PAINT APPLICATION

A PROJECT REPORT

Submitted by

ABIRAMI M (REG.NO:9517202109003)

BHUVANIKA S (REG.NO:9517202109011)

RAJAKUMARI S (REG.NO:9517202109042)

SUJI S (REG.NO:9517202109051)

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

MEPCO SCHLENK ENGINEERING COLLEGE,SIVAKASI

ANNA UNIVERSITY : CHENNAI 600 025

APRIL 2024

MEPCO SCHLENK ENGINEERING COLLEGE,

SIVAKASI AUTONOMOUS

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

BONAFIDE CERTIFICATE

This is to certify that it is the bonafide work of **BHUVANIKA S(9517202109011), RAJA KUMARI S (9517202109042), SUJI S (9517202109051)** for the mini project titled **"Paint Application"** in **PRINCIPLES OF SOFTWARE ENGINEERING** during the sixth semester December 2024 –April 2024 under my supervision.

SIGNATURE

Dr.P.Thendral M.E.,Ph.D

Associate Professor,

Artificial Intelligence and Data Science

Mepco Schlenk Engineering College

Sivakasi - 626 005

Virudhunagar District

SIGNATURE

Dr.J.Angela Jennifa Sujana M.E.,Ph.D

Professor & Head,

Artificial Intelligence and Data Science

Mepco Schlenk Engineering College

Sivakasi – 626 005

Virudhunagar District

Submitted for the project viva-voce examination to be held on _____.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

Software Requirements Specification

for

Paint Application

Version 1.0 approved

Prepared by Abirami M

Bhuvanika S

RajaKumari S

Suji S

Mepco Schlenk Engineering College

28/02/2024

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The purpose of the paint application system is to provide a user-friendly platform for children and to explore their interests through digital painting. This aims to offer a variety of tools and features, allowing them to experiment with colors, shapes, and textures in a playful environment. The scope of this product includes the development of a standalone application or web-based tool specifically designed for children aged [specify age range], offering features such as drawing tools, color palettes, templates, and save options suitable for their age and skill level.

1.2 Document Conventions

The application must have the document conventions as componentization as React is a component-based library, the components must be organized in correct order. JSX syntax also provides a clear explanation of code structure. Key terms can be highlighted using bold or italicized text to draw attention to important information or requirements

1.3 Intended Audience and Reading Suggestions

The intended audience are the Developers, Project managers, Designers, Testers, Marketing staff,And Documentation Writers. The reading suggestion follows the sequence as Introduction, scope of the project, Functional Requirements, Non-functional Requirements, User Interface design, System Architecture, Project Management, Testing Requirement, User Documentation and Appendices.

1.4 Product Scope

The scope of the product is to create a digital tool that enables the users to expose themselves in various painting features. It must provide a safe and user-friendly platform to experiment with the different colors, shapes, brushes etc. and to support various other helpful field such as education.

1.5 References

https://www.w3schools.com/react/
https://timetoprogram.com/create-image-magnifier-react-js/
https://www.freecodecamp.org/news/learn-react-hooks-by-building-a-paint-app/
https://reactscript.com/magic-painter-app/

2. Overall Description

2.1 Operating Environment

The hardware platform required to operate the application is the Laptop or a computer. The app supports multiple OS such as Linux, Windows etc.. The web browser to support for various platform is the Microsoft Edge and the software component required for the project is react framework with JSX for designing. The application is designed to run on either online as well as offline mode

2.2 Design and Implementation Constraints

The application's memory usage should be optimized to ensure smooth performance on devices with varying levels of Random Access Memory. The application is built using React by applying constraints related to the use of React components, JSX syntax, and other methods. Support for older browsers may be limited due to compatibility issues with modern JavaScript features or CSS properties. The application may need to be updated to leverage new features or performance improvements introduced in future versions of React

2.3 User Documentation

Comprehensive user documentation will be provided to guide users on how to use and troubleshoot the appliction effectively.The documentation will include instructions for drawing, painting and colour by numbers.

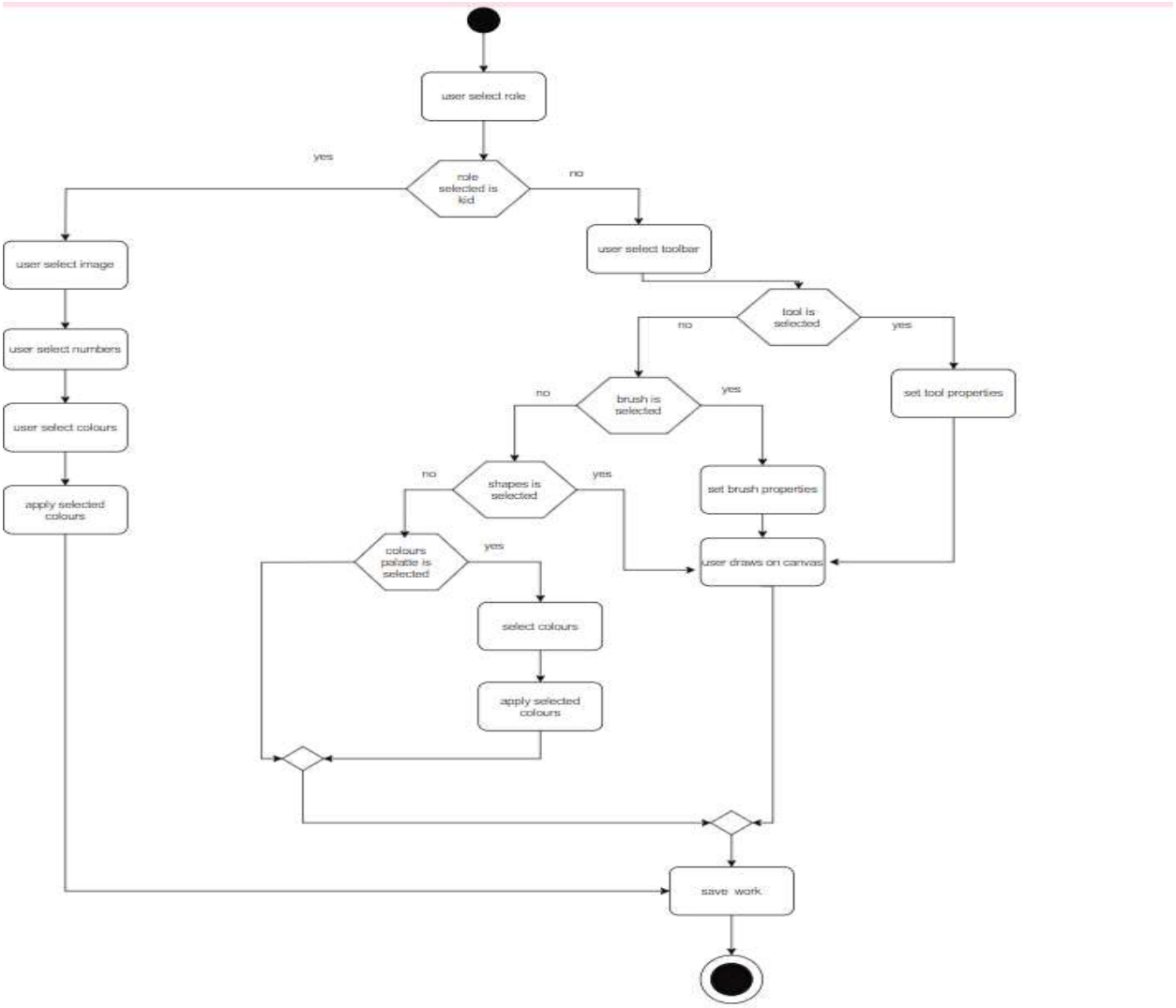
2.4 Assumptions and Dependencies

For the paint application development using react we need a assumption over stable react environment and several resources for the deployment. The dependencies for the project include the compatibility with the browser and the good server infrastructure. The operating system is also important for the dependencies.

3. Design

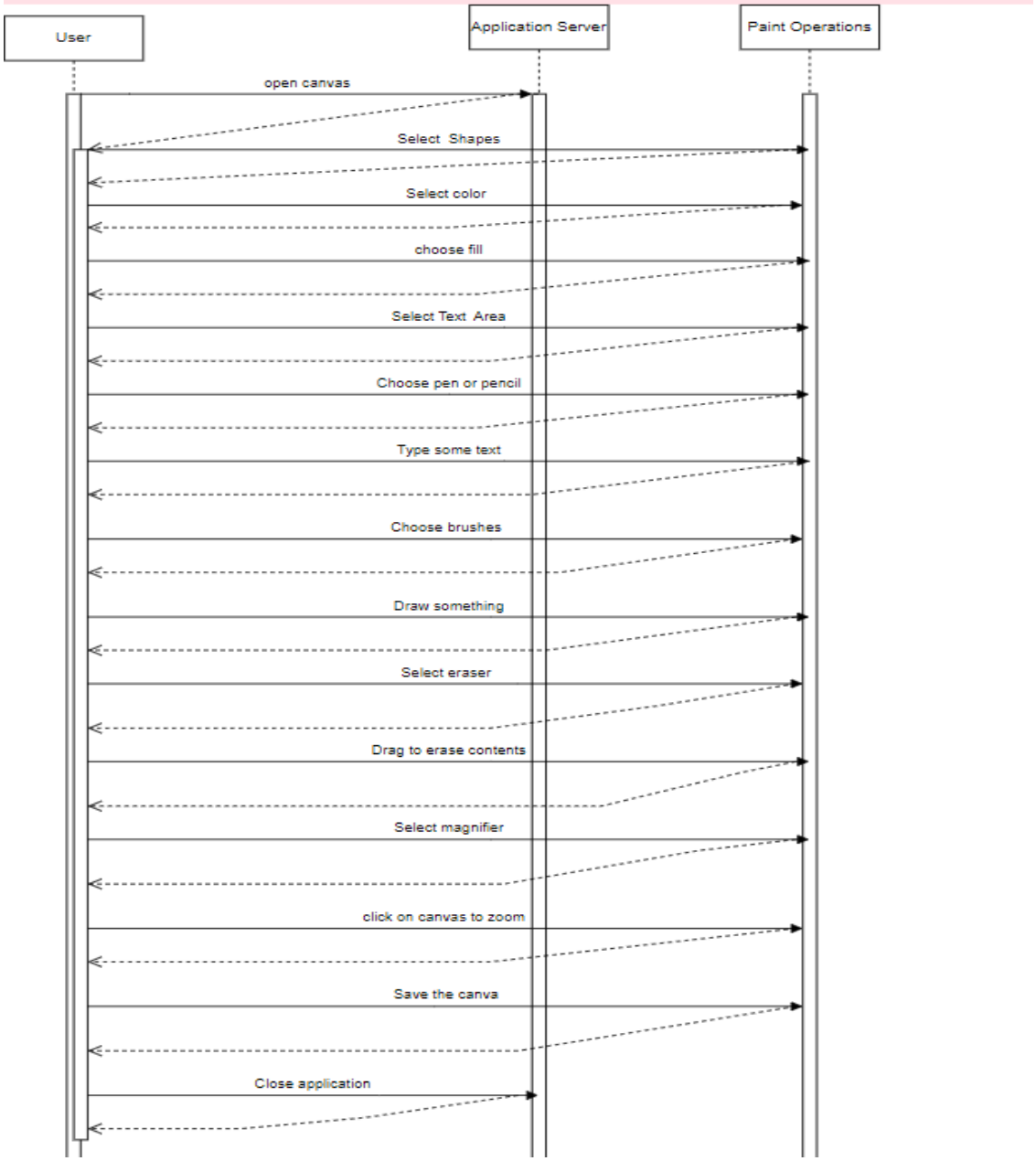
3.1 Activity Diagram

Activity diagrams in UML visually map processes and system behaviors, representing tasks, transitions, decision nodes, and merge symbols. They aid in system analysis, optimization, and software design, fostering clarity and collaboration among stakeholders.



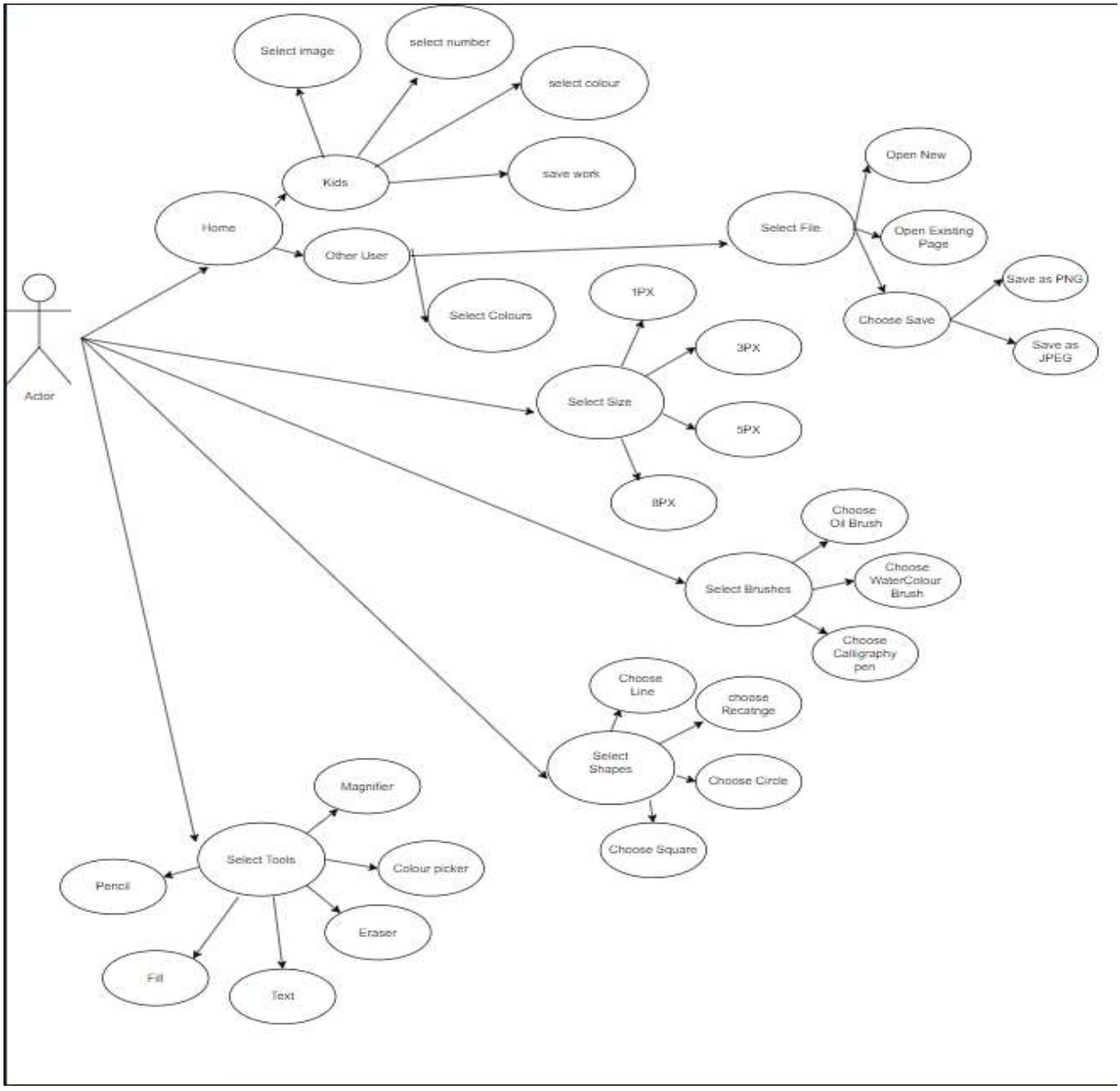
3.2 Sequence Diagram

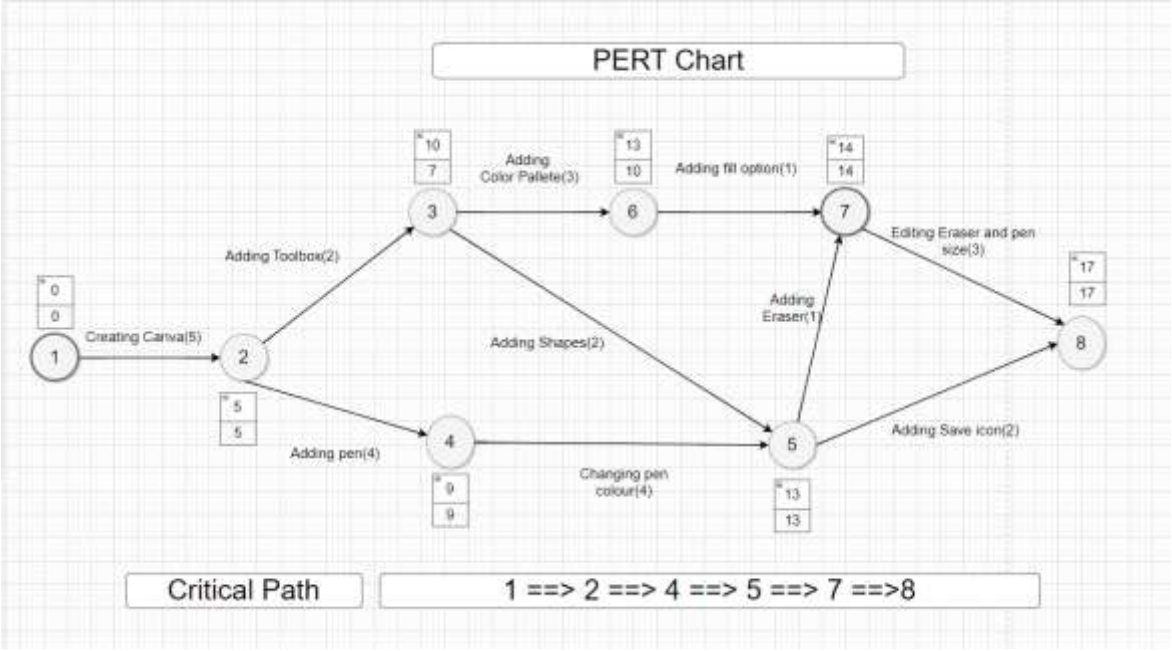
A sequence diagram is a Unified Modeling Language (UML) diagram that illustrates the sequence of messages between objects in an interaction. A sequence diagram consists of a group of objects that are represented by lifelines, and the messages that they exchange over time during the interaction.



3.3 Use Case Diagram

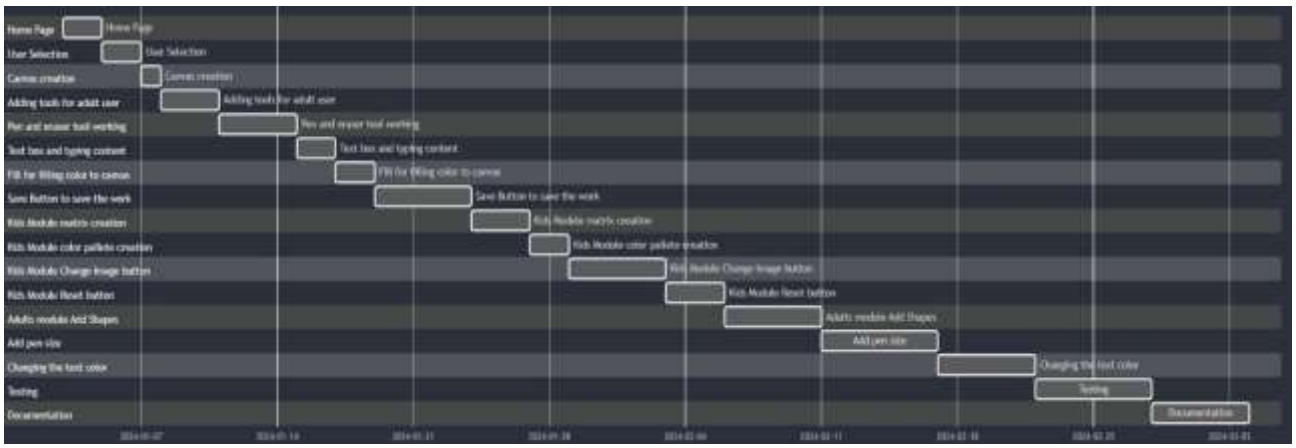
Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally.





3.4 GANTT Chart

A Gantt chart is a project management tool that illustrates work completed over a period of time in relation to the time planned for the work. It typically includes two sections: the left side outlines a list of tasks, while the right side has a timeline with schedule bars that visualize work.



3.5 PERT Chart

A PERT chart, also known as a PERT diagram, is a tool used to schedule, organize, and map out tasks within a project. PERT stands for program evaluation and review technique. It provides a visual representation of a project's timeline and breaks down individual tasks.

4. External Interface Requirements

4.1 User Interfaces

The user interface of the paint application is designed in a user -friendly manner. Users interact with the system through a web-based interface provided by React. The interface also allows users to navigate through various tools available and draw several things they like to do and allows you the save the workspace.

4.2 Hardware Interfaces

The Paint Application does not have specific hardware requirements beyond those typical of devices used to access web applications. Users can access the system from desktop computers, laptops, tablets with or without internet connectivity. The system is designed to be compatible with a wide range of hardware configurations, ensuring accessibility for users across different devices

4.3 Software Interfaces

The primary software Interface used by the application is React. Basically React is the most popular javascript library for building user interfaces. It is fast, flexible and it also has a strong community sitting online to help you every time. Its main application is in building single-page applications (SPAs), allowing smooth content updates without page reloads. With a focus on reusable components and a user-friendly syntax, React simplifies UI development, excelling in real-time applications and seamlessly integrating with backends.

4.4 Communications Interfaces

Users also communicate with the system through interacting with the user interface provided by React. The communication standard used here is HTTP which is IP based communication protocol that is used to deliver data from server to client or vice-versa.

5. System Features

5.1 System Feature 1: Drawing Tools

Description: This feature includes various tools that allow users to draw on the canvas.
Components:

- Pen tool: Allows freehand drawing with adjustable stroke size and color.
- Eraser tool: Enables users to erase parts of the drawing.
- Shapes tool: Provides options for drawing basic shapes like rectangles, circles, and lines.
- Text tool: Allows users to add text to the canvas, with options for font, size, and color.
- Brush tool: Offers different brush presets for artistic effects.

Functionality:

- Users can select different drawing tools from a toolbar.
- Tools should have customizable settings such as color, size, opacity, etc.
- Users can draw, erase, and manipulate shapes and text on the canvas.

4.2 System Feature 2: Canvas Manipulation

Description: This feature involves functionalities related to manipulating the canvas and its content.
Components:

- Zoom: Allows users to zoom in and out of the canvas for detailed work.
- Undo/redo: Provides the ability to undo and redo actions performed on the canvas.
- Layers: Allows users to work with multiple layers, rearrange them, and adjust their properties.
- Selection tool: Enables users to select and move individual elements on the canvas.

Functionality:

- Users can zoom in and out of the canvas using controls or gestures.
- Undo/redo functionality tracks user actions and allows them to revert or redo changes.
- Layers functionality enables users to organize and manage complex compositions.
- Selection tool allows users to select, move, resize, and manipulate individual elements on the canvas.

6. Other Nonfunctional Requirements

6.1 Performance Requirements

- Responsiveness: The application should respond promptly to user actions, ensuring smooth drawing experiences even with complex designs.
- Efficiency: It should utilize system resources efficiently to minimize lag and ensure a seamless user experience, even on low-spec devices.
- Scalability: The application should be able to handle increasing user loads without significant degradation in performance.

6.2 Safety Requirements

- Data Safety: Ensure that user data (such as saved drawings) is securely stored and protected from loss or corruption.
- Error Handling: Implement robust error handling mechanisms to gracefully handle unexpected errors and prevent crashes or data loss.

- User Protection: Provide safeguards to prevent accidental data loss, such as confirmation prompts before discarding unsaved changes.

6.3 Security Requirements

- Data Encryption: Implement encryption protocols to protect sensitive user data, especially during transmission over the internet.
- Authentication: If applicable (e.g., for user accounts), implement secure authentication mechanisms to ensure only authorized users can access the application and its features.
- Authorization: Enforce appropriate access controls to restrict users' actions based on their roles and permissions

6.4 Software Quality Attributes

- Maintainability: Ensure that the code base is well-structured, documented, and modular, facilitating easy maintenance and future enhancements.
- Reliability: The application should be robust and stable, minimizing the occurrence of bugs, crashes, or unexpected behavior.
- Usability: Design an intuitive and user-friendly interface, with clear instructions and guidance for users of all skill levels.
- Accessibility: Ensure that the application is accessible to users with disabilities, following relevant accessibility standards and guidelines

6.5 Business Rules

Define specific business rules governing the use of the application, such as terms of service, copyright policies for user-generated content, and any restrictions on commercial use or distribution. Ensure compliance with legal regulations and industry standards relevant to the application's domain (e.g., data privacy laws, copyright regulations)

7. Black Box Testing

Black-box testing is a type of software testing in which the tester is not concerned with the internal knowledge or implementation details of the software but rather focuses on validating the functionality based on the provided specifications or requirements.

Nonfunctional Testing:

Non-functional testing is a software testing technique that checks the non-functional attributes of the system. Non-functional testing is defined as a type of software testing to check non-functional aspects of a software application. It is designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing. Non-functional testing

is as important as functional testing. Non-functional testing is also known as NFT. This testing is not functional testing of software. It focuses on the software's performance, usability, and scalability.

Functional Testing:

Functional testing is defined as a type of testing that verifies that each function of the software application works in conformance with the requirement and specification. This testing is not concerned with the source code of the application. Each functionality of the software application is tested by providing appropriate test input, expecting the output, and comparing the actual output with the expected output. This testing focuses on checking the user interface, APIs, database, security, client or server application, and functionality of the Application Under Test. Functional testing can be manual or automated. It determines the system's software functional requirements.

Testing	Testing Description	Sample I/P	Expected O/P	O/P	Status
Non-Functional Testing					
Compatibility Testing	To verify that the application works with other systems	Test the app with other system	The app should work properly with all the functionalities	The app works properly with all the functionalities	Passed
Usability Testing	To verify that the application is easy to use	Working with the application	The app should be easy to use	The app is easy to use	Passed

Testing	Testing Description	Sample I/P	Expected O/P	O/P	Status
Functional Testing					
Input Testing	To verify that the pen tool works properly	Click and drag on the canvas area	The dragged area with the pen markings	The dragged area with the pen markings	Passed
	To verify that the eraser works properly	Click and drag on the canvas area	The dragged area with the markings are removed	The dragged area with the markings are removed	Passed
	To verify that the fill color works properly	Click on the fill icon and drag over the canvas	The canvas would be filled by colors	The canvas would be filled by colors	Passed
	To verify that the shapes works properly	Click on the shapes icon	The selected shapes would be visible	The selected shapes would be visible	Passed
	To verify that text area works properly	Click on the text icon and type some text	The entered text would be visible	The entered text would be visible	Passed
	To verify that the pen tool is clicked	Click on the pen icon	The icon is clicked	The icon is clicked	Passed
	To verify that the eraser tool is clicked	Click on the eraser icon	The icon is clicked	The icon is clicked	Passed
	To verify that the fill color is clicked	Click on the fill color icon	The icon is clicked	The icon is clicked	Passed
	To verify the shapes icon is clicked	Click on the shapes icon	The shapes icon is clicked	The shapes icon is clicked	Passed
	To verify that the pen color changes	Click on the color and draw over the canvas	The dragged area with the color in the pen markings	The dragged area with the color in the pen markings	Passed
	To verify that the shape size is increased	Click and drag the shapes on the canvas	The shape size is increased	The shape size is increased	Passed
	To verify that the shape size is decreased	Click and drag the shapes on the canvas	The shape size is decreased	The shape size is decreased	Passed
	To verify that the pen and eraser width works properly	Choose the required width	The width of the eraser and pen would be increased	The width of the eraser and pen would be increased	Passed

8. Analysis Models

COCOMO MODEL

Let's assume we'll have 2000 lines of code

For Organic Model,

- Effort=a*(LOC)^b
- Time=c*(Effort)^d
- No of People Required = Effort/Time

Effort = 2.4*(2000)^1.05
=5165.244

Time = 2.5*(5165.244)^0.38

=30.395

Persons-required = 5165.244/30.395

= 169.754

For Semi-Detached Model,

- Effort = a*(LOC)^b * EAF
- Time = c*(Effort)^d
- Persons-required = Effort/Time

Effort = 3.0*(2000)^1.12 * 1

=8964.258

Time = 2.5*(8964.258)^0.35

=43.107

Persons-required = 8964.258/43.107

=207.935

For Embedded Model,

- Effort = 3.6*(2000)^1.20*1

  = 12469.491

- Time = 2.5*(12469.491)^0.32

  = 53.847

- Persons-required = 12469.491 / 53.847

= 231.695

Function Point Analysis

Functional Components:

- ColorPicker Component
- DrawingCanvas Component
- ToolSelector Component
- SaveButton Component
- Magnifier Component

Complexity Assessment:

- ColorPicker: Medium
- DrawingCanvas: High
- ToolSelector: Medium
- SaveButton: Low
- Magnifier: Medium

Function Point Categories:

- External Inputs (EIs): ColorPicker, ToolSelector, SaveButton, Magnifier
- External Outputs (EOs): DrawingCanvas
- External Inquiries (EQs): None
- Internal Logical Files (ILFs): None
- External Interface Files (EIFs): None

Function Point Counts:

- EIs: 4 (ColorPicker, ToolSelector, SaveButton, UndoRedo)
- EOs: 1 (DrawingCanvas)
- EQs: 0
- ILFs: 0
- EIFs: 0

Weights:

Let’s Assign weights to each function point category based on complexity as

- EIs: Medium = 7 points each
- EOs: High = 5 points each
- EQs: Low = 3 points each

- ILFs: Not applicable
- EIFs: Not applicable

Calculation:

- EIs: 4 * 7 = 28
- EOs: 1 * 5 = 5
- EQs: 0
- ILFs: 0
- EIFs: 0

Total Function Points:

- Total = EIs + EOs + EQs + ILFs + EIFs
- Total = 28 + 5 + 0 + 0 + 0 = 33 function points

9. Other Requirements

- Database Requirements: Specify the database technology to be used for storing user data, such as drawings, user profiles (if applicable), and application settings.
- Internationalization Requirements: If the application will be available in multiple languages, outline requirements for supporting internationalization, such as language selection options and localization of user interface elements.
- Legal Requirements: Identify any legal considerations relevant to the project, such as compliance with data privacy regulations (e.g., GDPR), copyright laws for user-generated content, and any licensing requirements for third-party libraries or components.
- Reuse Objectives: Define objectives for reusing code or components within the project, such as utilizing reusable React components, libraries, or design patterns to streamline development and maintenance.
- Integration Requirements: Specify any external systems or services that the application needs to integrate with, such as social media platforms for sharing artwork or cloud storage services for saving drawings.
- Performance Monitoring: Define requirements for monitoring and optimizing the application's performance over time, such as implementing performance tracking tools and conducting regular performance audits.
- Accessibility Requirements: Detail requirements for ensuring the application is accessible to users with disabilities, including compliance with accessibility standards (e.g., WCAG) and testing procedures for accessibility features.

Appendix: Code

App.js

```
import { BrowserRouter, Routes, Route } from 'react-router-dom';
import HomePage from './pages/HomePage';
import Nextpage from './pages/Nextpage';
import Kid from './pages/Kid';

import Appli from './components/Appli';

function App() {
  return (
    <>
      <BrowserRouter>
        <Routes>
          <Route path='/' element={<HomePage />} />
          <Route path='/nextpage' element={<Nextpage />} />
          <Route path='/kid' element={<Kid />} />


          <Route path='/appli' element={<Appli />} />



        </Routes>
      </BrowserRouter>
    </>
  );
}

export default App;
```

HomePage.js

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
  }

  /* Apply basic styles */
  body {
    font-family: Arial, sans-serif;
    background-color: #fff;
    margin: 0;
    padding: 0;
  }
  h6 {
```

```
  font-size: 14px;
  text-align: center;
  color: #000000;
}

header {
  background-color: #333;
  color: #fff;
  padding: 20px 0;
}

nav ul {
  list-style-type: none;
  display: flex;
  justify-content: center;
}

nav ul li {
  margin: 0 20px;
}

nav ul li a {
  text-decoration: none;
  color: #fff;
  font-weight: 600;
  transition: color 0.3s ease;
}

nav ul li a:hover {
  color: #ff6600;

}

.hero {
  text-align: center;
  padding: 250px 0;
  background-image: url('./1816874.jpg');
  background-size: cover;
  color: #fff;
}

.hero h1 {
  font-size: 36px;
  margin-bottom: 20px;
}

.hero p {
  font-size: 18px;
  margin-bottom: 40px;
}

.btn {
  display: inline-block;
  padding: 10px 20px;
```

```
  background-color: #ff6600;
  color: #fff;
  text-decoration: none;
  border-radius: 5px;
  font-weight: 600;
  transition: background-color 0.3s ease;
}

.btn:hover {
  background-color: #ff9933;
}

.about {
  text-align: center;
  padding: 40px 0;
}

.about h2 {
  font-size: 24px;
  margin-bottom: 20px;
}

footer {
  background-color: #333;
  color: #fff;
  text-align: center;
  padding: 20px 0;
}
.dropdown {
  position: relative;
  display: inline-block;
}

.dropdown-toggle {
  background: none;
  border: none;
  cursor: pointer;
}

.dropdown-menu {
  list-style: none;
  padding: 0;
  margin: 0;
  background-color: #fff;
  border: 1px solid #ccc;
  position: absolute;
  top: 100%;
  left: 0;
  display: none;
  z-index: 1;
}


.dropdown.open .dropdown-menu {
```

```
    display: block;
  }

  .dropdown-menu li {
    padding: 10px;
    border-bottom: 1px solid #ccc;
  }

  .dropdown-menu li:last-child {
    border-bottom: none;
  }
```

NextPage.js

```
import React from 'react';
import { Link } from 'react-router-dom';
import '../styles/NextPage.css';

function NextPage() {
  return (
    <div className="next-page-container">
      <div className="left-background"></div>
      <div className="content-container">
        <h1>You Are .....!?</h1>
        <Link to="/kid">
        <h1><button className="role-button">Kid</button></h1>
        </Link>
        <Link to="/appli">
          <h1><button className="role-button">Teen</button></h1>
        </Link>
        {/* <Link to="/fetch">
        <h1><button className="role-button">  Fetch  </button></h1>
        </Link> */}

      </div>
      {/* Decorative elements */}
      <div className="decoration decoration2"></div>
      <div className="decoration decoration3"></div>
      <div className="decoration decoration4"></div> {/* Add a new decoration */}
      {/* You can continue adding more decorative elements */}
    </div>
  );
}

export default NextPage;
```

NextPage.css

```
.next-page-container {
    display: flex;
    justify-content: space-between;
    align-items: center;
    min-height: 100vh;
    /* padding: 20px; */
```

```
  }


  .left-background {
    width: 50%; /* Adjust the width as needed */
    background-image: url('./1816847.jpg'); /* Replace with the URL of your background image */
    background-size: cover;
    background-repeat: no-repeat;
    background-position: left center;
    min-height: 100vh;
  }



  .content-container {
    width: 50%;
    text-align: center;
  }

  h1 {
    font-size: 36px;
    margin-bottom: 20px;
  }

  .role-button {
    padding: 15px 20px;
    background-color: #ff6600;
    color: #fff;
    border: none;

    border-radius: 15px;
    font-weight: 1000;
    margin: 5px;
    cursor: pointer;
  }

  /* Decorative elements */
  .decoration {
    position: absolute;
    background-color: #ff6600; /* Decoration color */
    border-radius: 50%;
  }

 /* NextPage.css */

/* ... Previous styles ... */

/* Decorative elements */
.decoration1 {
    width: 20px;
    height: 20px;
    top: 20%; /* Adjusted to be closer to decoration2 */
    left: 10%; /* Adjusted to be closer to decoration2 */
```

```
  }

  .decoration2 {
    width: 15px;
    height: 15px;
    top: 20%;
    left: 80%;
  }

  .decoration3 {
    width: 25px;
    height: 25px;
    top: 50%; /* Adjusted to be closer to decoration4 */
    left: 65%; /* Adjusted to be closer to decoration4 */
  }

  .decoration4 {
    width: 18px;
    height: 18px;
    top: 70%;
    left: 70%;
  }

  .decoration5 {
    width: 22px;
    height: 22px;
    top: 50%; /* Adjusted to be closer to decoration4 */
    left: 15%; /* Adjusted to be closer to decoration4 */
  }
```

Palettes.js

```
import React from 'react';

const rgbValueArrays = [
  // Corresponds to palette 'primary' or 0
  [
    [255, 255, 255], [72, 72, 72],
    [255, 42, 42], [252, 187, 66],
    [255, 255, 76], [78, 137, 53],
    [31, 31, 255], [231, 153, 251],
    [170, 235, 255], [254, 213, 255]
  ],
  // Corresponds to palette 'bluey' or 1
  [
    [255, 255, 255], [25, 25, 25],
    [48, 89, 138], [115, 177, 232],
    [98, 200, 110], [241, 240, 233],
    [226, 122, 55], [242, 210, 135],
    [227, 108, 108], [44, 44, 87]
  ],
  []
];
```

```
// Pass in rgbValueArrays[puzzleIndex.paletteId] - then return an array of 0 or 1 (black/white)
values
export const determineFontColor = arr => {
  let tmpArr = [];
  for (let i = 0; i < 10; i++) {
    if ((arr[i][0] * 0.299 + arr[i][1] * 0.587 +
      arr[i][2] * 0.114) > 150) tmpArr.push(0); // black font
    else tmpArr.push(1); // white font
  }
  return tmpArr;
};

const Palettes = ({ puzzleIndex }) => {
  const selectedPalette = rgbValueArrays[puzzleIndex.paletteId];
  const fontColorArray = determineFontColor(selectedPalette);

  return (
    <div>
      <h1>Color Palette</h1>
      <div>
        <p>Selected Palette: {puzzleIndex.paletteId}</p>
        <div style={{ display: 'flex', justifyContent: 'space-around' }}>
          {selectedPalette.map((color, index) => (
            <div
              key={index}
              style={{
                backgroundColor: rgb(${color[0]}, ${color[1]}, ${color[2]}),
                color: fontColorArray[index] === 0 ? 'black' : 'white',
                padding: '10px',
                margin: '5px',
              }}
            >
              Color {index + 1}
            </div>
          ))}
        </div>
      </div>
    </div>
  );
};

export default {
    rgbValueArrays,
    determineFontColor
  };
```

Puzzle.js

```
export const puzzles = [{ name: 'kite',
                    palette: 'primary',
                    paletteId: 0,
                    legend: [ 0, 0, 0, 0, 0, 0, 0, 5, 5, 0, 0, 0, 0, 0, 0, 0,
                              0, 0, 0, 0, 0, 0, 0, 5, 5, 0, 0, 0, 0, 0, 0, 0,
```

```
        0, 0, 0, 0, 0, 0, 5, 6, 6, 5, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 5, 0, 6, 6, 0, 5, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 5, 0, 0, 6, 6, 0, 0, 5, 0, 0, 0, 0,
        0, 0, 0, 5, 0, 0, 0, 6, 6, 0, 0, 0, 5, 0, 0, 0,
        0, 0, 5, 0, 0, 0, 0, 6, 6, 0, 0, 0, 0, 5, 0, 0,
        0, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 0,
        2, 0, 2, 0, 0, 0, 0, 6, 6, 0, 0, 0, 0, 2, 0, 2,
        0, 2, 0, 5, 0, 0, 0, 6, 6, 0, 0, 0, 5, 0, 2, 0,
        0, 0, 0, 0, 5, 0, 0, 6, 6, 0, 0, 5, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 5, 0, 6, 6, 0, 5, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 5, 6, 6, 5, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 4, 4, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 4, 0, 0, 4, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 4, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, ]
},
{ name: 'butterfly',
  palette: 'primary',
  paletteId: 0,
  legend:[9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9,
        9, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 9,
        9, 8, 8, 8, 8, 8, 1, 8, 1, 8, 8, 8, 8, 8, 8, 9,
        9, 4, 4, 8, 8, 8, 8, 1, 8, 8, 8, 8, 4, 4, 8, 9,
        9, 4, 2, 4, 8, 8, 8, 1, 8, 8, 8, 4, 2, 4, 8, 9,
        9, 4, 2, 2, 4, 8, 8, 1, 8, 8, 4, 2, 2, 4, 8, 9,
        9, 4, 2, 2, 2, 4, 8, 1, 8, 4, 2, 2, 2, 4, 8, 9,
        9, 4, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 4, 8, 9,
        9, 8, 8, 8 ,8, 4, 5, 1, 5, 4, 8, 8, 8, 8, 8, 9,
        9, 8, 8, 8, 4, 5, 5, 1, 5, 5, 4, 8, 8, 8, 8, 9,
        9, 8, 8, 4, 5, 5, 5, 1, 5, 5, 5, 4, 8, 8, 8, 9,
        9, 8, 4, 5, 5, 5, 5, 1, 5, 5, 5, 5, 4, 8, 8, 9,
        9, 4, 4, 4, 4, 4, 4, 1, 4, 4, 4, 4, 4, 4, 8, 9,
        9, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 9,
        9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9,
        9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9]


},
{ name: 'popsicle',
  palette: 'primary',
  paletteId: 0,
  legend:[3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3, 3, 3, 3, 3, 3,
        3, 3, 3, 3, 3, 3, 4, 9, 9, 9, 4, 3, 3, 3, 3, 3,
        3, 3, 0, 0, 0, 4, 9, 9, 9, 9, 9, 4, 0, 0, 3, 3,
        3, 3, 0, 0, 4, 9, 9, 9, 9, 9, 9, 9, 4, 0, 3, 3,
        3, 3, 0, 0, 4, 9, 9, 9, 9, 9, 9, 9, 4, 0, 3, 3,
        3, 3, 0, 0, 4, 9, 9, 9, 9, 9, 9, 9, 4, 0, 3, 3,
        3, 3, 0, 0, 4, 9, 9, 9, 9, 9, 9, 9, 4, 0, 3, 3,
        3, 3, 0, 0, 4, 9, 9, 9, 9, 9, 9, 9, 4, 0, 3, 3,
        3, 3, 0, 0, 4, 9, 9, 9, 9, 9, 9, 9, 4, 0, 3, 3,
        3, 3, 0, 0, 4, 9, 9, 9, 9, 9, 9, 9, 4, 0, 3, 3,
        3, 3, 0, 0, 4, 9, 9, 9, 9, 9, 9, 9, 4, 0, 3, 3,
        3, 3, 0, 0, 4, 4, 4, 4, 4, 4, 4, 4, 4, 0, 3, 3,
        3, 3, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 3, 3,
```

```
            3, 3, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 3, 3,
            3, 3, 3, 3, 3, 3, 3, 3, 1, 3, 3, 3, 3, 3, 3, 3,
            3, 3, 3, 3, 3, 3, 3, 3, 1, 3, 3, 3, 3, 3, 3, 3,]

        },
        { name: 'bluey',
          palette: 'bluey',
          paletteId: 1,
          legend: [ 5, 5, 5, 5, 5, 9, 5, 5, 5, 5, 9, 5, 5, 5, 5, 5,
                    5, 5, 5, 5, 9, 9, 5, 5, 5, 5, 9, 9, 5, 5, 5, 5,
                    5, 5, 5, 9, 9, 7, 9, 5, 5, 9, 7, 9, 9, 5, 5, 5,
                    5, 5, 5, 9, 7, 7, 7, 9, 9, 7, 7, 7, 9, 5, 5, 5,
                    5, 5, 5, 9, 9, 9, 9, 3, 3, 3, 9, 9, 9, 5, 5, 5,
                    5, 5, 5, 9, 9, 0, 0, 0, 3, 0, 0, 0, 9, 5, 5, 5,
                    5, 5, 5, 9, 9, 0, 1, 0, 3, 0, 1, 0, 9, 5, 5, 5,
                    5, 5, 5, 9, 9, 0, 1, 0, 3, 0, 1, 0, 9, 5, 5, 5,
                    5, 5, 5, 9, 9, 0, 0, 0, 7, 7, 7, 7, 7, 1, 1, 5,
                    5, 5, 5, 9, 9, 3, 3, 3, 7, 7, 7, 7, 7, 7, 1, 5,
                    5, 5, 5, 3, 3, 3, 3, 7, 7, 7, 7, 7, 7, 7, 7, 5,
                    5, 5, 5, 3, 3, 3, 3, 7, 7, 7, 7, 7, 7, 7, 7, 5,
                    5, 5, 5, 3, 3, 3, 3, 7, 7, 0, 0, 0, 0, 7, 7, 5,
                    5, 5, 5, 3, 3, 3, 3, 7, 7, 8, 1, 1, 1, 5, 5, 5,
                    5, 5, 5, 3, 3, 3, 3, 7, 7, 8, 8, 1, 1, 5, 5, 5,
                    5, 5, 5, 3, 3, 3, 3, 7, 7, 7, 7, 7, 7, 5, 5, 5 ]

        }
    ];
```

Kids.js

```
import './kid.css';
import palettes from './palettes';
import { handleMouseDown, updateBrush, handleMouseMove, handleMouseUp,
handleTouchStart, handleTouchMove, handleTouchEnd } from './events';
import { puzzles } from './puzzles';
import {  determineFontColor } from './palettes';
import React, { useEffect, useState } from 'react';

// Define app function
function App() {
  const [puzzleIndex, setPuzzleIndex ] = useState(0);
  let pixelArr = [];
  let swatchArr = [];
  const rgbValueArrays = palettes.rgbValueArrays;

  // Create function to update puzzle
  const renderNewPuzzle = (event) => {
    if (puzzleIndex === 3) {
        setPuzzleIndex(0)
    } else {
        setPuzzleIndex(puzzleIndex + 1)
```

```
    }
    let pixels = document.querySelectorAll('.pixel');
    pixels.forEach(pixel => {
      pixel.style.backgroundColor = '';
    });
    let resetBrushColor = document.getElementById("brush");
    resetBrushColor.style.backgroundColor = '';
  };

  const resetPuzzle = (event) => {
    let pixels = document.querySelectorAll('.pixel');
    pixels.forEach(pixel => {
      pixel.style.backgroundColor = '';
    });
    let resetBrushColor = document.getElementById("brush");
    resetBrushColor.style.backgroundColor = '';
  }

  // Render legend for the canvas (256 pixel elements)
  for (let i = 0; i < 256; i++) {
    pixelArr.push(
      <div
        className="pixel"
        onMouseMove = { handleMouseMove }
        onMouseDown= { handleMouseDown}
        onMouseUp={ handleMouseUp }
        onTouchStart = { handleTouchStart }
        onTouchMove = { handleTouchMove }
        onTouchEnd = { handleTouchEnd } >
          { puzzles[puzzleIndex].legend[i] }
      </div>);
  };

  // Populate legend for the palette (10 swatches)
  for (let i = 0; i < 10; i++) {
    let currId = "color" + i;
    swatchArr.push(<div  className="swatch"  id={  currId  }  onClick={  updateBrush
}>{i}</div>);
  };

  // Store an array of dynamically determined palette swatch font colors (white or black)
  let                                    swatchFontColorsArr                                    =
determineFontColor(rgbValueArrays[puzzles[puzzleIndex].paletteId]);

  // After page renders, load palette specified in puzzles array from palette array
  useEffect(() => {
    let paletteArr = document.querySelectorAll('.swatch');
    for (let i = 0; i < 10; i++) {
      paletteArr[i].style.backgroundColor                =                'rgb('                +
rgbValueArrays[puzzles[puzzleIndex].paletteId][i][0] +
                                    ', ' +  rgbValueArrays[puzzles[puzzleIndex].paletteId][i][1] +
                                    ', ' + rgbValueArrays[puzzles[puzzleIndex].paletteId][i][2] + ')';
      if (swatchFontColorsArr[i] === 0) {
        paletteArr[i].style.color = 'black';
```

```
      } else {
        paletteArr[i].style.color = 'white';
      }
    }
  });

  return (
    <div className="app" >
      <header>Color By Numbers</header>

      <div className="container">

        <div className="canvas-brush-container" >
          <div className="canvas">
            { pixelArr }
          </div>
          <div className="paintbrush-container" >
            <img src="paintbrush-solid.svg" alt='paintbrush' />
            <div id="brush" className="selected-color"></div>
          </div>
        </div>

          <div className="palette-container" >
            <img src="palette-solid.svg" alt='artists palette' />
            <div className="palette">
              { swatchArr }
            </div>
          </div>

          <div className="btn-container">
            <button                    id="new-picture-btn"                  className="new-picture-btn"
onClick={renderNewPuzzle}>New Picture</button>
            <button                   id="reset-picture-btn"                 className='reset-picture-btn'
onClick={resetPuzzle}>Reset Image</button>
          </div>

      </div>
    </div>
  );
}

export default App;
```

kid.css

```
* {
  margin: 0;
  padding: 0;
  font-family: 'Inter', sans-serif;
}

html, body, .app {
  height: 100%;
  width: 100%;
```

```
  font-size: 16px;
}

header {
  width: 100%;
  height: 3.7rem;
  border-bottom: 1px darkgray solid;
  box-shadow: 0px 0.5px 4px gray;
  color: rgb(131, 131, 131);
  display: flex;
  justify-content: center;
  align-items: center;
  font-size: 1.9rem;
}

.container {
  padding: 1.2rem;
  display: flex;
  flex-direction: column;
}

.canvas-brush-container {
  display: flex;
  flex-wrap: nowrap;
  justify-content: center;
}

.canvas {
  min-width: 31.6rem;
  max-width: 31.6rem;
  display: grid;
  grid-template-columns: repeat(16, 1fr);
  flex-wrap: wrap;
  box-shadow: 0px 0.5px 4px gray;
  margin-right: 1rem;
  padding: 0;
  background-color: rgb(169, 169, 169);
}

#brush {
  margin-top: 0.4rem;
}

.pixel {
  aspect-ratio: 1 / 1;
  border: solid 1px darkgray;
  background-color: white;
  color: rgb(150, 150, 150);
  display: flex;
  justify-content: center;
  align-items: center;
  font-size: 1.25rem;
  user-select: none;
  margin: 0px;
```

```
}

/* PALETTE */

.palette-container {
  display: flex;
  flex-direction: row;
  justify-content: center;
  align-items: center;
  margin-right: 4.4rem;
  margin-top: 1rem;
}

img {
  height: 32px;
  width: 45px;
}

.palette {
  display: flex;
  flex-wrap: nowrap;
  height: 40px;
}

.swatch {
  margin: 0.12rem;
  border: solid darkgray 1px;
  height: 40px;
  min-width: 40px;
  border-radius: 50%;
  color: white;
  font-size: 1.5rem;
  display: flex;
  justify-content: center;
  align-items: center;
  user-select: none;
  box-shadow: 0px 0.5px 4px gray;
}

.selected-color {
  height: 40px;
  min-width: 40px;
  border-radius: 50%;
  border: 1px solid darkgray;
  box-shadow: 0px 0.5px 4px gray;
}

.canvas:hover, .palette:hover {
  cursor: pointer;
}

/* New Picture Button */

.btn-container {
```

```
  text-align: center;
  margin-top: 1.2rem;
}

.new-picture-btn {
  background-color: white;
  color: rgb(131, 131, 131);
  padding: 1rem;
  border: solid 1px darkgray;
  border-radius: 10px;
  box-shadow: 0px 0.5px 4px gray;
  font-size: 1rem;
  font-weight: 600;
  margin-right: 10px;
}

.new-picture-btn:hover {
  background-color: rgb(231, 231, 231);
  cursor: pointer;
}

.new-picture-btn:active {
  box-shadow: none;
}

.reset-picture-btn {
  background-color: white;
  color: rgb(131, 131, 131);
  padding: 1rem;
  border: solid 1px darkgray;
  border-radius: 10px;
  box-shadow: 0px 0.5px 4px gray;
  font-size: 1rem;
  font-weight: 600;
  margin-right: 10px;
}

.reset-picture-btn:hover {
  background-color: rgb(231, 231, 231);
  cursor: pointer;
}

.reset-picture-btn:active {
  box-shadow: none;
}

/* Media Queries */

@media only screen and (max-width: 600px)

 {

}
```

event.js

```
let isColoring = false;
let currColor = '';

export function updateBrush(event) {
    let returnedStyleObj = getComputedStyle(event.currentTarget);
    let clickedColor = returnedStyleObj.backgroundColor;
    let brush = document.getElementById("brush");
    brush.style.backgroundColor = clickedColor;
    currColor = clickedColor;
};

export const handleMouseDown = event => {
    event.currentTarget.style.backgroundColor = currColor;
    isColoring = true;
};

export const handleMouseMove = event => {
    if (isColoring) {
       event.currentTarget.style.backgroundColor = currColor;
    };
};

export const handleMouseUp = event => {
    isColoring = false;
};

// Handle Touch Events

export const handleTouchStart = event => {
    event.preventDefault();
    event.currentTarget.style.backgroundColor = currColor;
    isColoring = true;
};

export const handleTouchMove = event => {
    event.preventDefault();
    if (isColoring) {
       event.currentTarget.style.backgroundColor = currColor;
    };
};

export const handleTouchEnd = event => {
    event.preventDefault();
    isColoring = false;
};
```

Canva.js

```
import React, { useState, useRef, useEffect } from "react";
import "./App.css";
```

```
const App = () => {
  const canvasRef = useRef(null);
  const [context, setContext] = useState(null);
  const [tool, setTool] = useState("pencil");
  const [color, setColor] = useState("#000000");
  const [fillColor, setFillColor] = useState("#ffffff");
  const [pencilWidth, setPencilWidth] = useState(2);
  const [isDrawing, setIsDrawing] = useState(false);
  const [text, setText] = useState("");
  const [startPoint, setStartPoint] = useState({ x: 0, y: 0 });
  const [endPoint, setEndPoint] = useState({ x: 0, y: 0 });
  const [isSaving, setIsSaving] = useState(false);
  const [isTextEditing, setIsTextEditing] = useState(false);
  const [textPosition, setTextPosition] = useState({ x: 0, y: 0 });

  useEffect(() => {
    const canvas = canvasRef.current;
    canvas.width = window.innerWidth * 0.8;
    canvas.height = window.innerHeight * 0.8;
    const ctx = canvas.getContext("2d");
    ctx.lineCap = "round";
    setContext(ctx);
  }, []);

  const startDrawing = ({ nativeEvent }) => {
    const { offsetX, offsetY } = nativeEvent;
    setStartPoint({ x: offsetX, y: offsetY });
    if (tool === "pencil" || tool === "eraser" || tool === "rectangle" || tool === "circle" || tool ===
"line") {
      setIsDrawing(true);
    } else if (tool === "text") {
      setTextPosition({ x: offsetX, y: offsetY });
      setIsTextEditing(true);
    }
  };

  const draw = ({ nativeEvent }) => {
    if (!isDrawing) return;
    const { offsetX, offsetY } = nativeEvent;
    switch (tool) {
      case "pencil":
        drawPencilLine(offsetX, offsetY);
        break;
      case "eraser":
        erase(offsetX, offsetY);
        break;
      case "rectangle":
        setEndPoint({ x: offsetX, y: offsetY });
        break;
      case "circle":
        setEndPoint({ x: offsetX, y: offsetY });
        break;
      case "line":
        setEndPoint({ x: offsetX, y: offsetY });
```

```
        break;
      default:
        break;
    }
  };

  const endDrawing = () => {
    setIsDrawing(false);
    if (tool === "rectangle") {
      drawRectangle();
    } else if (tool === "circle") {
      drawCircle();
    } else if (tool === "line") {
      drawLine();
    }
  };

  const drawPencilLine = (x, y) => {
    context.strokeStyle = color;
    context.lineWidth = pencilWidth;
    context.lineTo(x, y);
    context.stroke();
  };

  const erase = (x, y) => {
    context.clearRect(x - 5, y - 5, 10, 10);
  };

  const drawRectangle = () => {
    context.strokeStyle = color;
    context.strokeRect(startPoint.x, startPoint.y, endPoint.x - startPoint.x, endPoint.y -
startPoint.y);
  };

  const drawCircle = () => {
    const radius = Math.sqrt(Math.pow(endPoint.x - startPoint.x, 2) + Math.pow(endPoint.y -
startPoint.y, 2));
    context.beginPath();
    context.arc(startPoint.x, startPoint.y, radius, 0, 2 * Math.PI);
    context.stroke();
  };

  const drawLine = () => {
    context.strokeStyle = color;
    context.beginPath();
    context.moveTo(startPoint.x, startPoint.y);
    context.lineTo(endPoint.x, endPoint.y);
    context.stroke();
  };

  const saveImage = () => {
    setIsSaving(true);
    const link = document.createElement("a");
    link.download = "canvas_image.png";
```

```
    link.href = canvasRef.current.toDataURL();
    link.click();
    setIsSaving(false);
  };

  const handleFill = () => {
    context.fillStyle = fillColor;
    context.fillRect(0, 0, canvasRef.current.width, canvasRef.current.height);
  };

  const handleText = () => {
    setIsTextEditing(true);
  };

  const handleTextChange = (e) => {
    setText(e.target.value);
  };

  const handleTextSubmit = (e) => {
    e.preventDefault();
    context.font = "25px Arial";
    context.fillStyle = color;
    context.fillText(text, textPosition.x, textPosition.y + 16); // Adjusting y position for proper
text display
    setText("");
    setIsTextEditing(false);
  };

  const clearCanvas = () => {
    context.clearRect(0, 0, canvasRef.current.width, canvasRef.current.height);
  };

  return (
    <div className="App">
      <nav className="navbar">
        <button onClick={() => setTool("pencil")}>Pencil</button>
        <button onClick={() => setTool("eraser")}>Eraser</button>
        <input type="color" value={color} onChange={(e) => setColor(e.target.value)} />
        <input
          type="range"
          min="1"
          max="20"
          value={pencilWidth}
          onChange={(e) => setPencilWidth(e.target.value)}
        />
        <button onClick={() => setTool("rectangle")}>Rectangle</button>
        <button onClick={() => setTool("circle")}>Circle</button>
        <button onClick={() => setTool("line")}>Line</button>
        <button onClick={handleText}>Text</button>
        {isTextEditing && (
          <form onSubmit={handleTextSubmit}>
            <input type="text" value={text} onChange={handleTextChange} />
            <button type="submit">Submit</button>
          </form>
```

```
        )}
        <button onClick={handleFill}>Fill</button>
        <input type="color" value={fillColor} onChange={(e) => setFillColor(e.target.value)} />
        <button onClick={saveImage} disabled={isSaving}>
          Save
        </button>
        <button onClick={clearCanvas}>Clear</button>
      </nav>
      <canvas
        ref={canvasRef}
        onMouseDown={startDrawing}
        onMouseMove={draw}
        onMouseUp={endDrawing}
        onMouseLeave={endDrawing}
      />
    </div>
  );
};

export default App
```

canva.css

```
.App {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    height: 100vh;
  }

  canvas {
    border: 2px solid #000;
  }

  .navbar {
    display: flex;
    justify-content: center;
    align-items: center;
    margin-bottom: 10px;
  }

  button {
    margin: 0 5px;
  }

  input[type="color"] {
    margin: 0 5px;
  }

  input[type="range"] {
    width: 100px;
    margin: 0 5px;
```

}

OUTPUT:

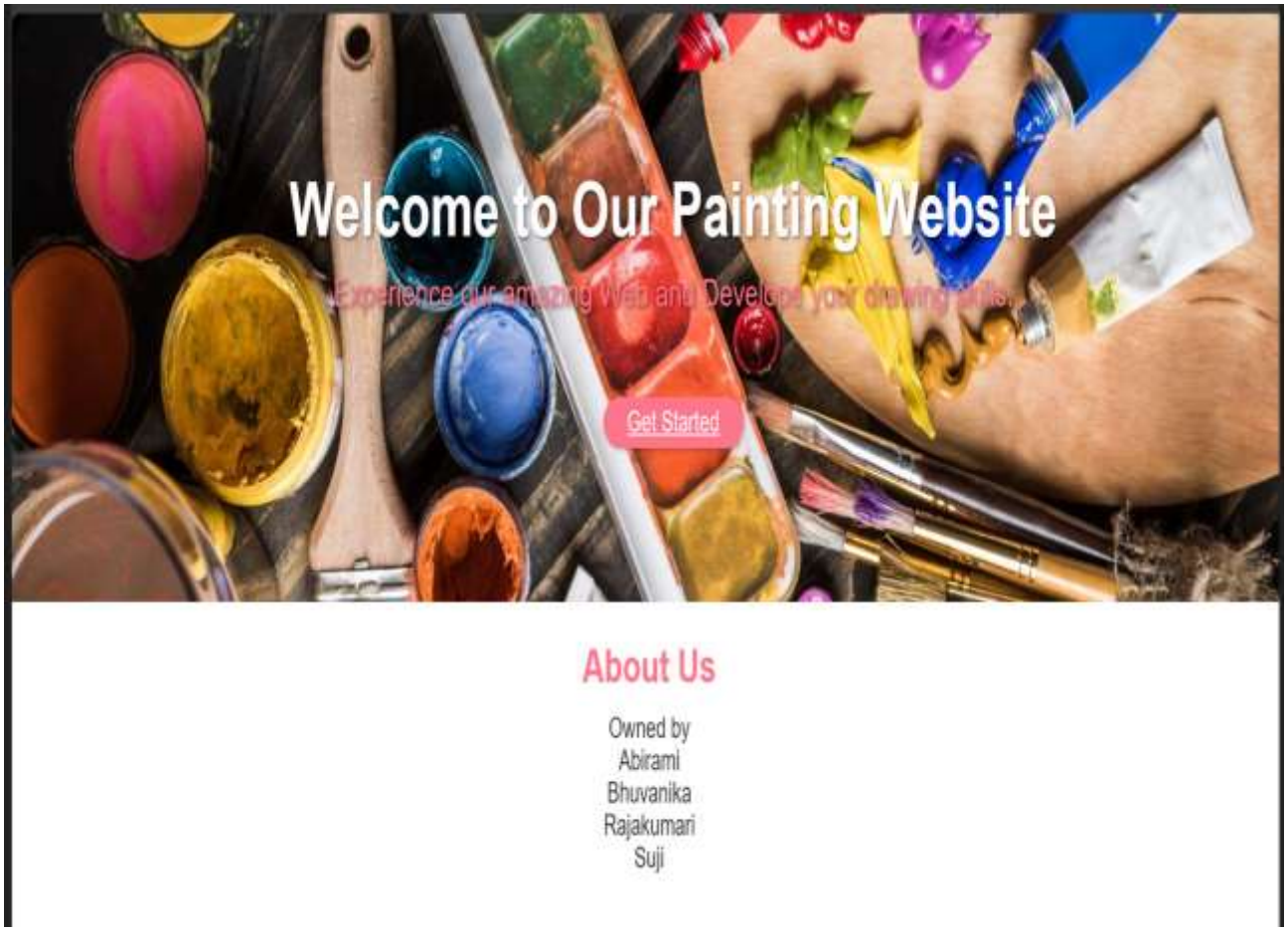


Fig-1 Home Page-1

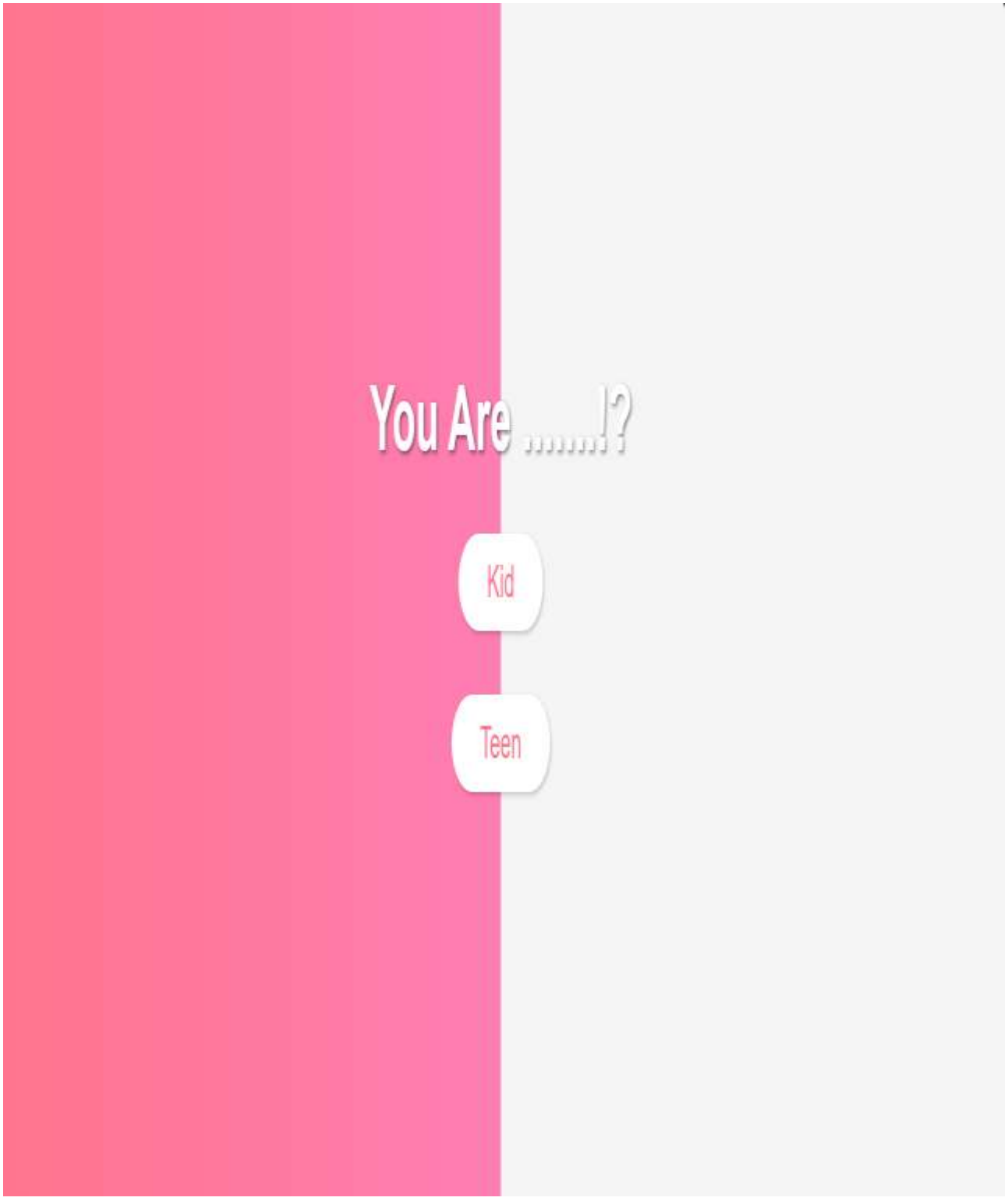


Fig-2 Home Page-2

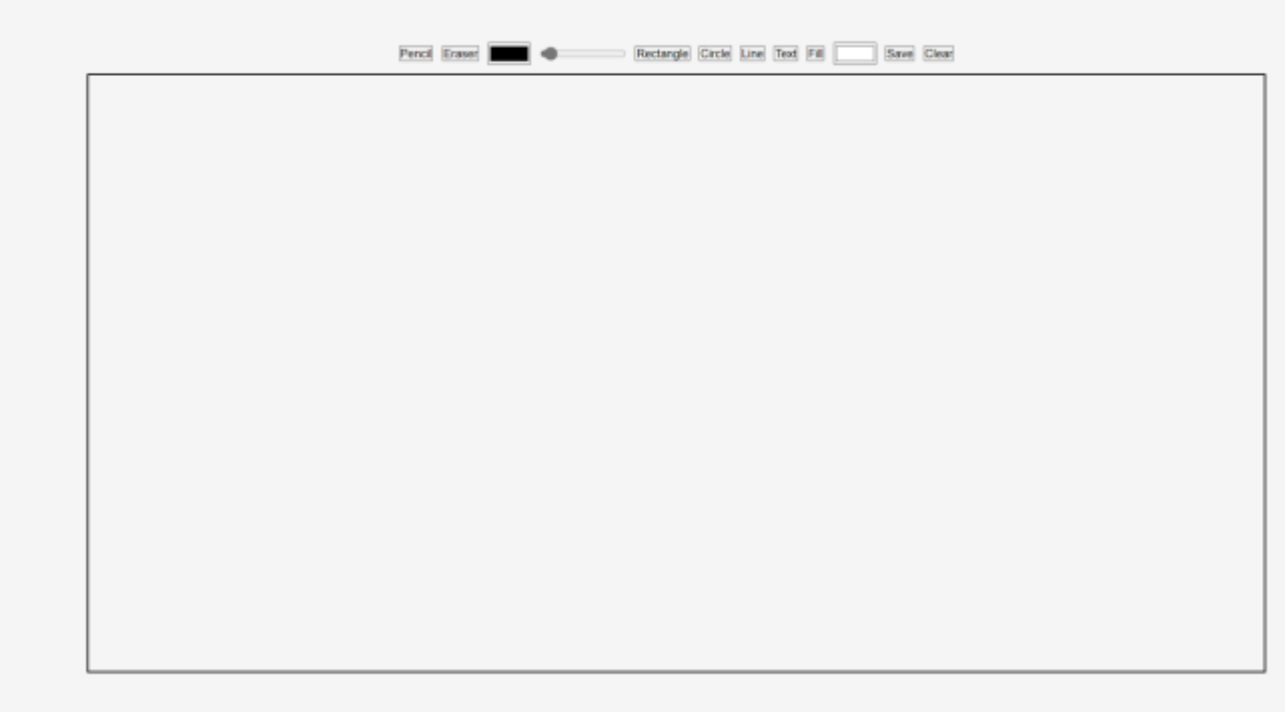


Fig-3 Canvas

Fig-4 Working of pen

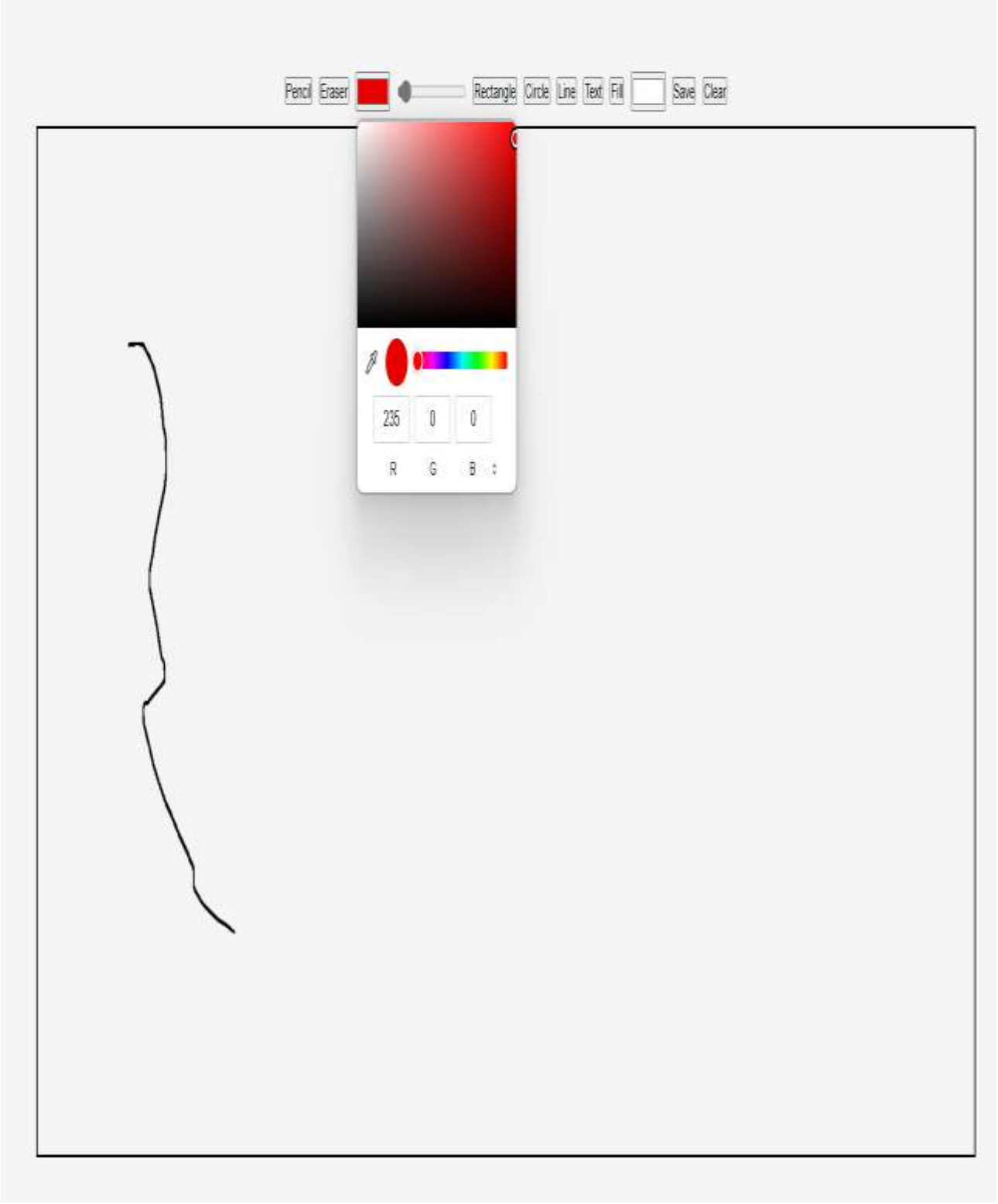


Fig-5 Color Palette

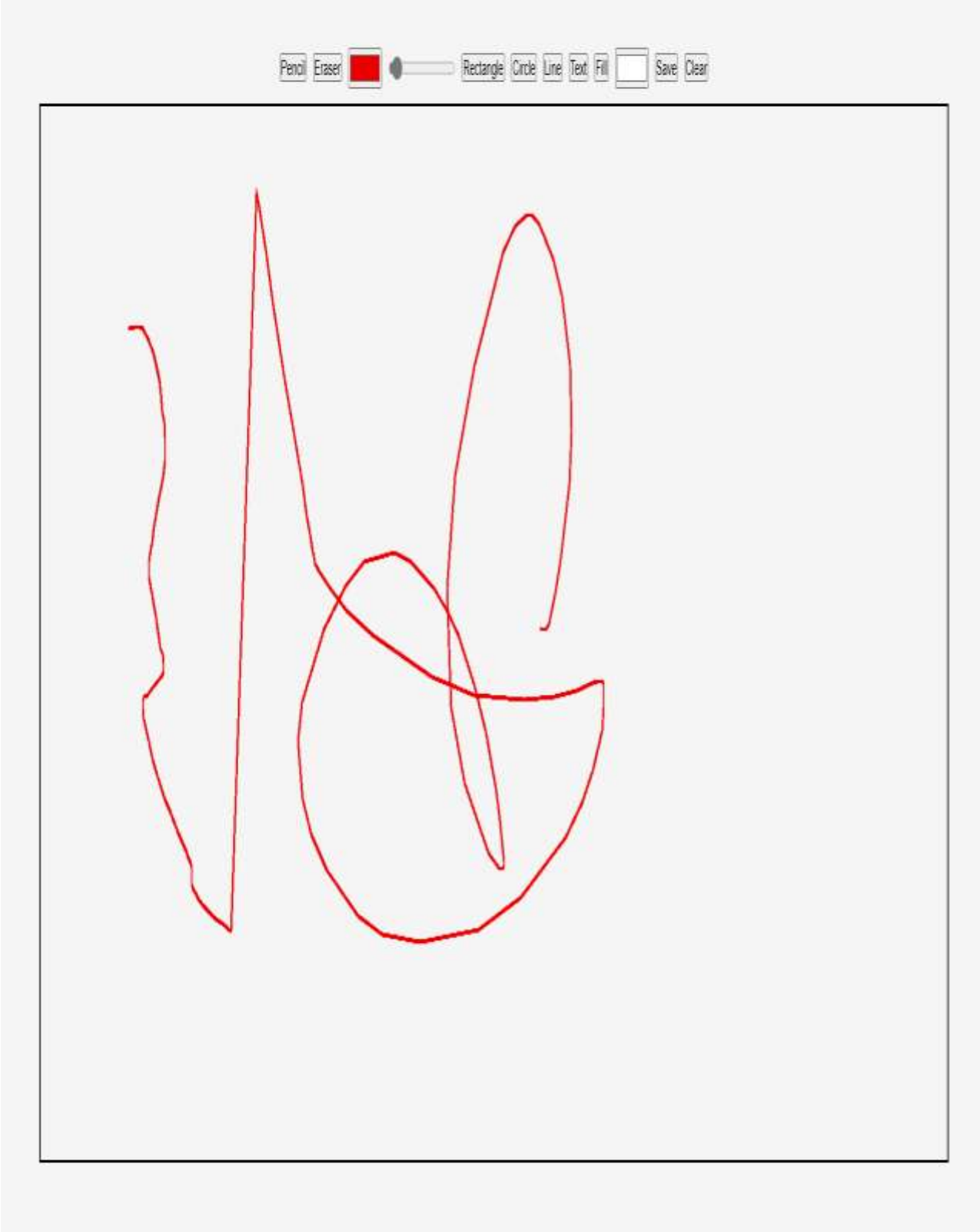


Fig-6 Working of Color Palette

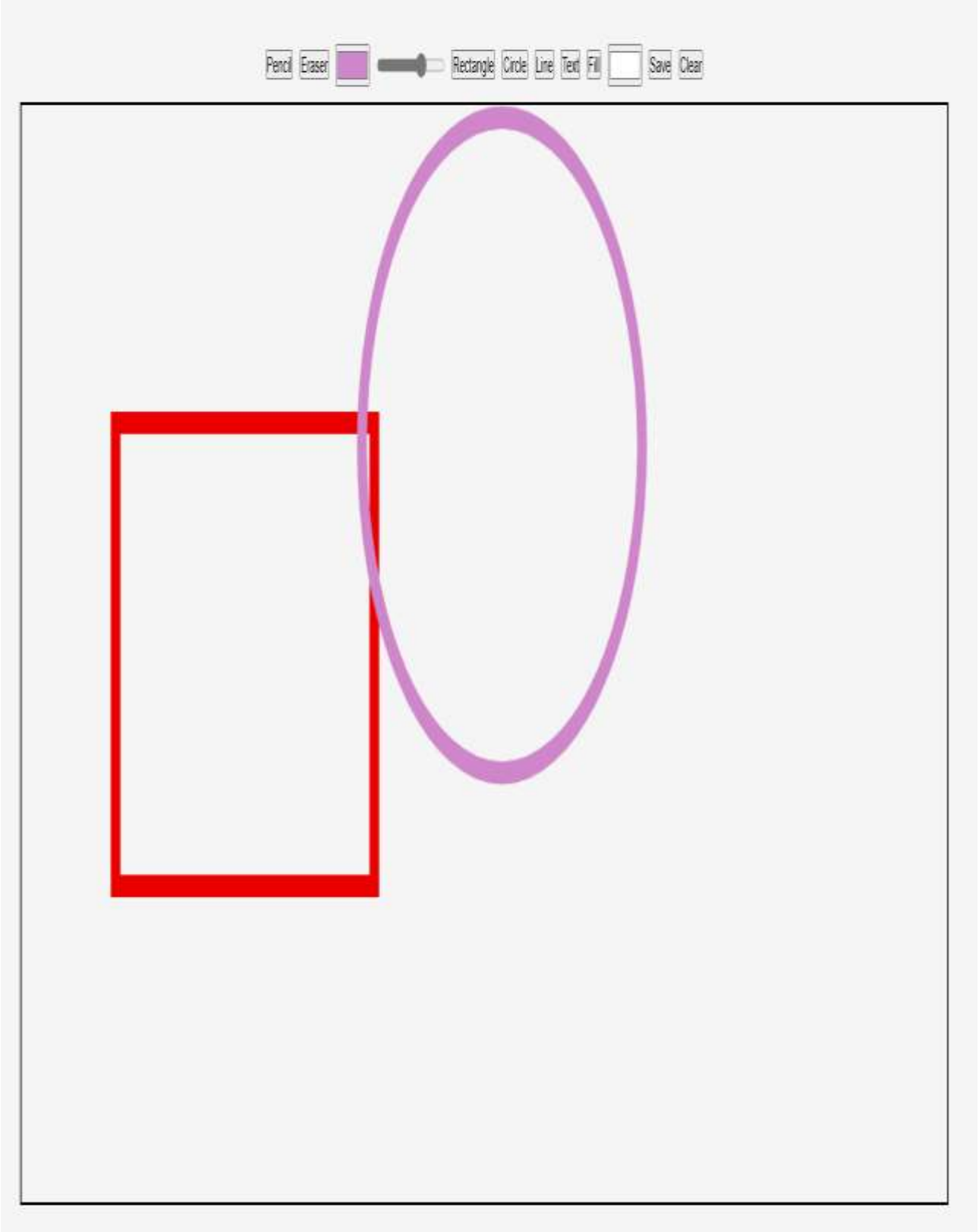


Fig-7 Working of shapes-1

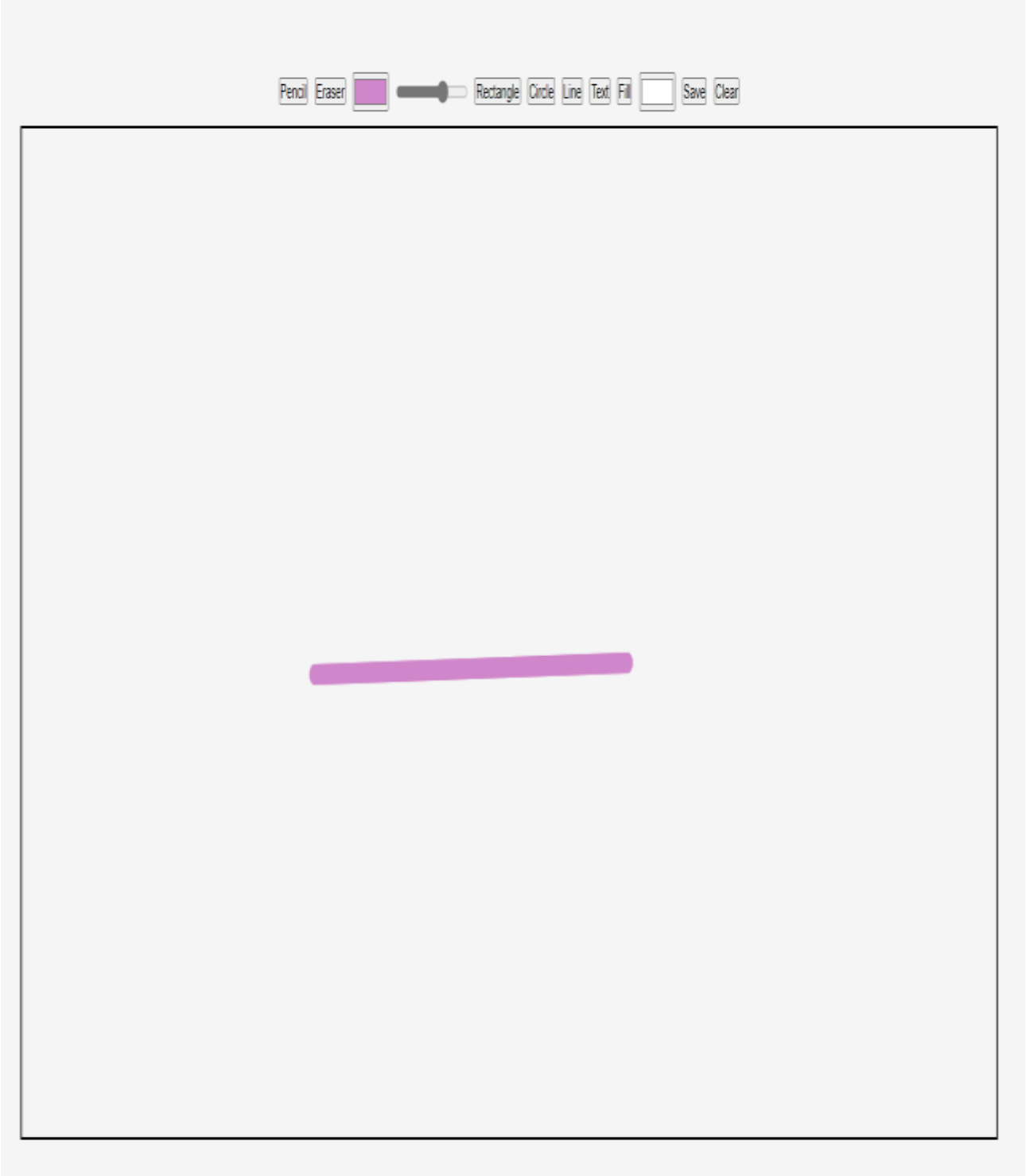


Fig-8 Working of shapes-2

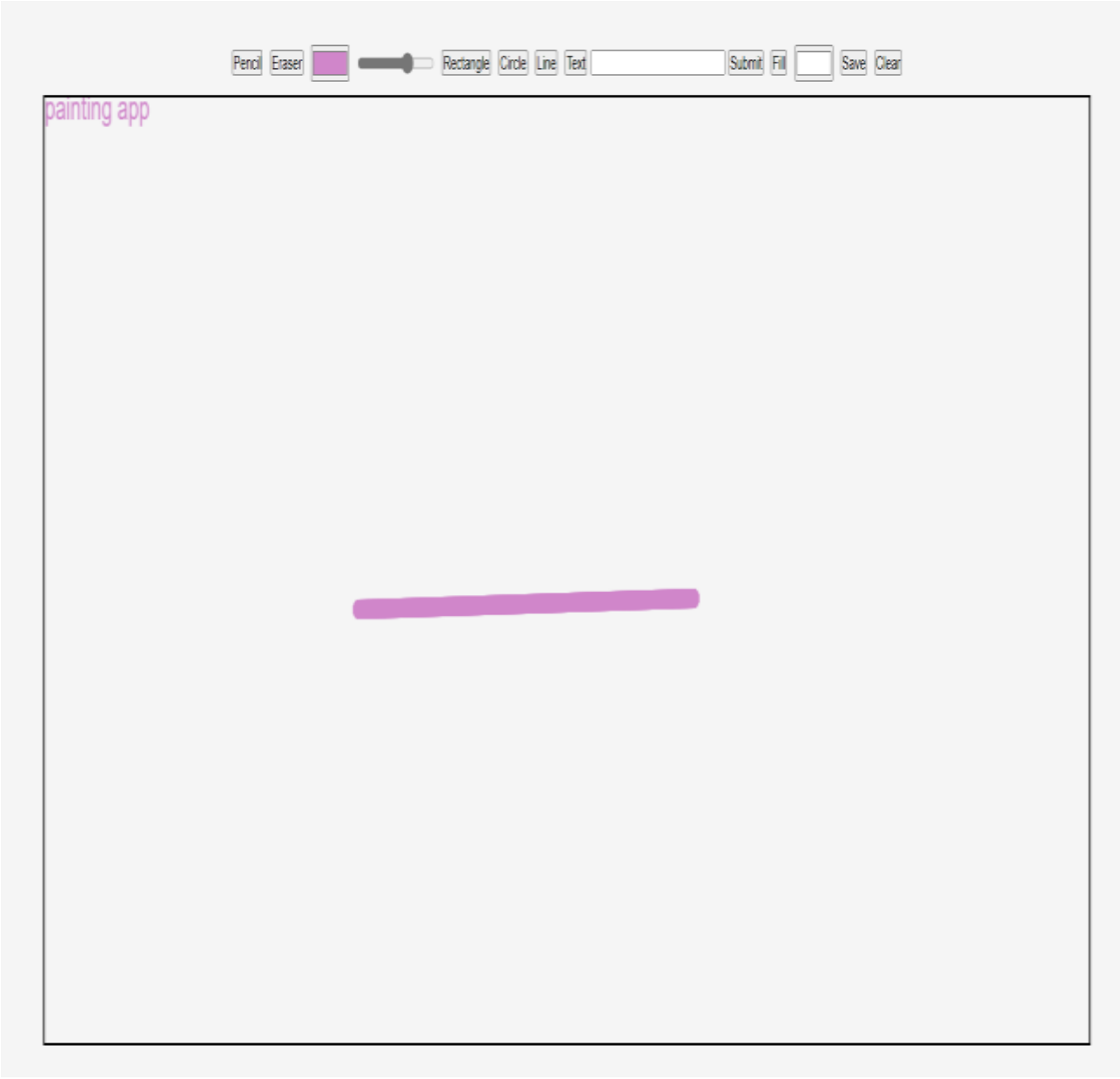


Fig-9 Working of Text

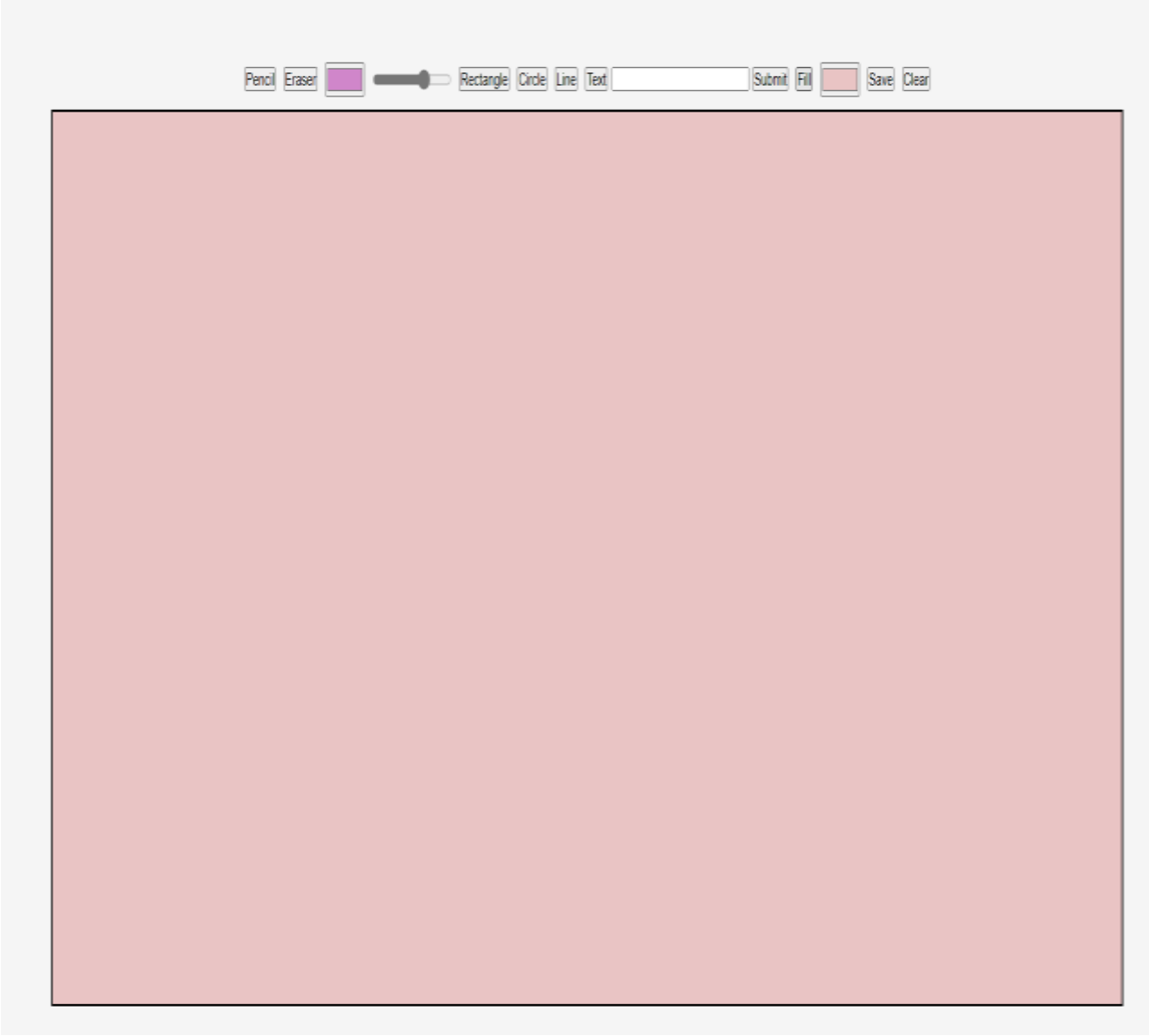


