background feature, we needed to adapt the eraser function to recognize when the user had changed the canvas colour. The function was edited to change the eraser tool's colour to the current background colour selected.

Code below:

eraser.js

```
function Eraser(){
    //set an icon and a name for the object
    this.icon = "assets/eraser.jpg";
    this.name = "eraser";

    //to smoothly draw we'll draw a line from the previous mouse location
    //to the current mouse location. The following values store
    //the locations from the last frame. They are -1 to start with because
    //we haven't started drawing yet.

    let previousMouseX = -1;
    let previousMouseY = -1;

    this.draw = function(){

        //set ink colour to colour selector value
        let bgColourInput = select('#bgColour');
        let bgColour = bgColourInput.value();
        stroke(bgColour);

        //set stroke weight to slider value
        let thickness = select('#penRange').value();
        strokeWeight(thickness);

        //if the mouse is pressed
        if(mouseIsPressed){
            //check if they previousX and Y are -1. set them to the current
            //mouse X and Y if they are.
            if (previousMouseX == -1){
                previousMouseX = mouseX;
                previousMouseY = mouseY;
            }
            //if we already have values for previousX and Y we can draw a line from
            //there to the current mouse location
            else{
                line(previousMouseX, previousMouseY, mouseX, mouseY);
                previousMouseX = mouseX;
                previousMouseY = mouseY;
            }
        }
        //if the user has released the mouse we want to set the previousMouse values
        //back to -1.
        else{
            previousMouseX = -1;
            previousMouseY = -1;
        }
    };
}
```

Colour selector:

To increase the user's options, the original colour pallete was replaced with a new colour selector. This changes the colour of all brush tools, including the mirror, stamp and spirograph tools. With the new colour selector, it could make things more difficult for users as it will be more difficult to find the exact colour they had used, unlike when using the original colour pallete. This problem is solved though adding an eyedropper tool, this allows users to take any colour already drawn to the canvas and instantly apply it to their brush.

Testing stage:

We tested each new function under various scenarios, e.g., with different colours applies or brush width, to ensure they all worked as planned without any unexpected errors. We were able to go through and correct any occuring issue to make sure the end result is free of problems or bugs. We also cleaned up the code, putting the tools under their own folder to further organize the files and to keep things easy to find. Comments were also added throughout the code to make it more comprehensible, this also made testing and re-editing code more efficient as it was easier to find a specific piece of code or to understand it again if we were revisiting a function we made earlier.

Self-Evaluation:

Overall, though adding new tools and features we were able to successfully expand on the original drawing app and improve user experience. After testing each newly implemented feature thoroughly, we confirmed that they work as intended without any bugs.

Users now have more options when drawing, through the additional tools such as the shape stamps, spirograph tools and background colour selector. This will make our app more interesting for potential users as there are more possibilities for what can be drawn.

The application has also been optimized though tools such as the eraser, colour selector, eyedropper tool and brush thickness slider. These give the user more control as well as reducing the limitations of the original code. Users are able to remove mistakes though the eraser tool. By replacing the previous colour pallete which had a limited choice of colour with our new colour selector feature and eyedropper tool, there is now a much bigger selection of colours to use.

While we were able to achieve our goal of expanding the scope of the drawing app, there are still aspects that we think can still be improved on. Overall, the option for freehanded brushes such as the freehand tool and spray can tool, is quite limited, which overall reduces what the user can create using our drawing tool.

If we were to revisit this project in the future, we would like to further expand on the code by adding more tools to either make the application less limiting and to make it more convenient for people to use.

Some of the features we would like to include would be a fill tool, to fill in gaps in the user's drawings. This would make the drawing application easier and faster to use as they can save the time of manually filling in a space with the freehand tool. In comparison to other drawing applications such as Adobe Photoshop or Procreate, we have a limited option for brushes, this may make our app less enjoyable compared to others. To change this, we would like to have more variety for the brushes available by implementing texture brushes, e.g., watercolour brush, stipple brush or a chalk brush. To give users more creative freedom, we would also like to implement an import image function so that people can move saved images in their files to the application’s canvas.