❖Names and IDs:

- Badr Elsayed - **22010664**

- Adham Anas - **22010601**

- Nour Khaled Mohamed - **22011319**

- Ali El-Deen Maher – **22010934**

❖Steps required to run code:

1. Backend:
   - Open the Paint_Backend folder using IntelliJ IDE or any other IDE, run the PaintBackendApplication.java class on port 9090.
2. Frontend:
   - Open the paint-frontend folder using visual studio IDE, then open the terminal of the IDE, and write npm install in the terminal.
   - Then write "npm run dev" in the terminal to open the project, on port "http://localhost:5173/"
3. Then you can use the paint application and start drawing.

❖UML diagram:

UML Class Diagram — Paint Application Backend

PaintBackendApplication
+main(String[] args):void;

Controller
-service: Service;
-factory: ShapeFactory;
+main(String[] args):void;
+drawShape (@RequestBody @NonNull DefaultShape s):void;
+editShape (@RequestBody @NonNull DefaultShape s):void;
+delete(@RequestParam String id):void;
+loadJson(@RequestParam String fileName, @RequestParam String path):SaveData;
+saveJson(@RequestParam String fileName, @RequestParam String path):SaveData;
+saveXml(@RequestParam String fileName, @RequestParam String path):SaveData;
+loadXml(@RequestParam String fileName, @RequestParam String path):SaveData;
+undo():SaveData;
+redo():SaveData;

Service
-shapes:Vector<Shape>;
-initialSaveData:SaveData;
-Undo:Stack<SaveData>;
-Redo:Stack<SaveData>;
-currentState:SaveData;
-zIndexTracker:int;
+toString():String;
+addShape (Shape s):void;
+edit (Shape s):void;
+delete(String id):void;
+saveJson (String filename, String path, int zIndexTracker):void;
+loadJson(String path):SaveData;
+saveXml(String filename, String path, int zIndexTracker):void;
+loadXml (String path):SaveData;
+ShapeToDefault(Shape s):DefaultShape;
+Undo():void;
+Redo():void;
+UndoRedoHandler():void;
+ShapeToDefaultList():Vector<DefaultShape>;
+defaultToShapes(Vector<DefaultShape> defaultShapes):void;

DefaultShape
- ID: String;
-type:String;
-x: double;
-y: double;
-fill_Colour:String;
-stroke_Colour:String;
-strokeWidth:double;
-scaleX:double;
-scaleY:double;
-width:double;
-height:double;
-radius:double;
-radiusX:double;
-radiusY:double;
-points:vector<double>;
-rotation:double;
-deleted:boolean;
-zIndex:int;
+DefaultShape();
+isDeleted():boolean;
+getDeleted():boolean;
+setDeleted():void;
+getZIndex():int;
+setZIndex(int zIndex):void;
+getID():String;
+setID(String ID):void;
+getType():String;
+setType(String type):void;
+getX():double;
+setX(double x):void;
+getY():double;
+setY(double y):void;
+getFill_Colour():String;
+setFill_Colour(String fill_Colour):void;
+getStroke_Colour():String;
+setStroke_Colour(String stroke_Colour):void;
+getStrokeWidth():double;
+setStrokeWidth(double strokeWidth):void;
+getScaleX():double;
+setScaleX(double scaleX):void;
+getScaleY():double;
+setScaleY(double scaleY):void;
+getRotation():double;
+setRotation(double rotation):void;
+getWidth():double;
+setWidth(double width):void;
+getHeight():double;
+setHeight(double height):void;
+getRadius():double;
+setRadius(double radius):void;
+getRadiusX():double;
+setRadiusX(double radiusX):void;
+getRadiusY():double;
+setRadiusY(double radiusY):void;
+getPoints():Vector<Double>;
+setPoints(Vector<Double> points):void;

Shape
- ID: String;
-type:String;
-x: double;
-y: double;
-fill_Colour:String;
-stroke_Colour:String;
-strokeWidth:double;
-scaleX:double;
-scaleY:double;
-rotation:double;
-deleted:boolean;
-zIndex:int;
+isDeleted():boolean;
+getDeleted():boolean;
+setDeleted():void;
+getZIndex():int;
+setZIndex(int zIndex):void;
+getID():String;
+setID(String ID):void;
+getType():String;
+setType(String type):void;
+getX():double;
+setX(double x):void;
+getY():double;
+setY(double y):void;
+getFill_Colour():String;
+setFill_Colour(String fill_Colour):void;
+getStroke_Colour():String;
+setStroke_Colour(String stroke_Colour):void;
+getStrokeWidth():double;
+setStrokeWidth(double strokeWidth):void;
+getScaleX():double;
+setScaleX(double scaleX):void;
+getScaleY():double;
+setScaleY(double scaleY):void;
+getRotation():double;
+setRotation(double rotation):void;
+getRadius():double;
+setRadius(double radius):void;
+clone():Shape;