Fig-10 Working of fill

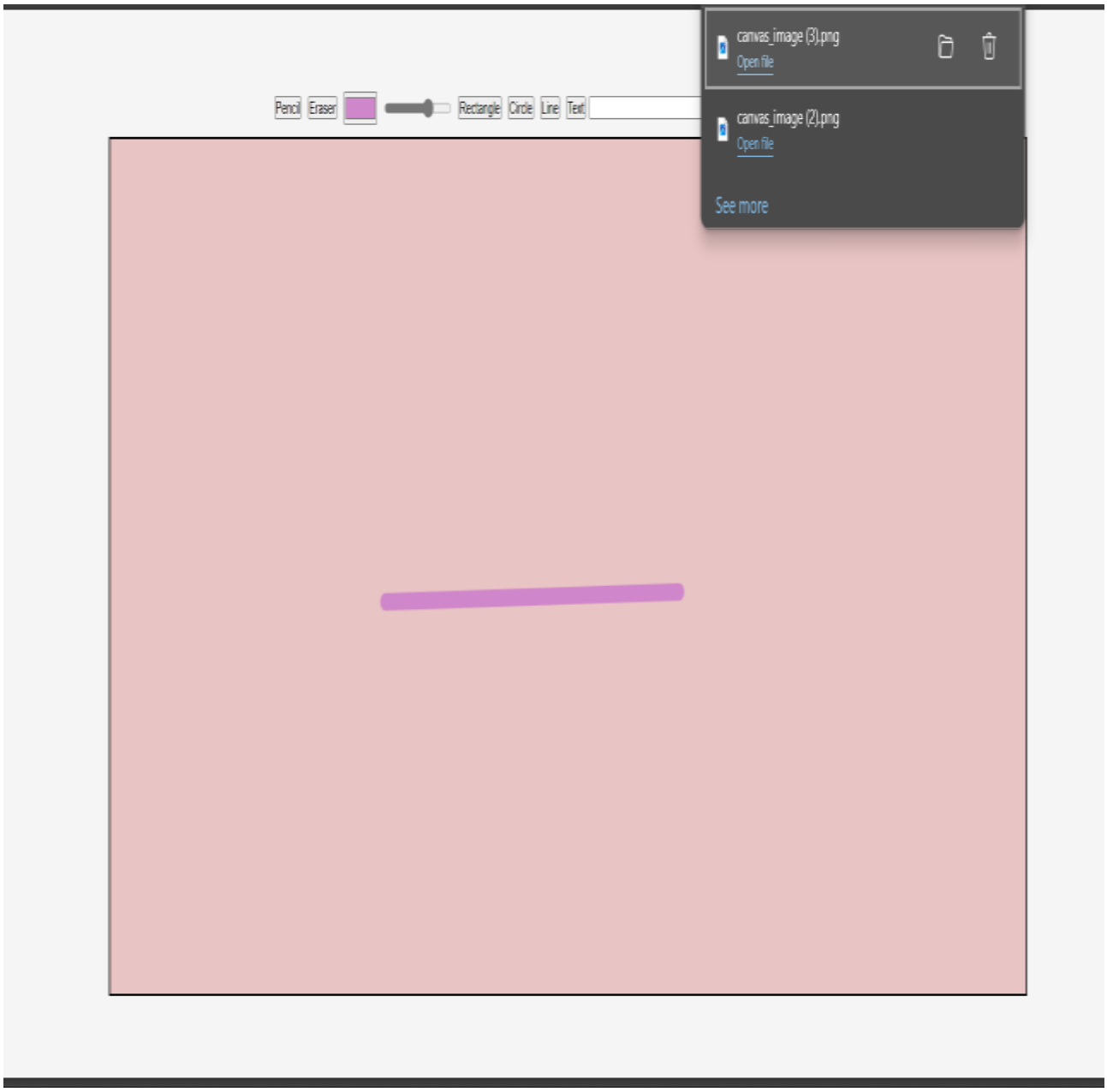


Fig-11 Working of save

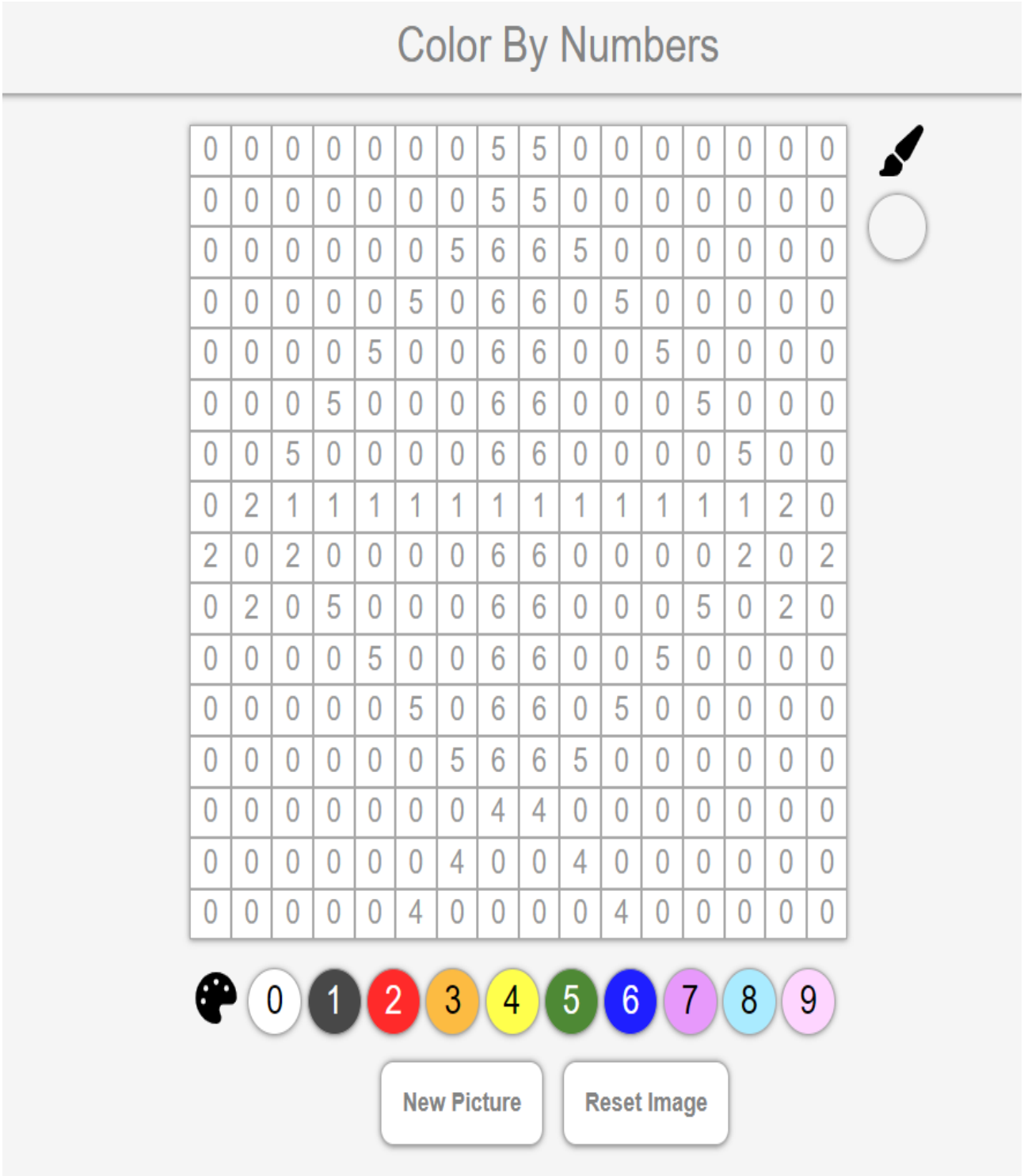


Fig-12 Grid Colouring

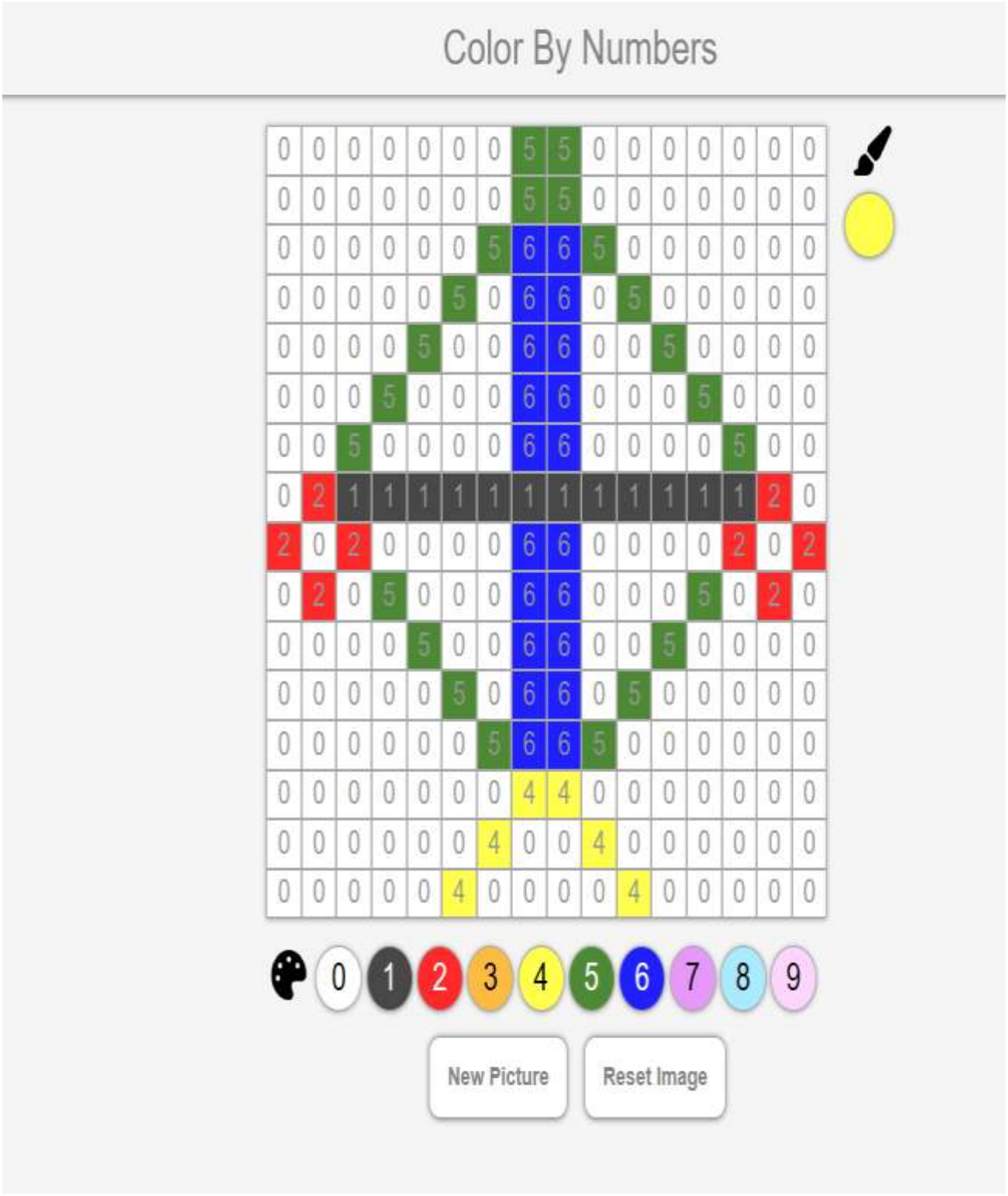


Fig-13 Color by number-1

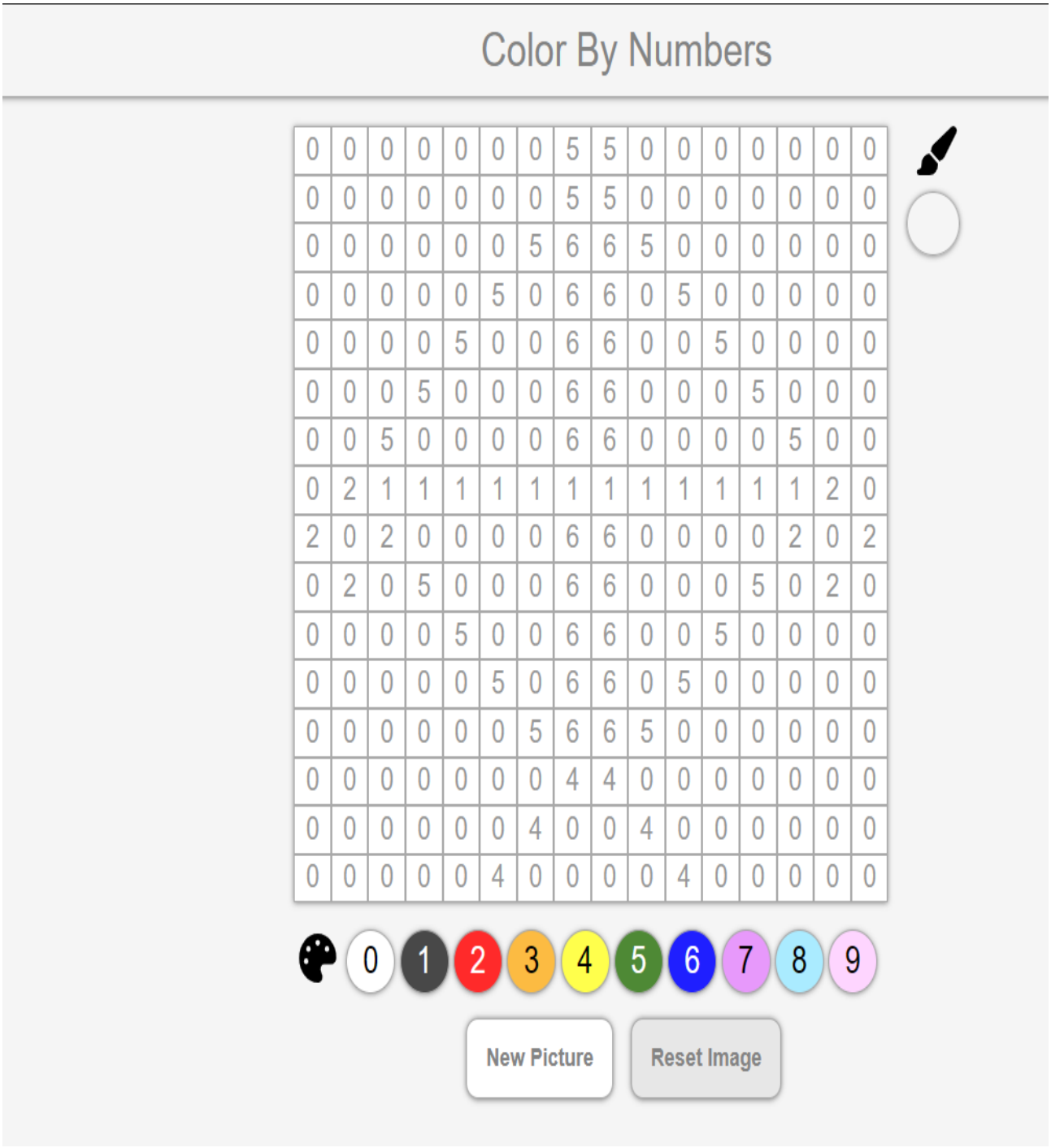


Fig-14 working of Reset image

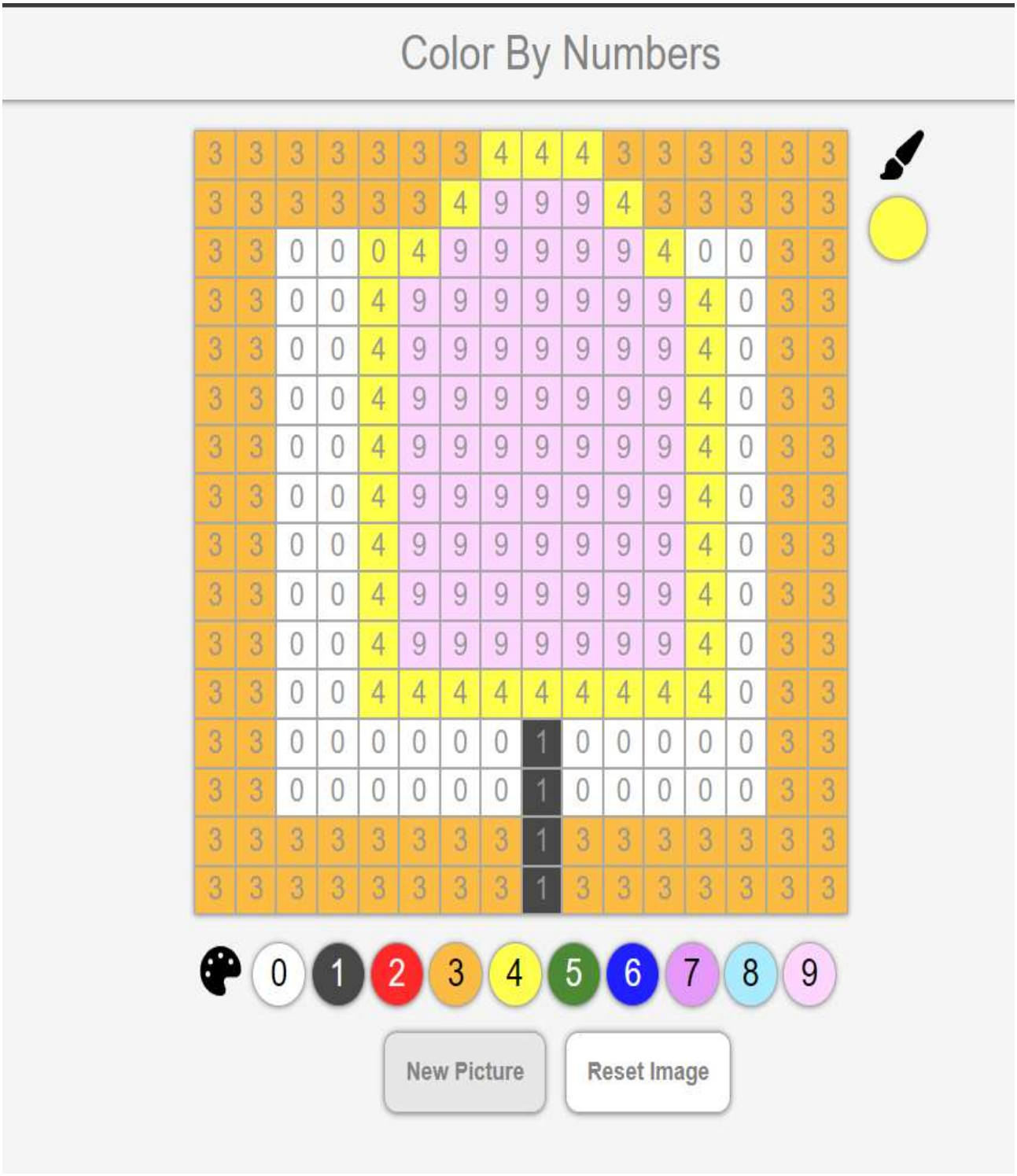


Fig-15 Working of new picture