RV College of Engineering®

Mysore Road, RV Vidyaniketan Post,Bengaluru - 560059, Karnataka, India

FULL STACK APPLICATION DEVELOPMENT

MCA261IA

SEE PROJECT BASED LABORATORY

MS PAINT

submitted by

Awais Ahmed	**1RV24MC020**
Ajay Kumar Kurni	**1RV24MC009**

under the guidance of

Name of the Guide

Prof. Prashanth K (A Section)
or
Dr. Savitha R (B Section)

Assistant Professor
Department of Master of Computer Applications
RV College of Engineering®

Department of Master of Computer Applications

2025-2026

RV College of Engineering®

Mysore Road, RV Vidyaniketan Post,Bengaluru - 560059, Karnataka, India

CERTIFICATE

Certified that the project entitled **'MS PAINT'**

on **Full Stack Application Development – MCA261IA** has been carried out by **AWAIS AHMED (1RV23MC020), AJAY KUMAR KURNI (1RV23MC009)** has successfully completed the project for the final SEE Lab Examination, incorporating all concepts of the course conducted by the Department of MCA, RV College of Engineering, Bengaluru.

Internal Guide

Prof. Prashanth K (A Section)

or

Dr. Savitha R (B Section)

Assistant Professor

Department of MCA

RV College of Engineering®

Head of the Department

Dr. Jasmine K S

Associate Professor & Director

Department of MCA,

RV College of Engineering®

External Viva Examination

Name of Examiners	**Signature with Date**
1.	
2.	

Table of Contents

1. Introduction

This project is a web-based recreation of the classic Microsoft Paint application.
The goal was to design and implement a lightweight drawing tool that runs entirely in the browser while preserving the familiar feel of legacy desktop paint software.

The application allows users to draw, edit, and save digital artwork using simple tools such as pencils, brushes, shapes, and color palettes — closely mirroring the original MS Paint interaction model.

This project demonstrates core concepts in **frontend engineering, UI state management, and HTML5 Canvas graphics programming**.

1.1 Overview of Problem Domain

Traditional image-editing software is often:

- too complex,
- overloaded with features, and
- requires installation.

However, many users only need **basic 2D drawing tools** for:

- sketching ideas
- annotating images
- teaching & demonstrations
- casual digital art

The problem addressed here is:

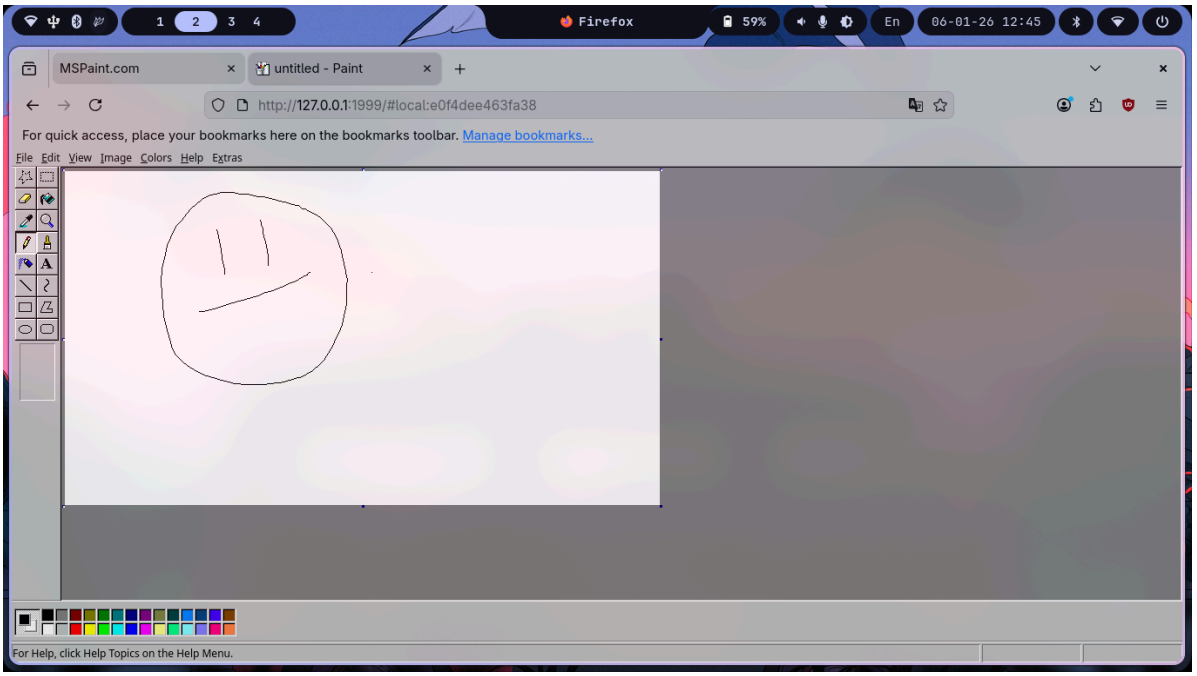
> *How can we provide an accessible, browser-based drawing tool that is fast, intuitive, and nostalgic — without heavy dependencies?*

3. Key Features and Analysis

The application recreates the essential toolkit of classic MS Paint, while keeping the interface intuitive and lightweight. Each tool is implemented using custom logic on top of the HTML5 Canvas API, ensuring smooth interaction and predictable behavior.

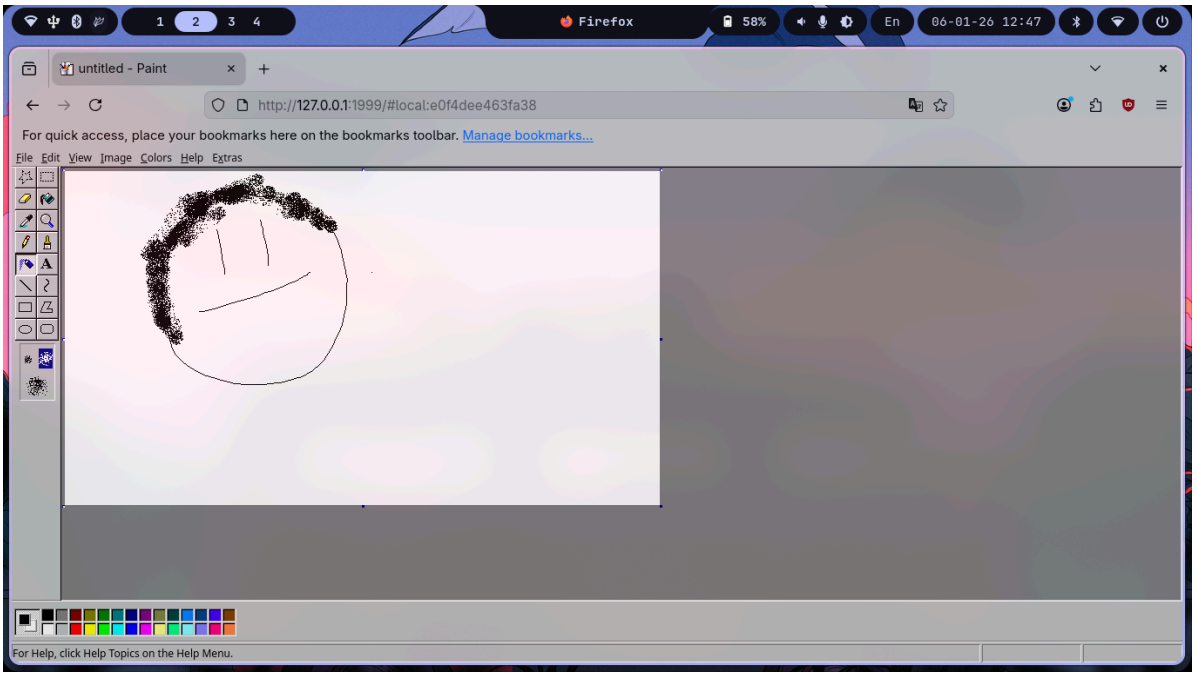
Drawing Tools (Pencil, Brush, Airbrush)

Pencil Tool: The pencil tool draws precise, single-pixel lines, making it suitable for sketching outlines and pixel-style artwork.



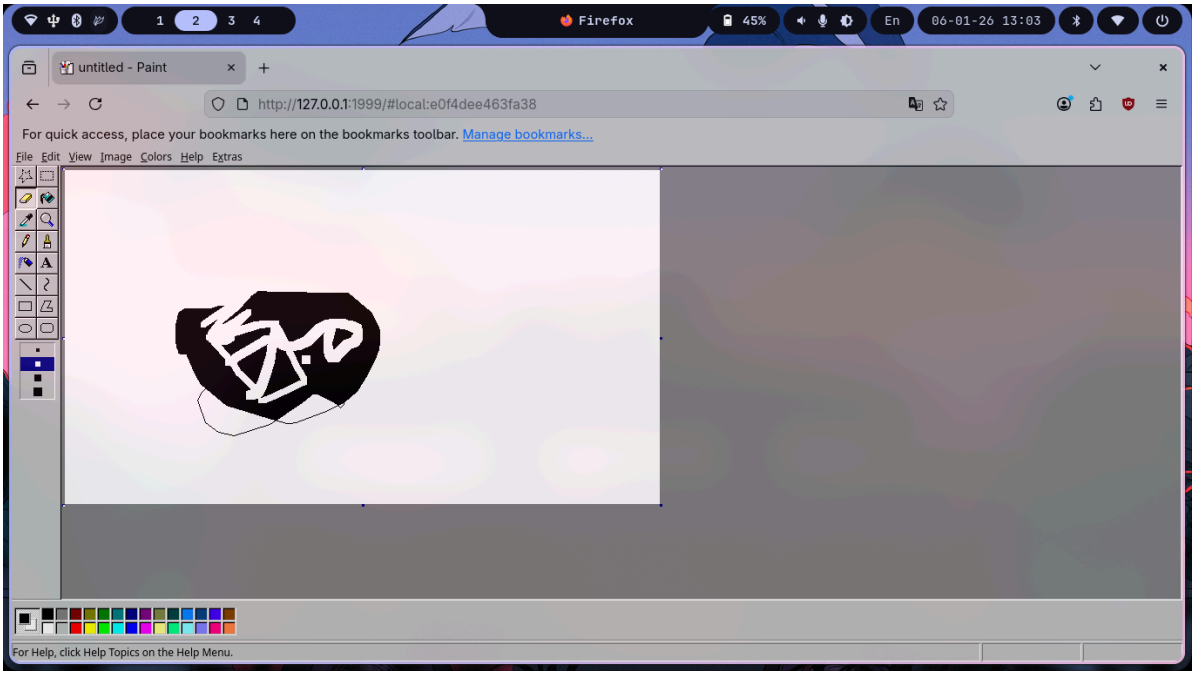
Brush Tool: The brush tool allows thicker, softer strokes, giving users more expressive and natural drawing control.

AirBrush: The airbrush simulates spray-paint behavior by dispersing random dots within a circular region, useful for shading and texture effects.



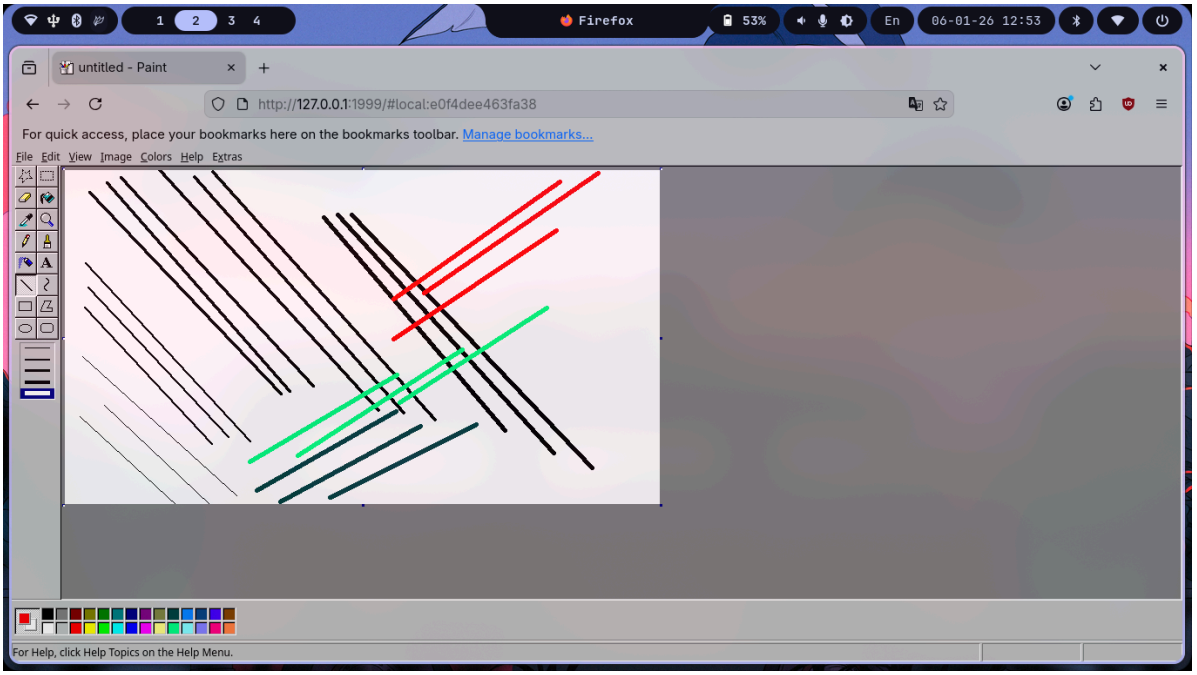
Eraser Tool

The eraser clears selected pixels from the canvas by replacing them with the background color. It behaves like a brush in reverse, allowing users to correct mistakes or refine shapes with accuracy.

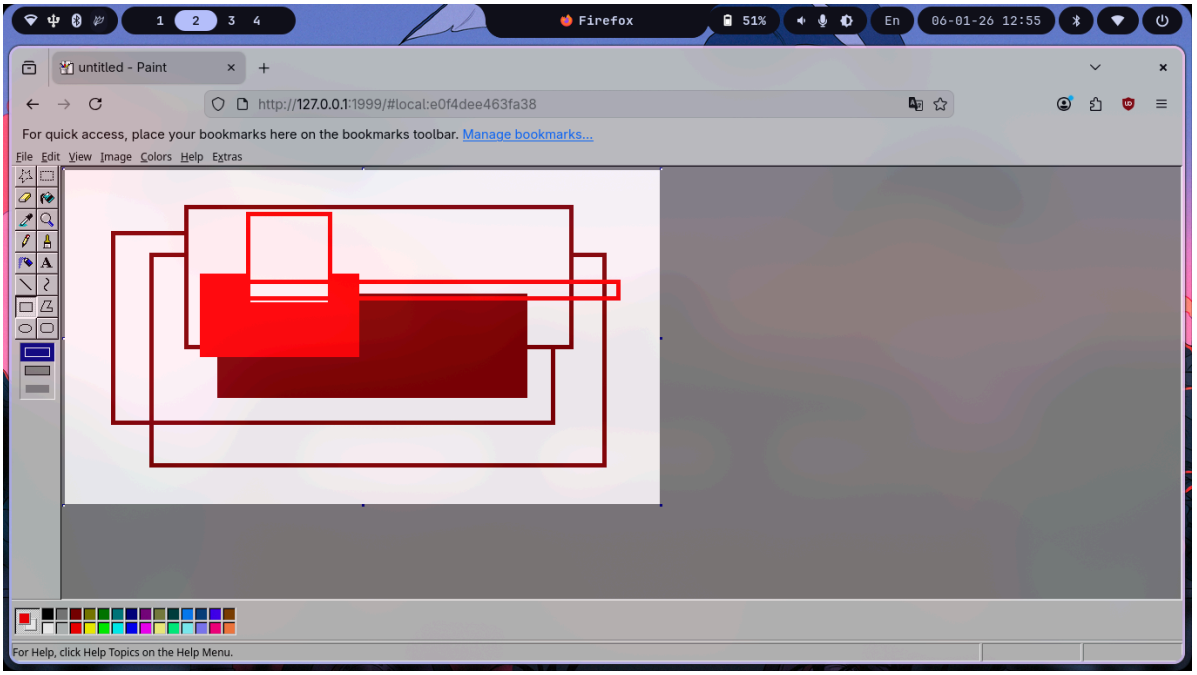


Shape Tools (Line, Rectangle, Ellipse, Polygon)

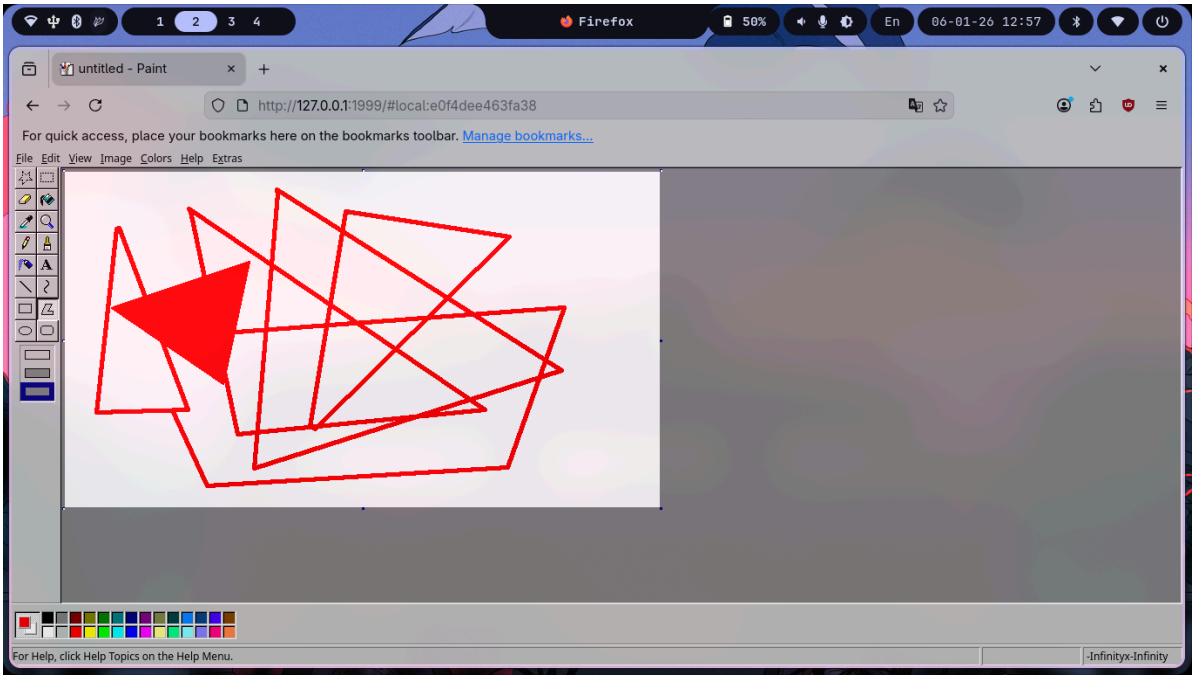
Line Tool: The line tool draws straight segments defined by two points, useful for diagrams and precise edges.



Rectangle Tool: The rectangle and ellipse tools support both outlined and filled shapes, enabling structured drawing and quick layout creation.

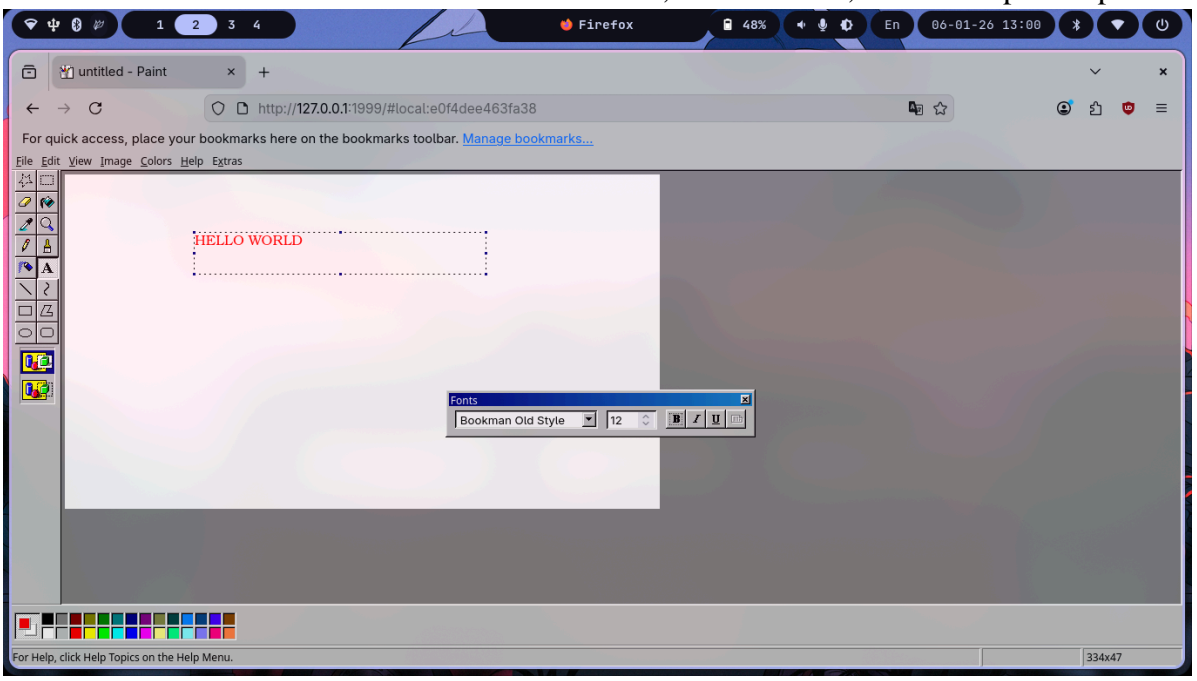


PolygonTool: The polygon tool lets users create multi-sided shapes by clicking successive points, offering flexibility beyond basic geometry.



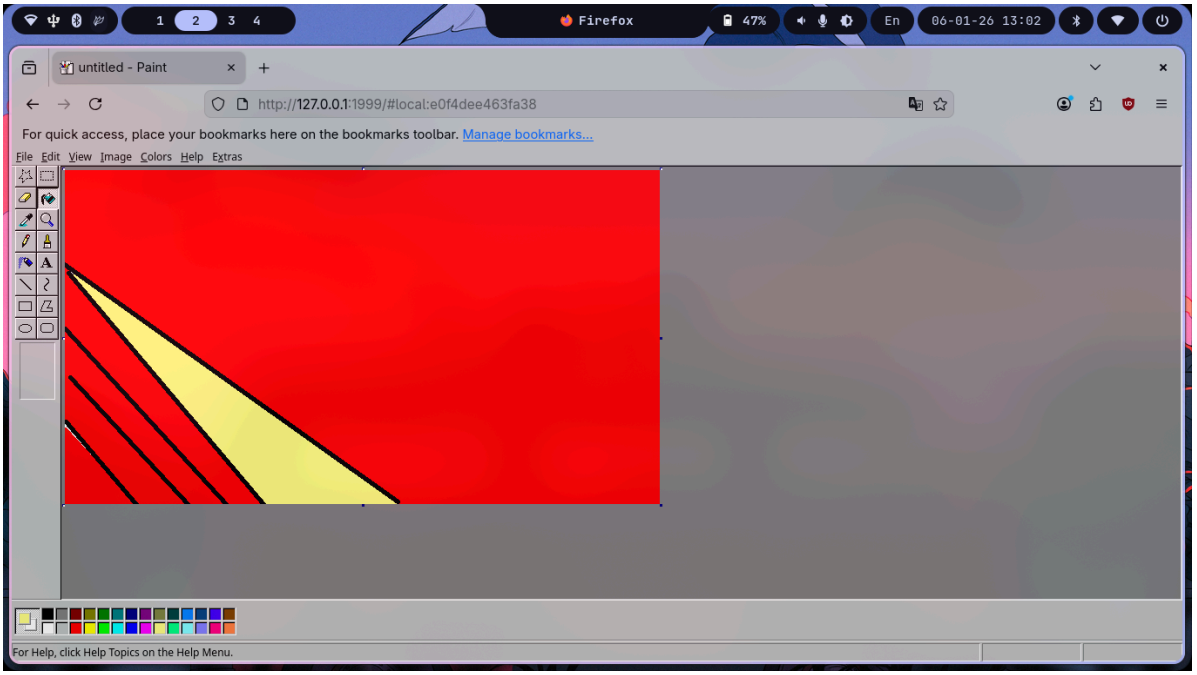
Text Tool

The text tool allows users to type directly on the canvas, positioning text elements wherever needed. This is useful for labels, annotations, and simple captions.



Fill Bucket (Flood-Fill)

The fill tool uses a flood-fill algorithm to recolor connected regions of similar color. It is especially effective for quickly coloring enclosed shapes and background areas.



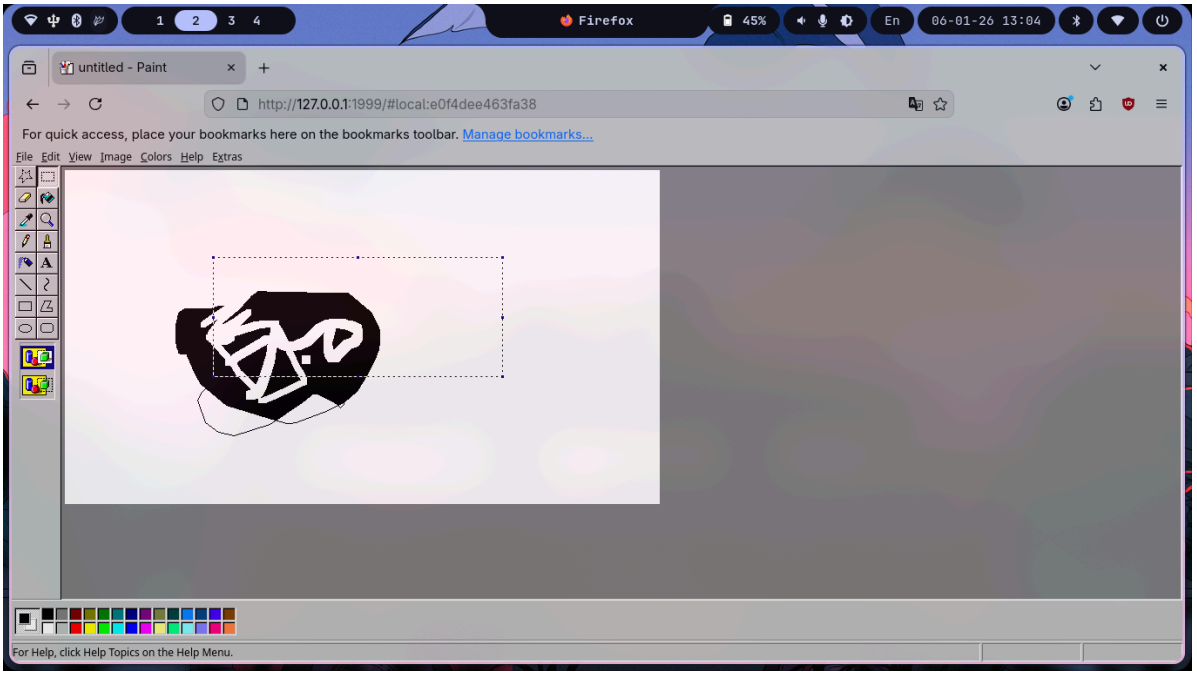
Eyedropper (Color Picker)

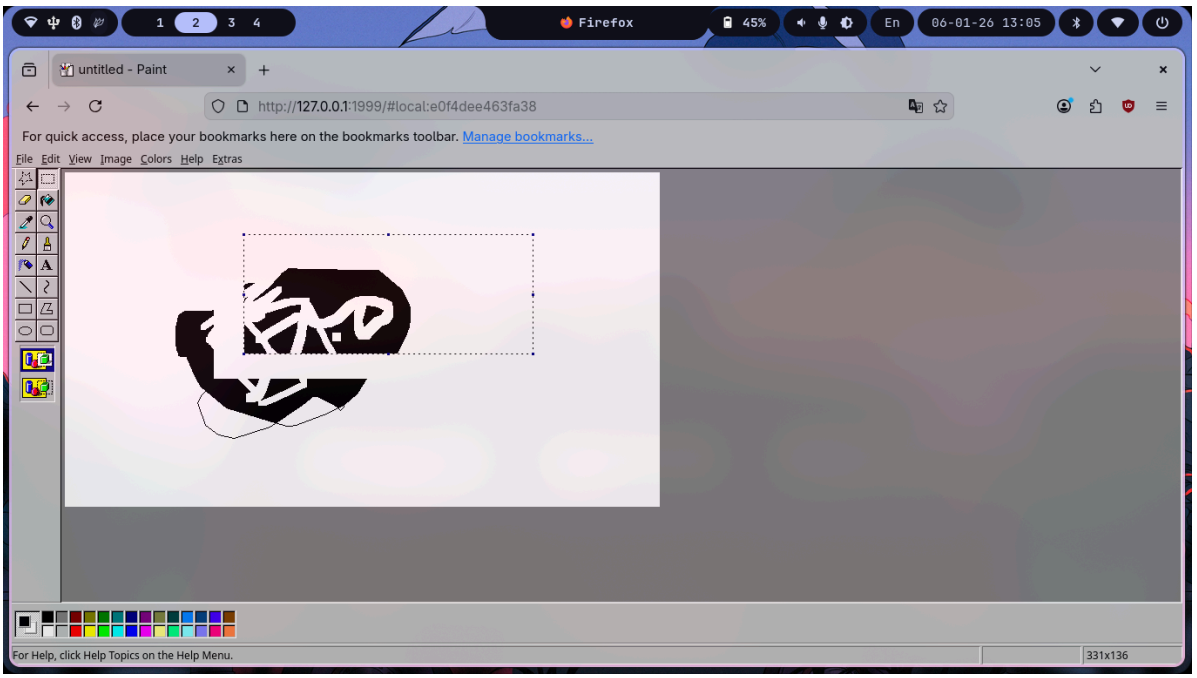
The eyedropper tool samples colors directly from the canvas, instantly setting them as the active drawing color. This makes it easy to match and reuse colors without guessing.

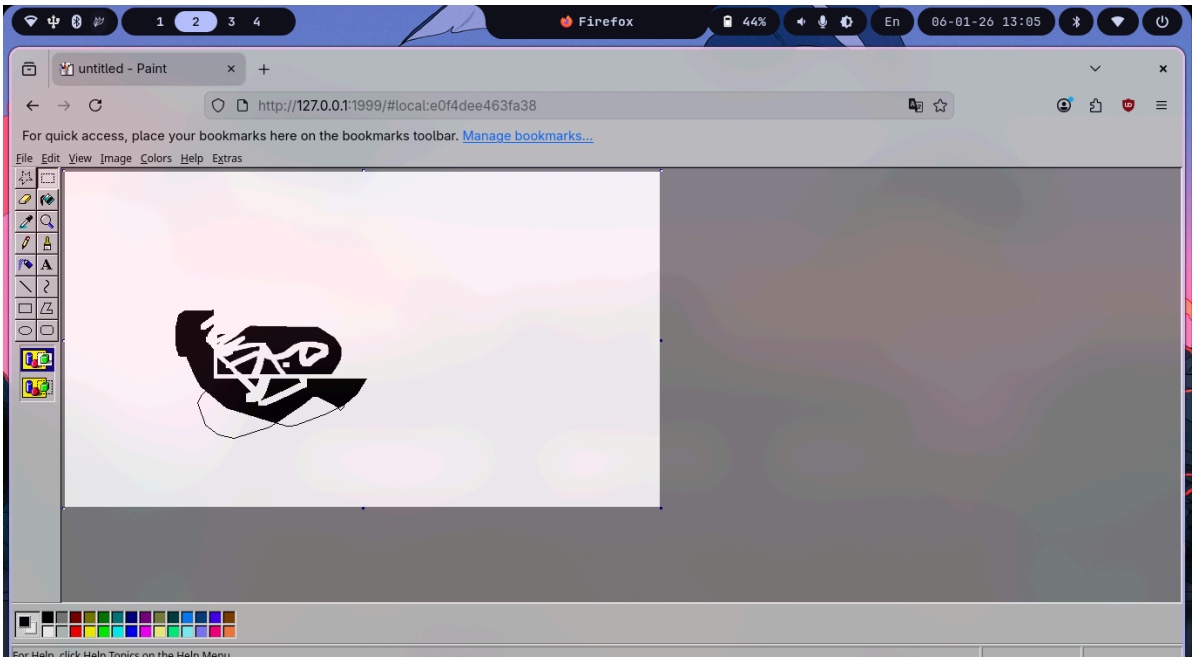
Selection Tools

Selection tools let you isolate specific parts of the canvas so you can move, edit, copy, or delete them without affecting the rest of the drawing. Instead of repainting something from scratch, you simply "grab" the exact area you want and work with it directly — like cutting out stickers from a page.

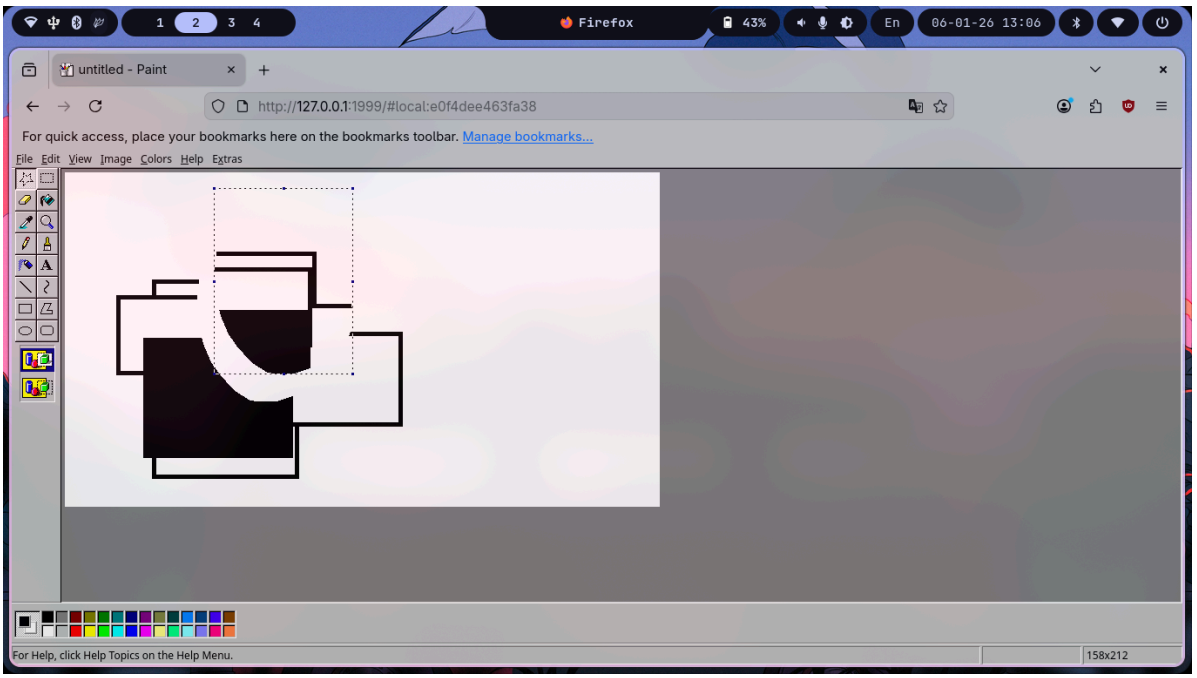
Rectangular Selection: Creates a clean, box-shaped selection. Perfect for structured shapes, UI elements, or anything roughly rectangular.

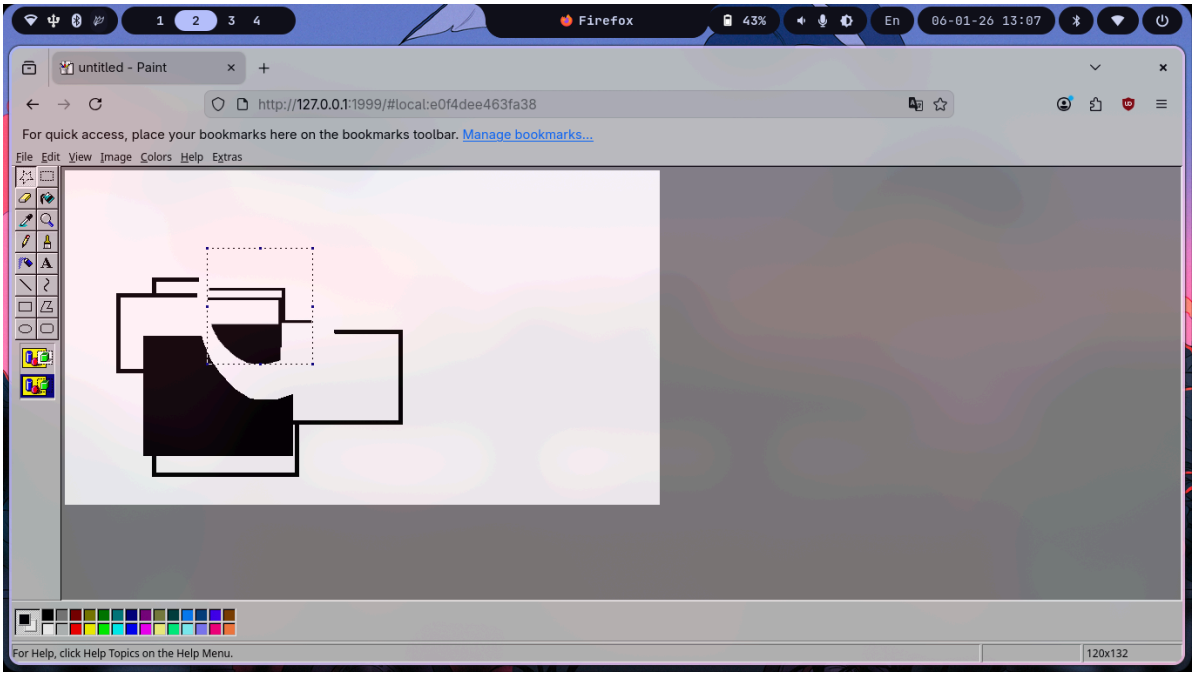






Freeform (Lasso) Selection: Lets you draw a custom outline around irregular shapes. It's great when precision matters — think tracing around a character, cloud, or anything not perfectly square.





Once an area is selected, you can: move it around like a draggable object, copy/paste it to duplicate artwork, resize or rotate it, or clear it entirely.

Some tools even support transparent selection, meaning only the colored pixels move — not the background behind them. Subtle, but powerful.

Undo / Redo System

The undo/redo mechanism stores snapshots of important canvas states. Users can safely experiment, revert mistakes, and restore previous work without losing progress.

Zoom and Magnifier

Zoom functionality enlarges specific regions of the canvas, assisting with detailed edits. The magnifier preserves drawing accuracy while navigating zoomed views.

Canvas Resize

Canvas resizing lets users adjust the workspace dimensions dynamically. Existing artwork is preserved as much as possible during resizing operations.

Open Image

Users can import external images onto the canvas, enabling tracing, editing, or annotation of pre-existing graphics.

Save as PNG

The application converts the current canvas into a downloadable PNG file. This allows artwork to be stored locally or shared across platforms.

Keyboard Shortcuts

Common commands such as **Ctrl+Z (Undo)** and other shortcuts improve workflow efficiency and mimic familiar desktop-software behavior.

Touch and Mouse Compatibility

All tools support both mouse and touch input, ensuring usability across laptops, tablets, and touchscreen devices without additional installations.

Technology Stack (MERN)

This project is designed using the MERN architecture, which combines powerful client-side rendering with efficient server-side data management. MERN ensures scalability, high performance, and smooth integration between all components.

1. MongoDB — Database Layer

MongoDB stores application data in a flexible JSON-like format.
It can store:

- saved drawings
- user profiles
- project history
- app settings

Its schema-less design makes it ideal for evolving features without complex migrations.

2. Express.js — Backend Framework

Express runs on top of Node.js and provides lightweight APIs for communication between the frontend and database.
It is responsible for:

- handling HTTP requests
- managing routes (save, load, delete images)
- authentication endpoints
- validation and security middleware

Express helps keep the backend clear, modular, and easy to scale.

3. React — Frontend Framework

React builds the user interface as reusable components.
It controls:

- the drawing tools
- canvas interactions
- menus, palettes, and layouts
- real-time state updates (colors, brush size, tools, undo/redo)

React ensures fast rendering and smooth interaction, even during complex drawing operations.

4. Node.js — Runtime Environment

Node.js powers the server and executes all backend logic.
It enables:

- JavaScript running on the server
- real-time operations
- efficient handling of multiple users
- communication with MongoDB

Node's event-driven model keeps performance strong and responsive.

5. Additional Tools

- **Vite / Webpack** – fast development builds and optimized production output
- **NPM** – dependency and script management
- **Git & GitHub** – version control, collaboration, and project hosting
- **CSS** – layout styling and UI consistency

Requirements & Specifications

This section defines what the application must do (functional requirements) and how it should behave (non-functional requirements). These requirements guided the design and development process.

---

1. Functional Requirements

The system must provide the following capabilities:

Drawing & Editing

- Users should be able to draw on the canvas using tools such as pencil, brush, and eraser.
- Users should be able to create shapes (line, rectangle, ellipse, polygon).
- Users should be able to insert text on the canvas.
- Users should be able to fill closed areas with color using the bucket tool.
- Users should be able to pick any color from the canvas using the eyedropper.

Canvas Management

- The system should allow resizing of the canvas.
- The system should support zooming in and out.
- The system should display cursor coordinates and canvas information.

File & Storage

- Users should be able to open images from their device.
- Users should be able to save artwork as a PNG file.
- Users should be able to save drawings to the database.
- Users should be able to load previously saved drawings from the database.

Undo / Redo

- The system should allow undo and redo actions for supported operations.

Compatibility

- The application should work with both mouse and touch input devices.

---

2. Non-Functional Requirements

These describe **quality expectations** of the system.

Performance

- Drawing actions should respond in real time without noticeable delay.
- Undo and redo operations should complete quickly.

Usability

- The interface should remain simple and intuitive, similar to classic MS Paint.
- Icons and menus should clearly represent their functions.

Reliability

- Saved drawings should not be lost during normal operation.
- Backend API requests should handle errors gracefully.

Scalability

- The backend should support multiple users saving and loading artwork.
- The database should handle growth in stored drawings.

Security

- User data and drawings must only be accessible to authenticated owners.
- Passwords must be stored securely in encrypted form.

Portability

- The application should run smoothly in modern web browsers across platforms.

---

3. System Specifications

Software Requirements

- Web Browser (Chrome, Firefox, Edge, etc.)
- Node.js and NPM
- MongoDB database
- Operating System: Windows / Linux / macOS

Hardware Requirements

- Minimum 4 GB RAM
- Dual-core processor
- Internet connectivity for database access and authentication

Implementation Details

The system is implemented using the MERN architecture, separating responsibilities between the frontend, backend, and database.

1. Frontend (React + Canvas)

The frontend is responsible for all user interactions.

- React components manage UI sections such as the toolbar, color palette, menus, and status bar.
- The HTML5 Canvas API is used for drawing operations.
- Each drawing tool triggers custom event handlers for **mousedown, mousemove, and mouseup.**

State variables are used to track:

- active tool
- brush size
- selected colors
- undo/redo stack
- zoom level
- current canvas image data

Undo and redo are implemented using a snapshot mechanism, where the canvas state is periodically stored and restored when needed.

---

2. Backend (Node.js + Express)

The backend exposes REST APIs that handle:

- user registration and login
- saving drawings to the database
- retrieving previously saved drawings
- deleting saved artwork

JSON Web Tokens (JWT) are used to authenticate requests, ensuring secure access control.

3. Database (MongoDB)

MongoDB stores application data using collections such as:

- **Drawings** – image data, title, timestamp, user reference

The flexible schema allows future expansion (tags, sharing, comments, etc.) without structural changes.

---

4. Integration Layer

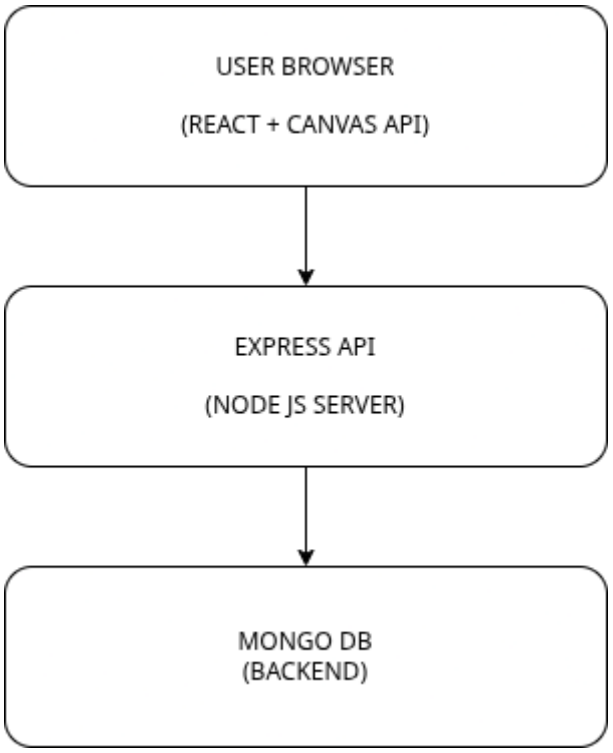
The frontend communicates with the backend using asynchronous HTTP requests (`fetch` / Axios).
Responses are handled gracefully to show messages such as:

- save success
- load failure
- authentication errors

This separation ensures the UI remains responsive while server operations occur in the background.

Architecture Diagram

Figure: System Architecture Overview



Demonstration

The application was tested through interactive drawing sessions that replicate real MS Paint workflows. Users can select tools, draw freely on the canvas, undo mistakes, change colors, and export their artwork as an image file.

The interface reacts instantly to mouse input, making the drawing experience feel natural and responsive — even on low-end systems.

Demonstration Link or Description

DEMO_VID.mp4

The demo showcases:

- Opening the web app in a browser
- Selecting the pencil and drawing freehand
- Using shapes to create structured drawings
- Changing colors using the color picker
- Selecting and moving parts of the drawing
- Saving the final artwork as an image

This walk-through highlights how the application mirrors the workflow of classic MS Paint while staying lightweight and web-based.

3. Key Features and Analysis

The application recreates the essential toolkit of classic MS Paint, while keeping the interface intuitive and lightweight. Each tool is implemented using custom logic on top of the HTML5 Canvas API, ensuring smooth interaction and predictable behavior.

Drawing Tools (Pencil, Brush, Airbrush)

Pencil Tool:

Draws sharp, single-pixel lines. Perfect for outlines, pixel art, or small details where precision matters.

Brush Tool:

Produces thicker, smoother strokes. It mimics real brush behavior, making it ideal for shading, coloring, and softer illustrations.

Airbrush Tool:

Simulates spray-paint by scattering tiny dots. This tool is useful for textures, gradients, and soft color blending.

Together, these tools cover everything from rough sketches to finished artwork.

Color Tools (Palette & Eyedropper)

Color Palette:

Users can pick from predefined colors or customize new ones. This ensures consistency in the artwork without constantly guessing shades.

Eyedropper (Color Picker):

The eyedropper tool samples colors directly from the canvas, instantly setting them as the active drawing color. This makes it easy to match and reuse colors without guessing — especially when working on edits or layered drawings.

Selection Tools (Rectangular & Freeform)

Selection tools allow users to isolate parts of their drawing and manipulate them independently.

Rectangular Selection:

Creates a box around the area you want to move or edit. Useful for structured objects, icons, or UI-like shapes.

Freeform (Lasso) Selection:

Lets the user draw a custom outline around irregular shapes. This is perfect for cutting out curved or organic drawings.

After selecting an area, users can:

- Move it around the canvas
- Copy and duplicate objects
- Delete mistakes
- Combine pieces creatively

This turns the drawing canvas into something more dynamic — not just "paint once," but edit, remix, rearrange.

Conclusion

This project shows that a capable, user-friendly drawing application can be created using only standard web technologies — without relying on heavy frameworks or specialized graphics software. By deliberately focusing on simplicity, familiar interactions, and smooth performance, the application delivers a nostalgic MS-Paint-style experience while embracing the flexibility of the modern web.

Throughout development, the emphasis was on making the tool feel natural: pick a brush, draw instantly, change colors effortlessly, select and move objects, and save artwork with a single click. These features come together to form a lightweight but meaningful creative environment. Instead of overwhelming users with complex options, the application proves that well-designed basics are often enough to support real creativity.

Beyond functionality, the project demonstrates several important ideas:

- **HTML5 Canvas is powerful enough** to support real-time drawing, interactions, and editing.
- **Web-based tools can meaningfully replace desktop utilities**, removing installation barriers and increasing accessibility.
- **Thoughtful UI design matters as much as code** — a clear interface can make simple features feel professional and enjoyable.

More importantly, the work highlights the deeper learning behind the implementation: understanding event handling, graphics rendering, state management, and user-experience tradeoffs. These lessons extend far beyond this single app and apply to a wide range of interactive web projects.

In the end, the project is not just a clone of MS Paint. It is a proof-of-concept that creative, responsive, and practical software can be built directly in the browser — open, portable, and available to anyone with an internet connection. And yeah — that's the kind of future the web was built for.