Computer & Systems Engineering Department

CSE 223: Programming 2

Assignment 3: Paint

Contributors:

i. Name: Adel Mahmoud Mohamed Abdelrahman ID: 20010769
ii. Name: Mohamed Hassan Sadek ID: 20011539
iii. Name: Mahmoud Attia Mohamed Abdelaziz Zian ID: 20011810
iv. Name: Mahmoud Ali Ahmed Ali Ghallab ID: 20011811

1. Problem Statement:

Part 1: Geometric Shapes Data Model:

1. Design an object-oriented model that covers the following geometric shapes: (Line Segments: Sectors), (Circle, Ellipses: Elliptical shapes), (Triangle, Rectangles, and Square: Polygons).

2. Draw a UML Class diagram that represents our model, showing all the classes, attributes and methods.

3. Apply the concepts of inheritance (Extend classes) and polymorphism (Overriding) to your design.

Part 2: Drawing and Painting Application:

1. Implement your design (Shapes modeling) from part 1.

2. Design and implement a GUI that allows the following functionalities for the user on all the shapes defined in part 1: Draw, Color, Resize, Move, Copy, and Delete. (Optional hint: check “Factory DP (For Draw), and Prototype DP (For Copy)”).

3. Implement your application such that it would allow the user to undo or redo any action performed.

4. The cursor should be used to select the location of a shape while drawing it, or moving it to another location, for resize for example, dialog boxes could be used, or you are free to implement it in a more user-friendly way of your choice. (Optional hint: **draw by mouse dragging**).

Part 3: Save and Load:

1. Provide an option in UI to save the drawing in XML (encoding: ISO-8859-1) and JSON file (You should implement both).

2. Provide an option to load previously saved drawings and modify the shapes.

3. Users must choose where to save the file.

---

2. Frameworks & technology used:

For the frontend part (view part), we used HTML, CSS, and typescript through angular framework.

For the backend (model and controller), we used Java language through spring framework.

---

3. Full list of the steps required to run our code:

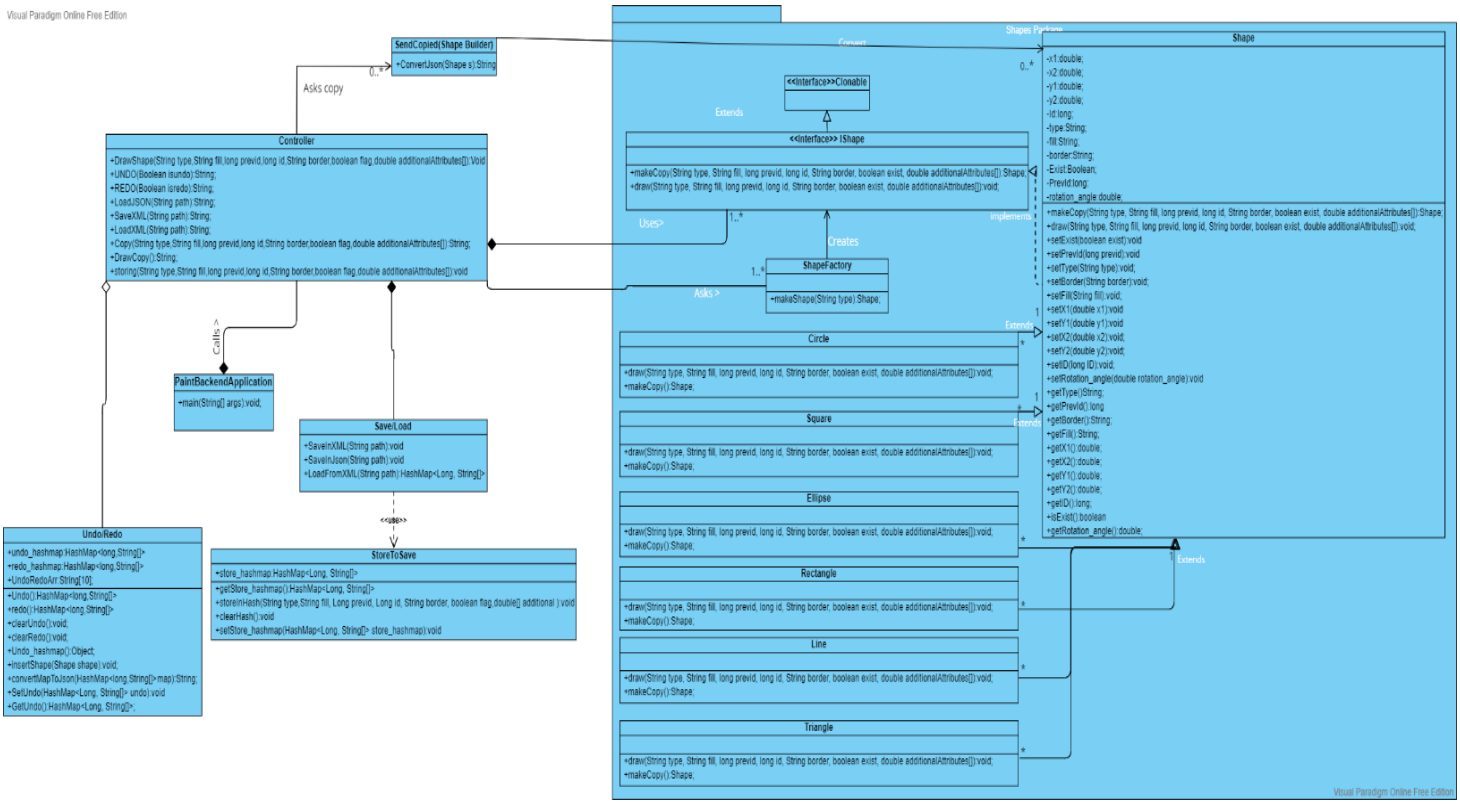
Note: Make sure you downloaded NodeJs and Angular-CLI.

- extract the compressed project folder.
- **Back-end part:**
  - Open the **paintBackend folder** using IntelliJ IDE or any other IDE, run the **PaintBackendApplication.java class** on port 9090.
  - you can change the port from the project resources → application.properties if the 9090 port was already used in your device but in this case, you will need to change it in all http requests in the app.components.ts file on the front-end folder.
- **Front-end part:**
  - 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 “ng serve --open” in the terminal to open the project, on port “http://localhost:4200/”-. Note: if you needed to change port 4200 as it was already in use then you will need to change it in the paintBackend folder in the **controller class**.
- Then you can use the paint application and draw whatever you want.

---

4. UML Project Diagram:



UML full diagram link:

https://online.visual-paradigm.com/w/skvbyjfy/app/diagrams/#diagram:workspace=skvbyjfy&proj=0&id=12

---

5. How we have applied the required design pattern in our code

1- Factory design pattern.

- We have an interface called **‘IShape’** that’s implemented by the superclass ‘**Shape’** which is inherited to all other shapes classes this interface is cloneable for the prototype DP.

- Then we have a super Class called ’**Shape’** that have the common methods and attributes for the shapes **with a purpose of inheritance and code reusability.**
- Shapes classes (circle, triangle,.) **extends** the **Shape** class.
- The factory creates an instance of any shape by using a key string which is the name of the desired shape to be formed then the factory returns an object with the type of Shape.

2- Prototype design pattern.

- In order to Copy shapes in the app, we create the copy method by prototype design pattern.
- First, we create an instance of Shape class (which implements the interface IShape which extends Cloneable interface) to call the function “makeCopy(desired shape attributes copied)” with all attributes of the desired shape to be copied as arguments.
- Then creates an instance of ‘SendCopied’ class to call ‘ConvertJson(shape)’ method that takes the shape as an argument to return a copy of the shape attributes as string then the frontend will draw the shape with these attributes.

