O.C Developer

DRAWING PAD

FOR ARTS STUDENTS

Group 4

Aquino, John Wayne
Balanay, Randall Ace
Bucod, Ancollete

GITHUB LINK:

To access the source code or file

Click Here!

https://github.com/Randallace05/Drawing-Pad-for-arts-students.git

Summary of the app

In a drawing pad mobile application, several essential functions enhance the user experience and creativity. The undo and redo functions allow users to easily correct mistakes and experiment with different ideas by reversing or reapplying the most recent actions. The clear canvas feature provides a quick way to start fresh by wiping the entire drawing area clean, making it easy to begin a new project or reset the workspace. To fine-tune their artwork, users can utilize the brush slider, which offers precise control over the size and opacity of the brush, allowing for both detailed and bold strokes. Additionally, the color picker is a vital tool that enables users to select and customize colors, whether through a preset palette, a full-color spectrum, or even an eyedropper feature to pick colors directly from the canvas. Together, these functions make the drawing pad intuitive and versatile, catering to a wide range of artistic needs.

Function

Redo: This function allows users to reapply the last undone action, effectively restoring any changes that were previously undone. It helps users revert back to their intended design after using the undo function.

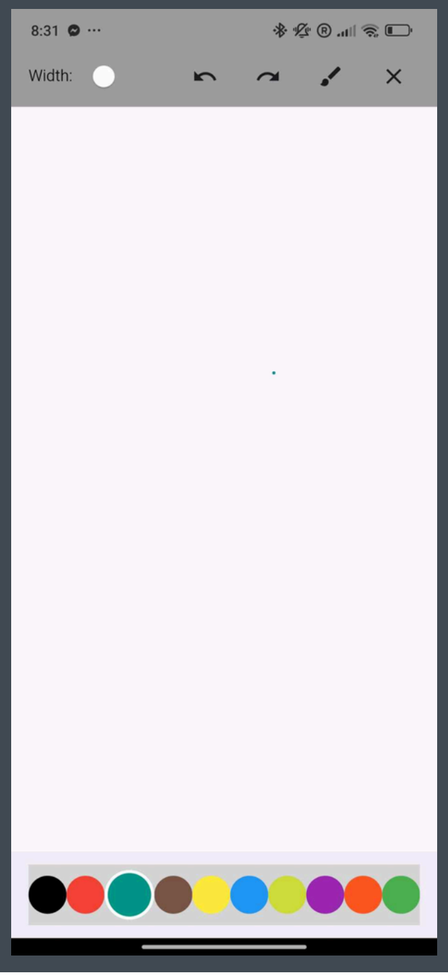
- **Undo:** The undo function lets users reverse the most recent action. This is useful for correcting mistakes or testing different creative ideas without committing to them.

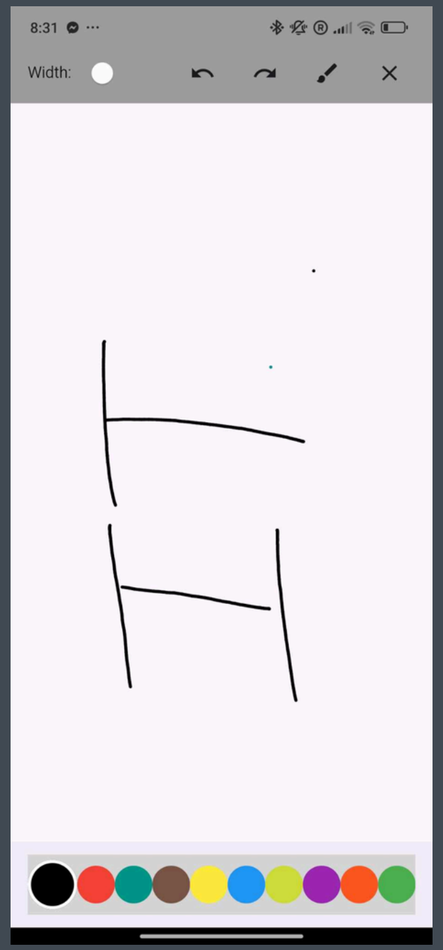
Clear Canvas: This function wipes the entire drawing area clean, allowing users to start fresh. It's an essential feature for beginning a new project or resetting the canvas.