Circle
#radius:double;
+clone():Circle;
+getRadius():double;
+setRadius(double radius):void;

Ellipse
- radiusX:double;
- radiusY:double;
+clone():Ellipse;
+getRadiusX():double;
+getRadiusY():double;
+setRadiusX(double radiusX):void;
+setRadiusY(double radiusY):void;

FreeDrawing
-points:Vector<Double>;
+clone():FreeDrawing;
+getPoints():Vector<Double>;
+setPoints(Vector<Double> points):void;

LineSegment
-points:Vector<Double>;
+clone():LineSegment;
+getPoints():Vector<Double>;
+setPoints(Vector<Double> points):void;

Rectangle
-width:double;
-height:double;
+clone():Rectangle;
+getWidth():double;
+setWidth(double width):void;
+getHeight():double;
+setHeight(double height):void;

Square
-width:double;
+clone():Square;
+getWidth():double;
+setWidth(double width):void;
+getHeight():double;

Triangle
-radius:double;
+clone():Triangle;
+getRadius():double;
+setRadius(double radius):void;

ShapeFactory
+createShape(boolean deleted,int zIndex,String ID,String type, double x, double y, String fill_Colour, String stroke_Colour,double strokeWidth, double scaleX, double scaleY, double rotation,double width, double height,double radius,double radiusX, double radiusY, Vector<Double> points):Shape;

Relationships: Calls, Asks, Uses, Extends, Creates

❖ Design Pattern Applied:

1. Factory design pattern:
   - To create the appropriate form that use several classes of shapes that inherit from an abstract class (Shape), we created a factory class(ShapeFactory). The factory class is used by another class (Controller), which is the client and contains all the information about the classes that will be constructed.

2. Prototype design pattern:
   - New shapes are created using the copiedShape as a prototype. A shape is saved as the copiedShape object, which acts as a template, when it is copied (handleCopy). The prototype shape is cloned during pasting (handlePaste), and properties such as id, x, y, and zindex are changed to suit the new context, creating a new shape (pastedShape).

❖ Design decisions:

1. Tool selection:
   - Implemented a tool selection mechanism to allow users to switch between drawing, selecting, and resizing shapes seamlessly.