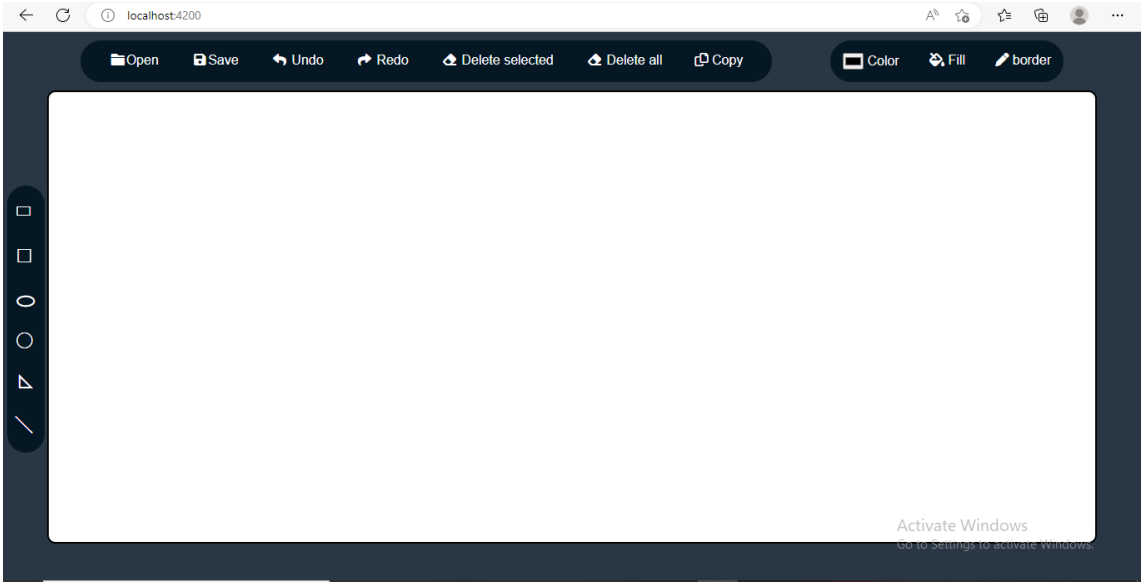
---

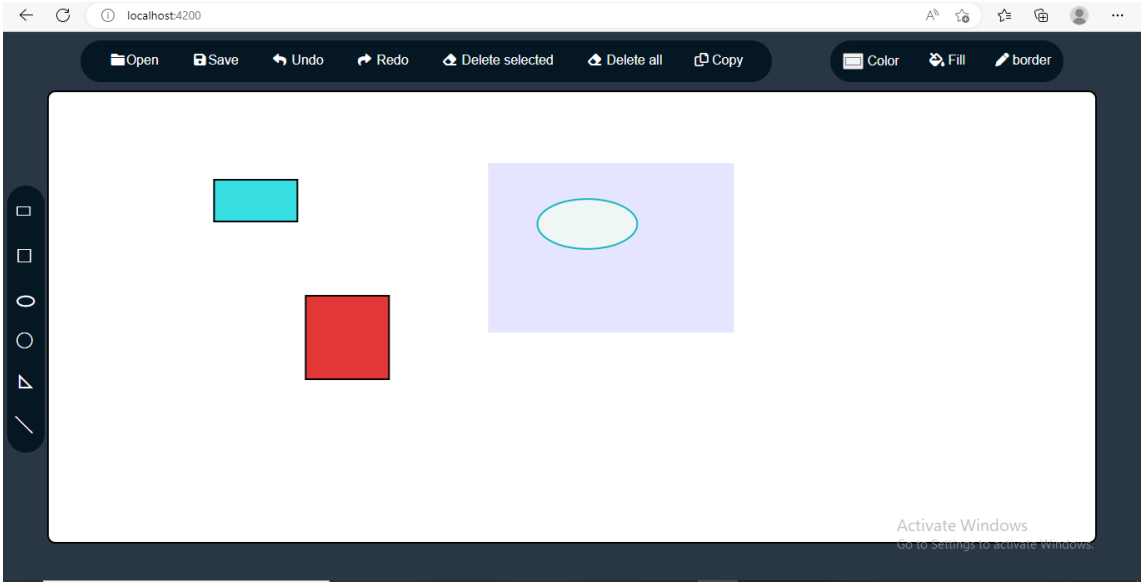
6. Any Design decision:

- We decided to make the OOP design patterns in the backend.
- We used the factory design pattern to create shapes by passing its name to the shape factory class.
- We used the prototype design pattern to copy shapes to help us take all attributes of the shape from one already created before.

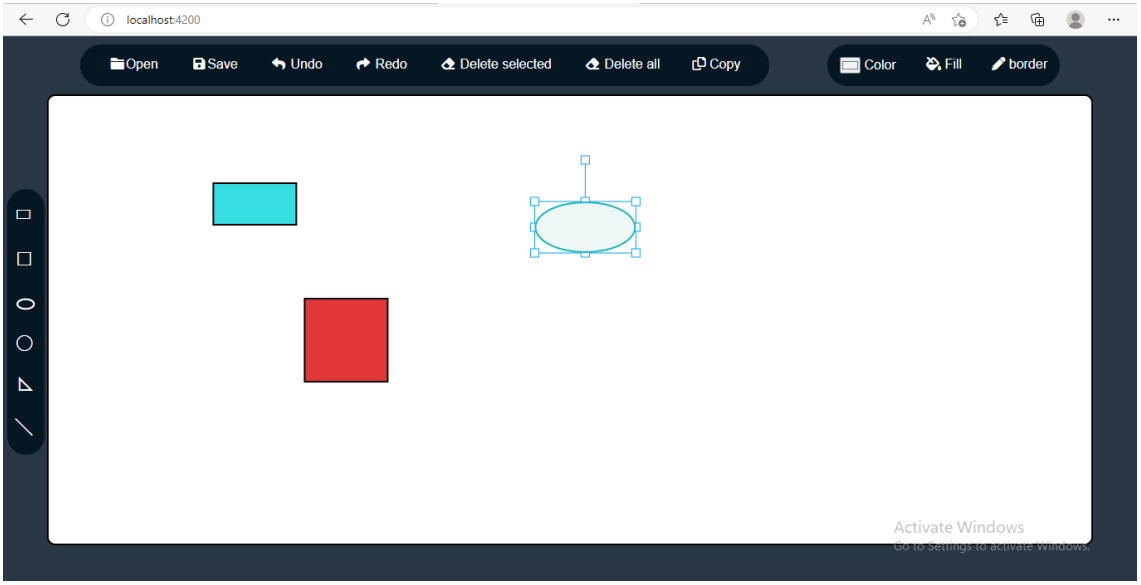
---

7. Snapshots of our UI and a user guide that explains how to use our application.





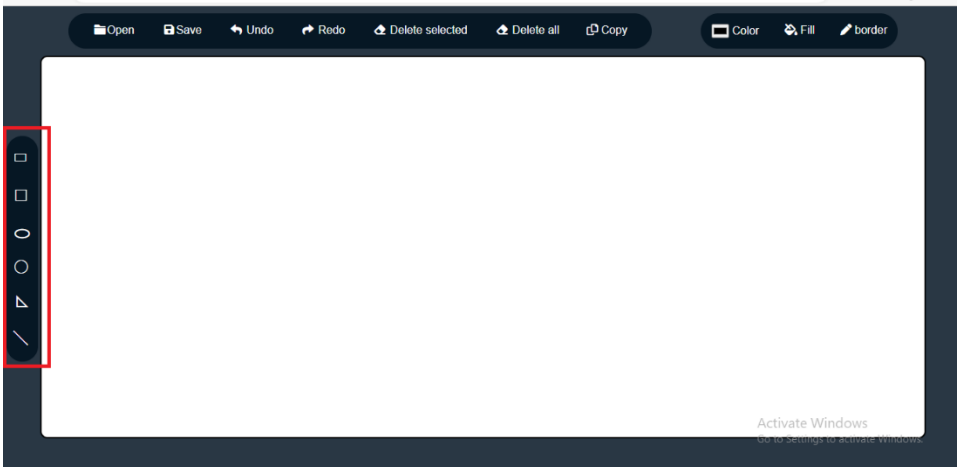
Leads to:



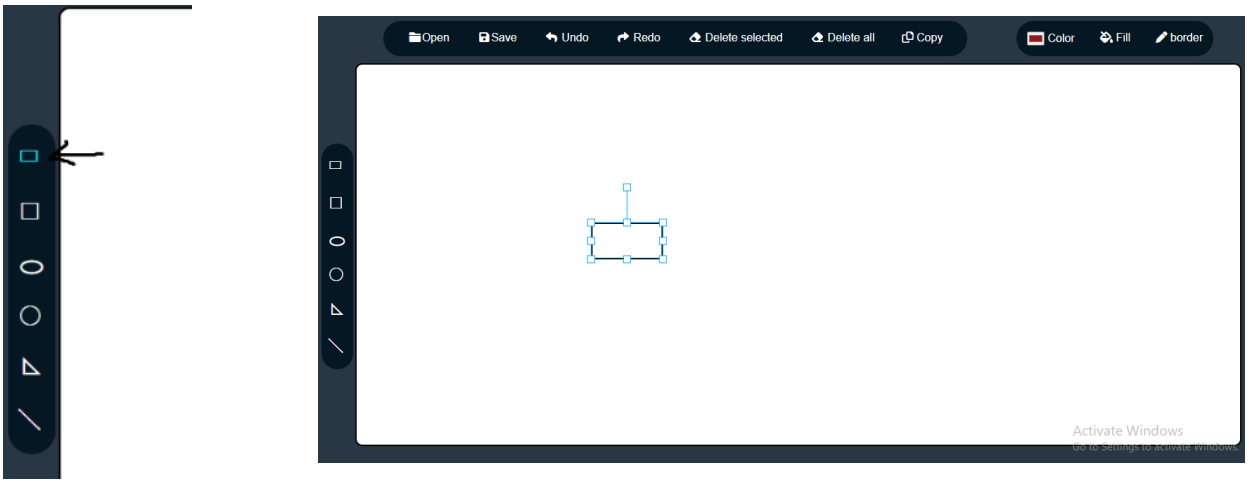
How to use the Application:

1. The left red rectangle specifies the shapes, which the app supports.

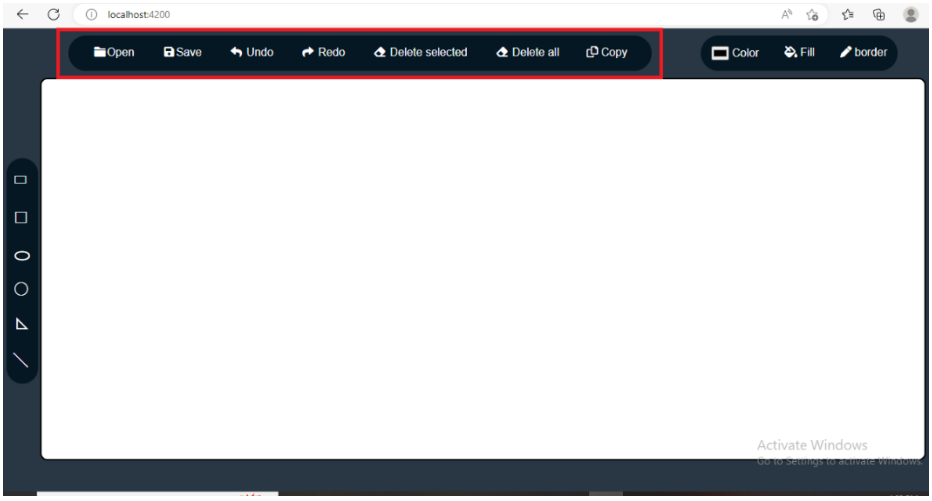
Click on the shape then click on the drawing field to draw the shape.