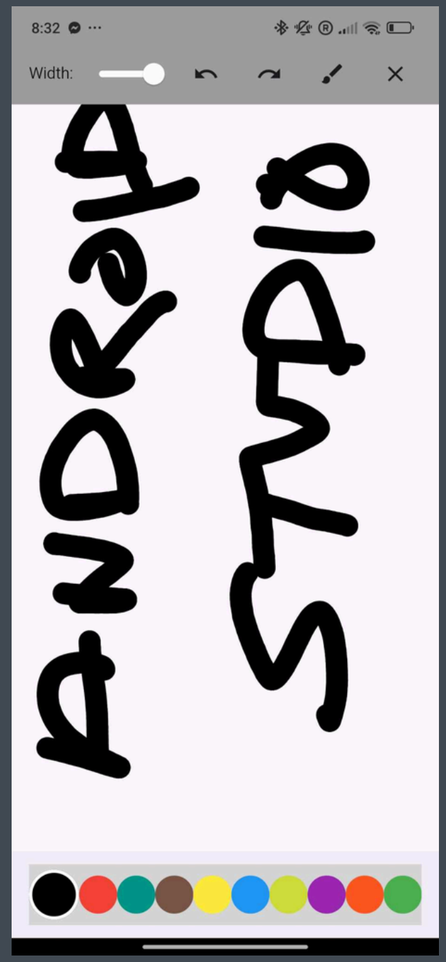
Brush Slider: The brush slider allows users to adjust the size and opacity of the brush. This feature provides precise control over the thickness and transparency of the strokes, enabling detailed or bold artwork.

Color Picker: The color picker provides a palette or spectrum for users to select and customize colors for their brush strokes. This feature might include options for choosing from a preset palette, a full-color wheel, or even eyedropper functionality to pick colors from the existing canvas.

Screenshot of Mobile Application







Source Code

```dart
import 'dart:ui';
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      home: const DrawingBoard(),
    );  // MaterialApp
  }
}

class DrawingBoard extends StatefulWidget {
  const DrawingBoard({super.key});

  @override
  State<DrawingBoard> createState() => _DrawingBoardState();
}

class _DrawingBoardState extends State<DrawingBoard> {
  Color selectedColor = Colors.black;
  double strokeWidth = 5;
  List<DrawingPoint?> drawingPoints = [];
  List<DrawingPoint?> undonePoints = [];
  List<Color> colors = [
    Colors.black,
    Colors.red,
    Colors.teal,
    Colors.brown,
    Colors.yellow,
    Colors.blue,
    Colors.lime,
    Colors.purple,
    Colors.deepOrange,
    Colors.green,
    Colors.white,
  ];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Colors.grey,
        title: Row(
          children: [
            const Text(
              "Width:",
              style: TextStyle(fontSize: 18, color: Colors.black),
            ),  // Text
            Expanded(
              child: Slider(
```

Source Code

```
            value: strokeWidth,
            min: 1.0,
            max: 20.0,
            onChanged: (val) => setState(() => strokeWidth = val),
            activeColor: Colors.white,
            inactiveColor: Colors.grey,
          ),  // Slider
        ),  // Expanded
        Row(
          children: [
            IconButton(
              icon: const Icon(Icons.undo),
              onPressed: undo,
              tooltip: 'Undo',
            ),  // IconButton
            const SizedBox(width: 10),
            IconButton(
              icon: const Icon(Icons.redo),
              onPressed: redo,
              tooltip: 'Redo',
            ),  // IconButton
            const SizedBox(width: 10),
            IconButton(
              icon: const Icon(Icons.brush),
              onPressed: () => setState(() => selectedColor = Colors.white),
              tooltip: 'Eraser',
            ),  // IconButton
            const SizedBox(width: 10), // Space between buttons
            IconButton(
              icon: const Icon(Icons.clear),
              onPressed: () => setState(() {
                drawingPoints.clear();
                undonePoints.clear();
              }),
              tooltip: 'Clear Canvas',
              iconSize: 24, // Smaller size for the Clear Canvas button
            ),  // IconButton
          ],
        ),  // Row
      ],
    ),  // Row
  ),  // AppBar
  body: GestureDetector(
    onPanStart: (details) {
      setState(() {
        undonePoints.clear(); // Clear undone points when a new stroke starts
        drawingPoints.add(
          DrawingPoint(
            details.localPosition,
            Paint()
              ..color = selectedColor
              ..isAntiAlias = true
              ..strokeWidth = strokeWidth
              ..strokeCap = StrokeCap.round,
          ),  // DrawingPoint
        );
      });
    },
```