❖Snapshots UI:
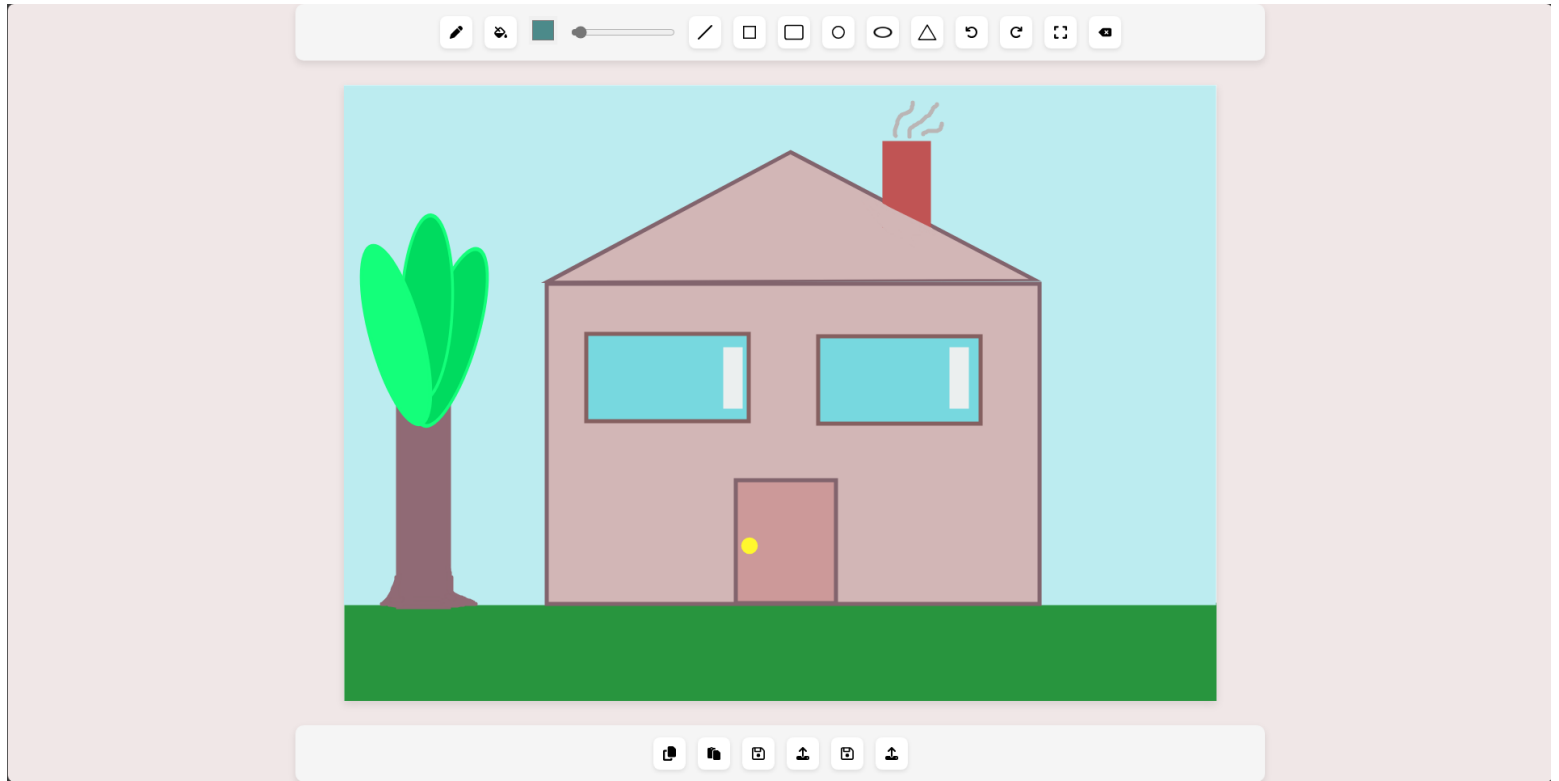
❖User guide:
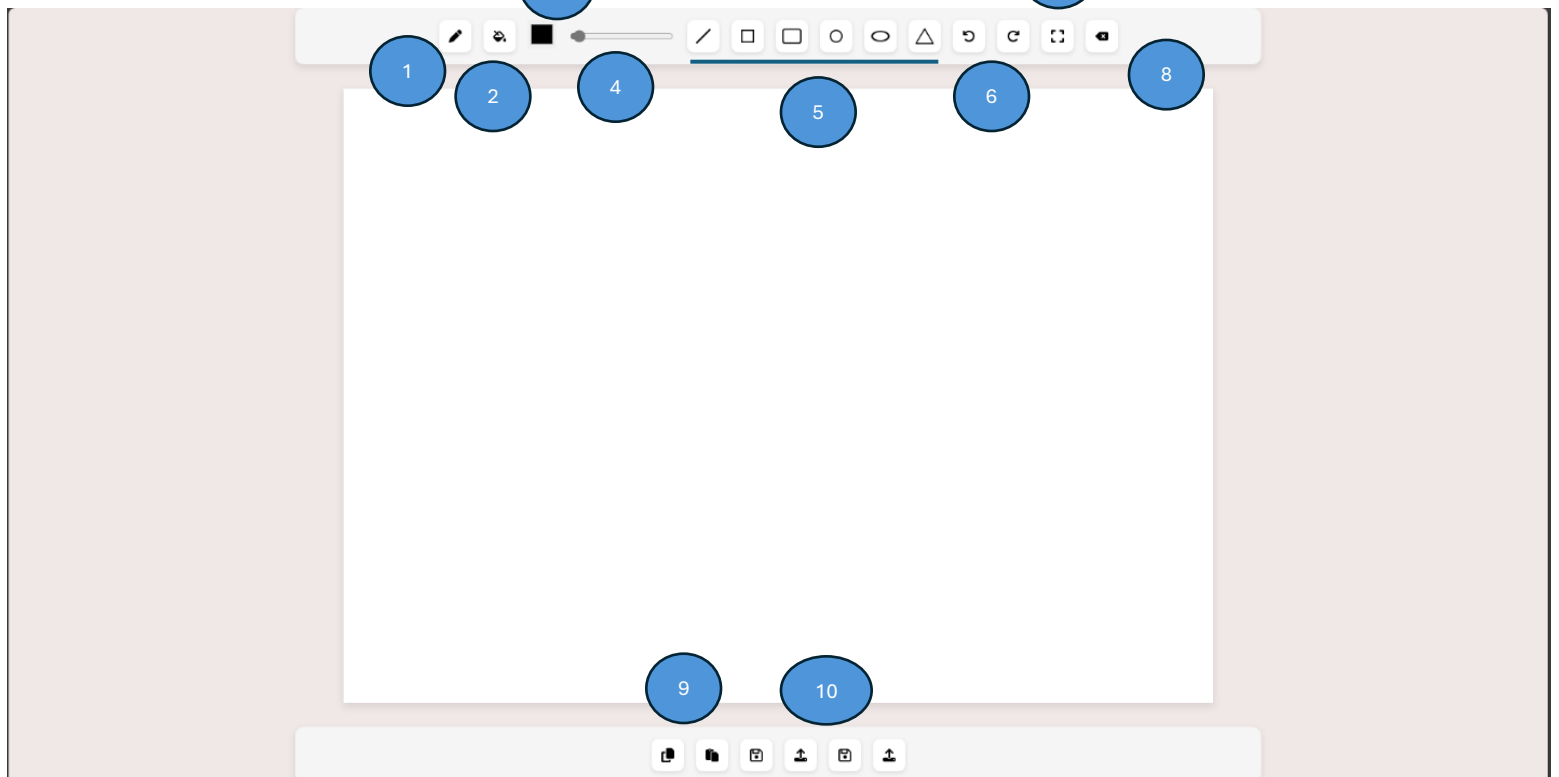
1. Free hand tool for drawing.