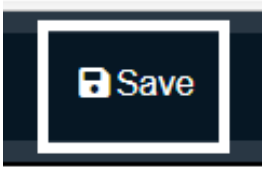
For Example:

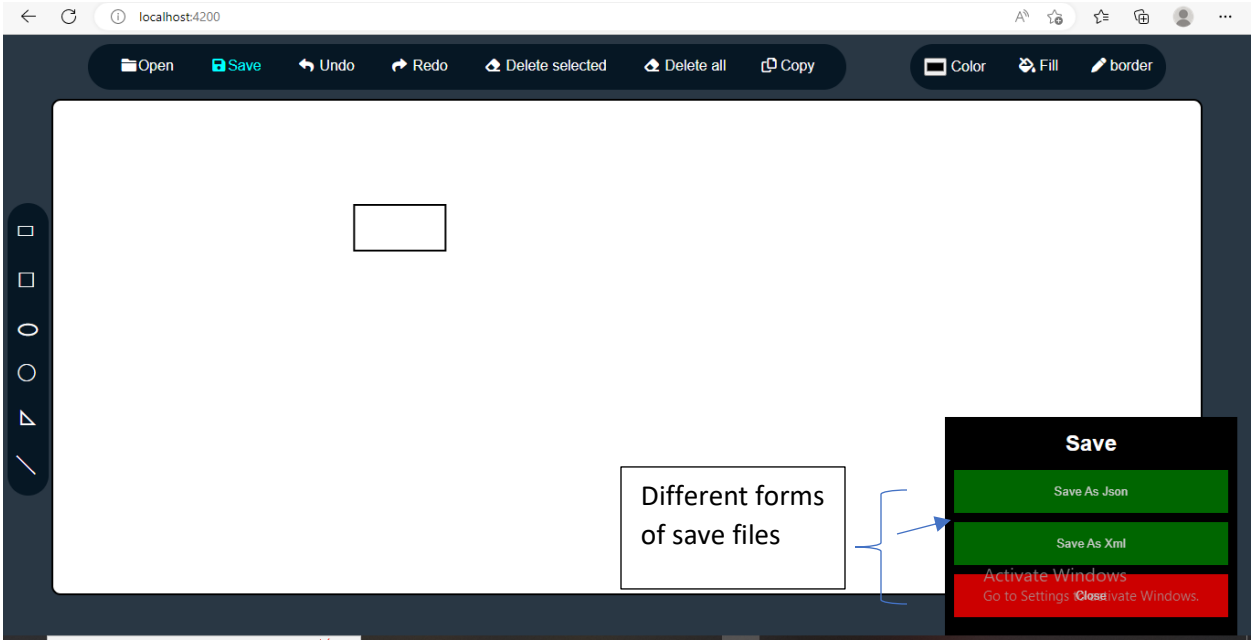


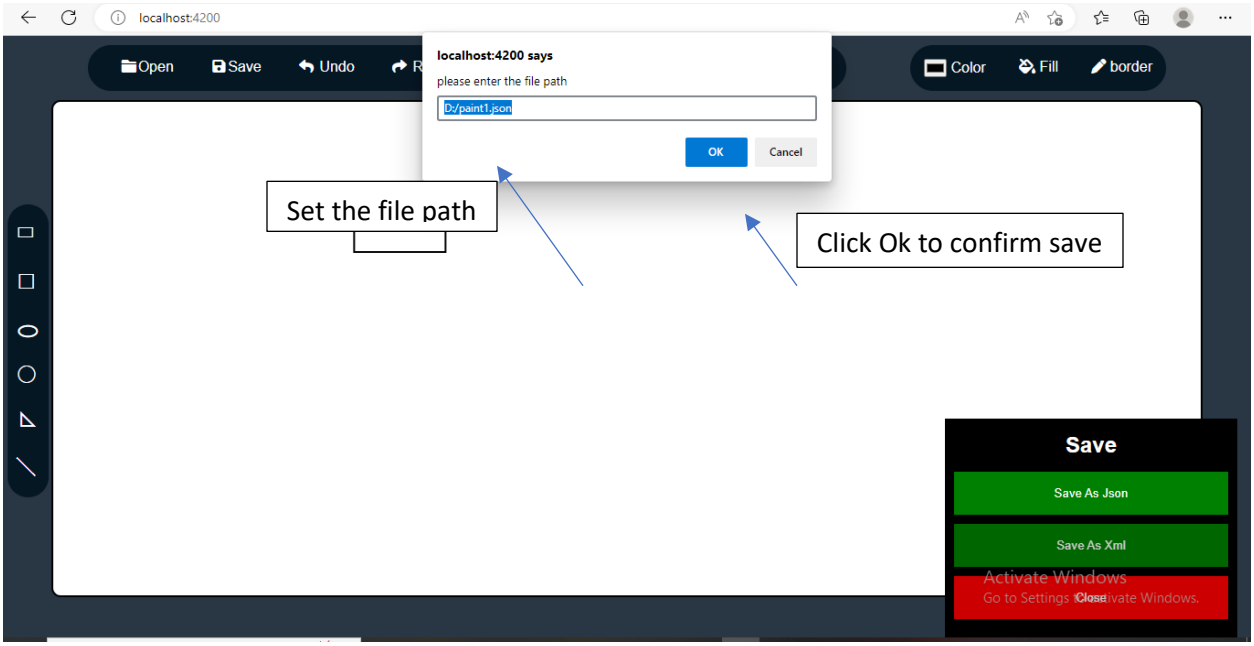
2. The above red rectangle states all operations can be done through your session in the application in addition to saving your drawing and opening previously saved one.



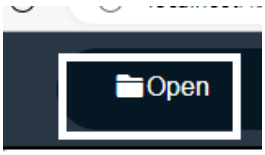
1. Save icon to save your painting, click on it then it will show you the different forms saving and you can choose any one of them and a pop up menu will appear to you to set the file name and path you want to save in.