Source Code

```
117           onPanUpdate: (details) {
118             setState(() {
119               drawingPoints.add(
120                 DrawingPoint(
121                   details.localPosition,
122                   Paint()
123                     ..color = selectedColor
124                     ..isAntiAlias = true
125                     ..strokeWidth = strokeWidth
126                     ..strokeCap = StrokeCap.round,
127                 ),  // DrawingPoint
128               );
129             });
130           },
131           onPanEnd: (details) {
132             setState(() {
133               drawingPoints.add(null);
134             });
135           },
136           child: CustomPaint(
137             painter: _DrawingPainter(drawingPoints),
138             child: Container(
139               height: MediaQuery.of(context).size.height,
140               width: MediaQuery.of(context).size.width,
141             ),  // Container
142           ),  // CustomPaint
143         ),  // GestureDetector
144         bottomNavigationBar: BottomAppBar(
145           child: Container(
```

```
146             color: Colors.grey[350],
147             child: SingleChildScrollView(
148               scrollDirection: Axis.horizontal,
149               child: Row(
150                 mainAxisAlignment: MainAxisAlignment.spaceEvenly,
151                 children: List.generate(
152                   colors.length,
153                       (index) => _buildColorChose(colors[index]),
154                 ),  // List.generate
155               ),  // Row
156             ),  // SingleChildScrollView
157           ),  // Container
158         ),  // BottomAppBar
159       );  // Scaffold
160     }
161
162     void undo() {
163       setState(() {
164         if (drawingPoints.isNotEmpty) {
165           DrawingPoint? lastPoint = drawingPoints.removeLast();
166           while (lastPoint != null && drawingPoints.isNotEmpty) {
167             undonePoints.add(lastPoint);
168             lastPoint = drawingPoints.removeLast();
169           }
170           undonePoints.add(null); // Mark the end of the removed stroke
171         }
172       });
173     }
174
```

Source Code

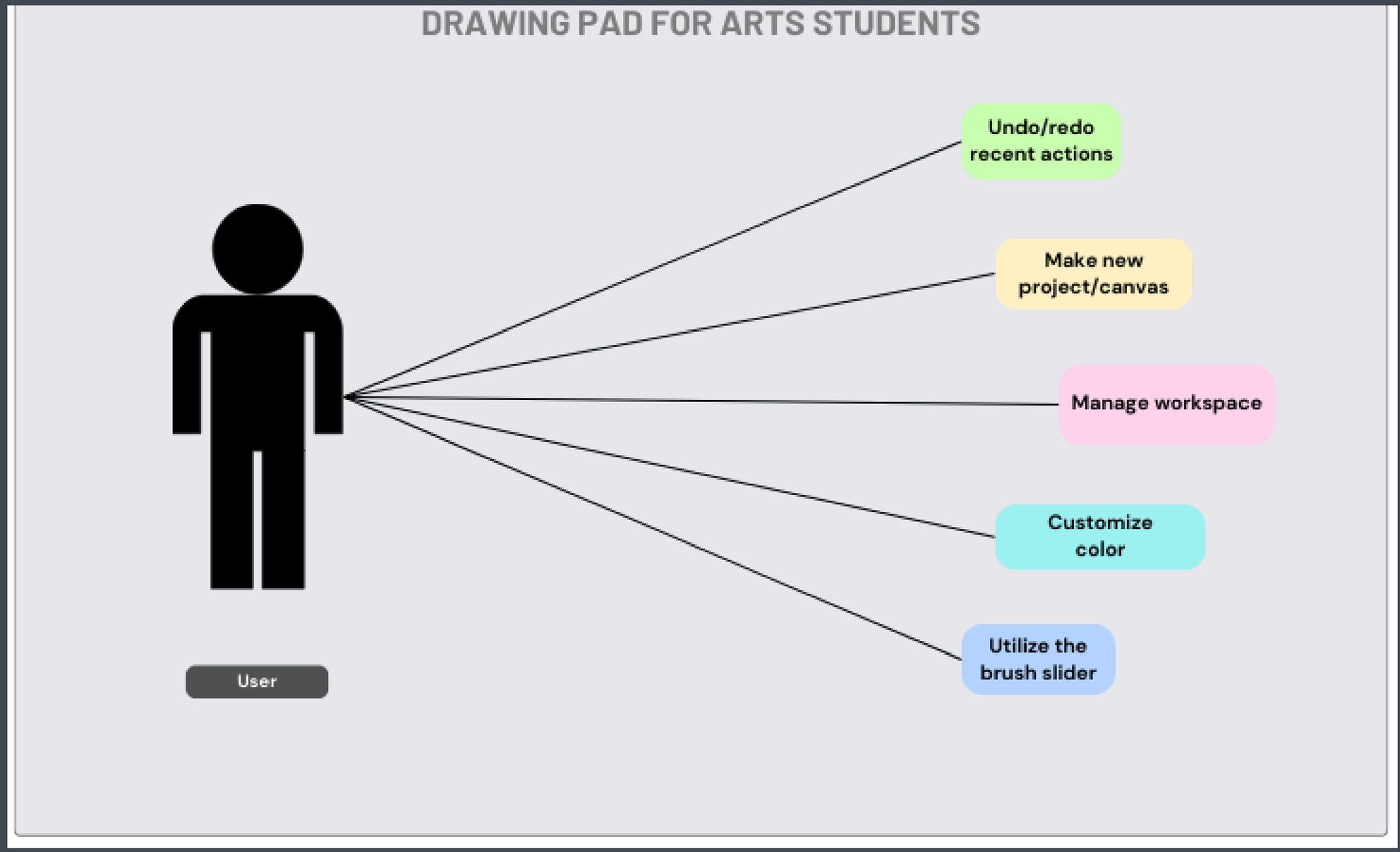
```
175   void redo() {
176     setState(() {
177       if (undonePoints.isNotEmpty) {
178         DrawingPoint? lastUndonePoint = undonePoints.removeLast();
179         while (lastUndonePoint != null && undonePoints.isNotEmpty) {
180           drawingPoints.add(lastUndonePoint);
181           lastUndonePoint = undonePoints.removeLast();
182         }
183         drawingPoints.add(null); // Mark the end of the restored stroke
184       }
185     });
186   }
187
188   Widget _buildColorChose(Color color) {
189     bool isSelected = selectedColor == color;
190     return GestureDetector(
191       onTap: () => setState(() => selectedColor = color),
192       child: Container(
193         height: isSelected ? 46 : 35,
194         width: isSelected ? 46 : 35,
195         decoration: BoxDecoration(
196           color: color,
197           shape: BoxShape.circle,
198           border: isSelected
199               ? Border.all(
200             color: Colors.white,
201             width: 3,
202           )  // Border.all
203               : null,
```

```
204         ),  // BoxDecoration
205       ),  // Container
206     );  // GestureDetector
207   }
208 }
209
210 class _DrawingPainter extends CustomPainter {
211   final List<DrawingPoint?> drawingPoints;
212
213   _DrawingPainter(this.drawingPoints);
214
215   List<Offset> offsetsList = [];
216
217   @override
218   void paint(Canvas canvas, Size size) {
219     for (int i = 0; i < drawingPoints.length - 1; i++) {
220       if (drawingPoints[i] != null && drawingPoints[i + 1] != null) {
221         canvas.drawLine(
222           drawingPoints[i]!.offset,
223           drawingPoints[i + 1]!.offset,
224           drawingPoints[i]!.paint,
225         );
226       } else if (drawingPoints[i] != null && drawingPoints[i + 1] == null) {
227         offsetsList.clear();
228         offsetsList.add(drawingPoints[i]!.offset);
229
230         canvas.drawPoints(
231           PointMode.points,
232           offsetsList,
```

Source Code

```
233                drawingPoints[i]!.paint,
234              );
235            }
236          }
237        }
238
239        @override
240        bool shouldRepaint(covariant CustomPainter oldDelegate) => true;
241      }
242
243      class DrawingPoint {
244        Offset offset;
245        Paint paint;
246
247        DrawingPoint(this.offset, this.paint);
248      }
249
```

USE CASE


DRAWING PAD FOR ARTS STUDENTS
Undo/redo recent actions
Make new project/canvas
Manage workspace
Customize color
Utilize the brush slider
User