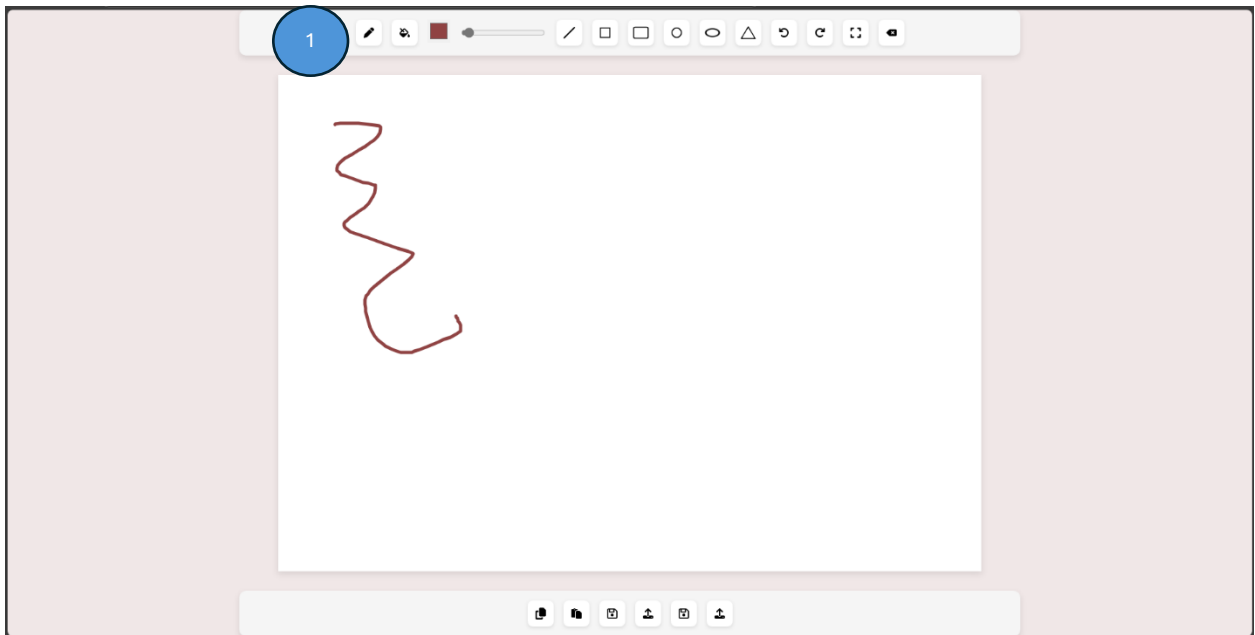
5. Choose any shape you want to draw then click and drag on canvas!
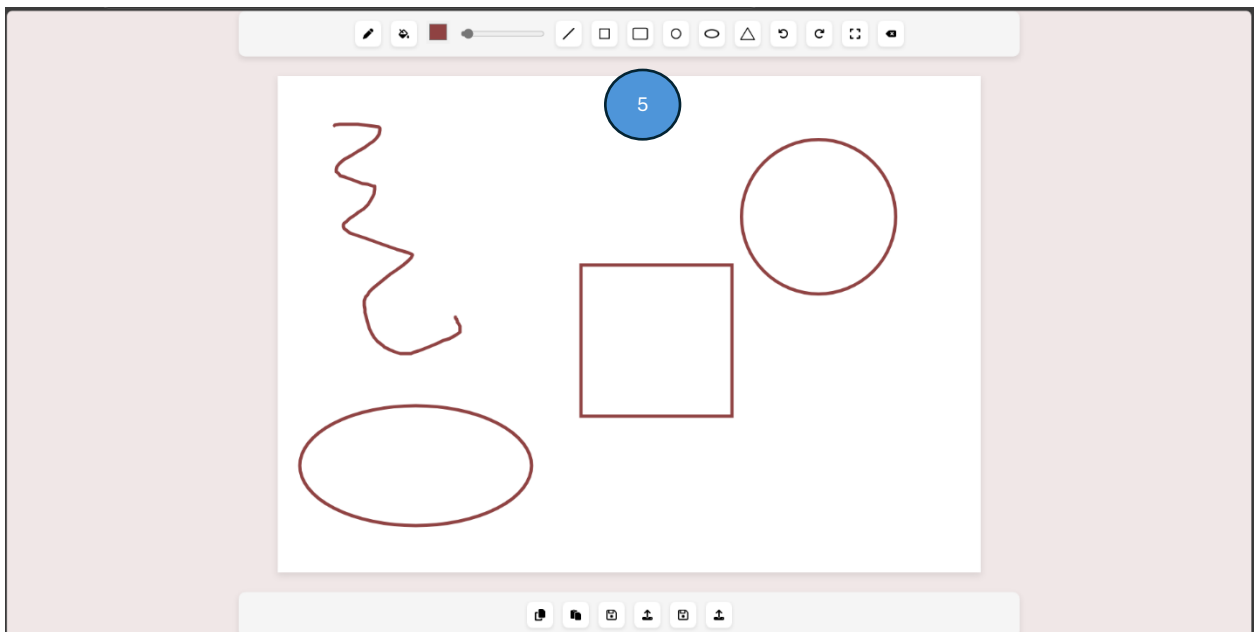
6. For when you want to undo/redo.
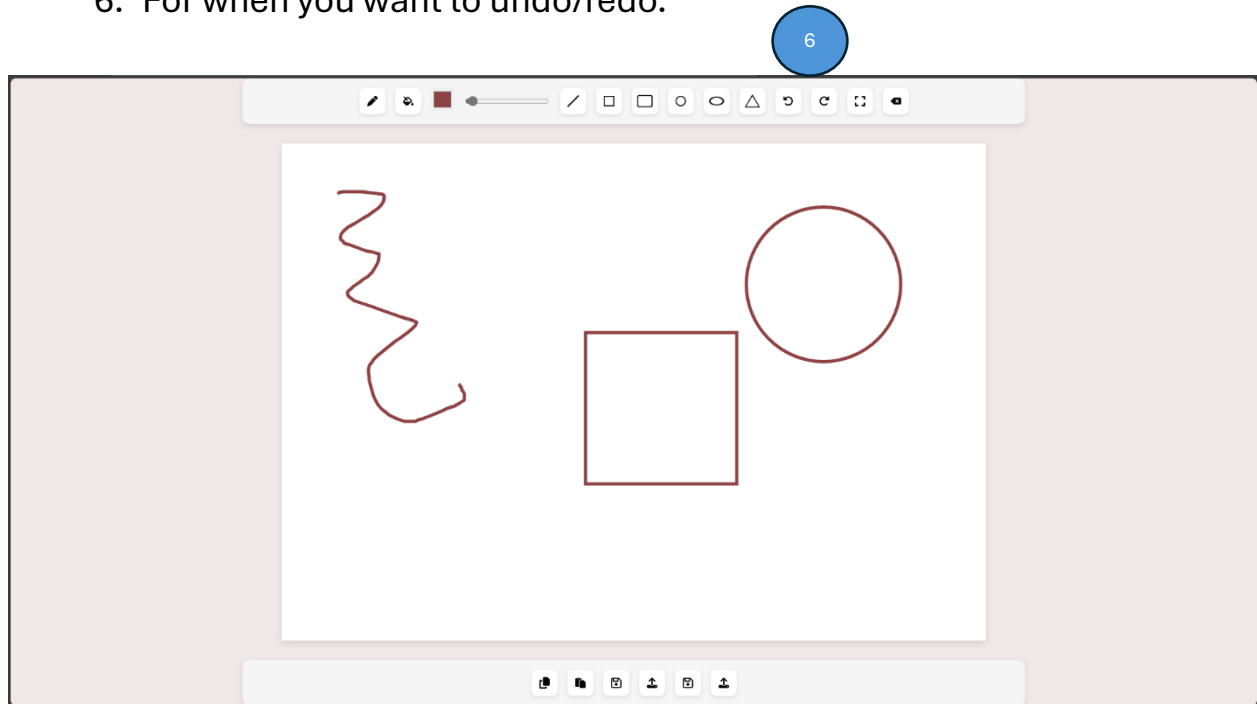
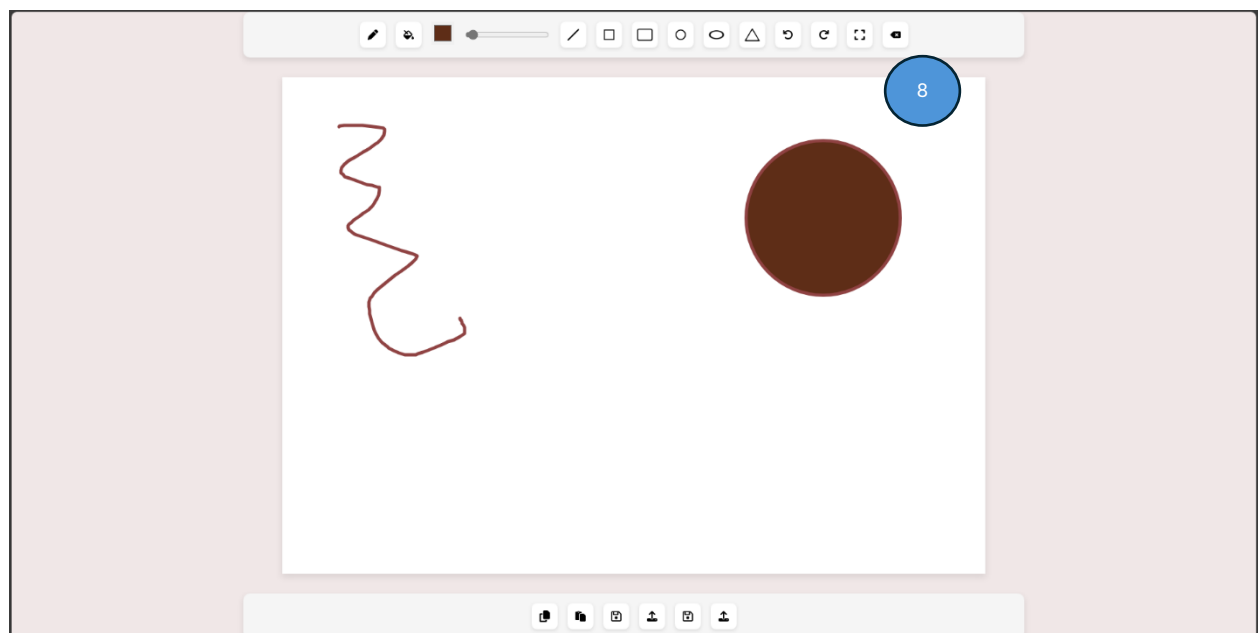2. 3.   To fill your shapes with any color of your choice.
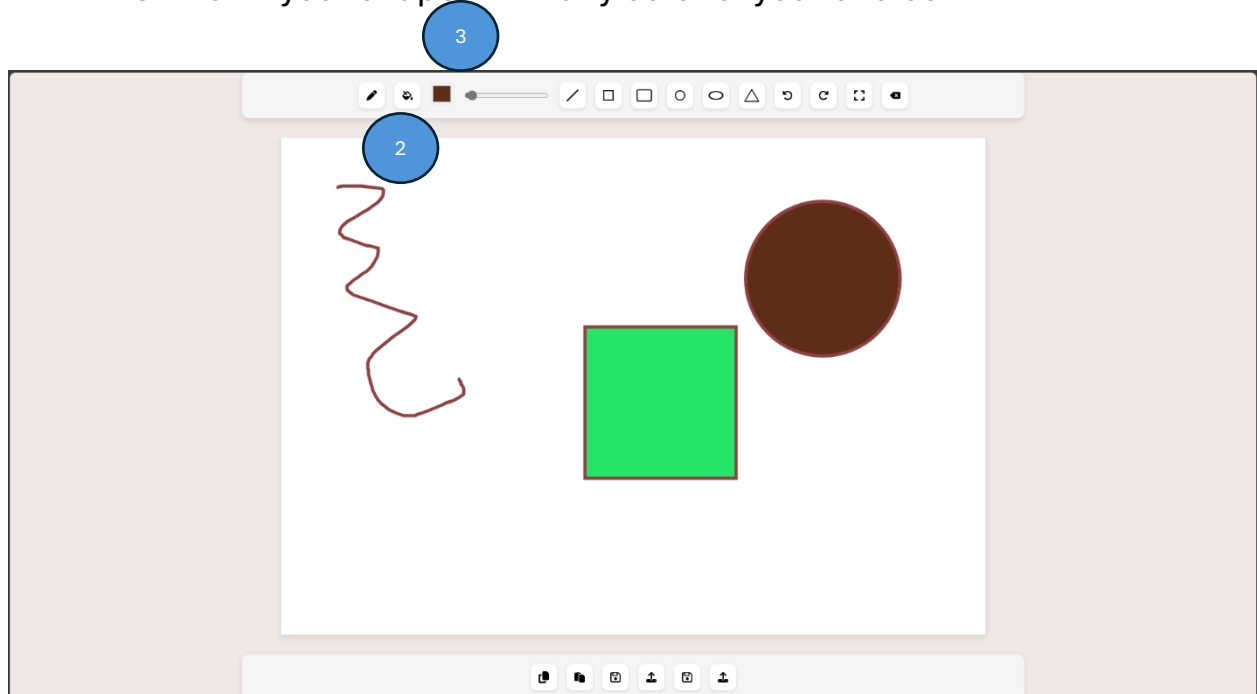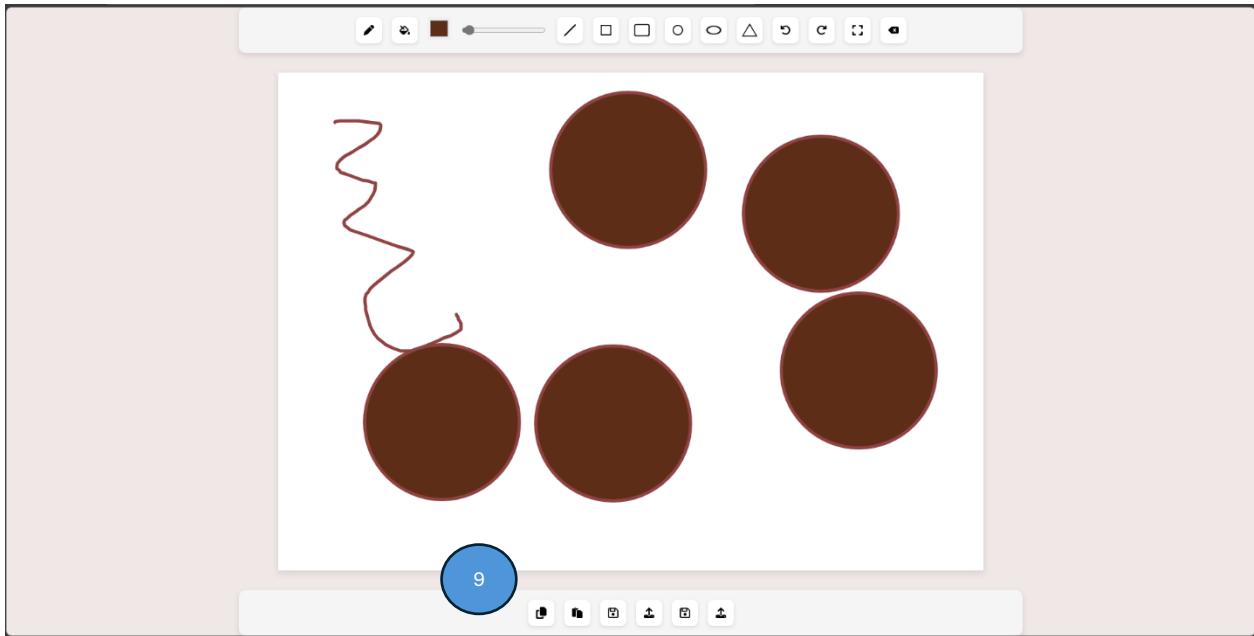
7.Resize or change positions by using the selector.
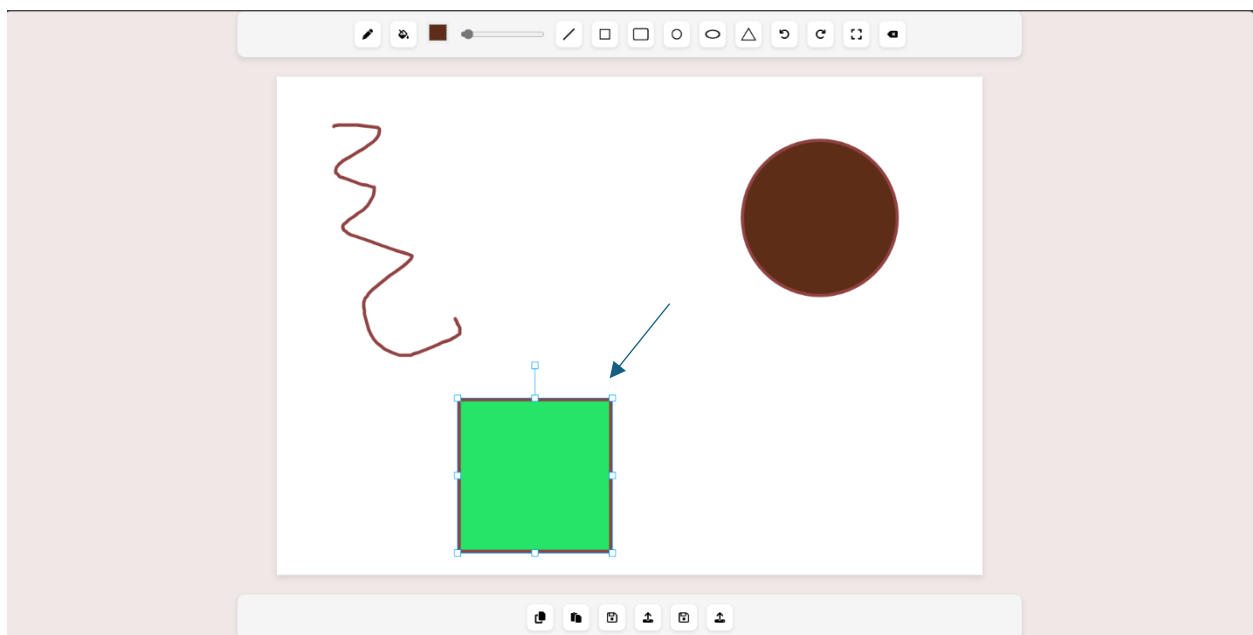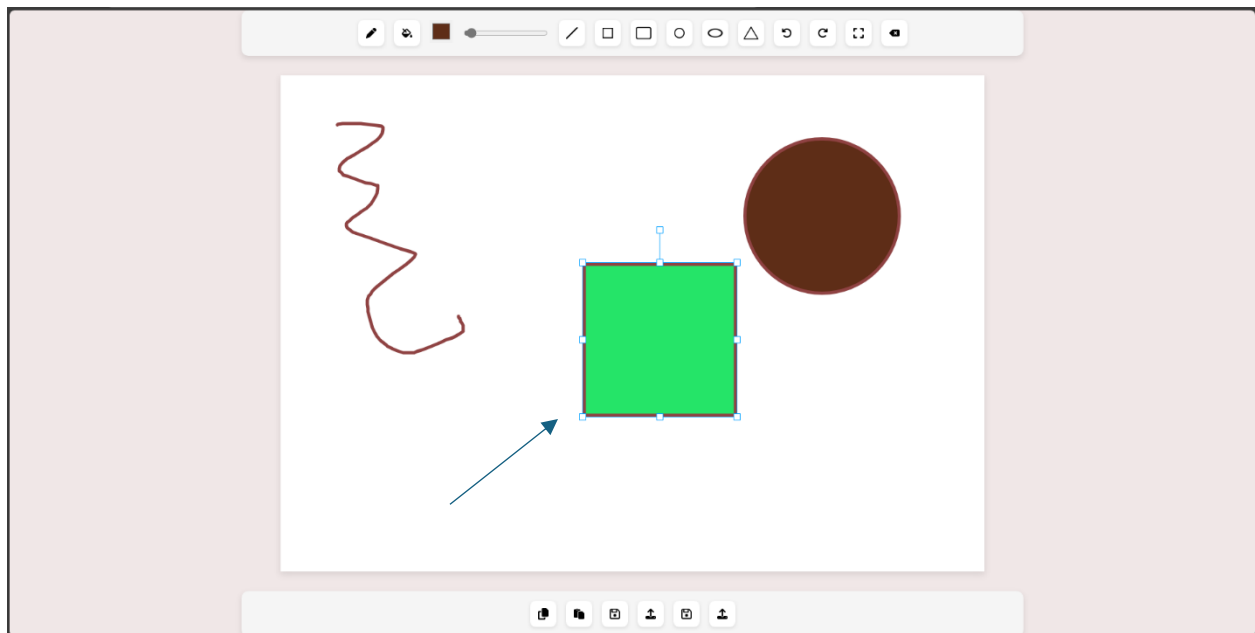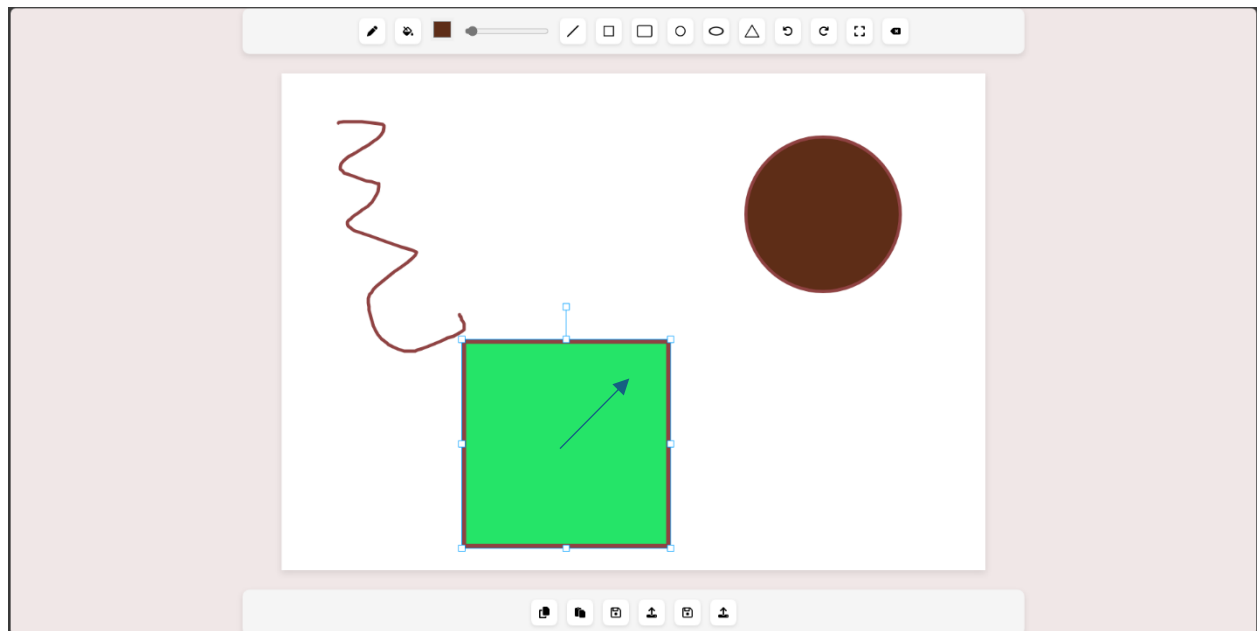
8. Don't like the shape? Delete by this tool!

9.Create a copy of any shape and paste it anywhere you like.

10.Save your drawing in XML or JSON files in any path you want and load it anytime.