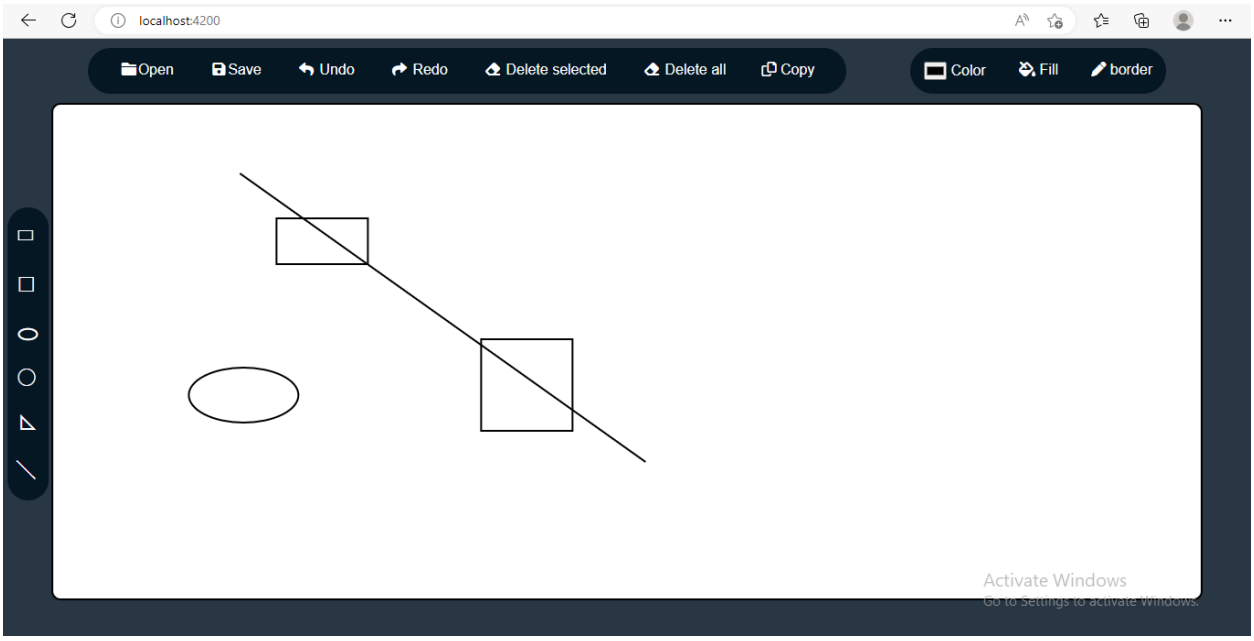


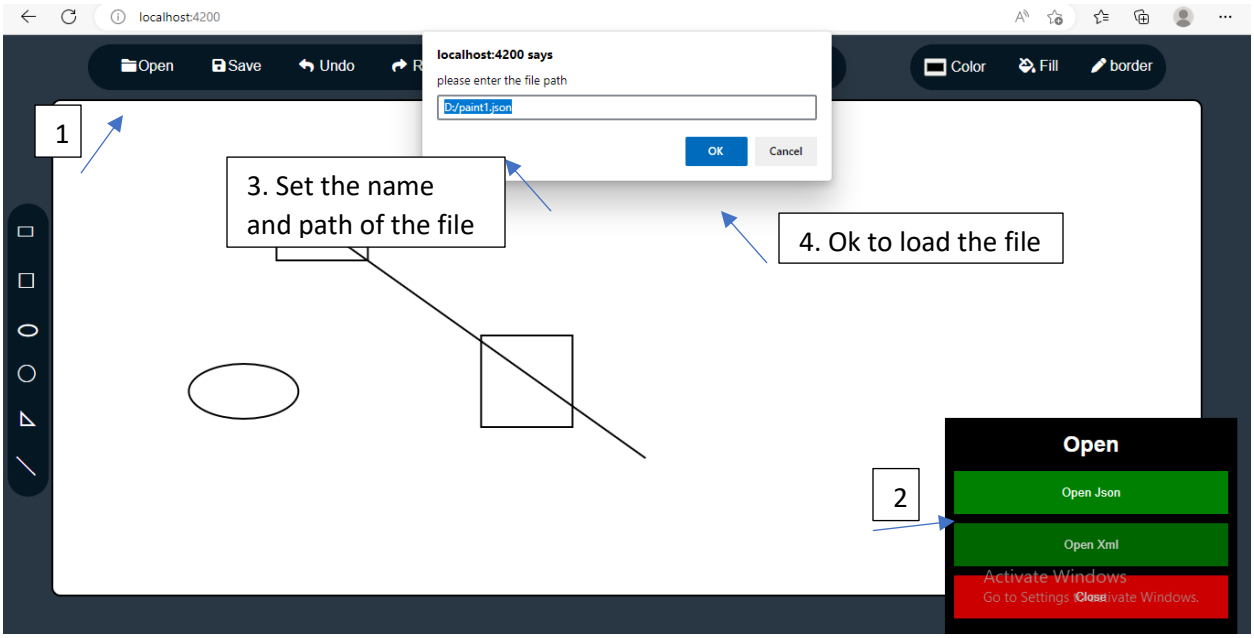


2. Open icon for opening loaded file in the saved folders

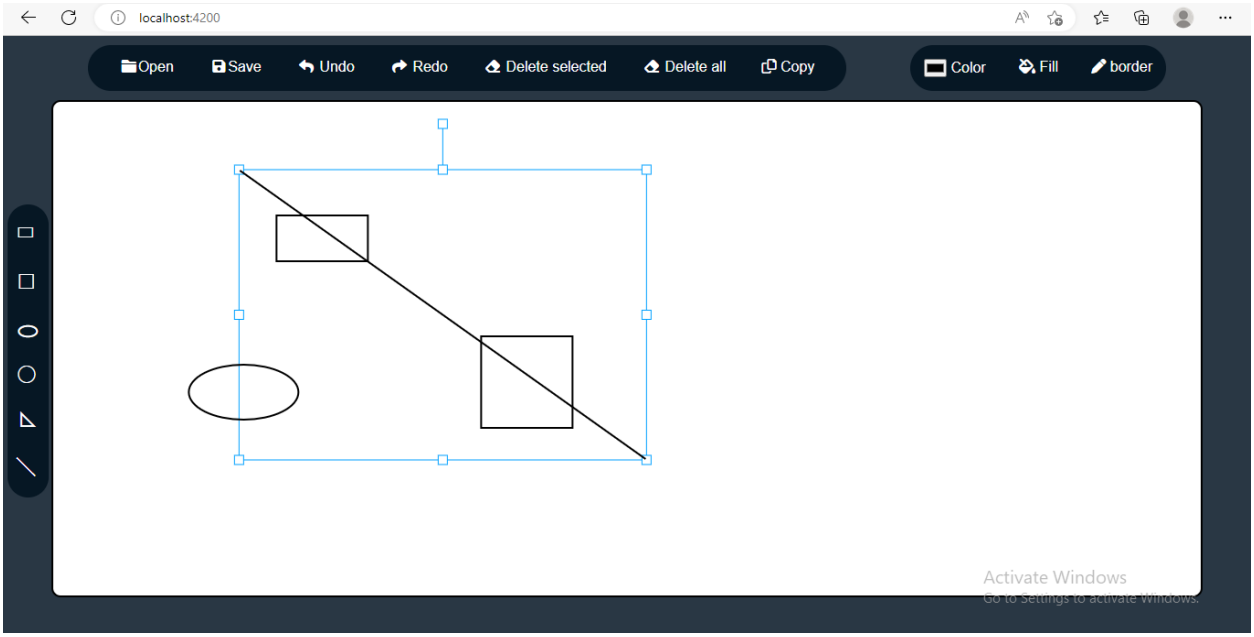


This is the saved file

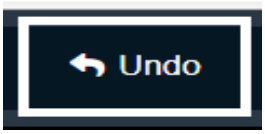

localhost:4200
Open
Save
Undo
Redo
Delete selected
Delete all
Copy
Color
Fill
border
Activate Windows


localhost:4200
Open
Save
Undo
Color
Fill
border
localhost:4200 says
please enter the file path
D:/paint1.json
OK
Cancel
1
3. Set the name and path of the file
4. Ok to load the file
2
Open
Open Json
Open Xml
Close
Activate Windows

Loaded file you saved using the same path you used to save:

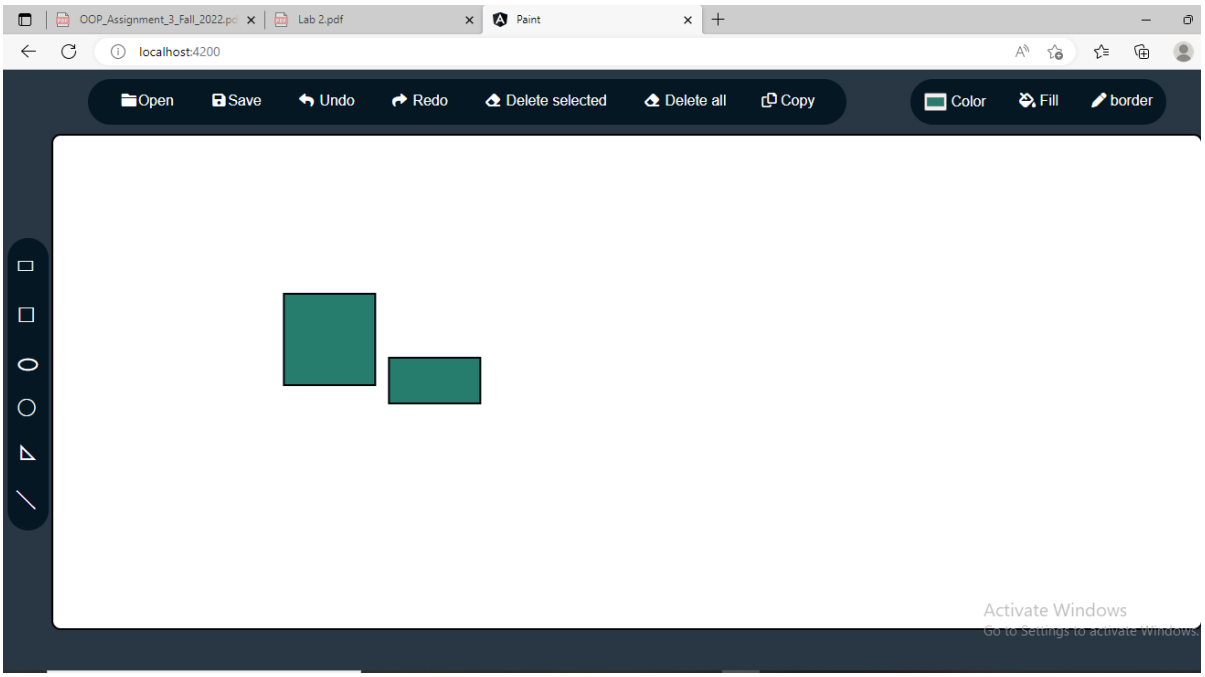


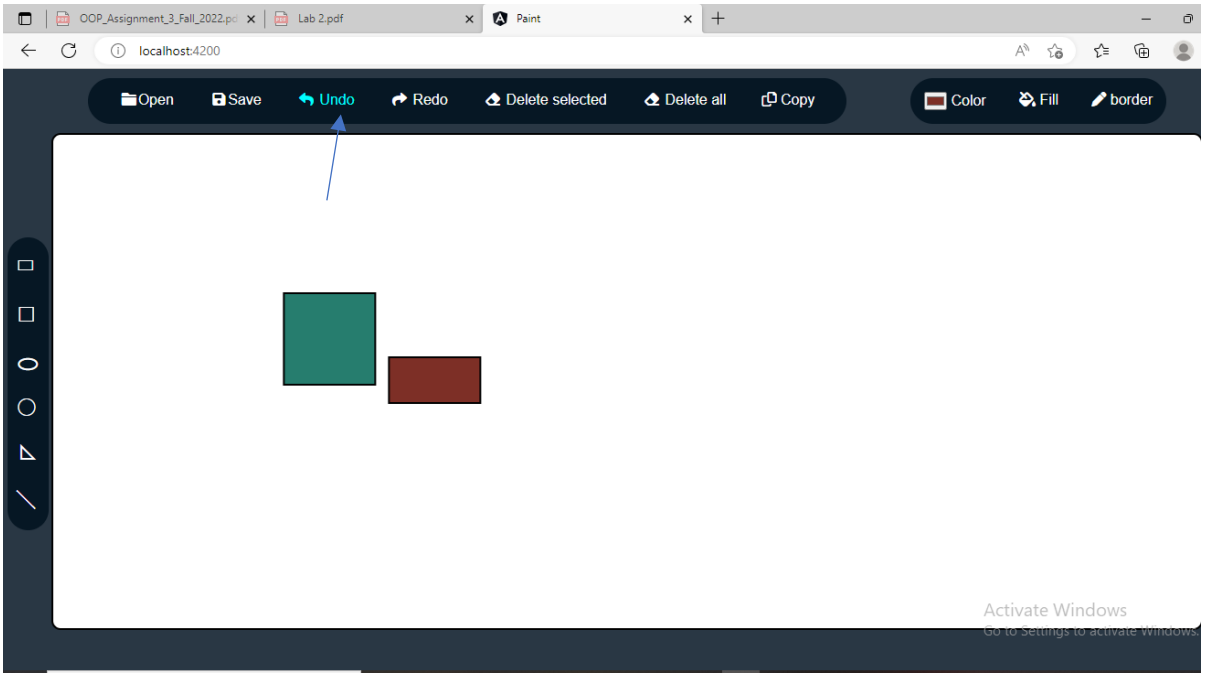
3. Click Undo to back to the previous step/s if you want that.

