**EXPERIMENT NO – 02**

**Aim :** Write an application that draws basic painting on the screen.

**Theory :**

Painting in Android Studio involves creating custom drawing and painting functionalities within your Android application. This can include drawing shapes, lines, text, and applying various effects like colors, gradients, and transparency. Here's the theory behind implementing painting in Android Studio:

**Canvas and Paint:**

**Canvas:** In Android, a Canvas is an object that provides the drawing surface for your application. You can draw onto a Canvas using various drawing methods.

**Paint:** Paint is an object that holds the style and color information about how to draw geometries, text, and bitmaps. It defines how to draw shapes, colors, and styles.

**Custom Views:**

Painting in Android is often implemented within custom views. A custom view allows you to create unique user interface elements with custom drawing logic.

You can create a custom view by subclassing existing views like View or SurfaceView and overriding their onDraw(Canvas canvas) method.

**Drawing Shapes and Lines:**

Use methods like drawRect(), drawCircle(), drawLine(), etc., provided by the Canvas object to draw basic shapes and lines.

You can customize the appearance of shapes by setting properties in the Paint object, such as stroke width, color, and style.

**Handling Touch Events:**

To enable users to draw on the screen, you need to handle touch events like onTouchEvent(MotionEvent event) in your custom view.

Capture touch events (e.g., ACTION\_DOWN, ACTION\_MOVE, ACTION\_UP) to track user interactions with the screen.

Use the touch input coordinates to draw paths or shapes on the Canvas as the user interacts with the screen.

**Implementing Undo/Redo Functionality:**

Allow users to undo or redo their drawing actions. Maintain a stack of drawing commands or paths.

When the user performs an undo action, remove the last drawing command from the stack and redraw the canvas.

For redo, restore the removed drawing command from the undo stack and redraw the canvas.

**Saving and Sharing Drawings:**

Implement functionality to save the user's drawing as an image file (e.g., PNG or JPEG) locally on the device.

Provide options for users to share their drawings via other applications like messaging apps or social media platforms.

**Optimizing Performance:**

Drawing complex graphics or animations can impact performance, especially on older devices.

Optimize performance by minimizing unnecessary drawing operations, caching reusable graphic elements, and using hardware acceleration where possible.

**Testing and Debugging:**

Thoroughly test your painting functionality on different screen sizes, resolutions, and Android versions to ensure compatibility.

Use debugging tools provided by Android Studio to identify and fix any issues with drawing logic or touch event handling.

By understanding these theoretical concepts, you can effectively implement painting functionality in your Android application using Android Studio. This allows you to create custom drawing and painting applications or integrate painting features into existing applications for various purposes like note-taking, sketching, or creative expression.

**GITHUB LINK : https://github.com/kartikParmar12/Mobile\_Computing**

**OUTPUT :**

****

**CONCLUSION**

Thus we have Implemented the application that draws basic painting on the screen.