

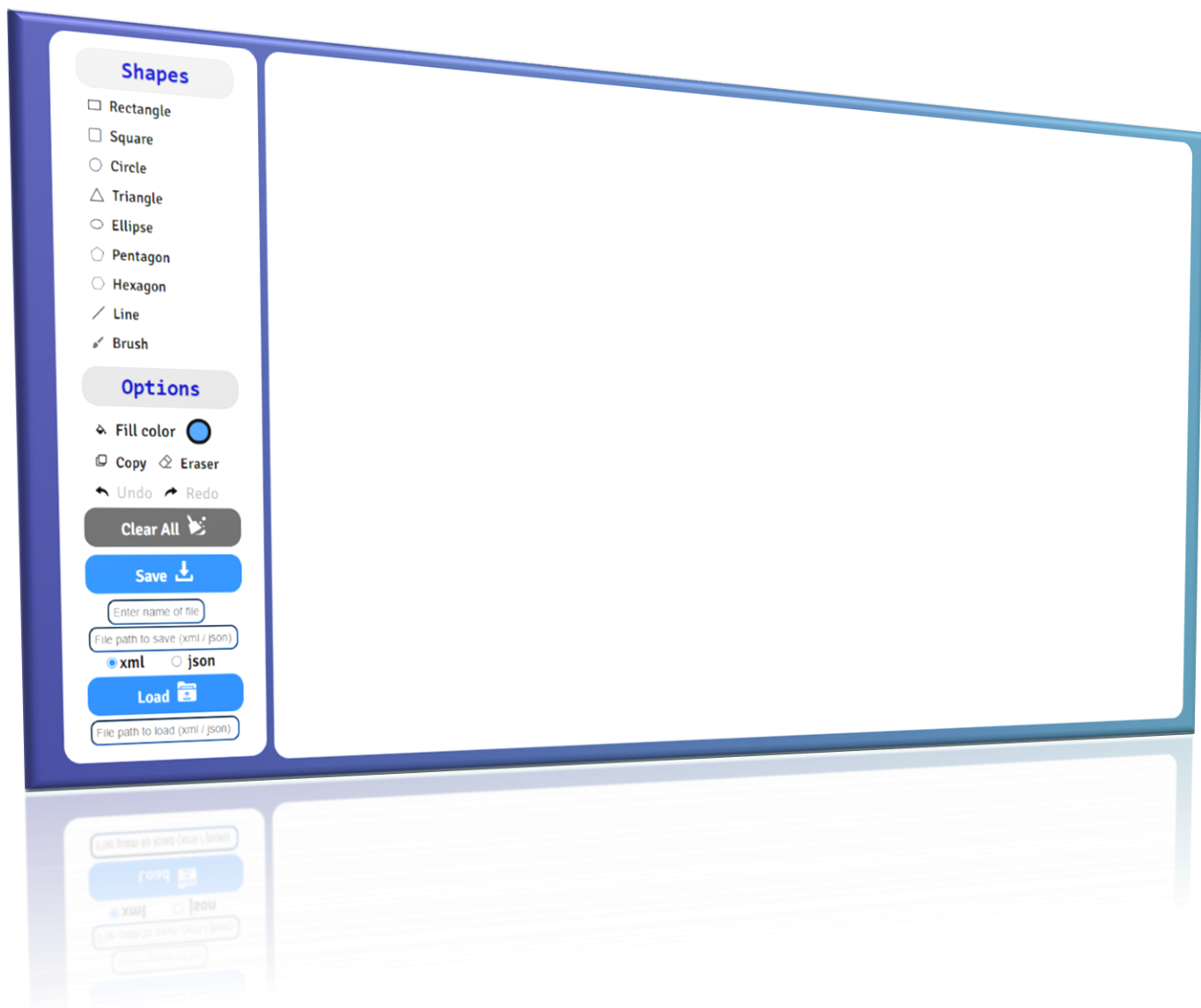
Student 1: karim Tarek Ibrahim (20011112)

Student 2: youssef Hossam Aboelwafa (20012263)

Student 3: Nagui Mostafa Nagui (20012069)

Student 4: Abdullah Mohamed Taman (20010906)

Assignment 3 report: (Paint)



Description:

- Our project is an **object-oriented** model for geometric shapes
 - We have applied OOP concepts that represent our model with an advanced User Interface with 2D graphics capabilities
 - Our project covers the following shapes:
(**Line, Circle, Ellipse, Triangle, Rectangle, Square, pentagon and Hexagon**) and a **brush tool** as an extra feature.
 - We have applied the concepts of **inheritance** and **polymorphism** to our design
 - This project is developed using **Java Spring Boot** and **Angular**
-

Features:

- Drawing
 - Coloring and fill shapes
 - Resizing
 - Moving
 - Copying
 - Deleting
 - Clear all
 - Erase
 - Undo
 - Redo
 - Taking notes with brush
 - Saving & loading sketches in (**xml**)/(**json**) format
-

Snippets of the project:

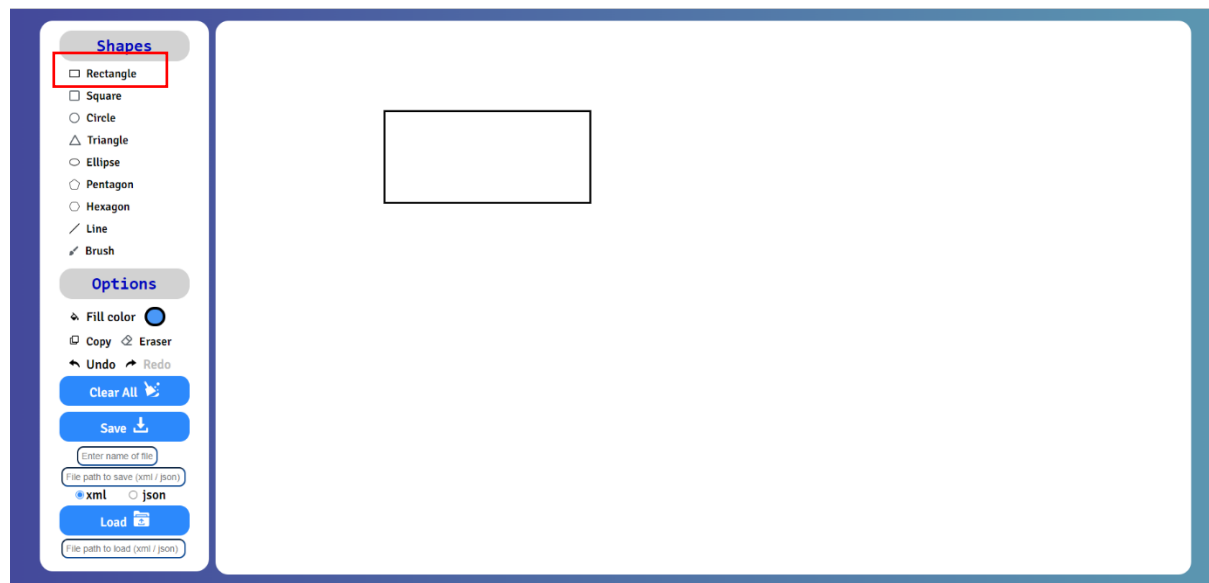
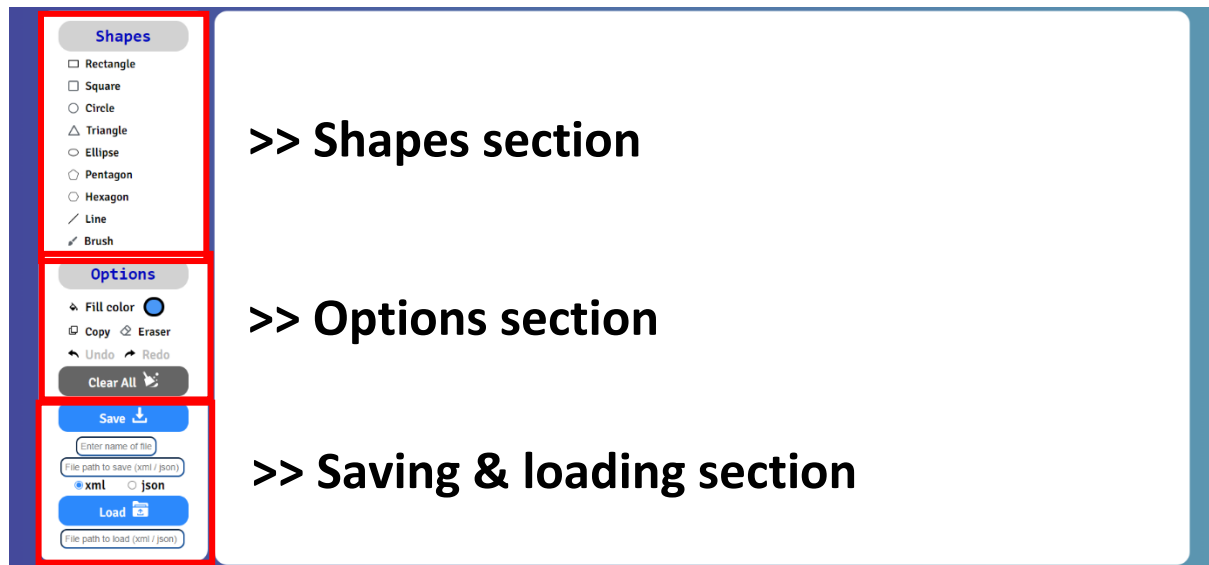


Figure 1 Drawing rectangle: to draw rectangle click on rectangle button one click and click on screen.

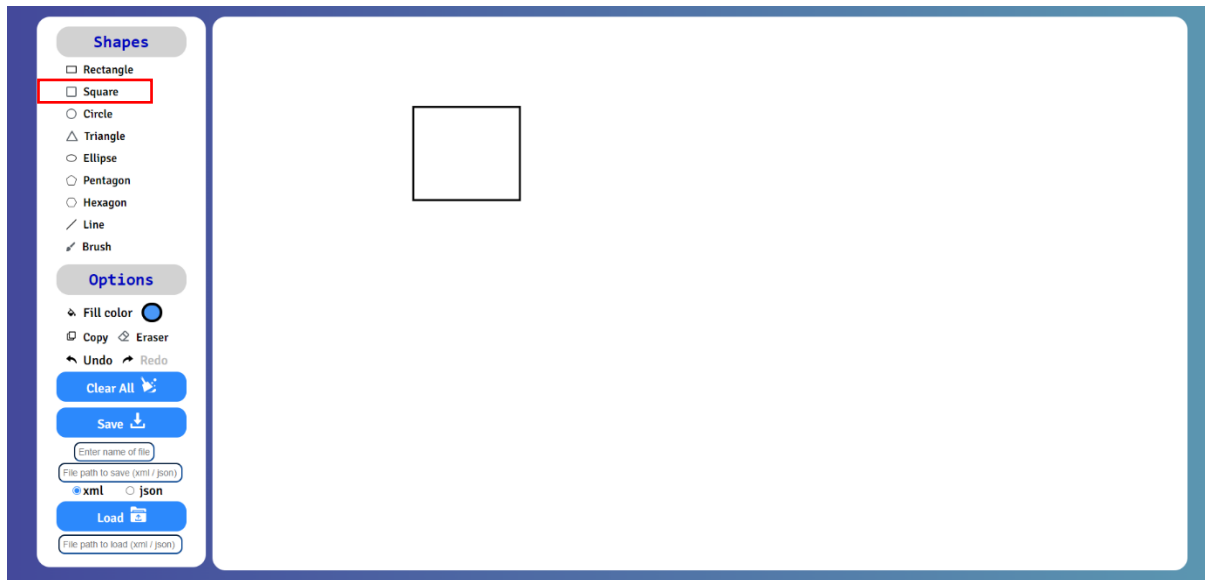


Figure 2 Drawing square: to draw square click on square button one click and click on screen.

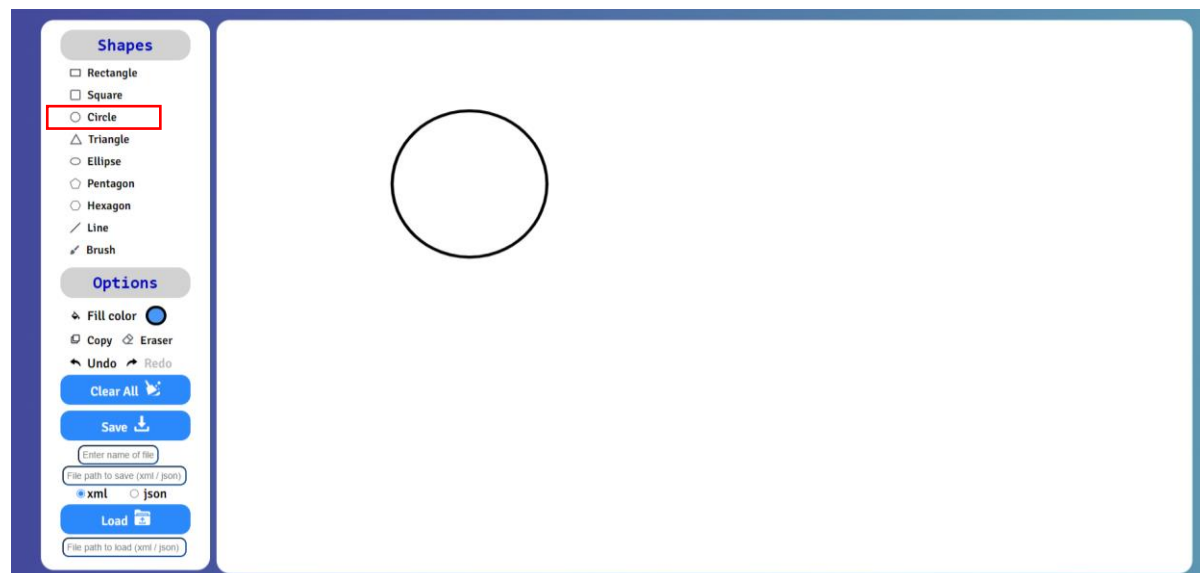


Figure 3 Drawing circle: to draw circle click on circle button one click and click on screen.

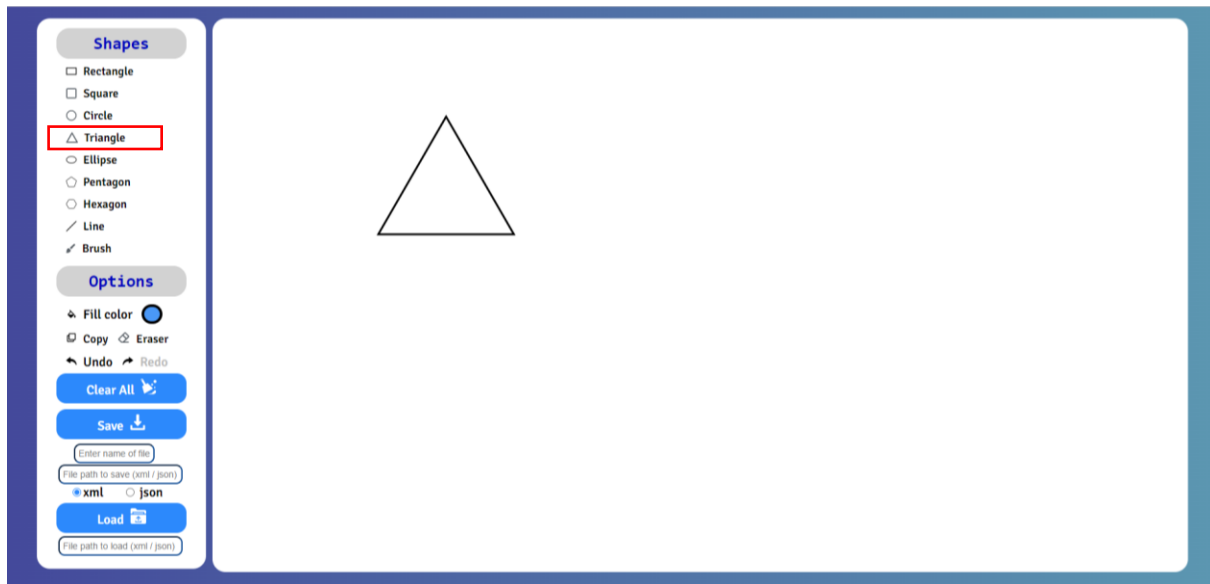


Figure 4 Drawing triangle: to draw triangle click on triangle button one click and click on screen.

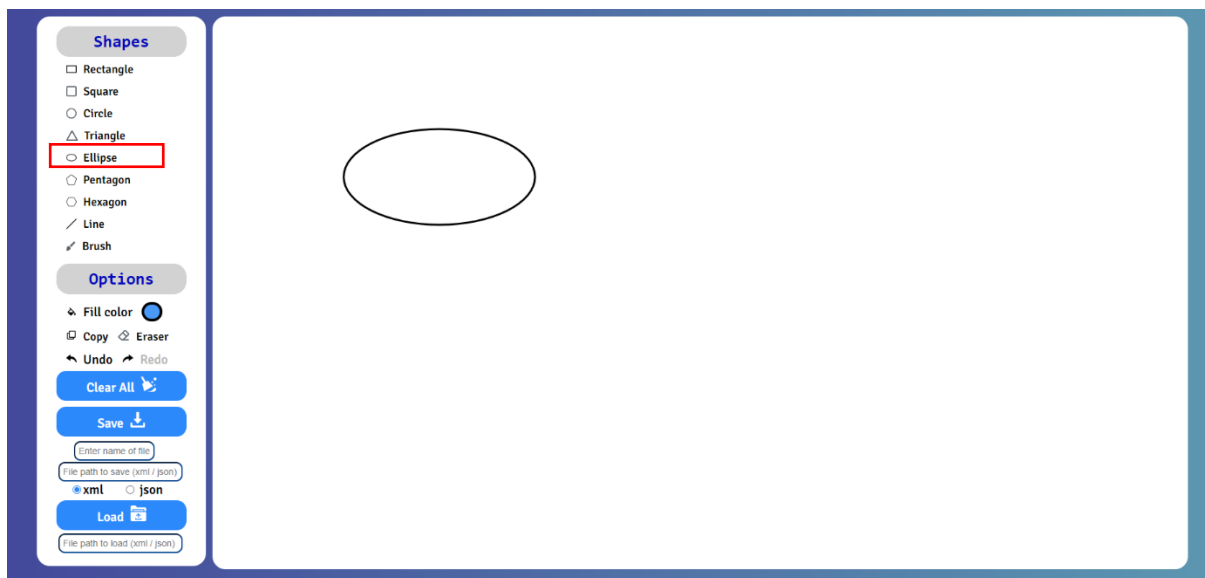


Figure 5: Drawing ellipse: to draw ellipse click on ellipse button one click and click on screen.

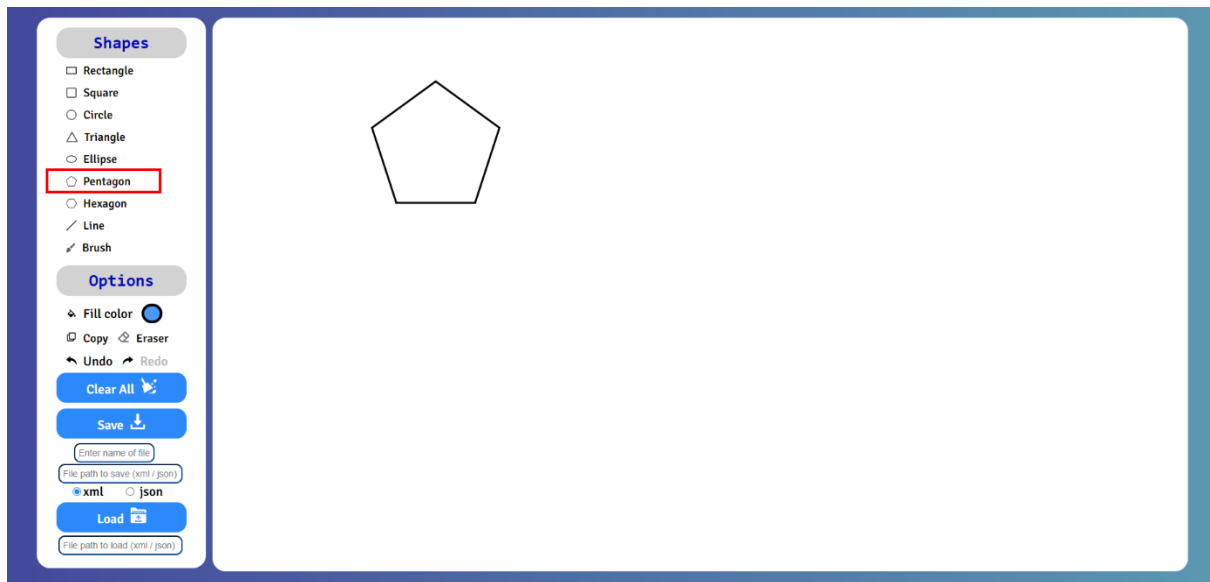


Figure 6 Drawing pentagon: to draw pentagon click on pentagon button one click and click on screen.

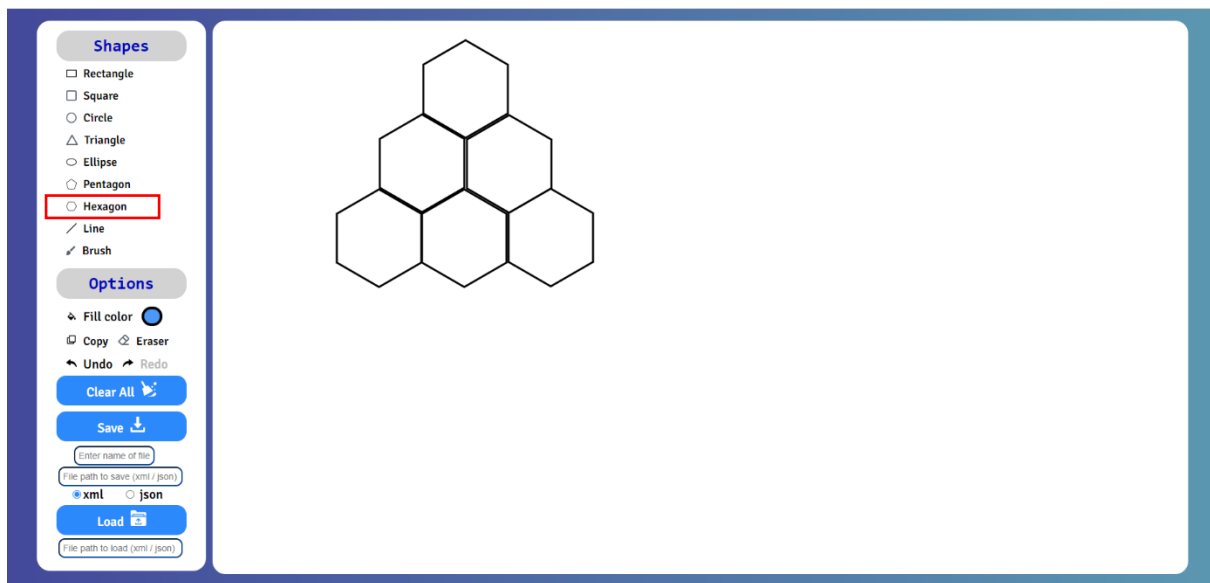


Figure 7 Drawing Hexagon: to draw hexagon click on Hexagon button one click and click on screen.

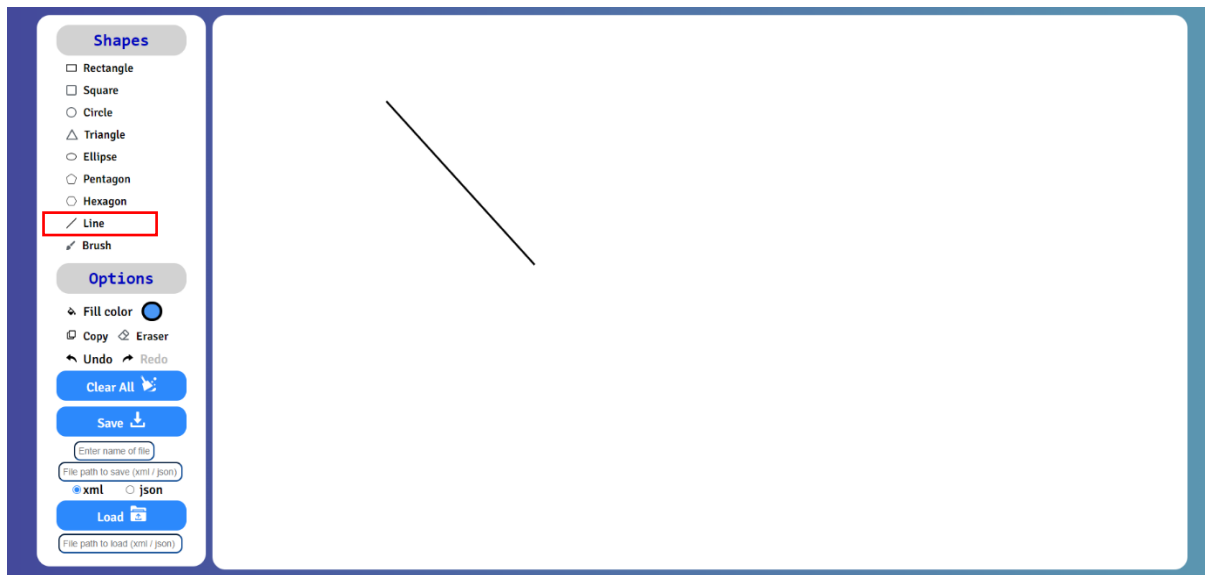


Figure 8 Drawing line segment: to draw pentagon click on Hexagon button one click and click on screen.

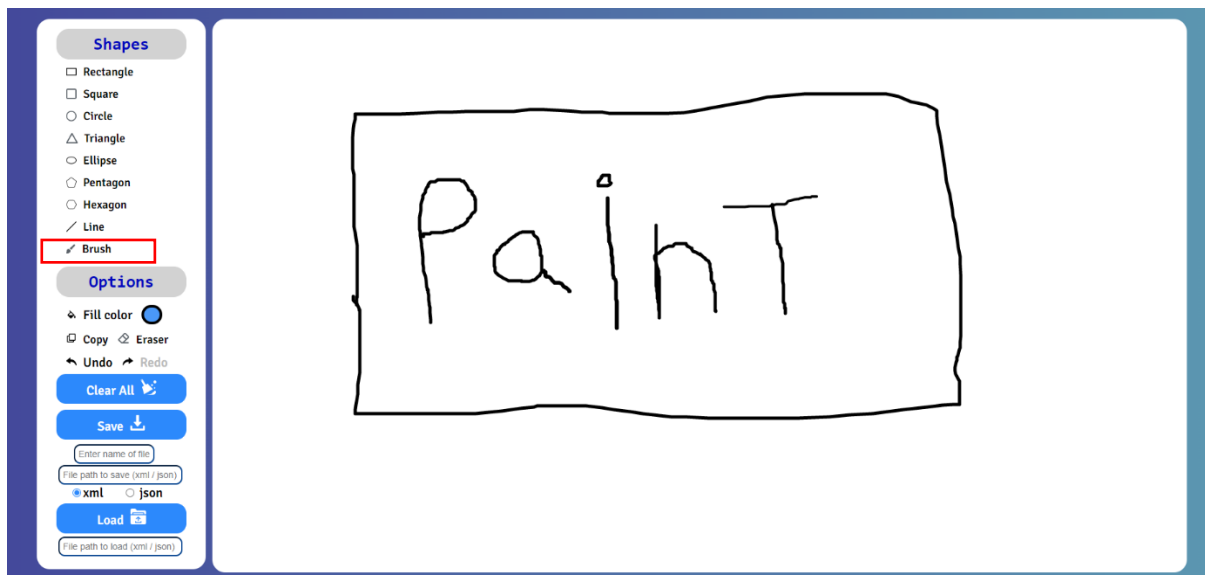


Figure 9 Drawing & taking notes using a brush: to take notes click on Brush button one click and click on screen.

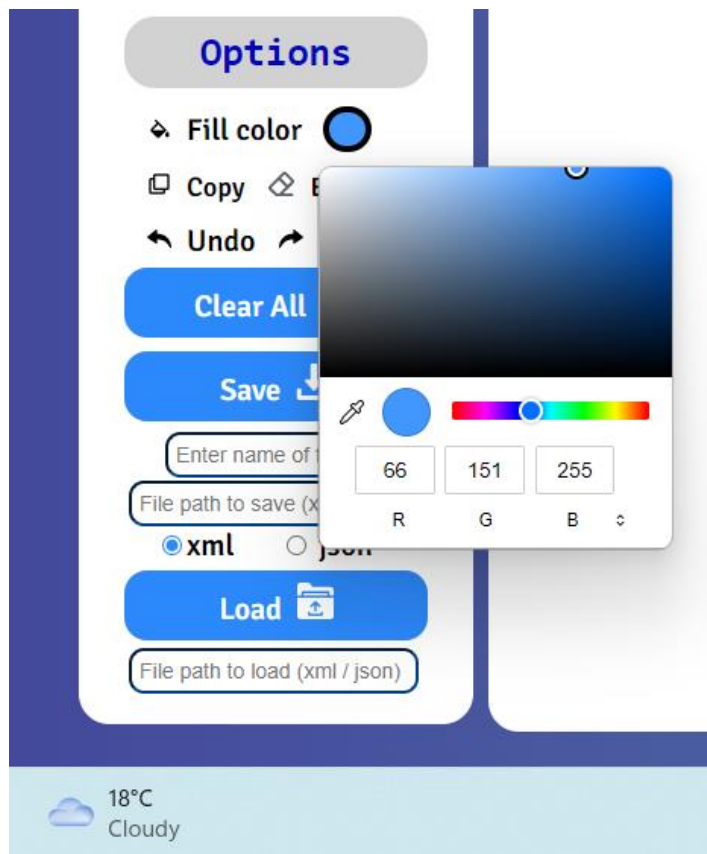


Figure 10 Filling Shapes using color picker: click on color picker and choose any color you want.

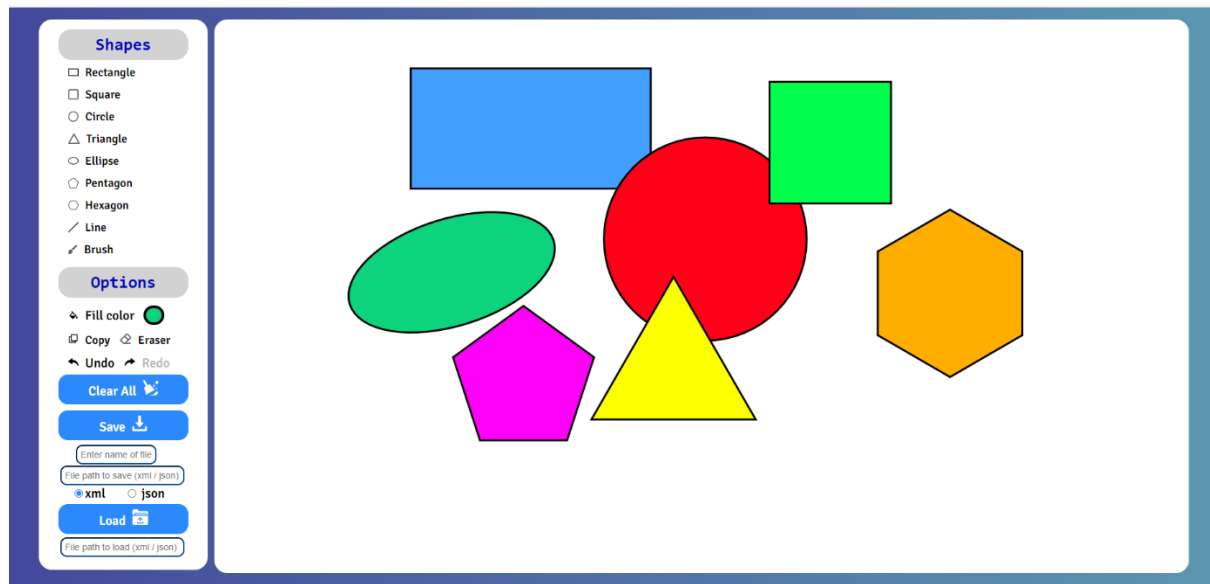


Figure 11 Filling shapes with colors: to fill shapes with colors after you choose it you should **DoubleClick** on shape

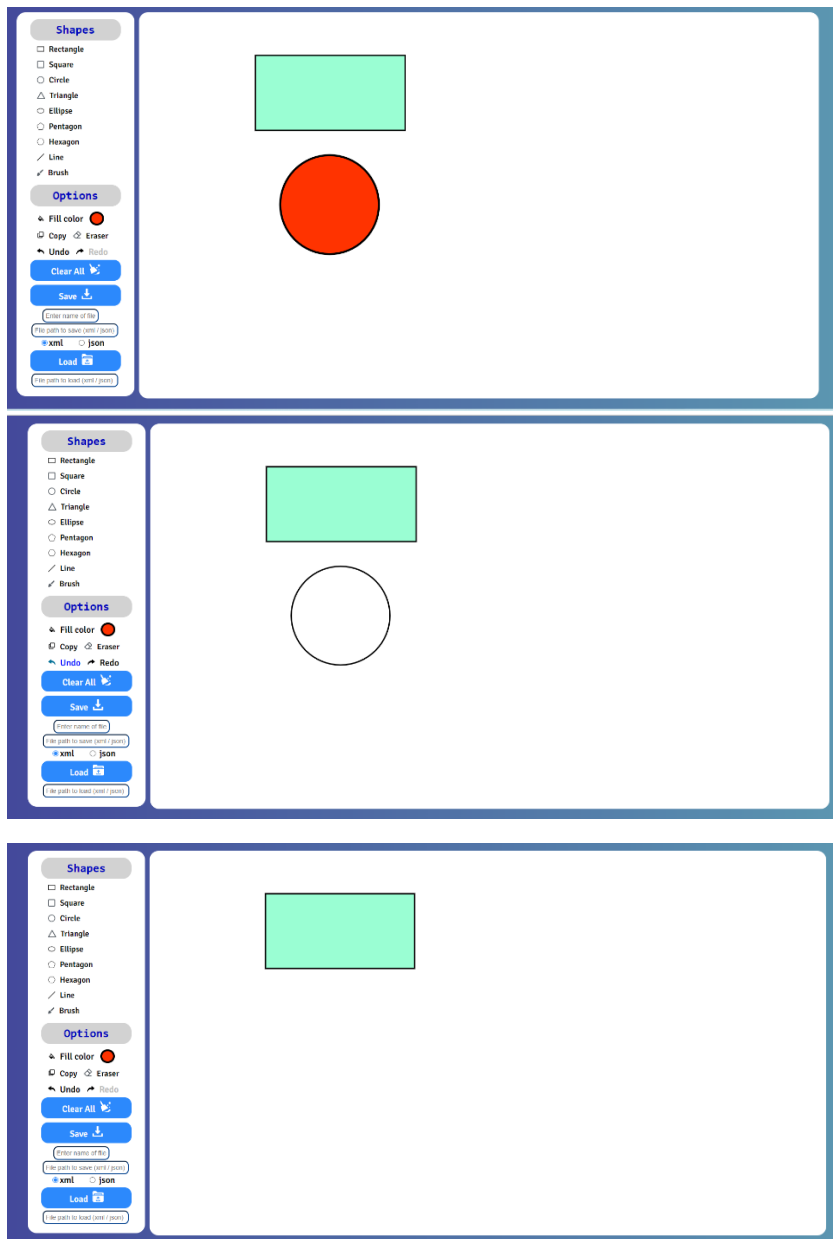


Figure 12: Undo & Redo

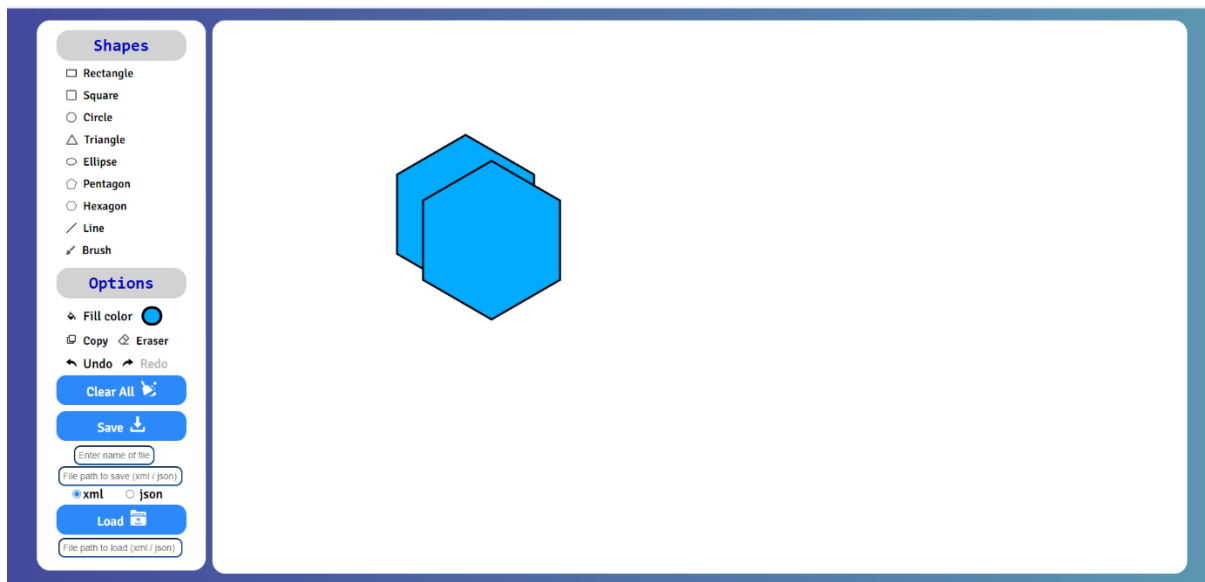


Figure 13 Copy shapes: to copy any shape click on copy button one click and click on the shape.



Figure 64: Undo/Redo/Clear all before using it in the beginning of the program

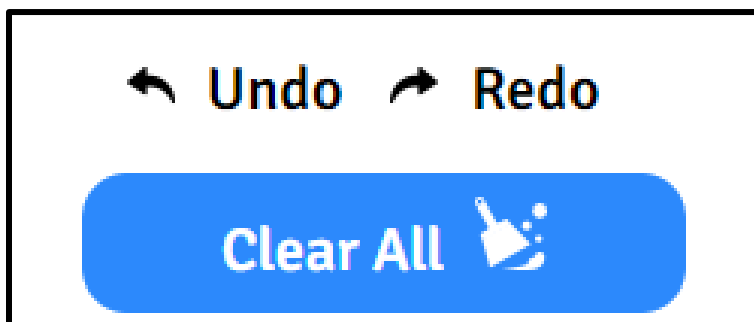
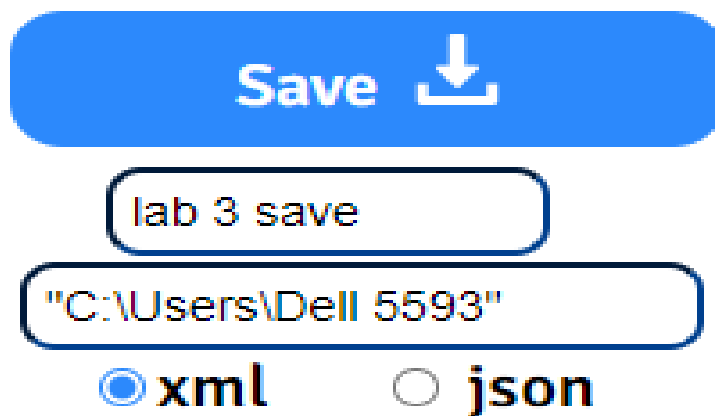
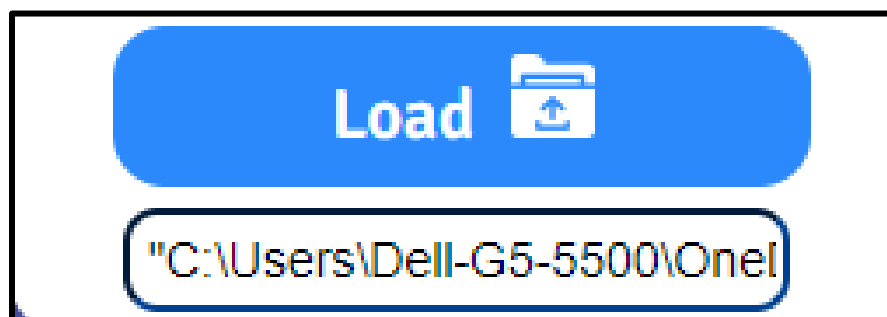


Figure 75: Undo/Redo/Clear all are activated



A blue button labeled "Save" with a white download icon. Below it is a text input field containing "lab 3 save". Below that is a text input field containing the file path "C:\Users\Dell 5593". At the bottom are two radio buttons: the first is selected and labeled "xml", the second is unselected and labeled "json".

Figure 16: Saving file and choosing the format you want (xml/json) giving the name of the file and the path you want to save your file in



A blue button labeled "Load" with a white folder and upload icon. Below it is a text input field containing the file path "C:\Users\Dell-G5-5500\OneI".

Figure 17: Loading a file using the path

Design Decisions for Backend:

- Spring Boot is used as a framework
 - Every shape is represented by a specific class that implements the interface Ishape.
 - Maximum number of attributes of a shape is 14 where every object of a class has a unique ID
 - All the logic is implemented in a class named Drawer
 - A Factory class is responsible of creating the objects
 - An array list (History) of array lists of objects from the classes that implements Ishape is created to save every operation the user executes, where every list of History represents a state of the program
 - A state contains all the shapes that are shown now
 - When the user draws a shape, a method (addShape) from the controller class is called. addShape receives an object from an implemented class called JsonShape (a JsonShape object has the same attributes as the shape but with no implemented methods) and its type (square, rectangle ...etc.) and returns an array list of shapes
 - The factory class then creates an object from a class based on the type of the JsonShape received, the JsonShape object attributes is copied to the created object through a method called Handle
 - A new array list, that contains all the shapes of the last state and the new added shape, is created, and added to the array list History.
 - The new state is then returned to be drawn.
 - If the last state contains a shape with the same ID of the added shape, then new state will contain the added shape and all the shapes of the previous state except for the one with the duplicated ID. (That means that the added shape is just an edited version of an already existed shape and not a new created shape)
 - The undo function simply returns the state that precedes the last state in the History array list
 - The redo function simply returns the last state in the History array list
 - The save function receives a string in the form of (filePath+filename+fileType) and based on the filetype the file is saved as a json or xml file, the saved file contains the last state
 - The Load function receives the path as a string. after the file is loaded the History array list is cleared and the loaded state is added to it
-

Design Decisions for Frontend:

- Angular is used as a framework.
 - the brush is considered as an extra feature, and it can't be saved or loaded or copy.
 - function copy sends a Request to back with old ID and new ID and received new copy shape with new id to draw it.
 - "Object_shape" class take an object of konva and return object with same attributes
 - "Service" class contain many functions to connect frontend to backend.
 - "Copy" class take two attributes ID and new ID.
 - When front make shape, copy, clear, erase, undo, redo, save or load the front. send request to the back to update data and receive the new update to draw it.
 - When user resize any shape after he finish, he should click out of shape on the screen to send new update to back.
 - To drag any shape, you should click on it and when you finish, you should click out of shape on the screen to send your update to back.
 - The name that back know about Hexagon, pentagon and triangle is "triangle"
-

How did we implement the Design patterns?

Factory DP:

We made a class called the factory, which is used by the drawer class. The drawer class handles all the requests. When the user wants to add shape, we just use this class which depends on the Ishape interface. Depending on the shape requested, we return a concrete class of the required class.

Prototype DP:

We made a clone function in the Ishape interface which was implemented in each concrete shape. We give this function the object we want to clone as a parameter and return its copy with the new ID. We take the new ID from the front end. We didn't make the classes extend the cloneable class, we just call the clone function and it handles it all.

A descriptive UML class diagram for the project:

https://drive.google.com/drive/folders/1OANuxLPhfcU5HITiD57Szg2sLJqu9bII?usp=share_link