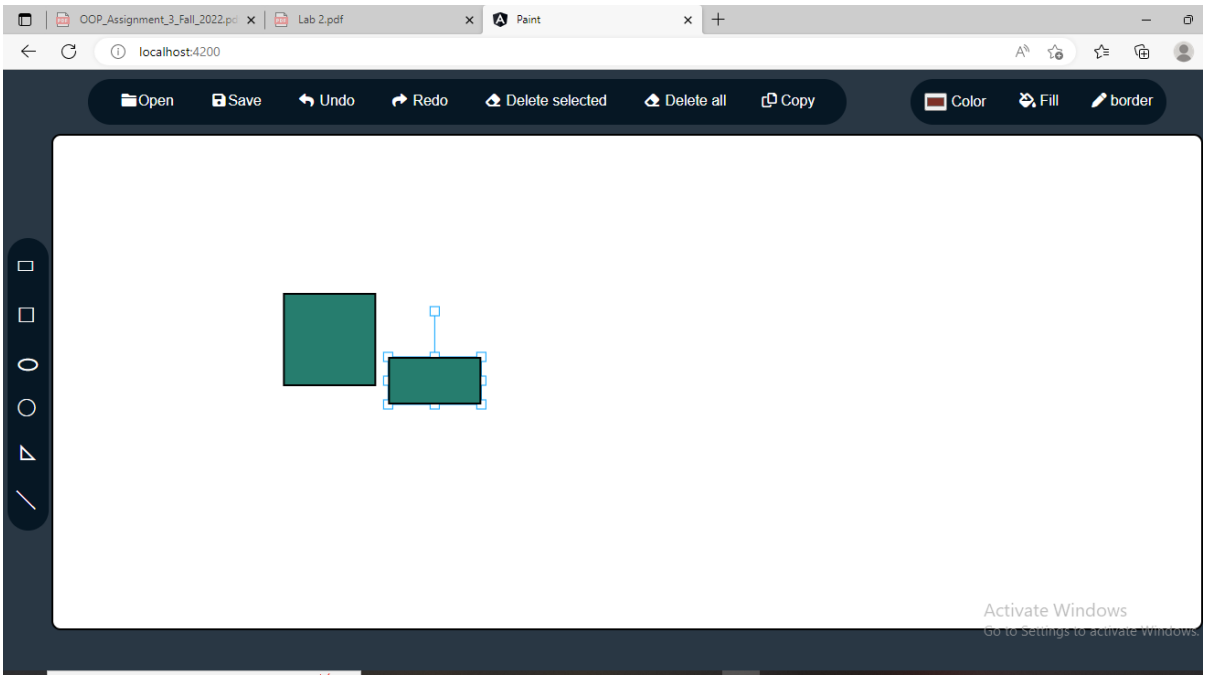


For example:

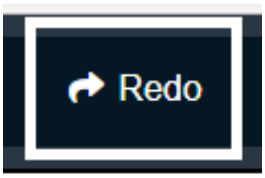
We draw this by coloring rectangle and square into green, then change the color of the rectangle to brown, clicking undo return the previous color to green of square.

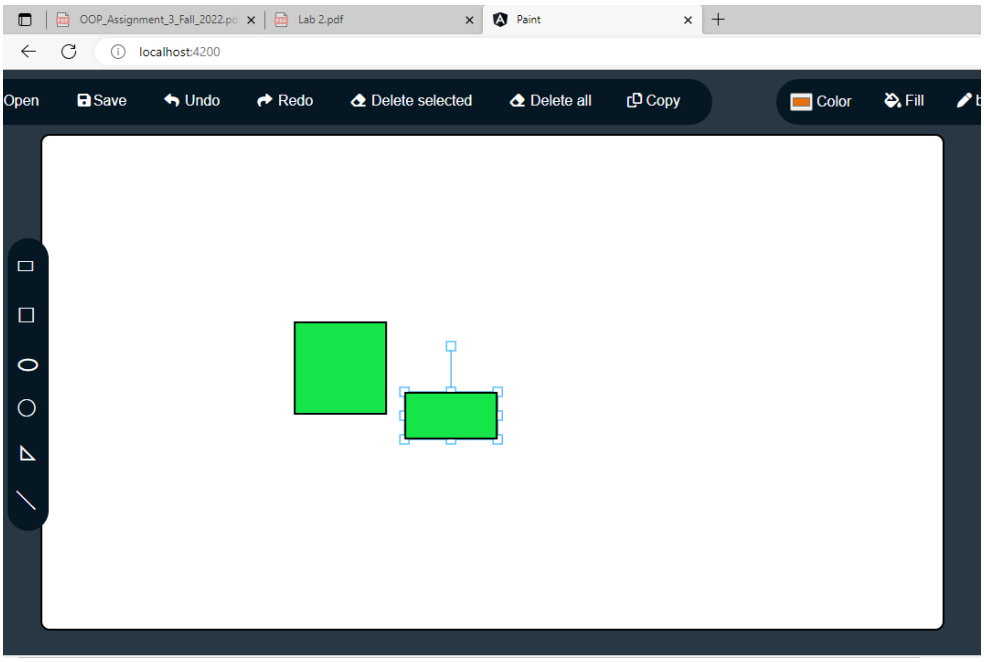

OOP_Assignment_3_Fall_2022.pd
Lab 2.pdf
Paint
localhost:4200
Open
Save
Undo
Redo
Delete selected
Delete all
Copy
Color
Fill
border
Activate Windows

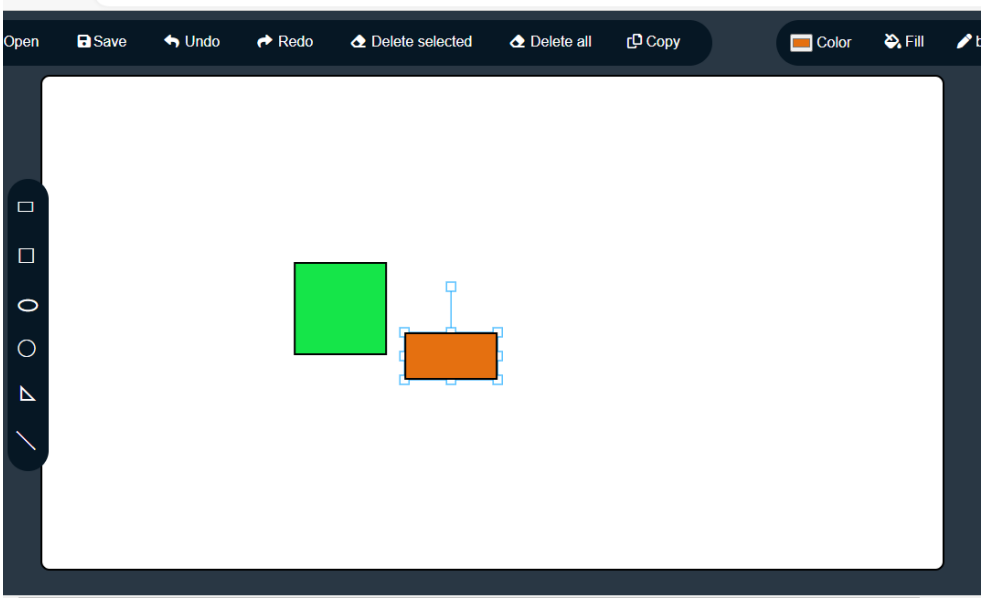

OOP_Assignment_3_Fall_2022.pd
Lab 2.pdf
Paint
localhost:4200
Open
Save
Undo
Redo
Delete selected
Delete all
Copy
Color
Fill
border
Activate Windows



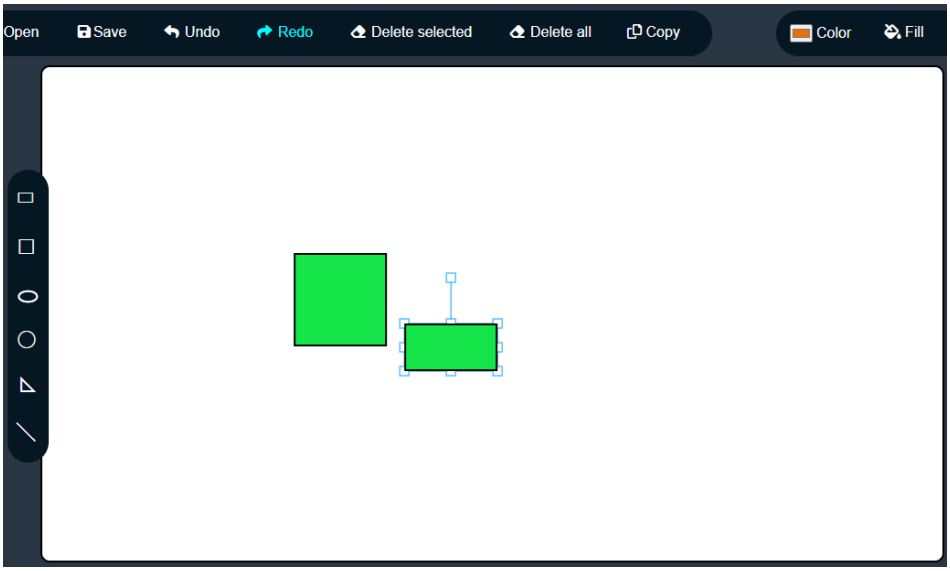
4. Clicking on this icon bring you the last change if you clicked undo before



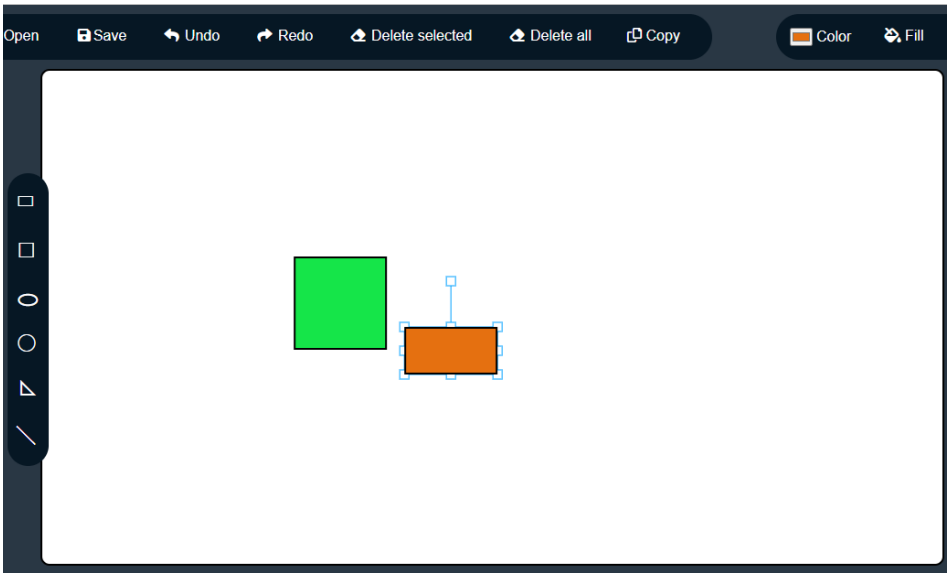




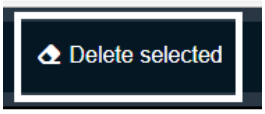
Clicking undo

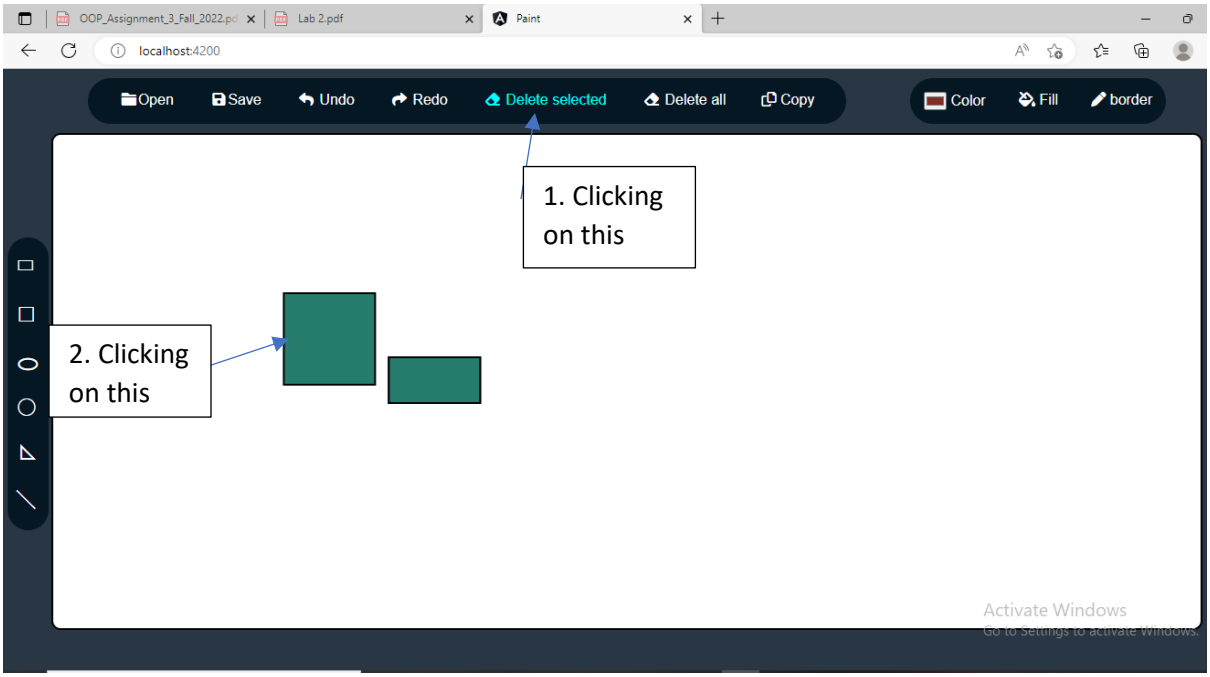


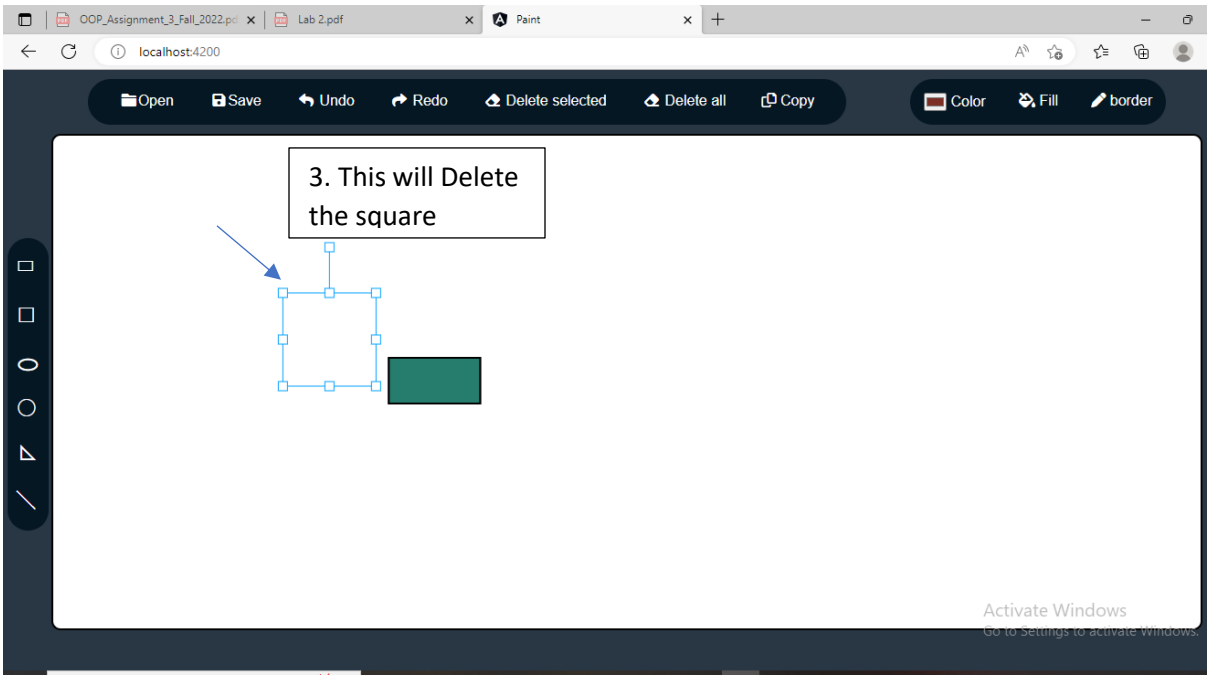
Clicking redo:



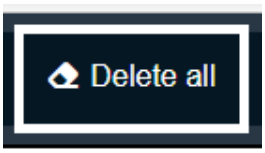
5. clicking on this will delete the selected shape/s

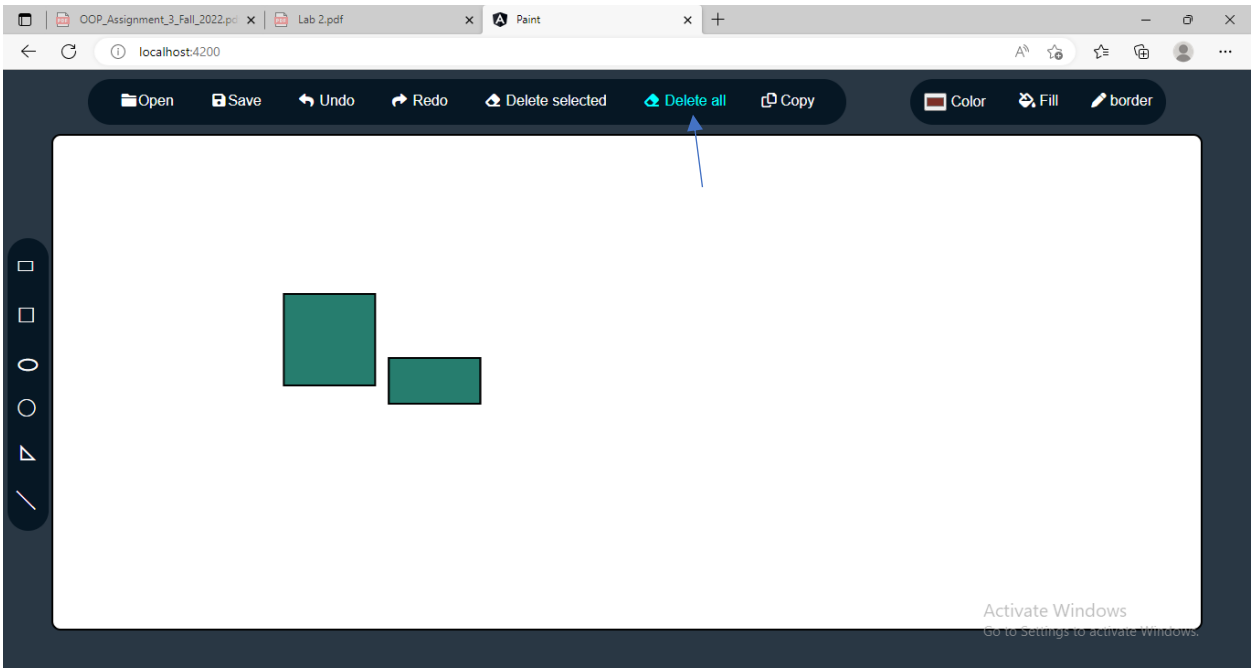


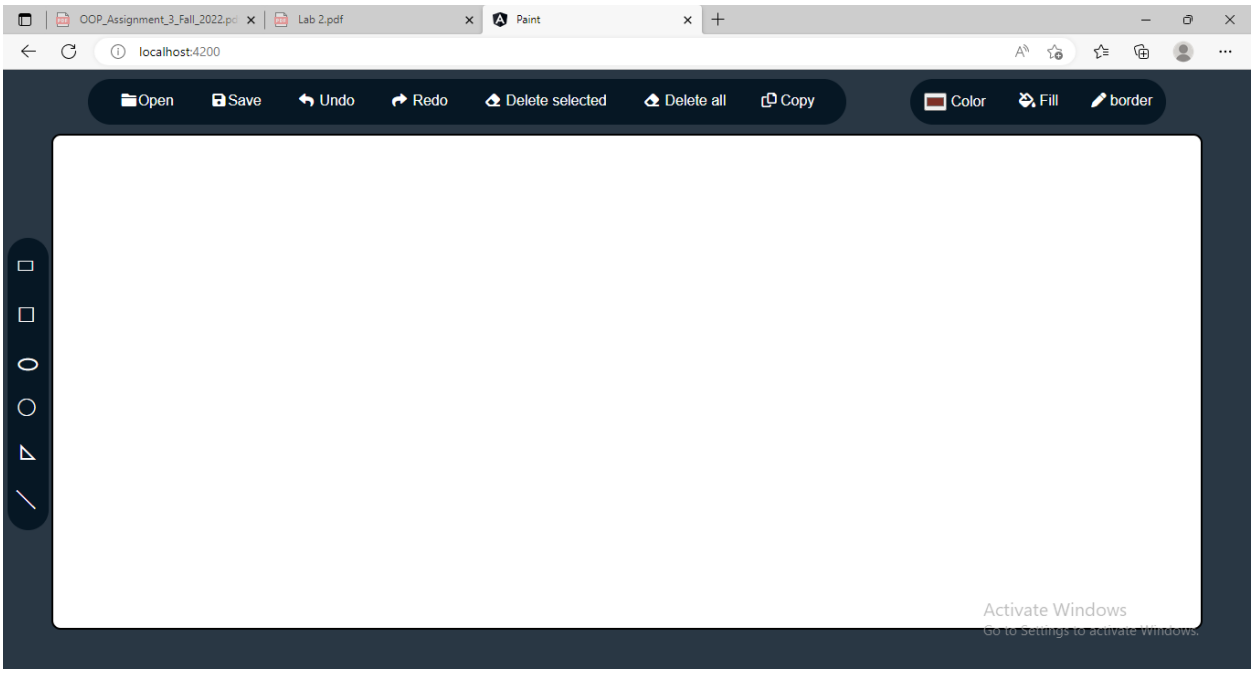




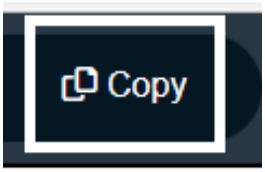
6. To delete all the painting.

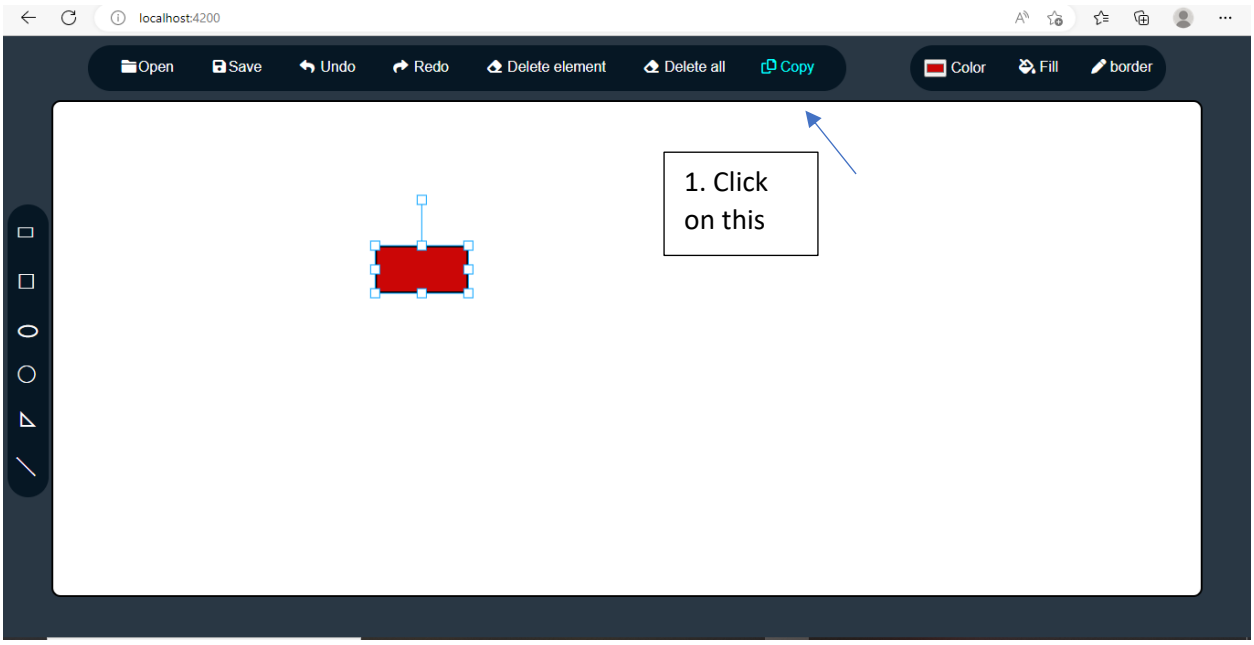


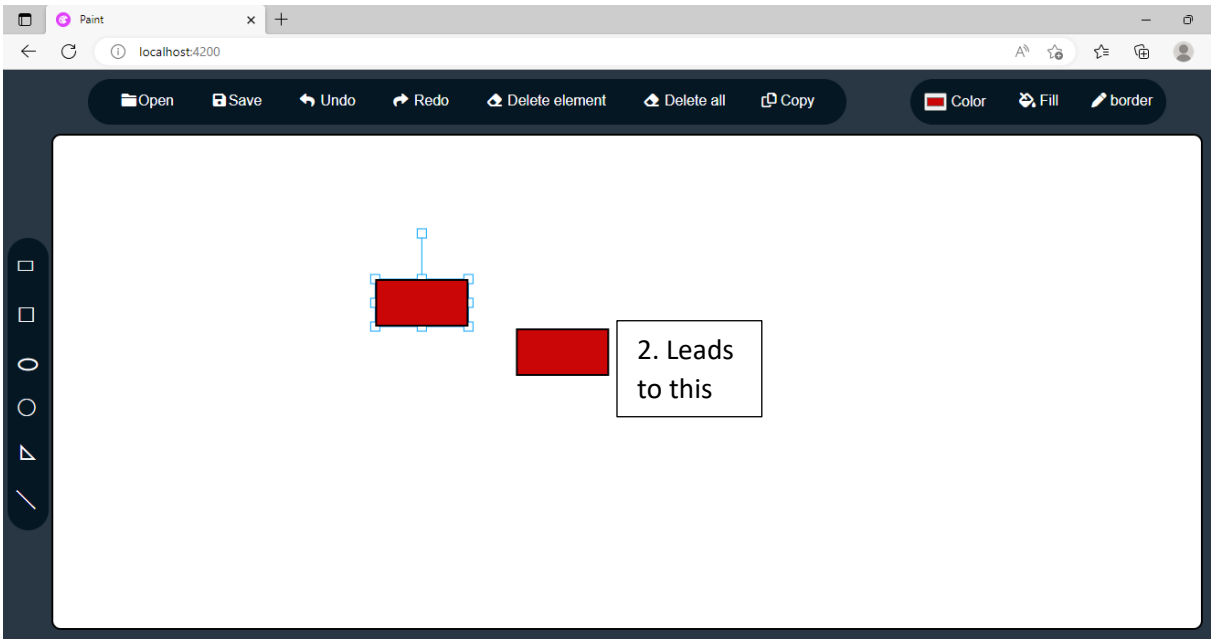




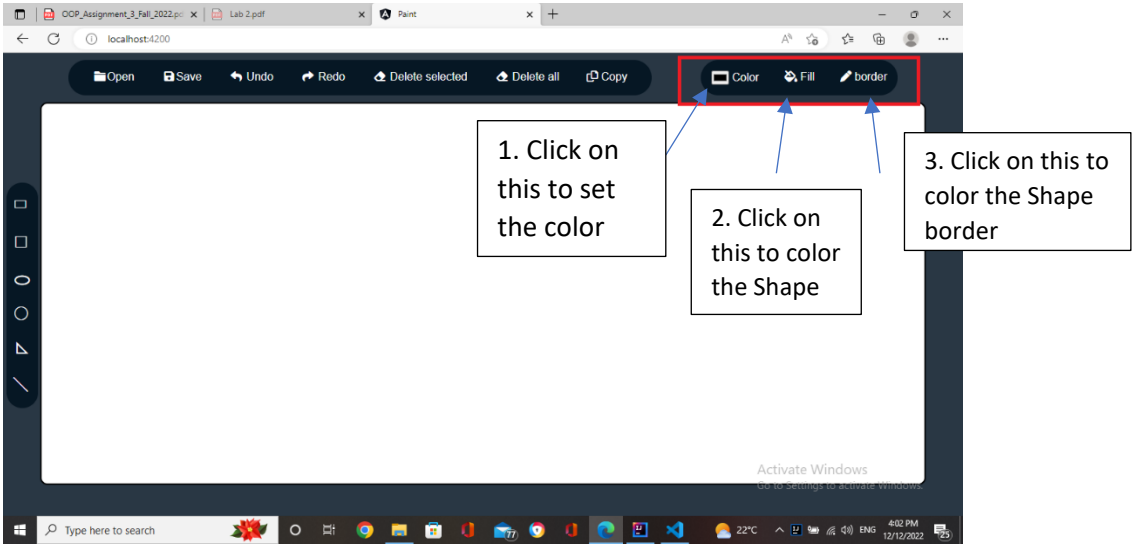
7. Click on this will Copy the shape:



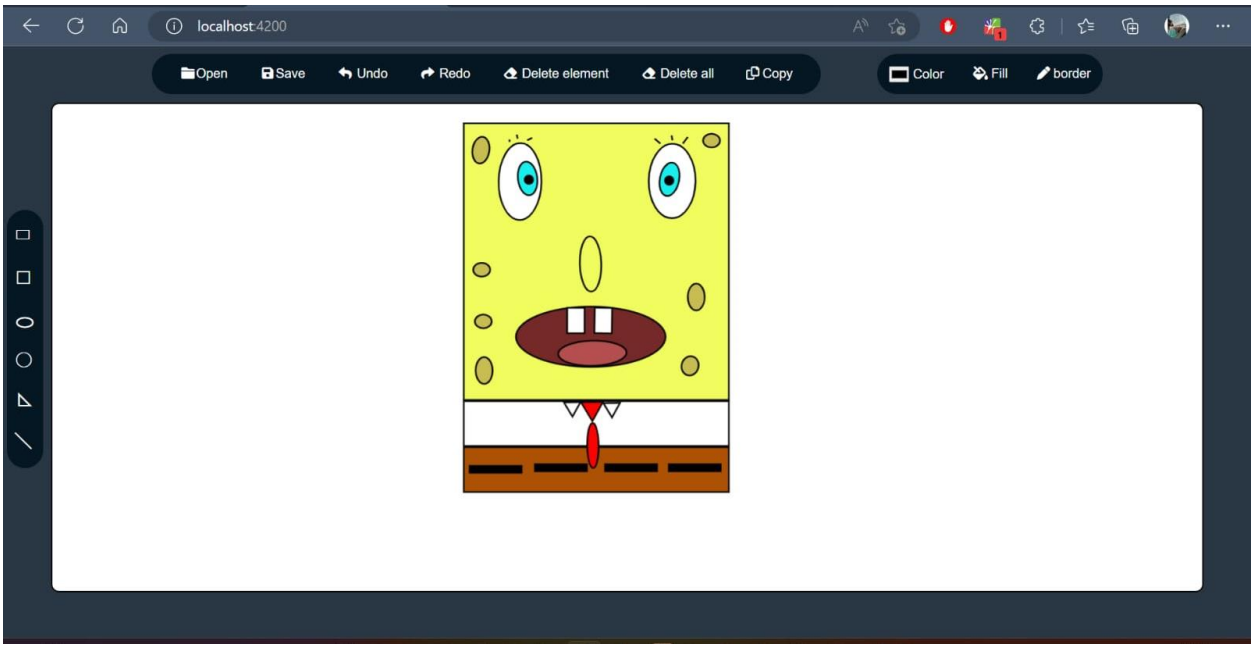


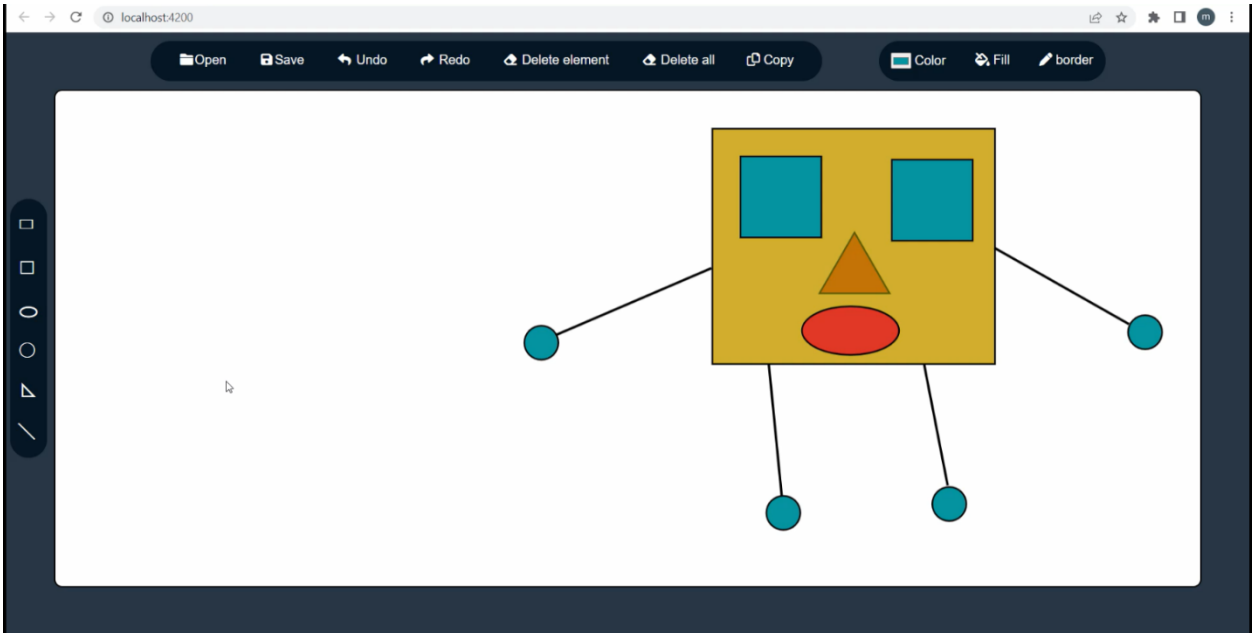


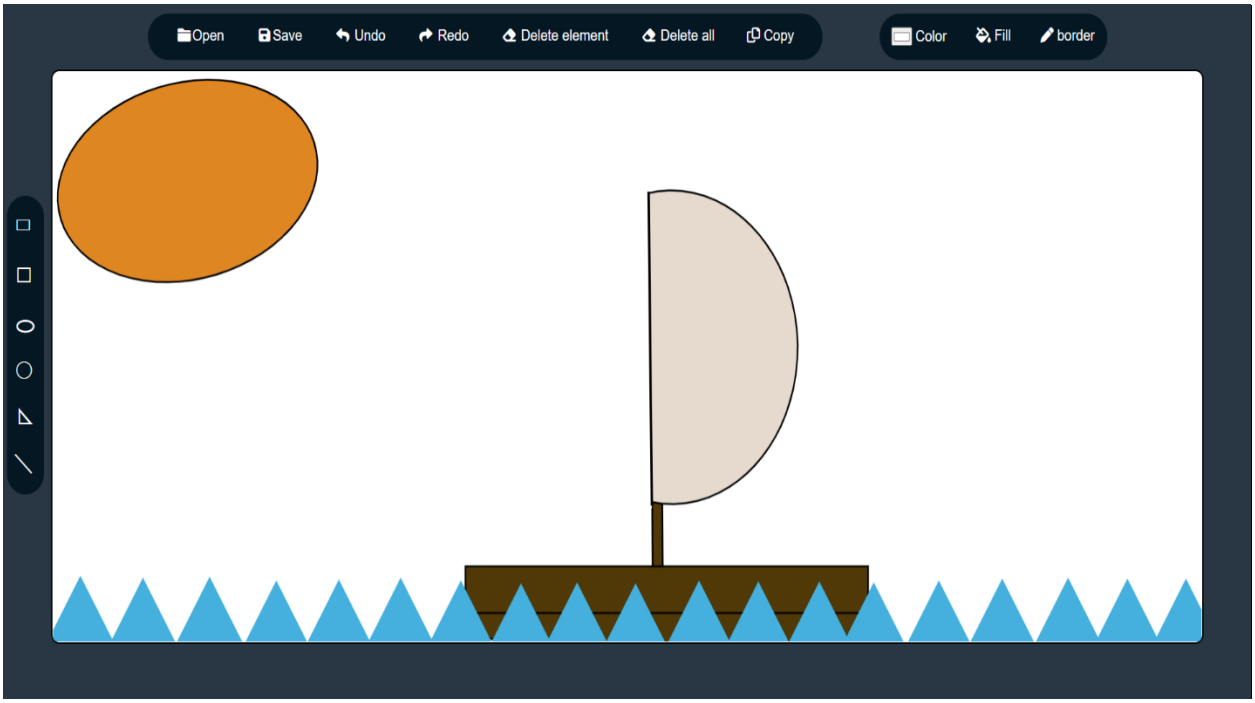
3. Then this last rectangle to color any selected shape. Color icon to specify the color that you want, fill to color inside the shape by this color after selecting the shape. And border icon to color the shape border after selecting the shape.



Some paints with our paint application:



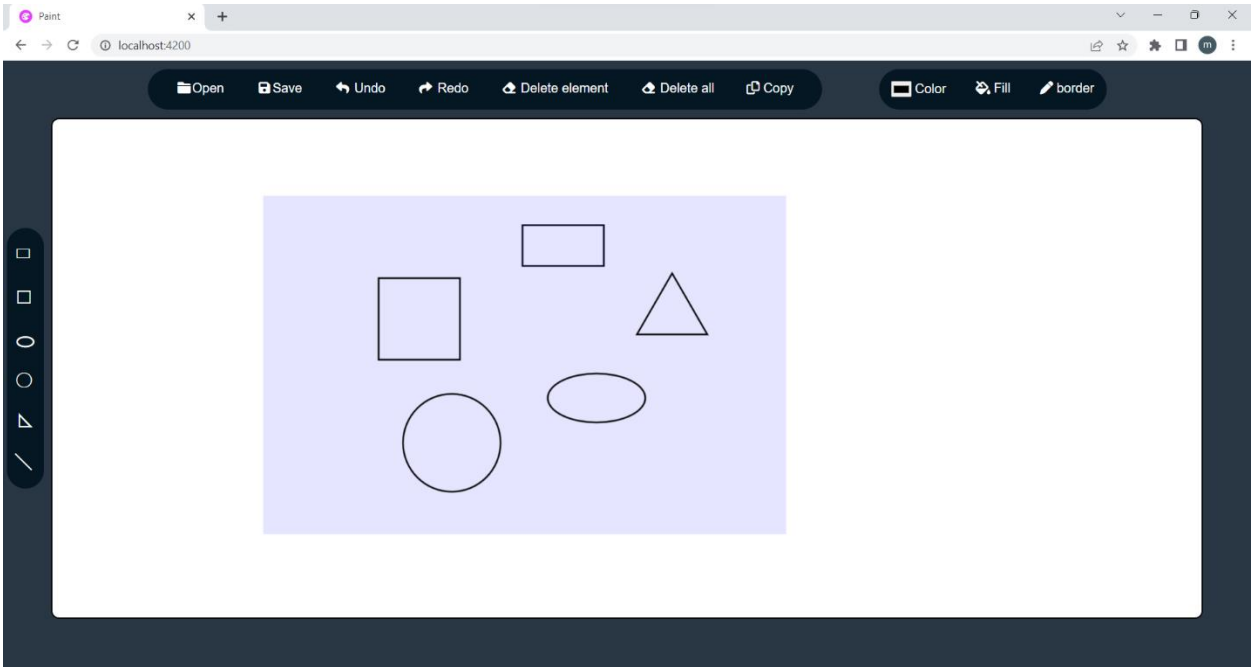




Additional features:

- We add the feature that the user can use the keyboard to copy, undo or redo:
    - Copy: ctrl + c
    - Undo: ctrl + z
    - Redo: ctrl + y

- We add the feature that the user can select more than one shape using the



selection rectangle: