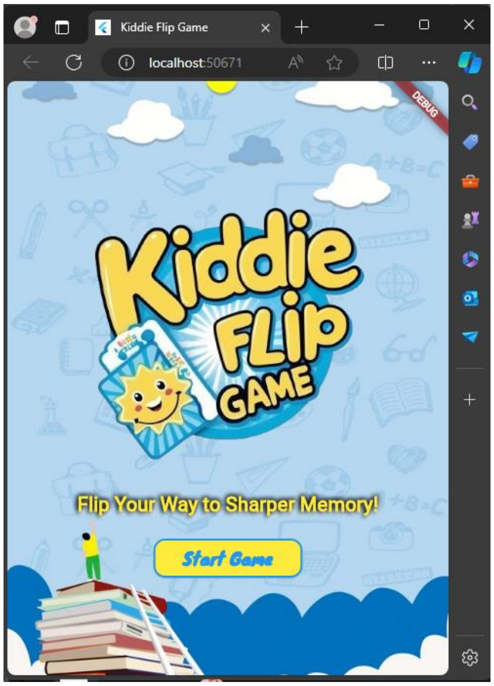
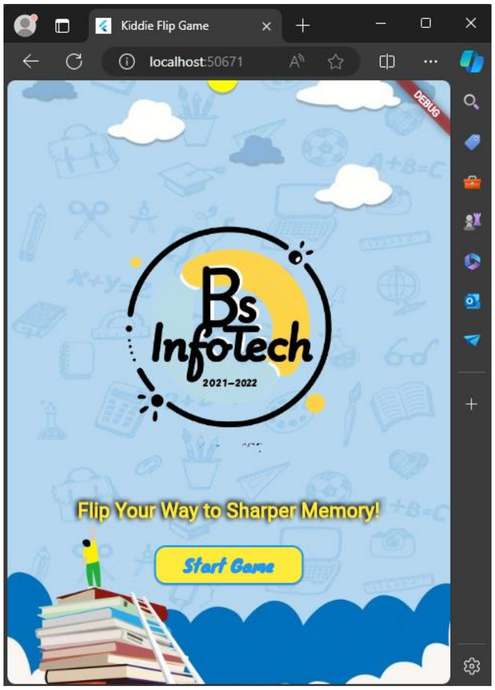
KIDDIE FLIP GAME
Flip Your Way to Sharper Memory!

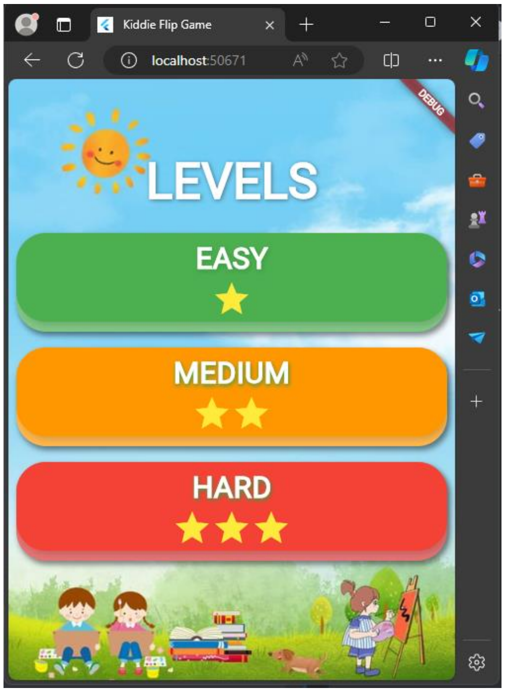
Bon Kristian De Vera

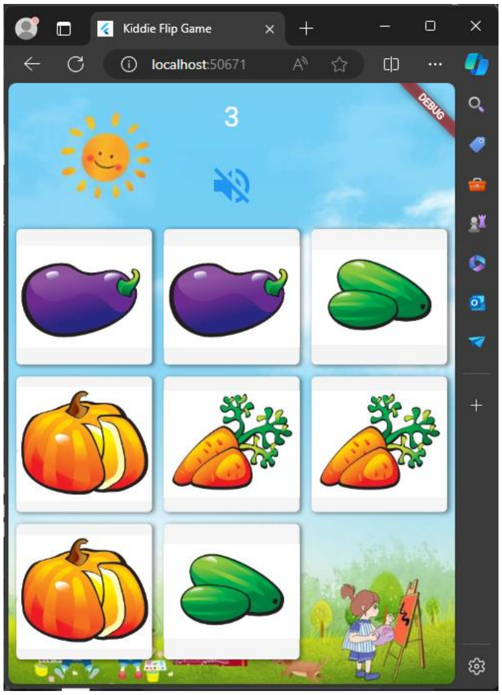
Ryan Degorio

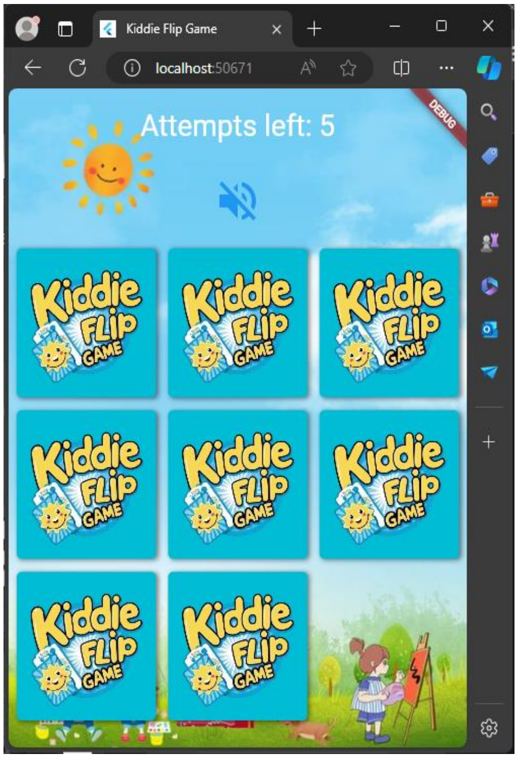
Blesse Grace S. Estorosos


Kiddie Flip Game
localhost:50671
DEBUG
Kiddie FLIP GAME
Flip Your Way to Sharper Memory!
Start Game

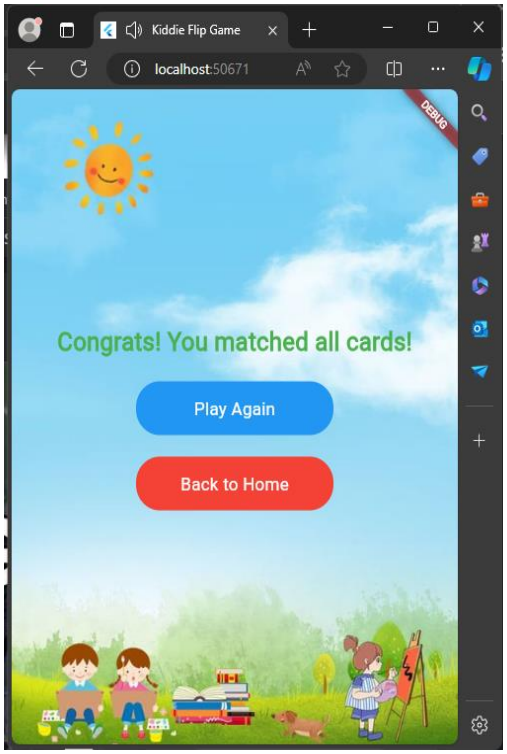

Kiddie Flip Game
localhost:50671
DEBUG
Bs Infotech
2021-2022
Flip Your Way to Sharper Memory!
Start Game


Kiddie Flip Game
localhost:50671
DEBUG
LEVELS
EASY
MEDIUM
HARD

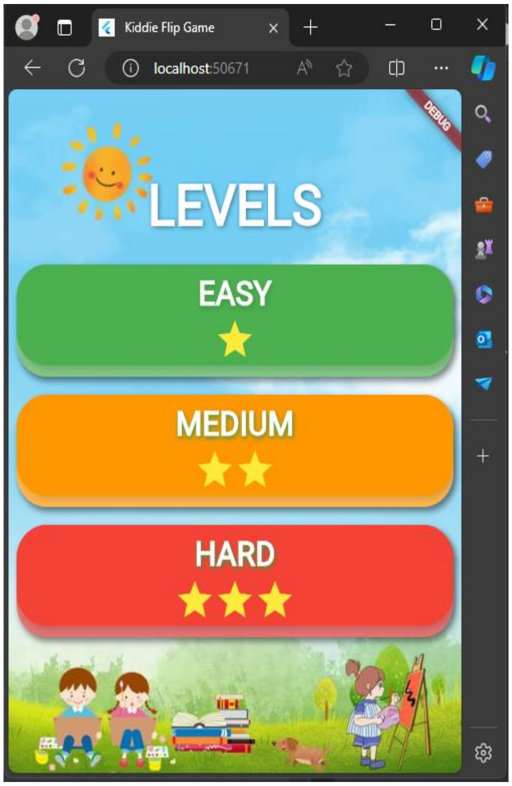

Kiddie Flip Game
localhost:50671
DEBUG
3

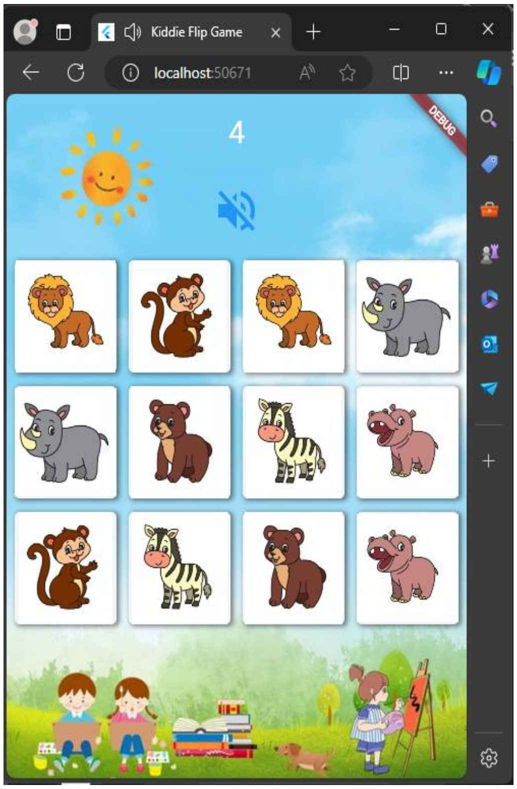


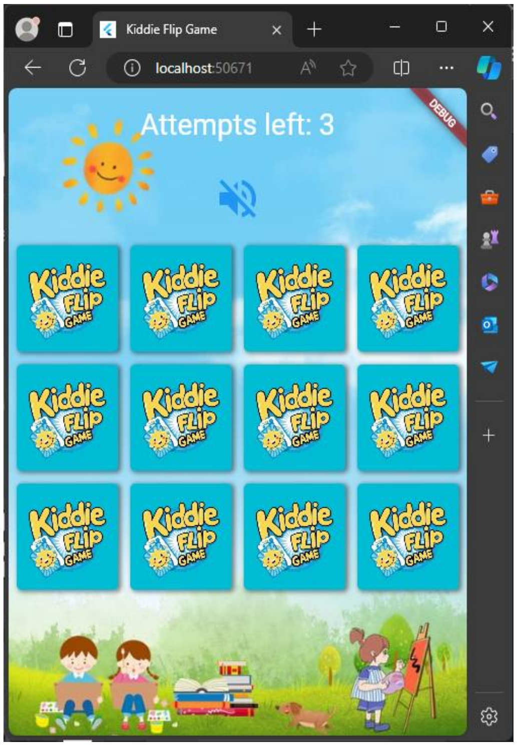
Attempts left: 5

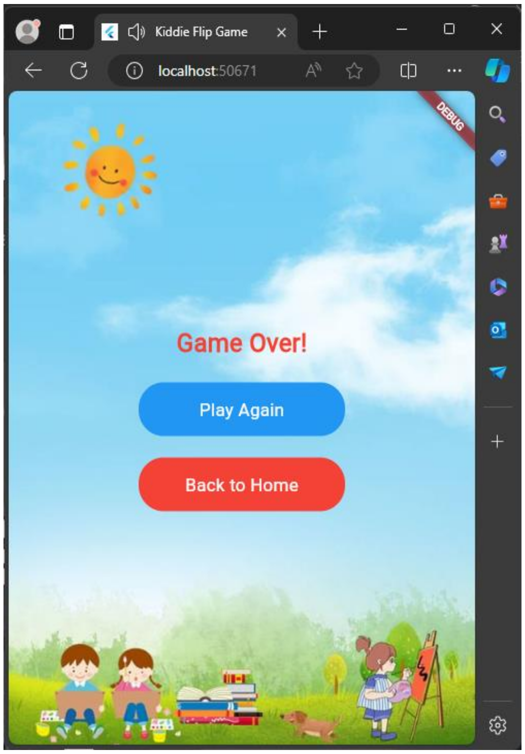


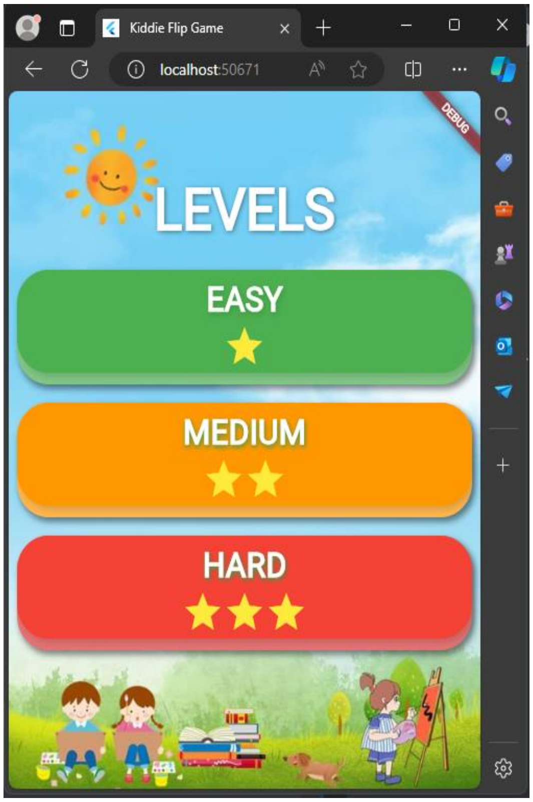
Congrats! You matched all cards!

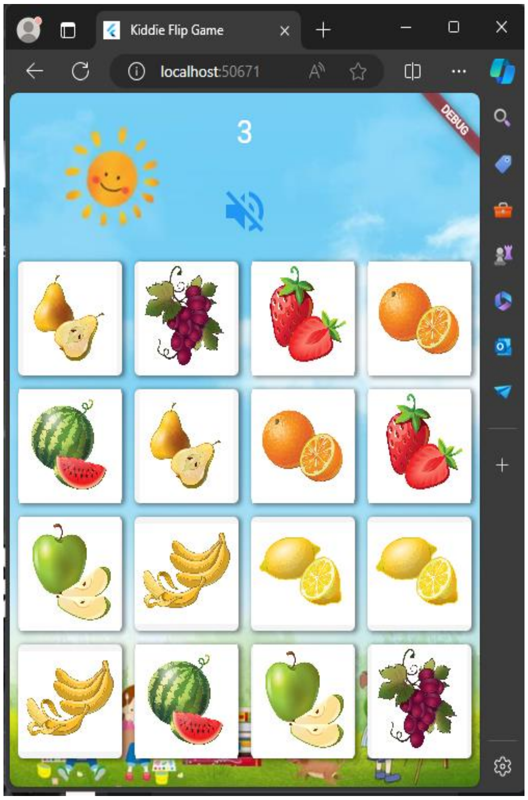


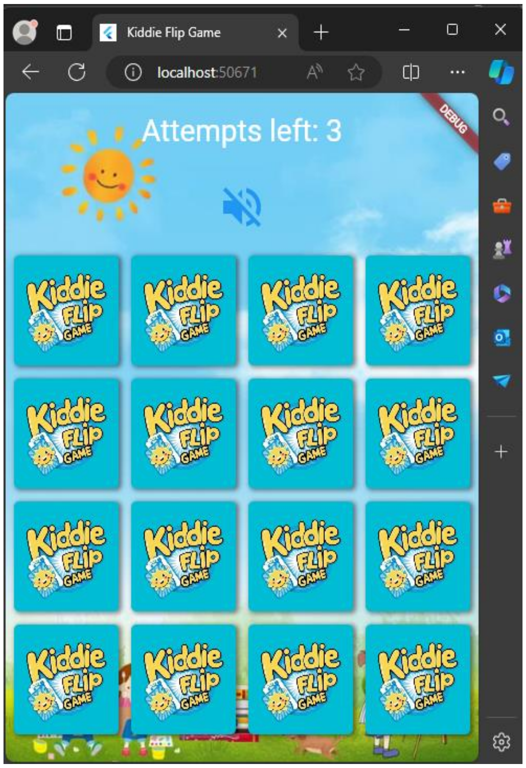



Kiddie Flip Game
localhost:50671
DEBUG
Attempts left: 3
Kiddie FLIP GAME

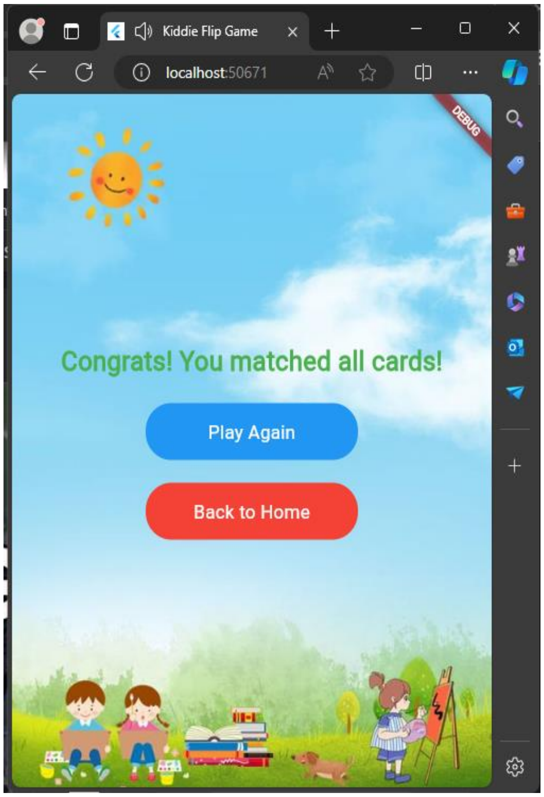

Kiddie Flip Game
localhost:50671
DEBUG
Game Over!
Play Again
Back to Home


Kiddie Flip Game
localhost:50671
DEBUG
LEVELS
EASY
MEDIUM
HARD


Kiddie Flip Game
localhost:50671
DEBUG
3

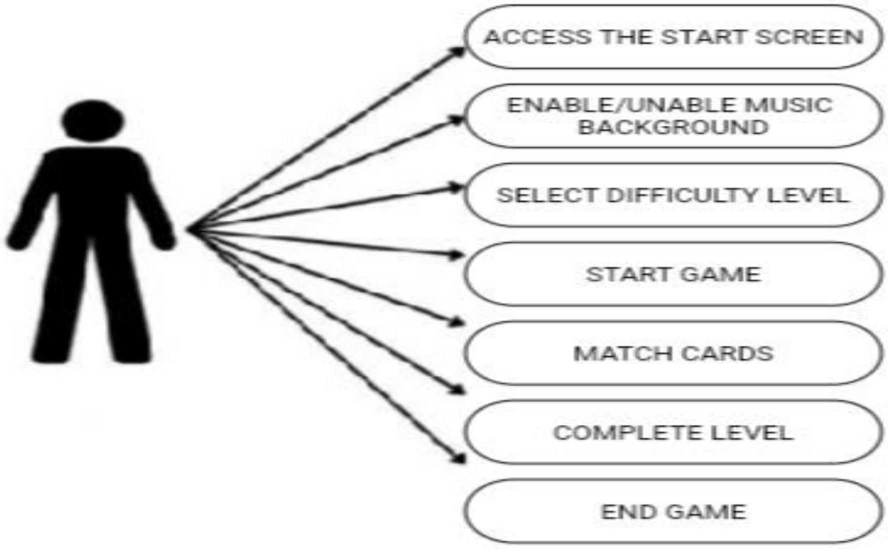


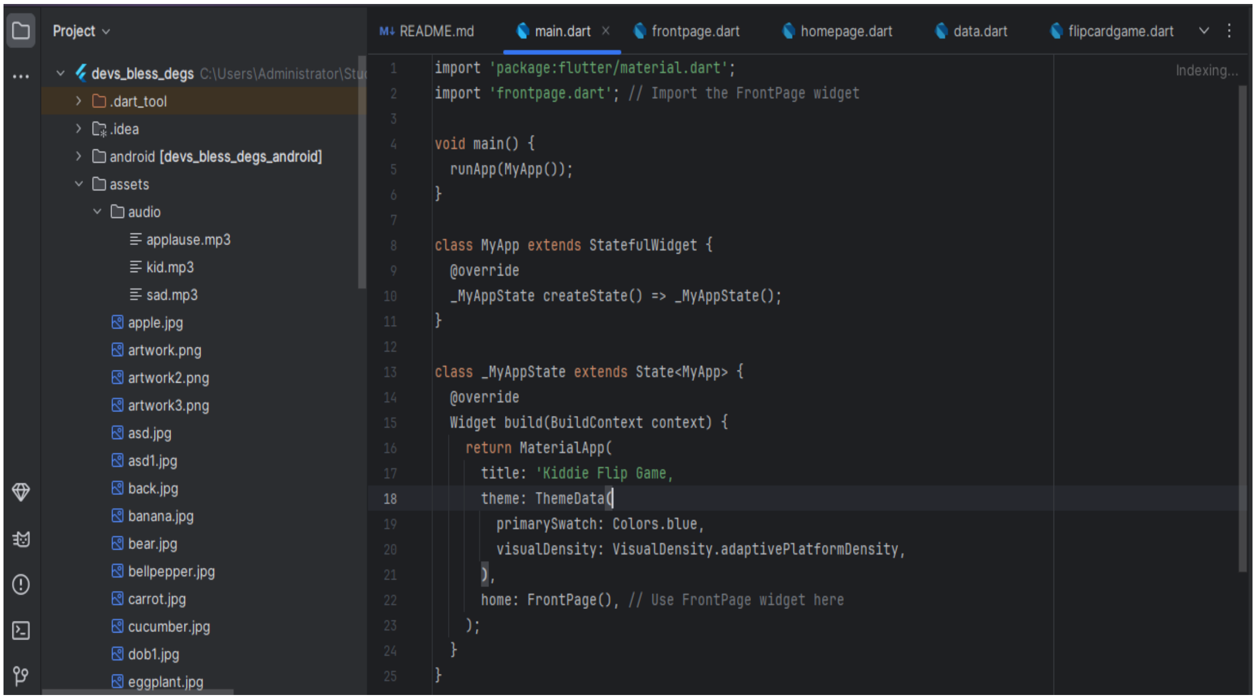
Attempts left: 3



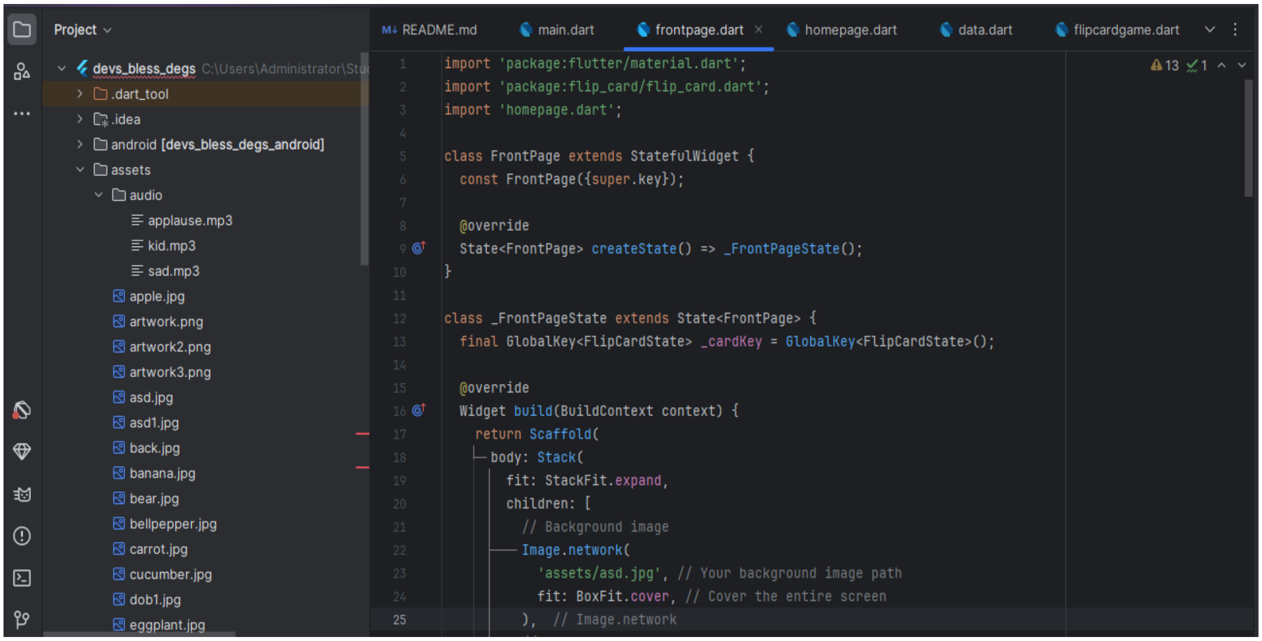
Congrats! You matched all cards!

Kiddie Flip Game is a fun and engaging memory game designed for players of all ages. It offers a simple yet challenging experience that sharpens your memory and reflexes. The game features three levels: easy, medium and hard. Each level comes with different card flip speeds, challenging you to rely on your quick reflexes and sharp memory to match the identical images before they disappear. The game also includes features such as the ability to turn the sound on or off, and the option to flip the logo on the front page of the application, adding a touch of customization to your gameplay.

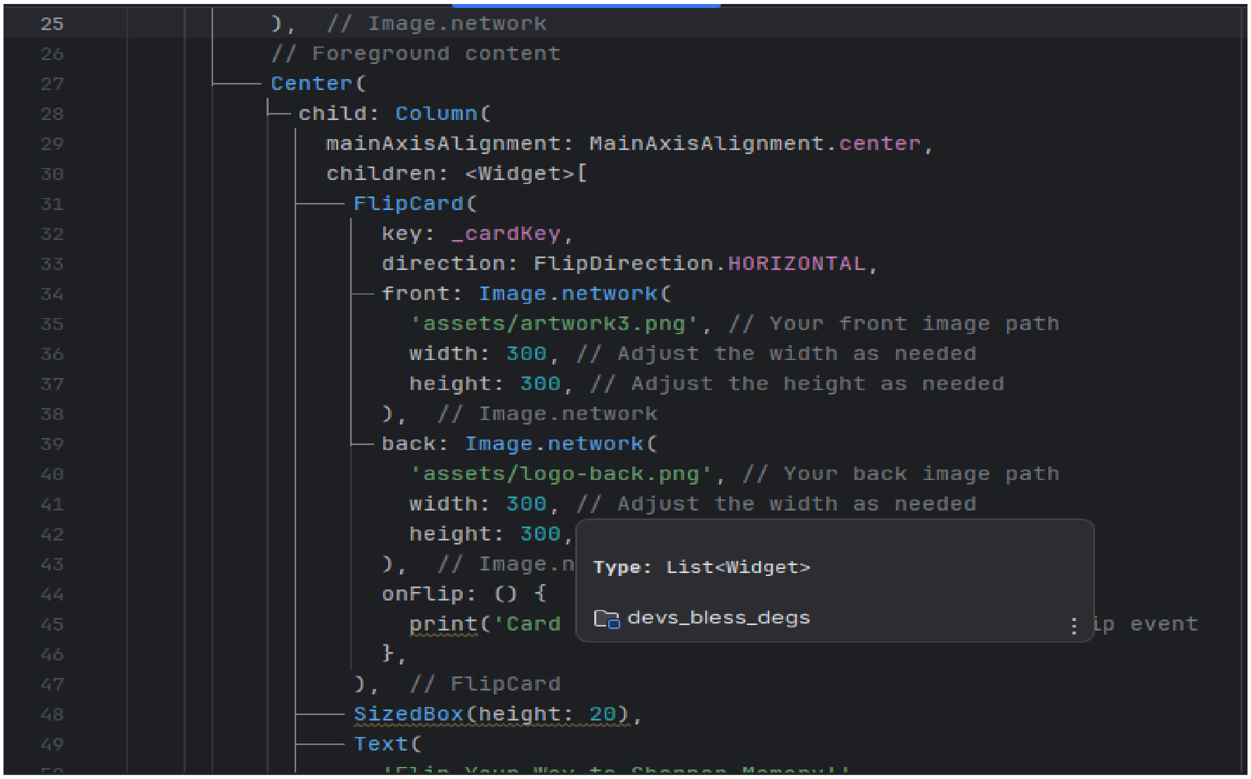




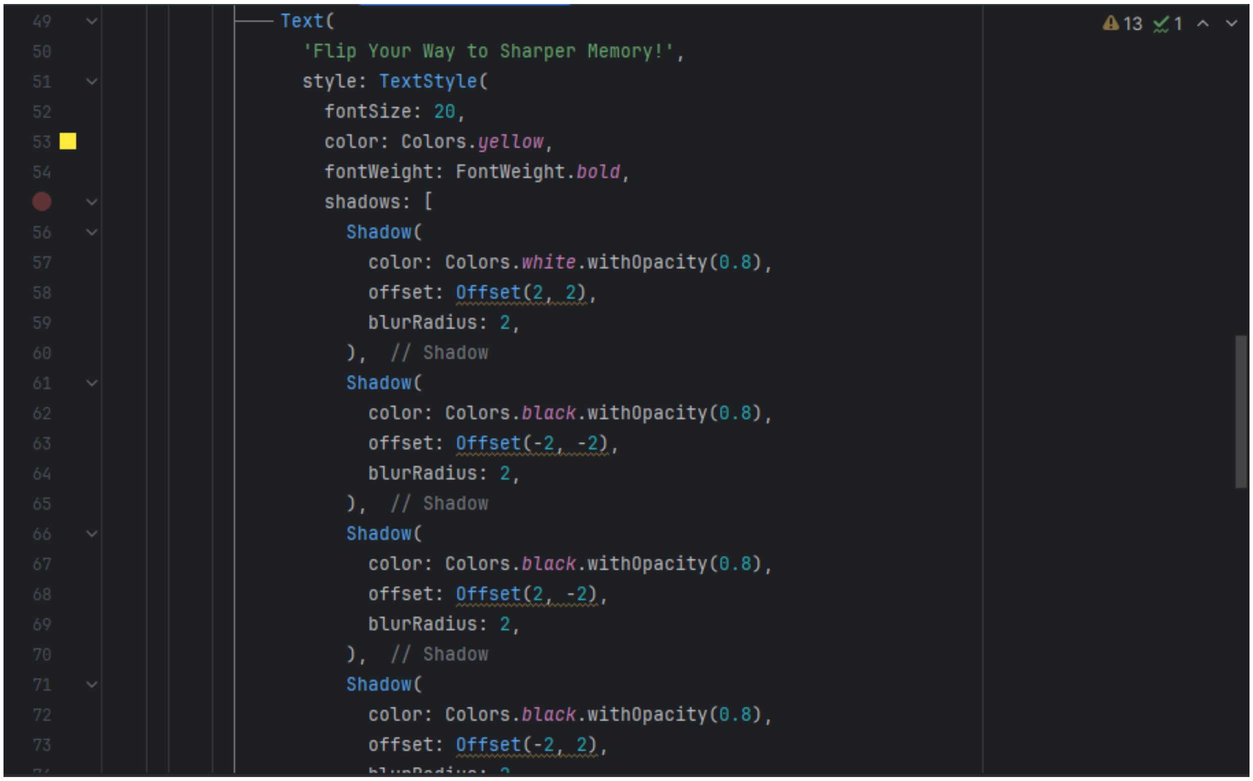
```dart
import 'package:flutter/material.dart';
import 'frontpage.dart'; // Import the FrontPage widget

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

class MyApp extends StatefulWidget {
  @override
  _MyAppState createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Kiddie Flip Game,
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ),
      home: FrontPage(), // Use FrontPage widget here
    );
  }
}
```



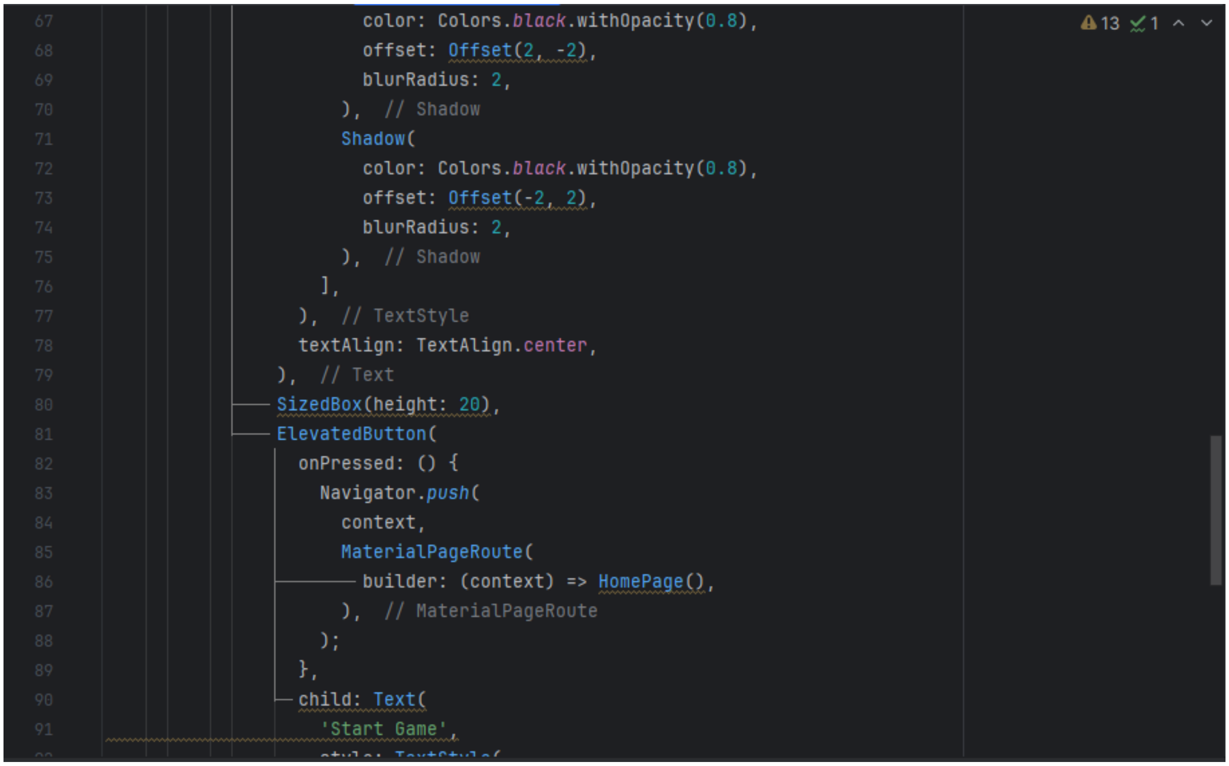
```dart
import 'package:flutter/material.dart';
import 'package:flip_card/flip_card.dart';
import 'homepage.dart';

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

  @override
  State<FrontPage> createState() => _FrontPageState();
}

class _FrontPageState extends State<FrontPage> {
  final GlobalKey<FlipCardState> _cardKey = GlobalKey<FlipCardState>();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Stack(
        fit: StackFit.expand,
        children: [
          // Background image
          Image.network(
            'assets/asd.jpg', // Your background image path
            fit: BoxFit.cover, // Cover the entire screen
          ),  // Image.network
```



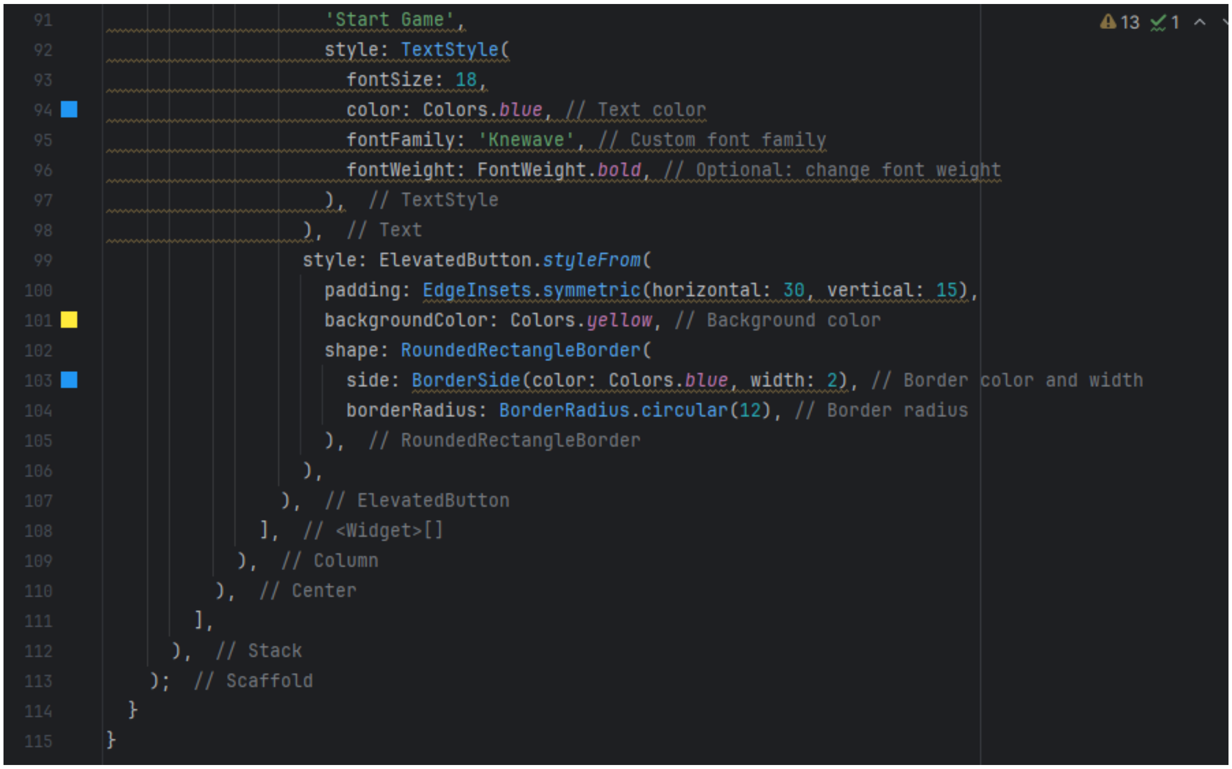
```
), // Image.network
// Foreground content
Center(
  child: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: <Widget>[
      FlipCard(
        key: _cardKey,
        direction: FlipDirection.HORIZONTAL,
        front: Image.network(
          'assets/artwork3.png', // Your front image path
          width: 300, // Adjust the width as needed
          height: 300, // Adjust the height as needed
        ), // Image.network
        back: Image.network(
          'assets/logo-back.png', // Your back image path
          width: 300, // Adjust the width as needed
          height: 300,
        ), // Image.n
        onFlip: () {
          print('Card                                      ip event
        },
      ), // FlipCard
      SizedBox(height: 20),
      Text(
        'Flip Your Way to Sharper Memory!'
```



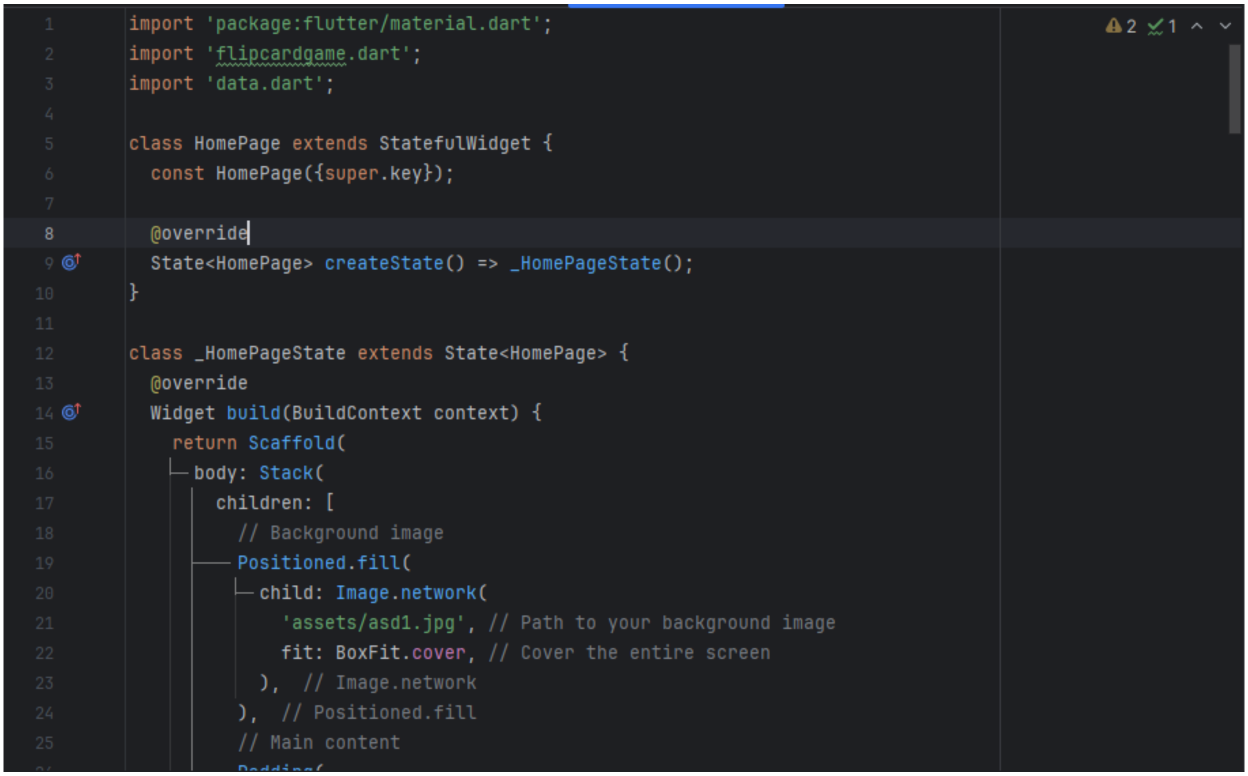
```
Text(
  'Flip Your Way to Sharper Memory!',
  style: TextStyle(
    fontSize: 20,
    color: Colors.yellow,
    fontWeight: FontWeight.bold,
    shadows: [
      Shadow(
        color: Colors.white.withOpacity(0.8),
        offset: Offset(2, 2),
        blurRadius: 2,
      ), // Shadow
      Shadow(
        color: Colors.black.withOpacity(0.8),
        offset: Offset(-2, -2),
        blurRadius: 2,
      ), // Shadow
      Shadow(
        color: Colors.black.withOpacity(0.8),
        offset: Offset(2, -2),
        blurRadius: 2,
      ), // Shadow
      Shadow(
        color: Colors.black.withOpacity(0.8),
        offset: Offset(-2, 2),
        blurRadius: 2
```



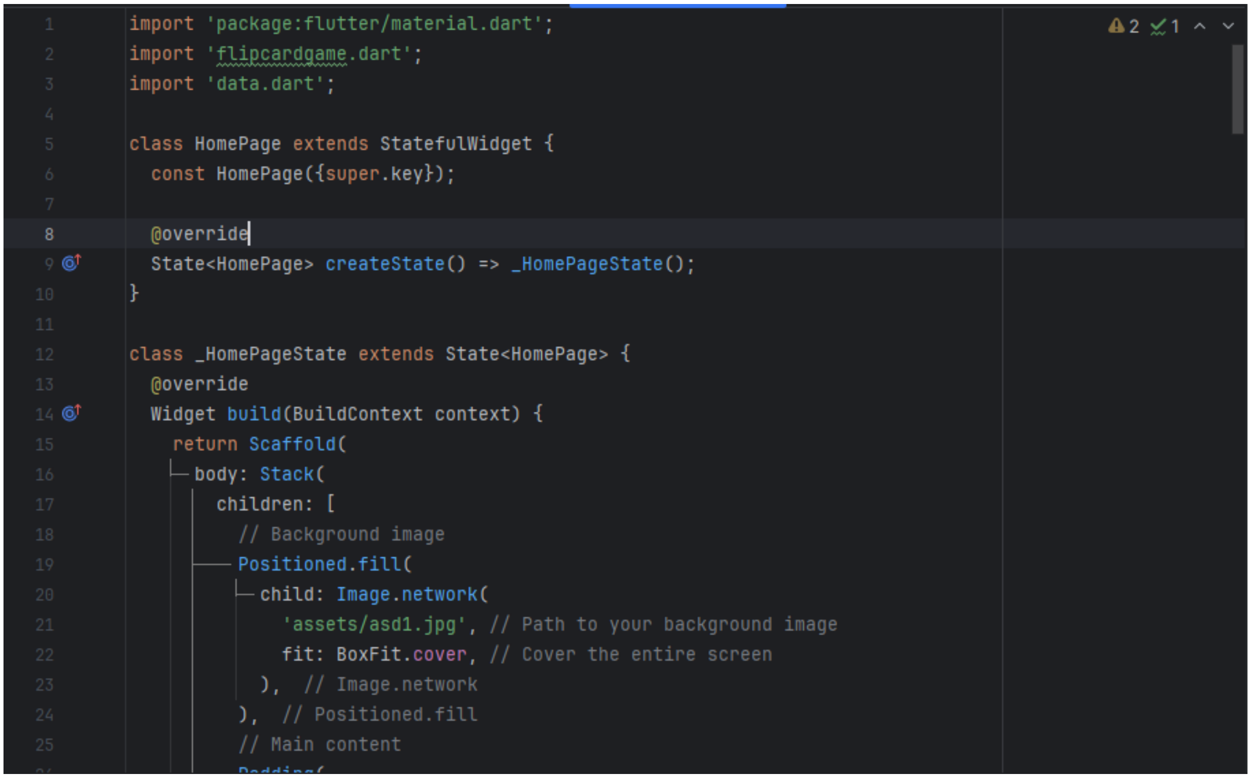
```
                        color: Colors.black.withOpacity(0.8),
                        offset: Offset(2, -2),
                        blurRadius: 2,
                      ),  // Shadow
                      Shadow(
                        color: Colors.black.withOpacity(0.8),
                        offset: Offset(-2, 2),
                        blurRadius: 2,
                      ),  // Shadow
                    ],
                  ),  // TextStyle
                  textAlign: TextAlign.center,
                ),  // Text
                SizedBox(height: 20),
                ElevatedButton(
                  onPressed: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(
                        builder: (context) => HomePage(),
                      ),  // MaterialPageRoute
                    );
                  },
                  child: Text(
                    'Start Game',
```



```
                    'Start Game',
                    style: TextStyle(
                      fontSize: 18,
                      color: Colors.blue, // Text color
                      fontFamily: 'Knewave', // Custom font family
                      fontWeight: FontWeight.bold, // Optional: change font weight
                    ),  // TextStyle
                  ),  // Text
                  style: ElevatedButton.styleFrom(
                    padding: EdgeInsets.symmetric(horizontal: 30, vertical: 15),
                    backgroundColor: Colors.yellow, // Background color
                    shape: RoundedRectangleBorder(
                      side: BorderSide(color: Colors.blue, width: 2), // Border color and width
                      borderRadius: BorderRadius.circular(12), // Border radius
                    ),  // RoundedRectangleBorder
                  ),
                ),  // ElevatedButton
              ],  // <Widget>[]
            ),  // Column
          ),  // Center
        ],
      ),  // Stack
    );  // Scaffold
  }
}
```

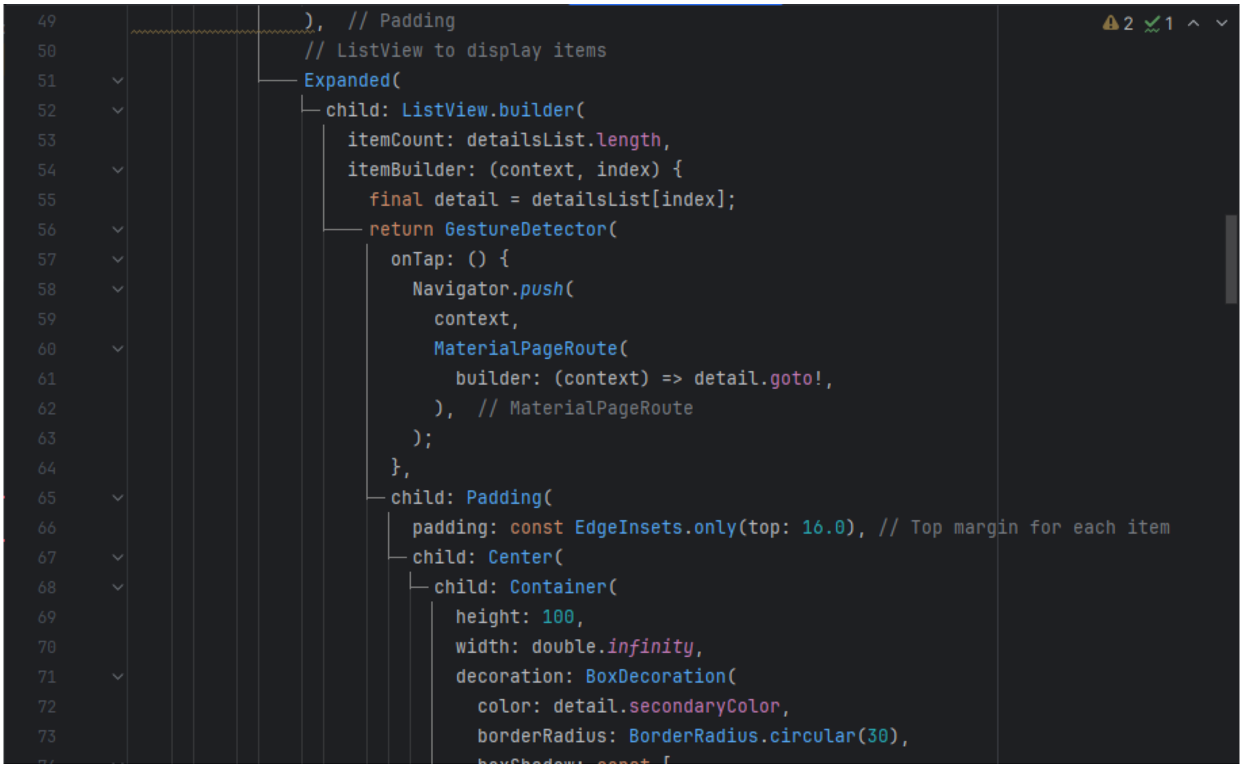


```dart
import 'package:flutter/material.dart';
import 'flipcardgame.dart';
import 'data.dart';

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

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Stack(
        children: [
          // Background image
          Positioned.fill(
            child: Image.network(
              'assets/asd1.jpg', // Path to your background image
              fit: BoxFit.cover, // Cover the entire screen
            ),  // Image.network
          ),  // Positioned.fill
          // Main content
          Padding(
```

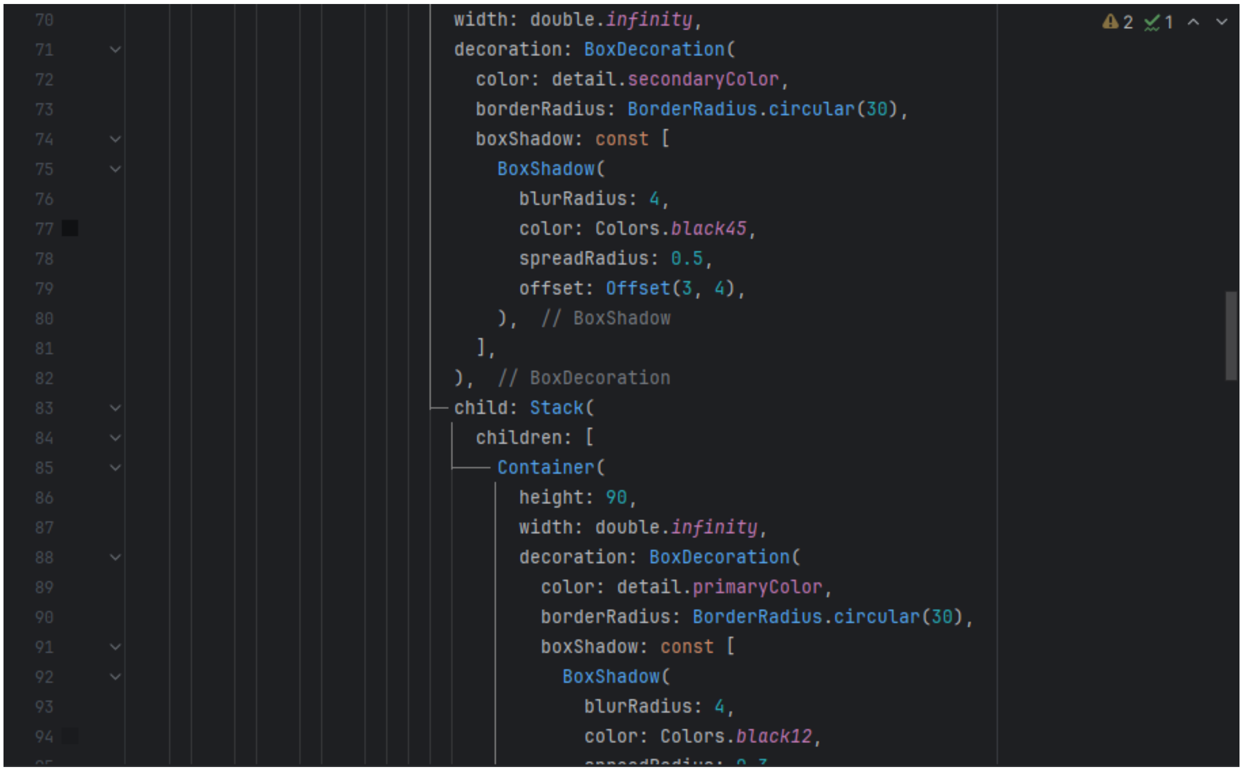


⚠2 ✓1

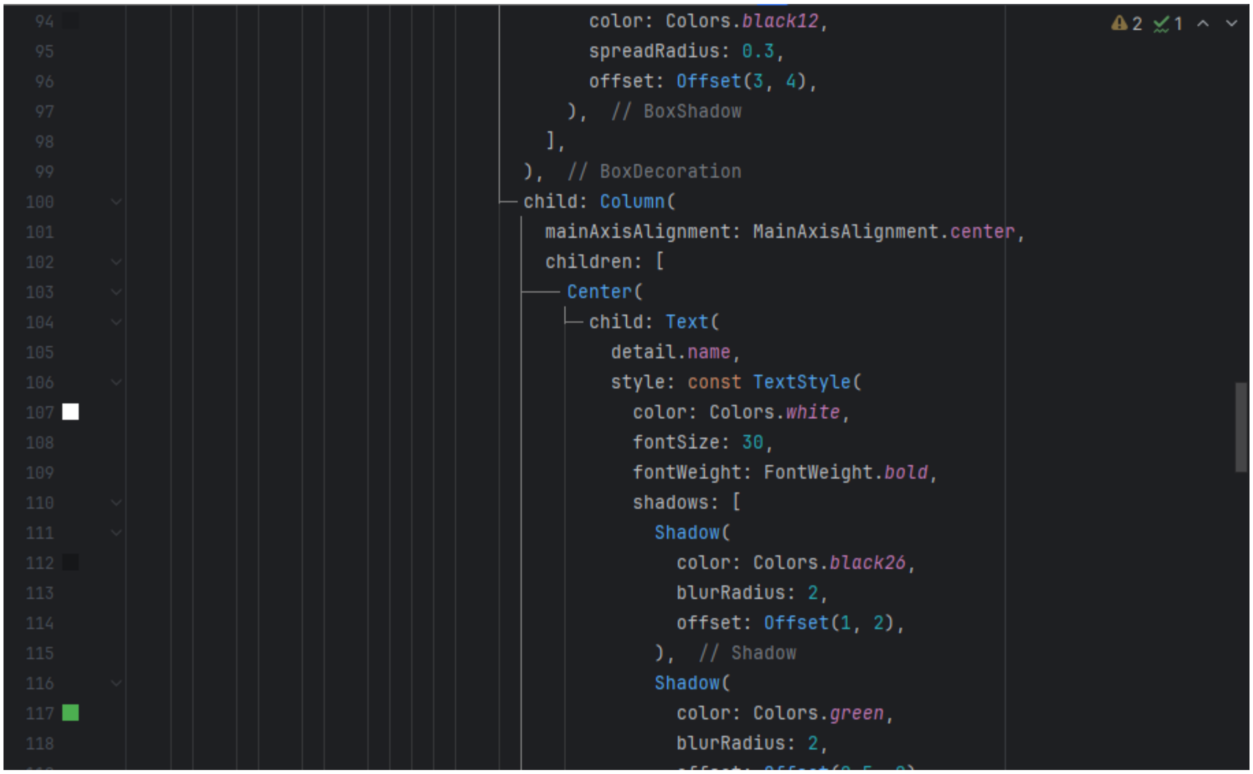
```dart
import 'package:flutter/material.dart';
import 'flipcardgame.dart';
import 'data.dart';

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

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Stack(
        children: [
          // Background image
          Positioned.fill(
            child: Image.network(
              'assets/asd1.jpg', // Path to your background image
              fit: BoxFit.cover, // Cover the entire screen
            ),  // Image.network
          ),  // Positioned.fill
          // Main content
          Padding(
```



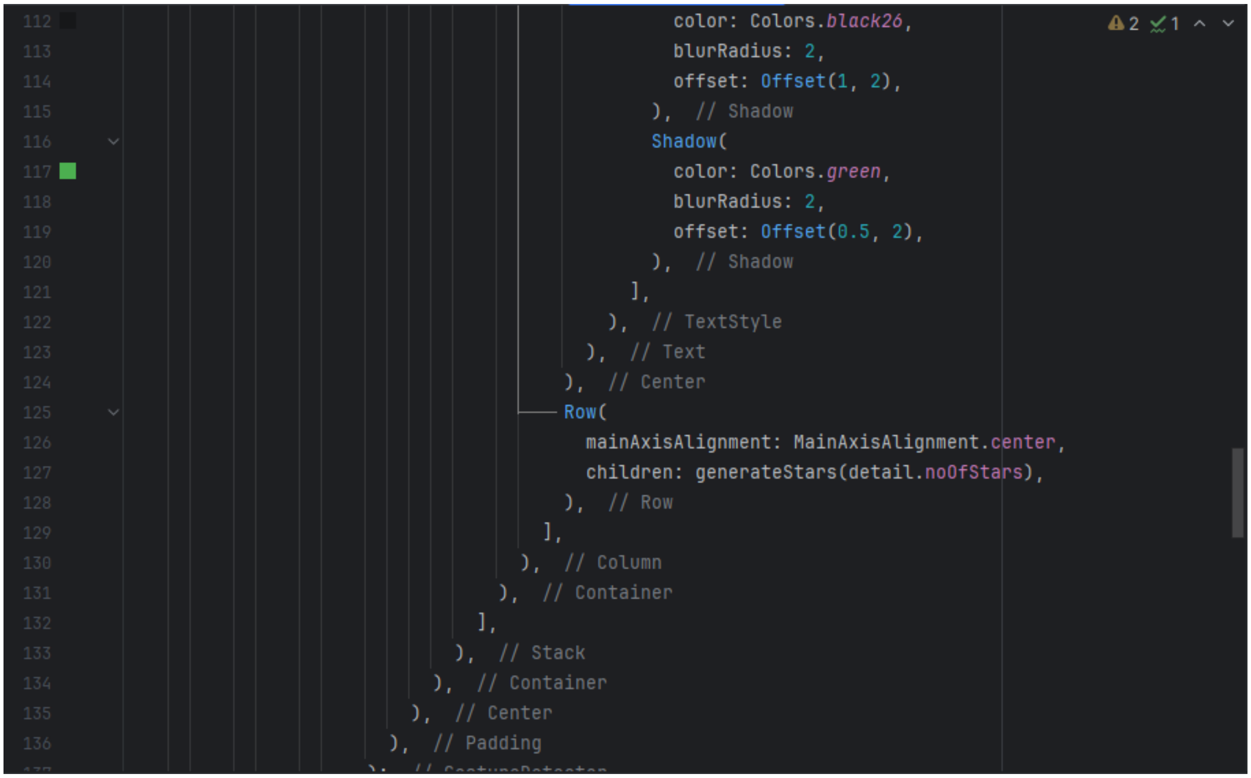
```
          ),  // Padding
          // ListView to display items
          Expanded(
            child: ListView.builder(
              itemCount: detailsList.length,
              itemBuilder: (context, index) {
                final detail = detailsList[index];
                return GestureDetector(
                  onTap: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(
                        builder: (context) => detail.goto!,
                      ),  // MaterialPageRoute
                    );
                  },
                  child: Padding(
                    padding: const EdgeInsets.only(top: 16.0), // Top margin for each item
                    child: Center(
                      child: Container(
                        height: 100,
                        width: double.infinity,
                        decoration: BoxDecoration(
                          color: detail.secondaryColor,
                          borderRadius: BorderRadius.circular(30),
```



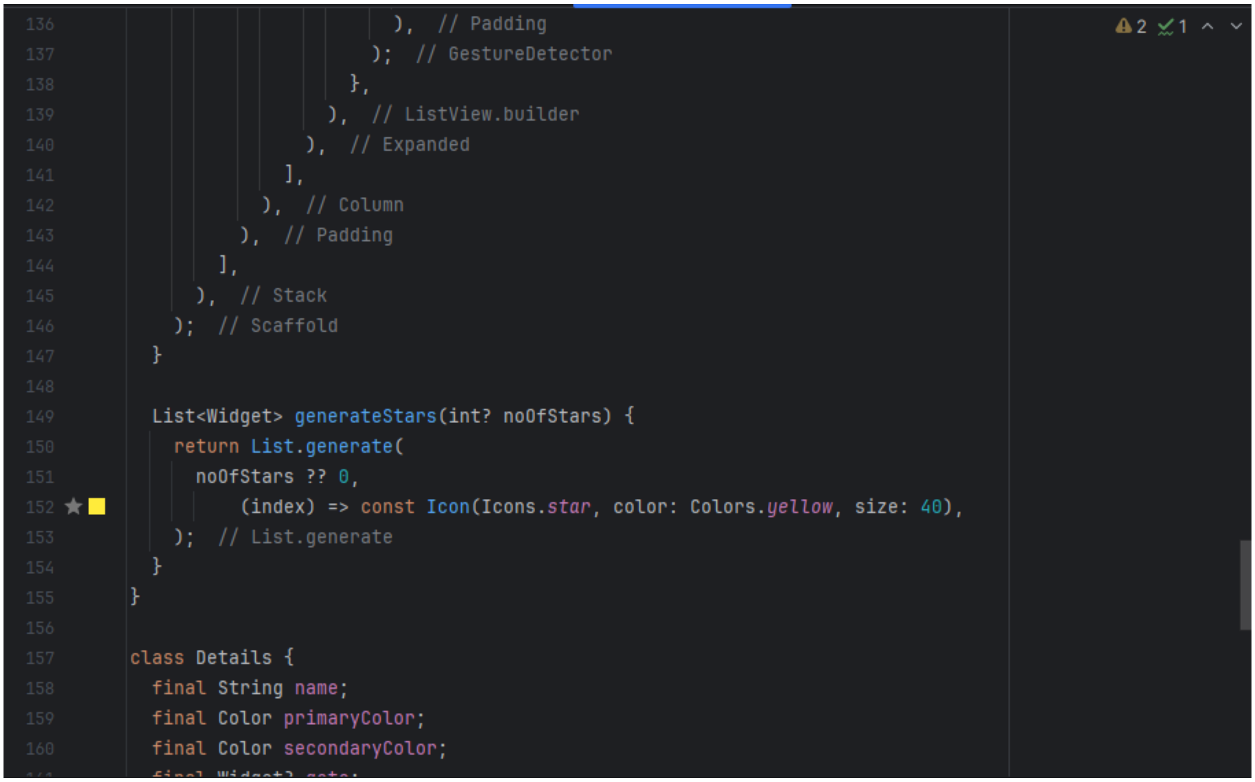
```
                        width: double.infinity,
                        decoration: BoxDecoration(
                          color: detail.secondaryColor,
                          borderRadius: BorderRadius.circular(30),
                          boxShadow: const [
                            BoxShadow(
                              blurRadius: 4,
                              color: Colors.black45,
                              spreadRadius: 0.5,
                              offset: Offset(3, 4),
                            ),  // BoxShadow
                          ],
                        ),  // BoxDecoration
                        child: Stack(
                          children: [
                            Container(
                              height: 90,
                              width: double.infinity,
                              decoration: BoxDecoration(
                                color: detail.primaryColor,
                                borderRadius: BorderRadius.circular(30),
                                boxShadow: const [
                                  BoxShadow(
                                    blurRadius: 4,
                                    color: Colors.black12,
```



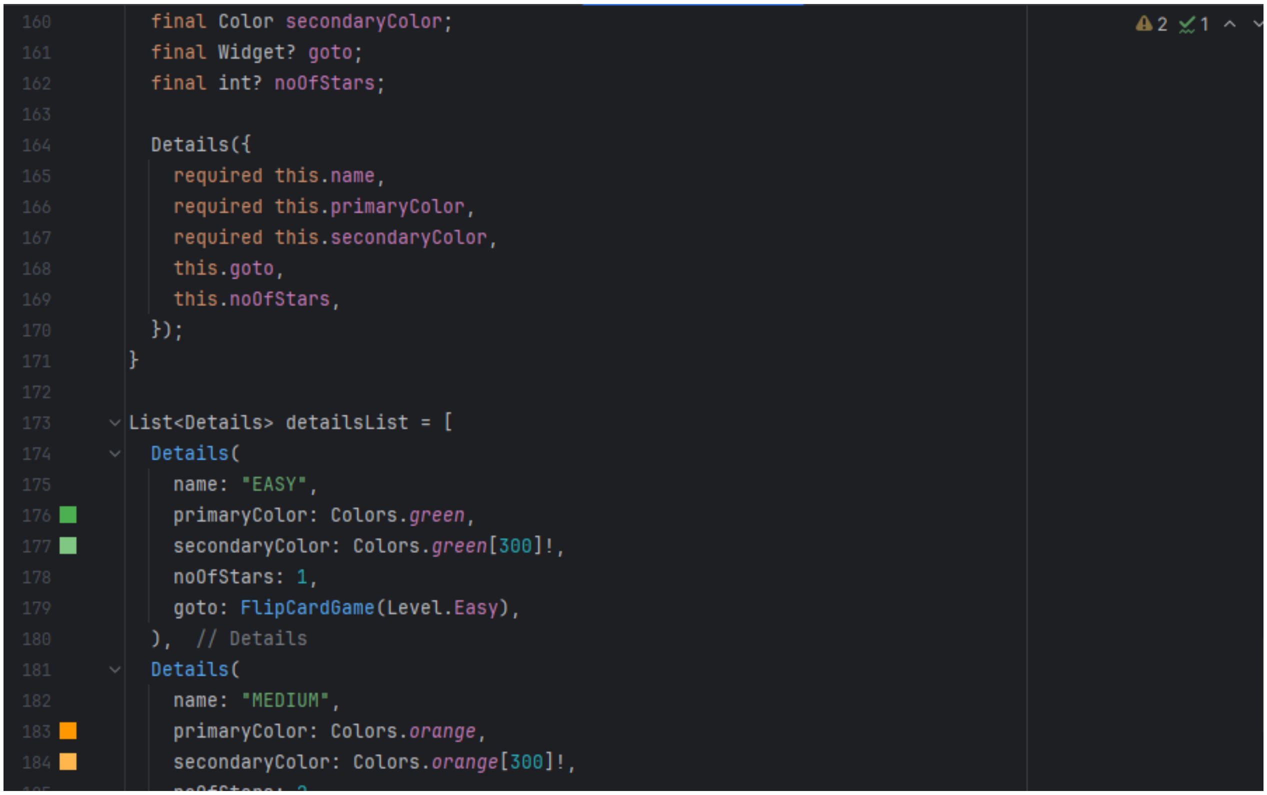
```
                    color: Colors.black12,
                    spreadRadius: 0.3,
                    offset: Offset(3, 4),
                  ), // BoxShadow
                ],
              ), // BoxDecoration
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  Center(
                    child: Text(
                      detail.name,
                      style: const TextStyle(
                        color: Colors.white,
                        fontSize: 30,
                        fontWeight: FontWeight.bold,
                        shadows: [
                          Shadow(
                            color: Colors.black26,
                            blurRadius: 2,
                            offset: Offset(1, 2),
                          ), // Shadow
                          Shadow(
                            color: Colors.green,
                            blurRadius: 2,
                            offset: Offset(0.5, 2),
```



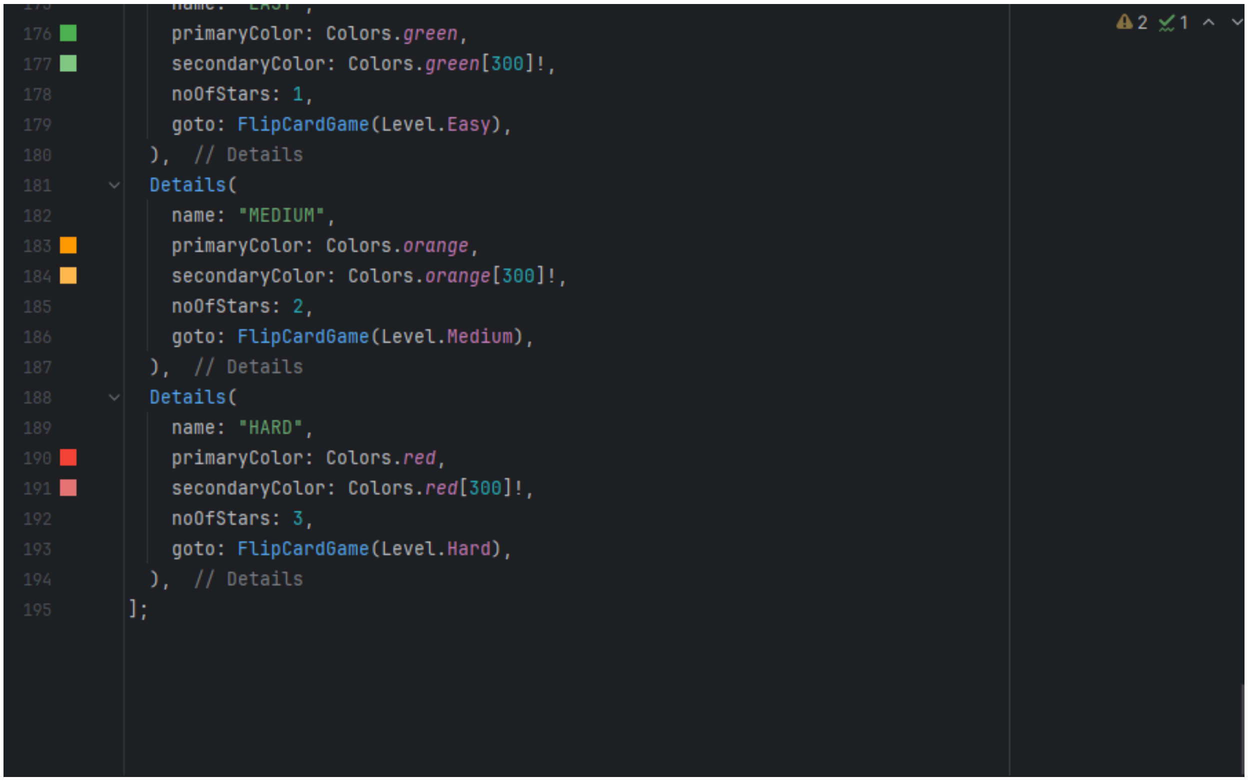
```
                            color: Colors.black26,
                            blurRadius: 2,
                            offset: Offset(1, 2),
                          ), // Shadow
                          Shadow(
                            color: Colors.green,
                            blurRadius: 2,
                            offset: Offset(0.5, 2),
                          ), // Shadow
                        ],
                      ), // TextStyle
                    ), // Text
                  ), // Center
                  Row(
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: generateStars(detail.noOfStars),
                  ), // Row
                ],
              ), // Column
            ), // Container
          ],
        ), // Stack
      ), // Container
    ), // Center
  ), // Padding
), // GestureDetector
```



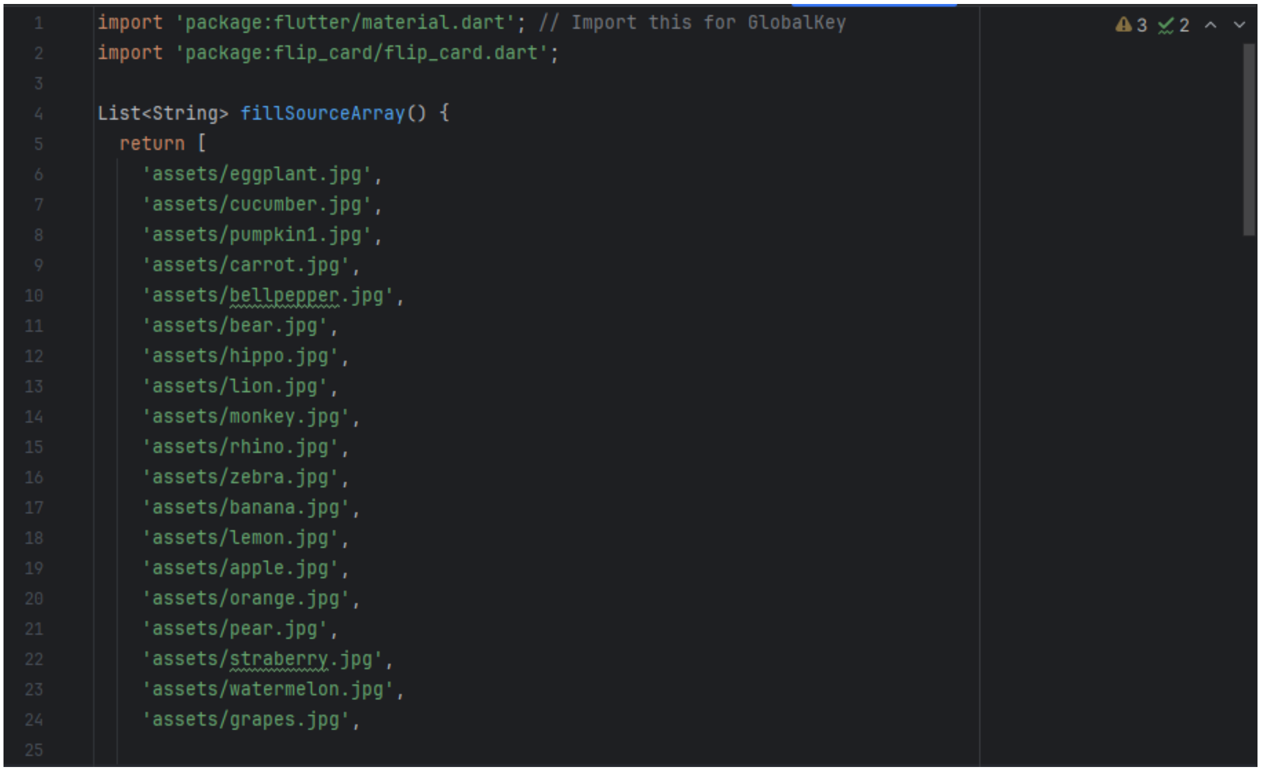
```dart
                        ),  // Padding
                      );  // GestureDetector
                    },
                  ),  // ListView.builder
                ),  // Expanded
              ],
            ),  // Column
          ),  // Padding
        ],
      ),  // Stack
    );  // Scaffold
  }

  List<Widget> generateStars(int? noOfStars) {
    return List.generate(
      noOfStars ?? 0,
          (index) => const Icon(Icons.star, color: Colors.yellow, size: 40),
    );  // List.generate
  }
}

class Details {
  final String name;
  final Color primaryColor;
  final Color secondaryColor;
  final Widget? goto;
```



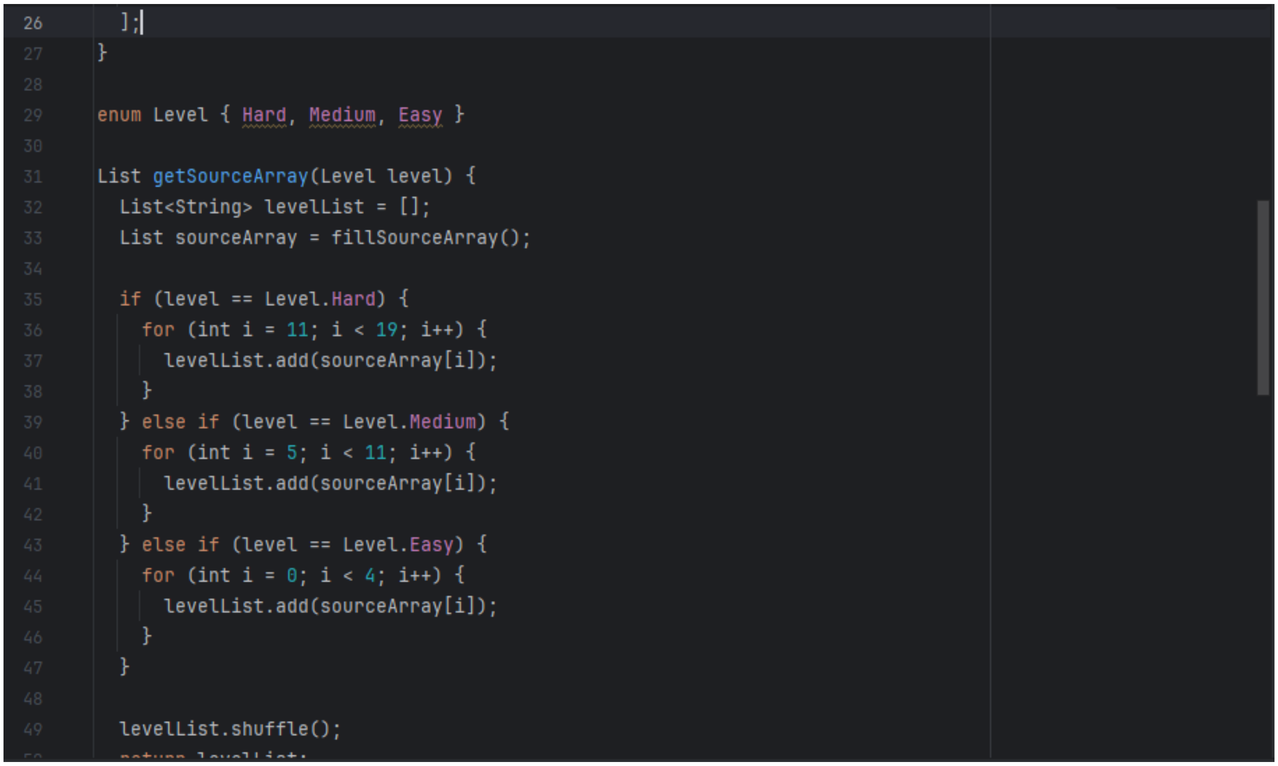
```dart
  final Color secondaryColor;
  final Widget? goto;
  final int? noOfStars;

  Details({
    required this.name,
    required this.primaryColor,
    required this.secondaryColor,
    this.goto,
    this.noOfStars,
  });
}

List<Details> detailsList = [
  Details(
    name: "EASY",
    primaryColor: Colors.green,
    secondaryColor: Colors.green[300]!,
    noOfStars: 1,
    goto: FlipCardGame(Level.Easy),
  ),  // Details
  Details(
    name: "MEDIUM",
    primaryColor: Colors.orange,
    secondaryColor: Colors.orange[300]!,
    noOfStars: 2,
```



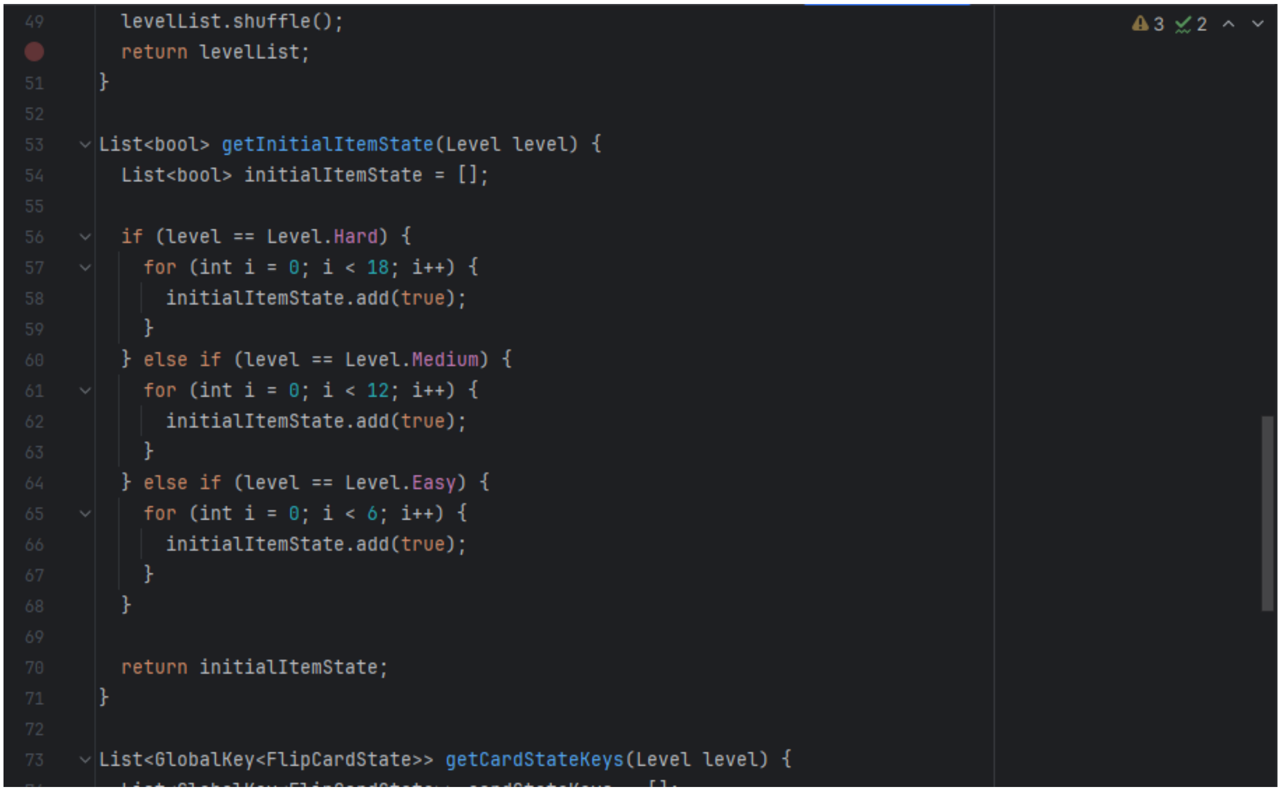
```dart
    name: "EASY",
    primaryColor: Colors.green,
    secondaryColor: Colors.green[300]!,
    noOfStars: 1,
    goto: FlipCardGame(Level.Easy),
  ), // Details
  Details(
    name: "MEDIUM",
    primaryColor: Colors.orange,
    secondaryColor: Colors.orange[300]!,
    noOfStars: 2,
    goto: FlipCardGame(Level.Medium),
  ), // Details
  Details(
    name: "HARD",
    primaryColor: Colors.red,
    secondaryColor: Colors.red[300]!,
    noOfStars: 3,
    goto: FlipCardGame(Level.Hard),
  ), // Details
];
```



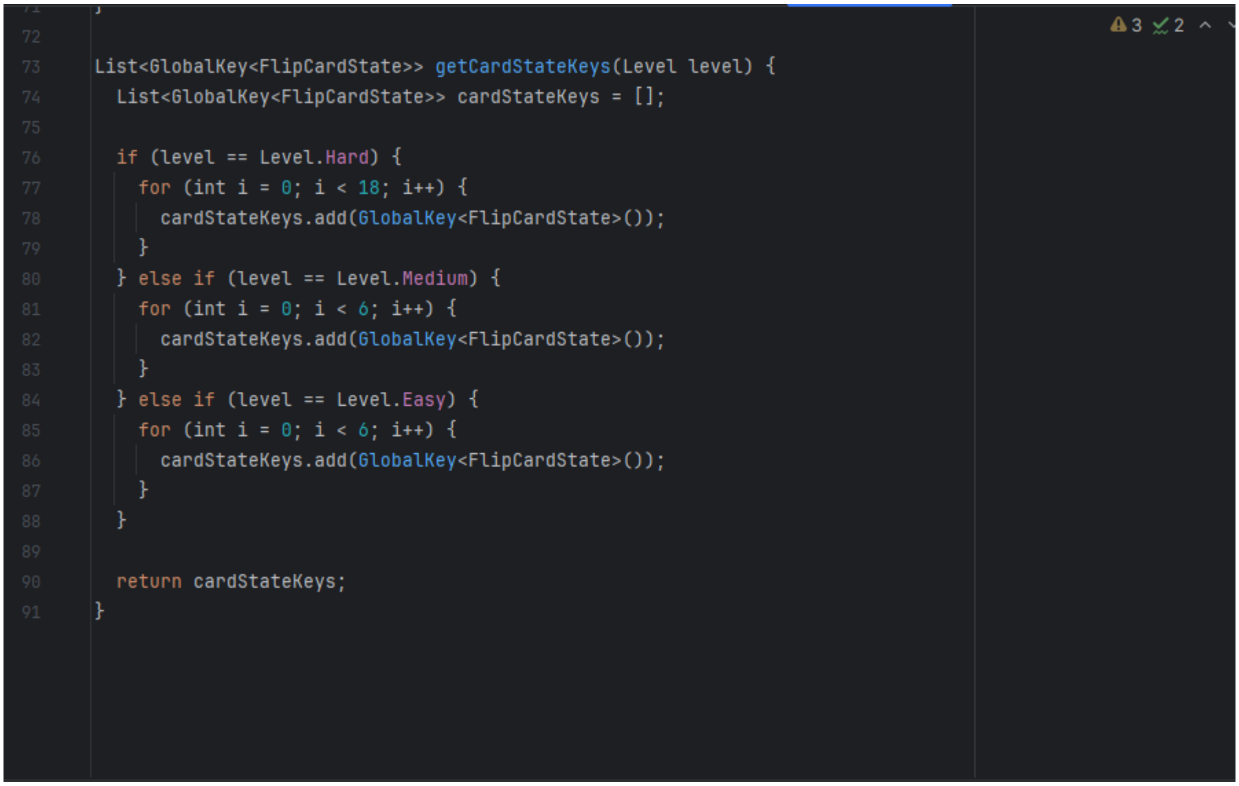
```dart
import 'package:flutter/material.dart'; // Import this for GlobalKey
import 'package:flip_card/flip_card.dart';

List<String> fillSourceArray() {
  return [
    'assets/eggplant.jpg',
    'assets/cucumber.jpg',
    'assets/pumpkin1.jpg',
    'assets/carrot.jpg',
    'assets/bellpepper.jpg',
    'assets/bear.jpg',
    'assets/hippo.jpg',
    'assets/lion.jpg',
    'assets/monkey.jpg',
    'assets/rhino.jpg',
    'assets/zebra.jpg',
    'assets/banana.jpg',
    'assets/lemon.jpg',
    'assets/apple.jpg',
    'assets/orange.jpg',
    'assets/pear.jpg',
    'assets/straberry.jpg',
    'assets/watermelon.jpg',
    'assets/grapes.jpg',

```



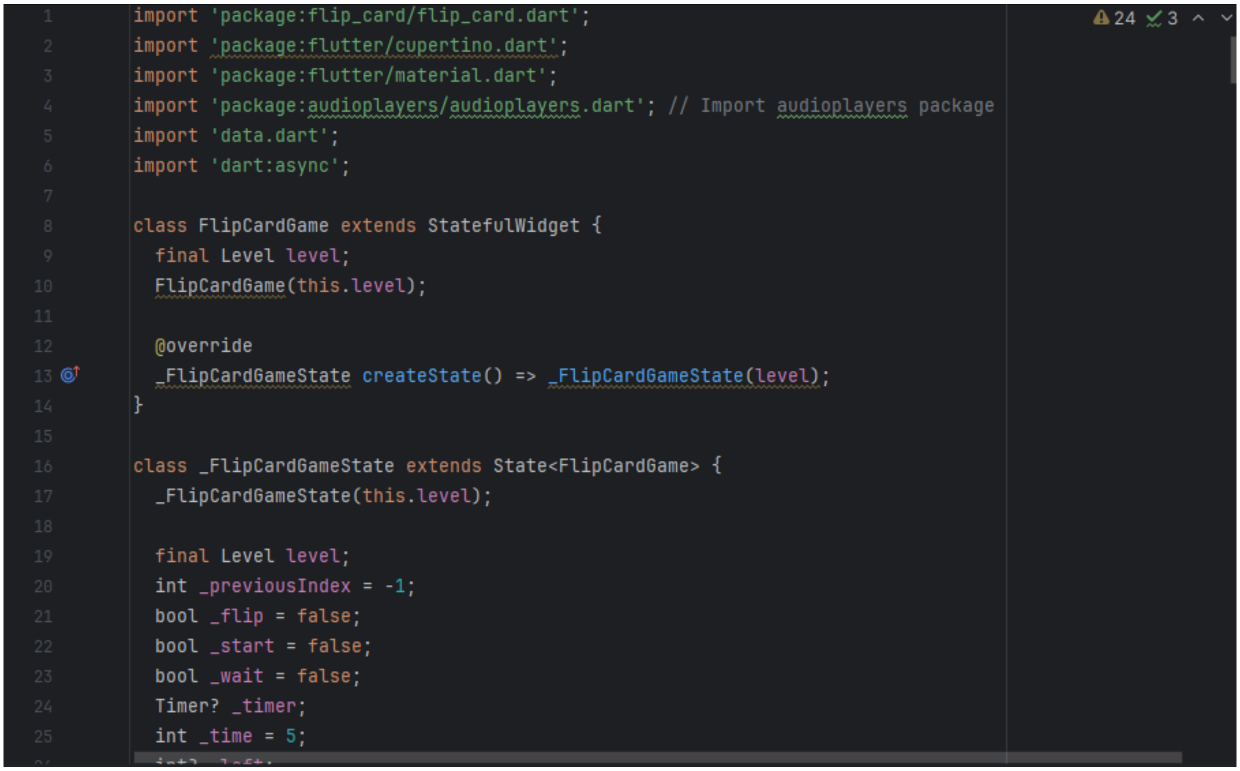
```
  ];
}

enum Level { Hard, Medium, Easy }

List getSourceArray(Level level) {
  List<String> levelList = [];
  List sourceArray = fillSourceArray();

  if (level == Level.Hard) {
    for (int i = 11; i < 19; i++) {
      levelList.add(sourceArray[i]);
    }
  } else if (level == Level.Medium) {
    for (int i = 5; i < 11; i++) {
      levelList.add(sourceArray[i]);
    }
  } else if (level == Level.Easy) {
    for (int i = 0; i < 4; i++) {
      levelList.add(sourceArray[i]);
    }
  }

  levelList.shuffle();
  return levelList;
```



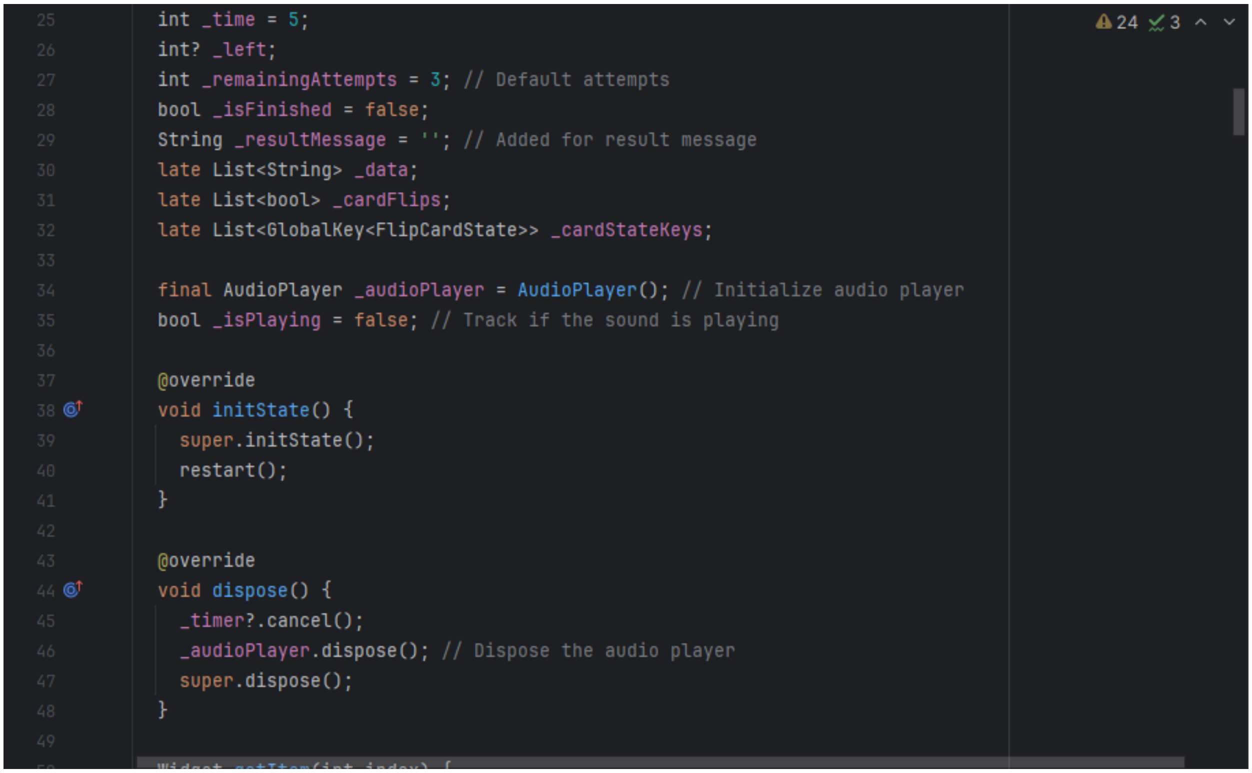
```
  levelList.shuffle();
  return levelList;
}

List<bool> getInitialItemState(Level level) {
  List<bool> initialItemState = [];

  if (level == Level.Hard) {
    for (int i = 0; i < 18; i++) {
      initialItemState.add(true);
    }
  } else if (level == Level.Medium) {
    for (int i = 0; i < 12; i++) {
      initialItemState.add(true);
    }
  } else if (level == Level.Easy) {
    for (int i = 0; i < 6; i++) {
      initialItemState.add(true);
    }
  }

  return initialItemState;
}

List<GlobalKey<FlipCardState>> getCardStateKeys(Level level) {
  List<GlobalKey<FlipCardState>> cardStateKeys = [];
```



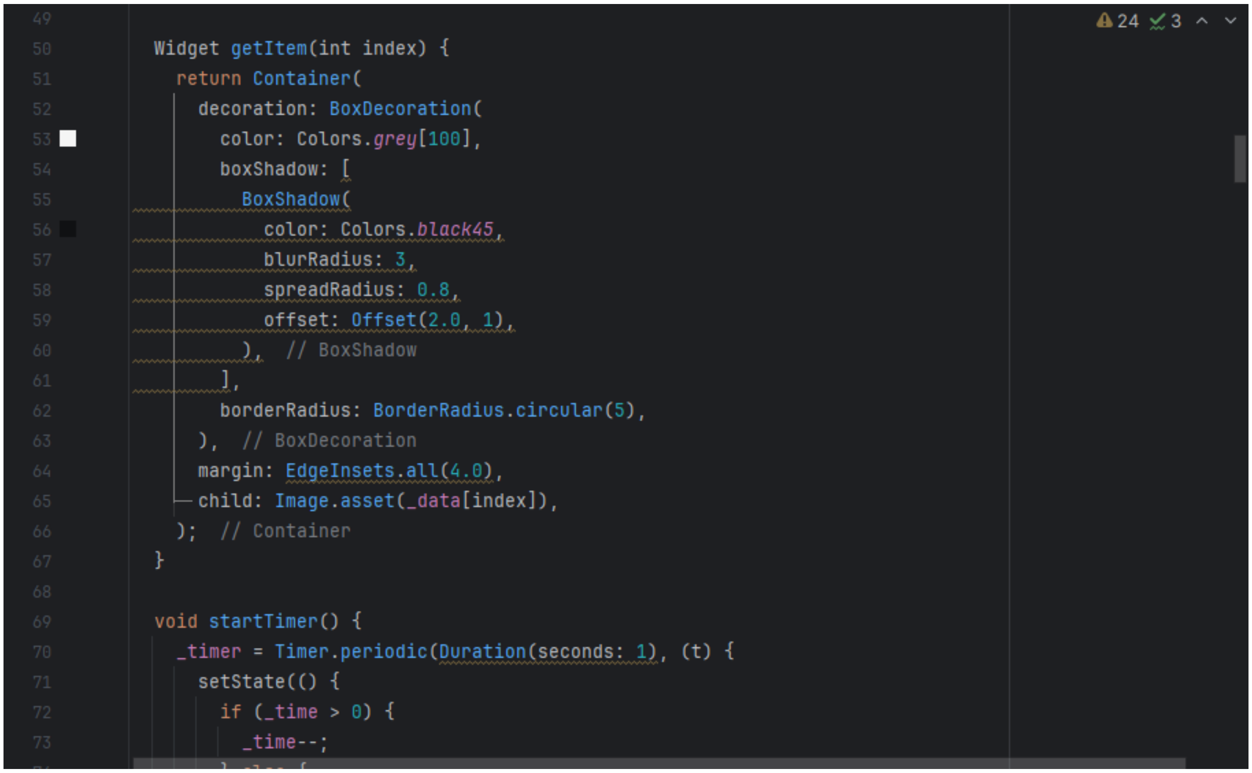
```
}

List<GlobalKey<FlipCardState>> getCardStateKeys(Level level) {
  List<GlobalKey<FlipCardState>> cardStateKeys = [];

  if (level == Level.Hard) {
    for (int i = 0; i < 18; i++) {
      cardStateKeys.add(GlobalKey<FlipCardState>());
    }
  } else if (level == Level.Medium) {
    for (int i = 0; i < 6; i++) {
      cardStateKeys.add(GlobalKey<FlipCardState>());
    }
  } else if (level == Level.Easy) {
    for (int i = 0; i < 6; i++) {
      cardStateKeys.add(GlobalKey<FlipCardState>());
    }
  }

  return cardStateKeys;
}
```



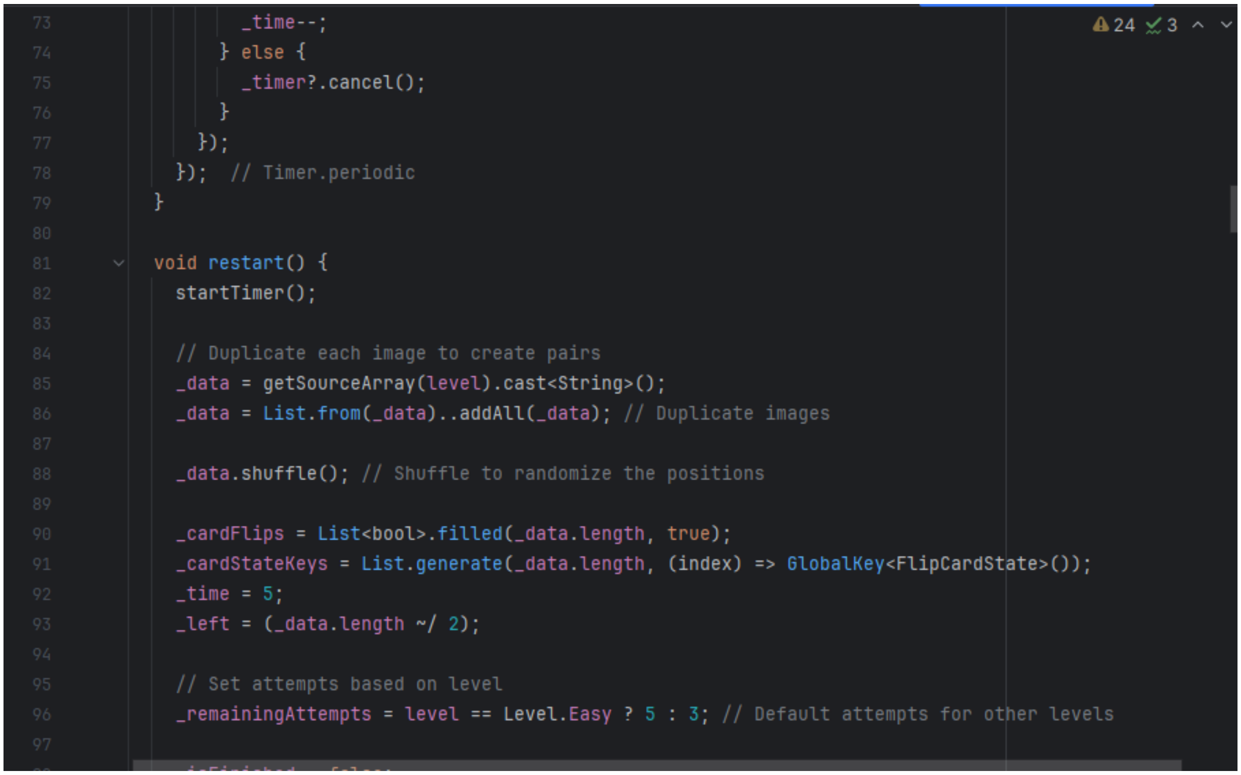
```
import 'package:flip_card/flip_card.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:audioplayers/audioplayers.dart'; // Import audioplayers package
import 'data.dart';
import 'dart:async';

class FlipCardGame extends StatefulWidget {
  final Level level;
  FlipCardGame(this.level);

  @override
  _FlipCardGameState createState() => _FlipCardGameState(level);
}

class _FlipCardGameState extends State<FlipCardGame> {
  _FlipCardGameState(this.level);

  final Level level;
  int _previousIndex = -1;
  bool _flip = false;
  bool _start = false;
  bool _wait = false;
  Timer? _timer;
  int _time = 5;
  int? _left;
```



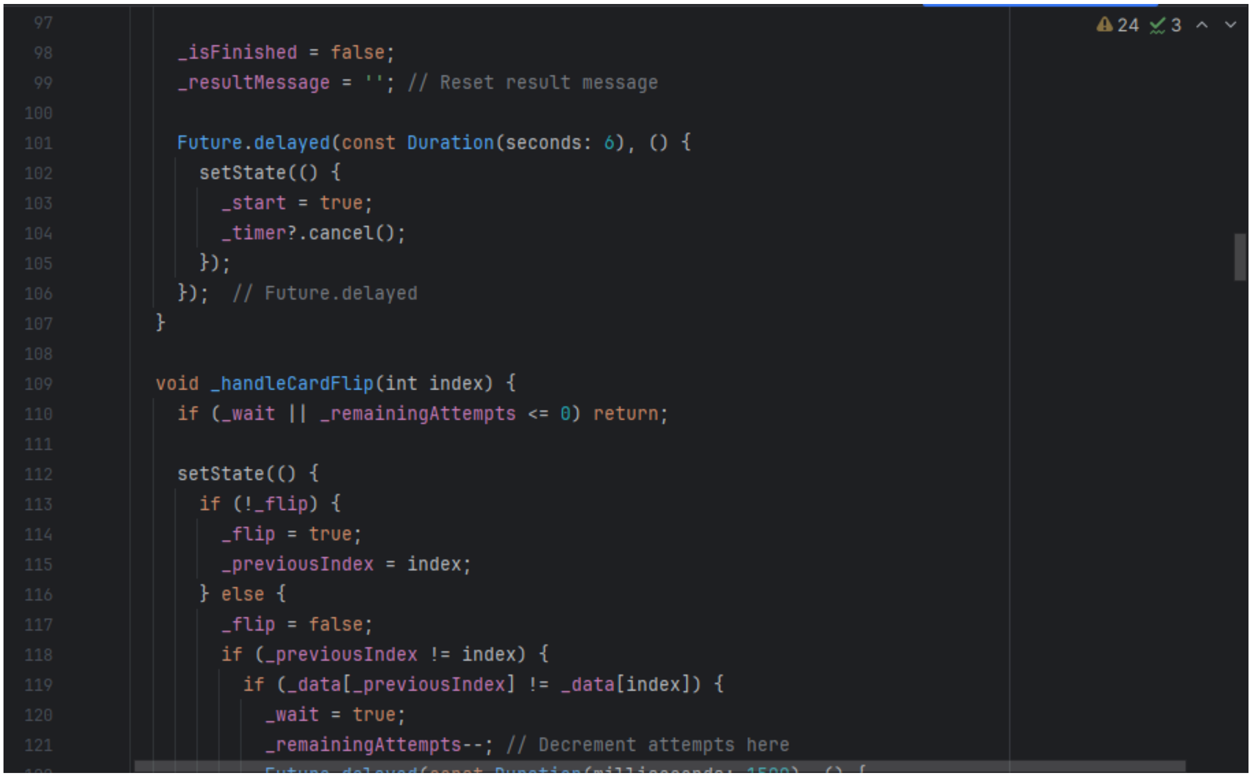
```
  int _time = 5;
  int? _left;
  int _remainingAttempts = 3; // Default attempts
  bool _isFinished = false;
  String _resultMessage = ''; // Added for result message
  late List<String> _data;
  late List<bool> _cardFlips;
  late List<GlobalKey<FlipCardState>> _cardStateKeys;

  final AudioPlayer _audioPlayer = AudioPlayer(); // Initialize audio player
  bool _isPlaying = false; // Track if the sound is playing

  @override
  void initState() {
    super.initState();
    restart();
  }

  @override
  void dispose() {
    _timer?.cancel();
    _audioPlayer.dispose(); // Dispose the audio player
    super.dispose();
  }

  Widget getItem(int index) {
```



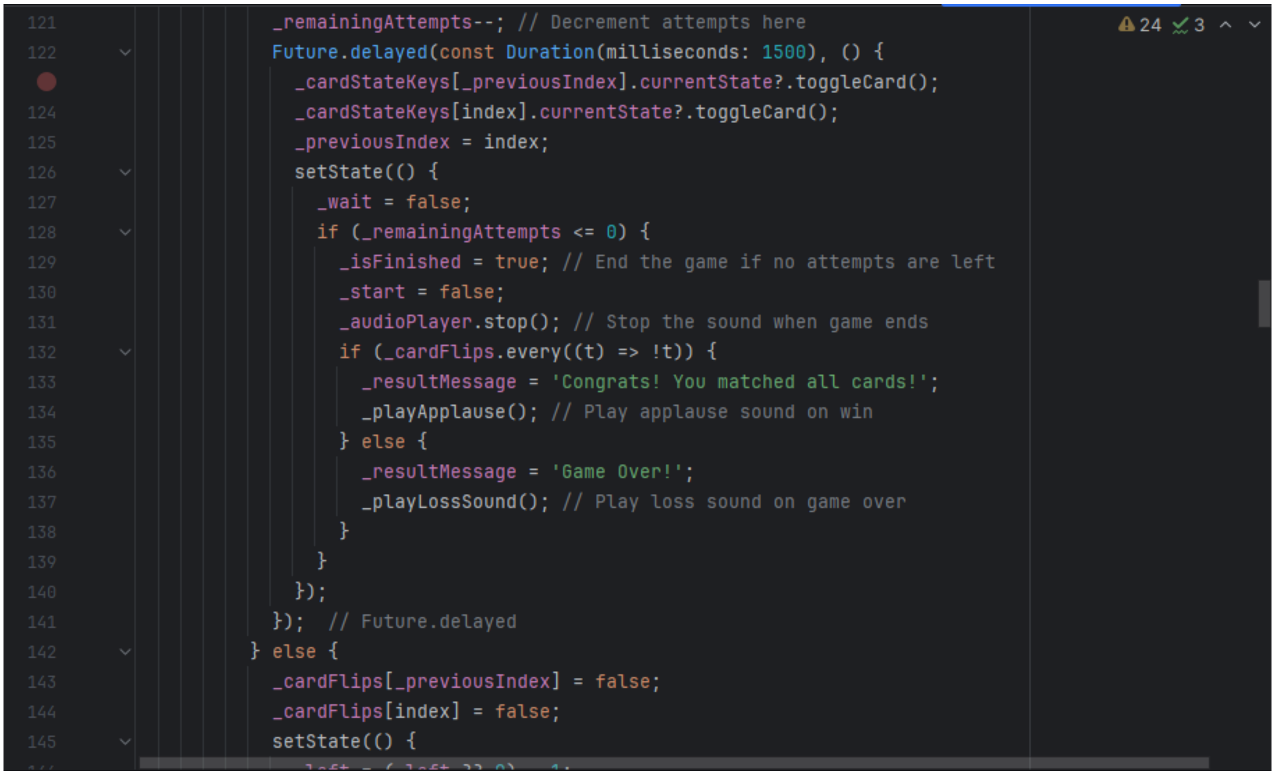
```
  Widget getItem(int index) {
    return Container(
      decoration: BoxDecoration(
        color: Colors.grey[100],
        boxShadow: [
          BoxShadow(
            color: Colors.black45,
            blurRadius: 3,
            spreadRadius: 0.8,
            offset: Offset(2.0, 1),
          ),  // BoxShadow
        ],
        borderRadius: BorderRadius.circular(5),
      ),  // BoxDecoration
      margin: EdgeInsets.all(4.0),
      child: Image.asset(_data[index]),
    );  // Container
  }

  void startTimer() {
    _timer = Timer.periodic(Duration(seconds: 1), (t) {
      setState(() {
        if (_time > 0) {
          _time--;
        } else {
```



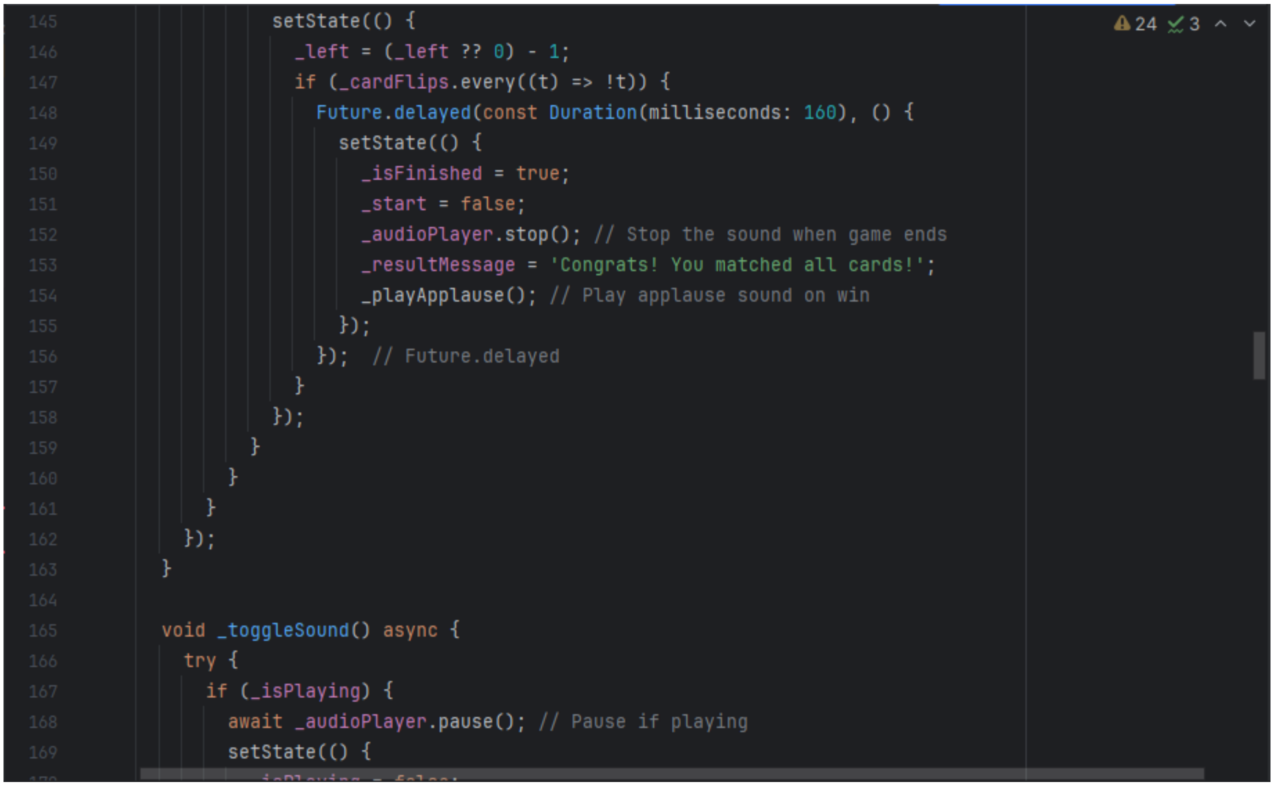
```
        _time--;
      } else {
        _timer?.cancel();
      }
    });
  });  // Timer.periodic
}

void restart() {
  startTimer();

  // Duplicate each image to create pairs
  _data = getSourceArray(level).cast<String>();
  _data = List.from(_data)..addAll(_data); // Duplicate images

  _data.shuffle(); // Shuffle to randomize the positions

  _cardFlips = List<bool>.filled(_data.length, true);
  _cardStateKeys = List.generate(_data.length, (index) => GlobalKey<FlipCardState>());
  _time = 5;
  _left = (_data.length ~/ 2);

  // Set attempts based on level
  _remainingAttempts = level == Level.Easy ? 5 : 3; // Default attempts for other levels

  _isFinished = false;
```



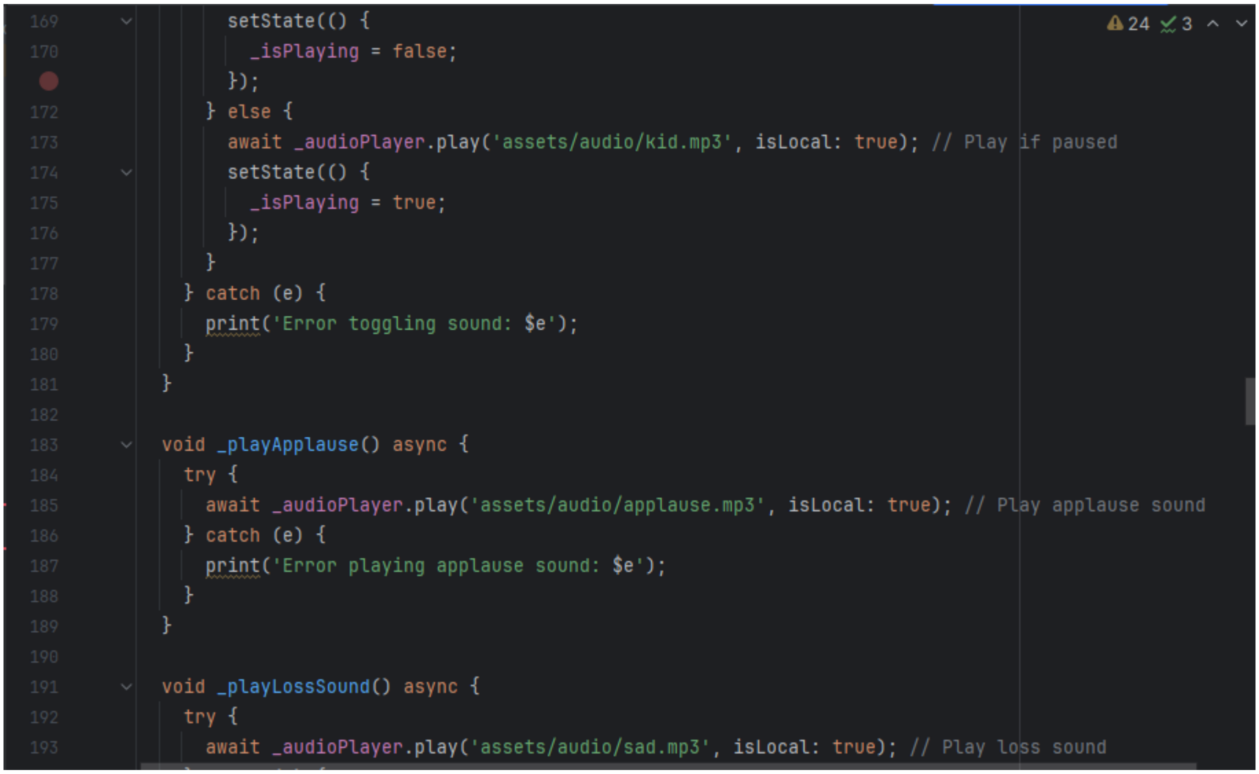
```

  _isFinished = false;
  _resultMessage = ''; // Reset result message

  Future.delayed(const Duration(seconds: 6), () {
    setState(() {
      _start = true;
      _timer?.cancel();
    });
  });  // Future.delayed
}

void _handleCardFlip(int index) {
  if (_wait || _remainingAttempts <= 0) return;

  setState(() {
    if (!_flip) {
      _flip = true;
      _previousIndex = index;
    } else {
      _flip = false;
      if (_previousIndex != index) {
        if (_data[_previousIndex] != _data[index]) {
          _wait = true;
          _remainingAttempts--; // Decrement attempts here
          Future.delayed(const Duration(milliseconds: 1500), () {
```



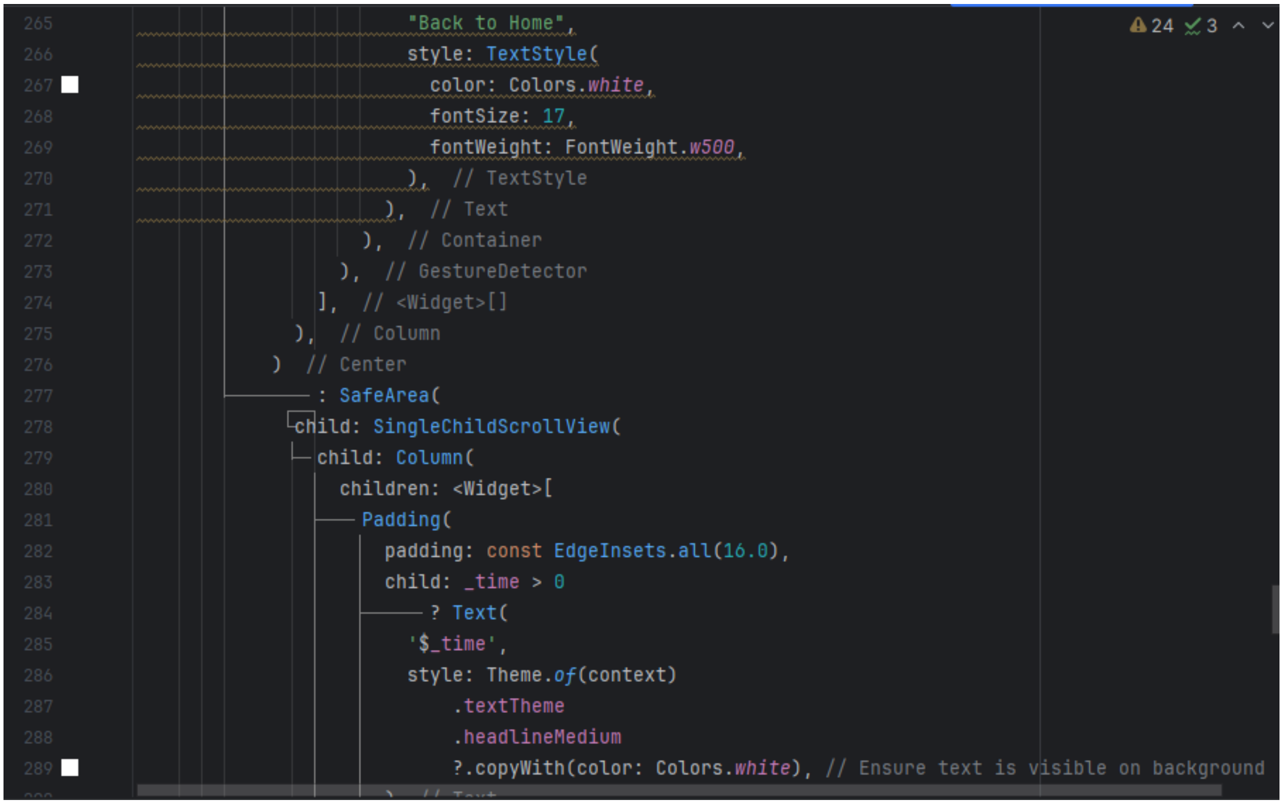
```
          _remainingAttempts--; // Decrement attempts here
          Future.delayed(const Duration(milliseconds: 1500), () {
            _cardStateKeys[_previousIndex].currentState?.toggleCard();
            _cardStateKeys[index].currentState?.toggleCard();
            _previousIndex = index;
            setState(() {
              _wait = false;
              if (_remainingAttempts <= 0) {
                _isFinished = true; // End the game if no attempts are left
                _start = false;
                _audioPlayer.stop(); // Stop the sound when game ends
                if (_cardFlips.every((t) => !t)) {
                  _resultMessage = 'Congrats! You matched all cards!';
                  _playApplause(); // Play applause sound on win
                } else {
                  _resultMessage = 'Game Over!';
                  _playLossSound(); // Play loss sound on game over
                }
              }
            });
          });  // Future.delayed
        } else {
          _cardFlips[_previousIndex] = false;
          _cardFlips[index] = false;
          setState(() {
```



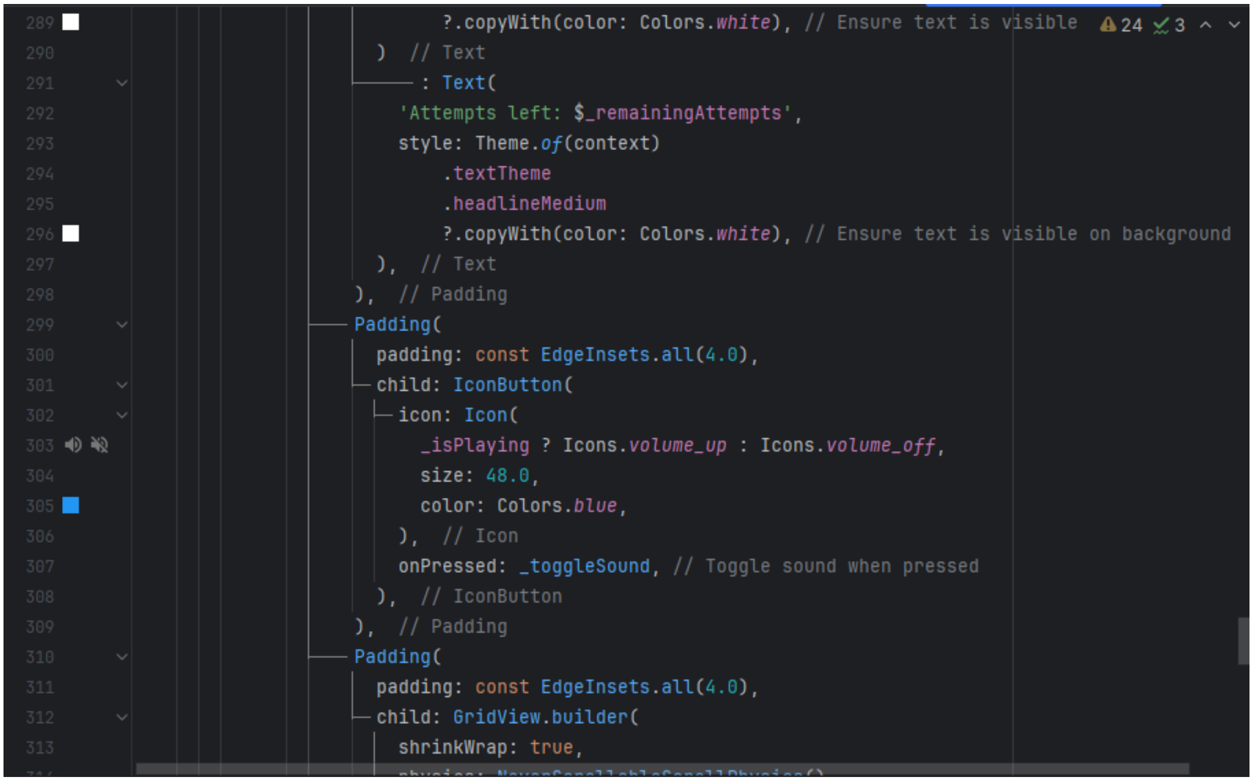
```
          setState(() {
            _left = (_left ?? 0) - 1;
            if (_cardFlips.every((t) => !t)) {
              Future.delayed(const Duration(milliseconds: 160), () {
                setState(() {
                  _isFinished = true;
                  _start = false;
                  _audioPlayer.stop(); // Stop the sound when game ends
                  _resultMessage = 'Congrats! You matched all cards!';
                  _playApplause(); // Play applause sound on win
                });
              });  // Future.delayed
            }
          });
        }
      }
    }
  });
}

void _toggleSound() async {
  try {
    if (_isPlaying) {
      await _audioPlayer.pause(); // Pause if playing
      setState(() {
```



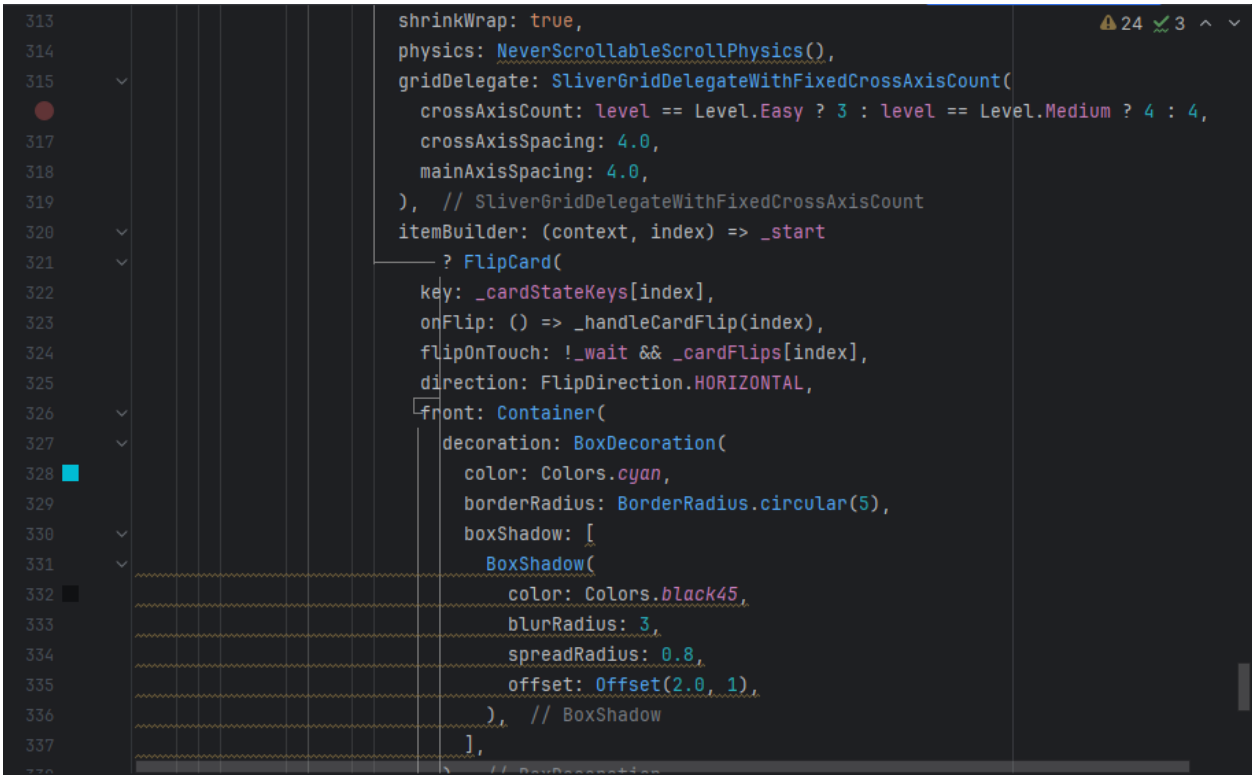
```
      setState(() {
        _isPlaying = false;
      });
    } else {
      await _audioPlayer.play('assets/audio/kid.mp3', isLocal: true); // Play if paused
      setState(() {
        _isPlaying = true;
      });
    }
  } catch (e) {
    print('Error toggling sound: $e');
  }
}

void _playApplause() async {
  try {
    await _audioPlayer.play('assets/audio/applause.mp3', isLocal: true); // Play applause sound
  } catch (e) {
    print('Error playing applause sound: $e');
  }
}

void _playLossSound() async {
  try {
    await _audioPlayer.play('assets/audio/sad.mp3', isLocal: true); // Play loss sound
```



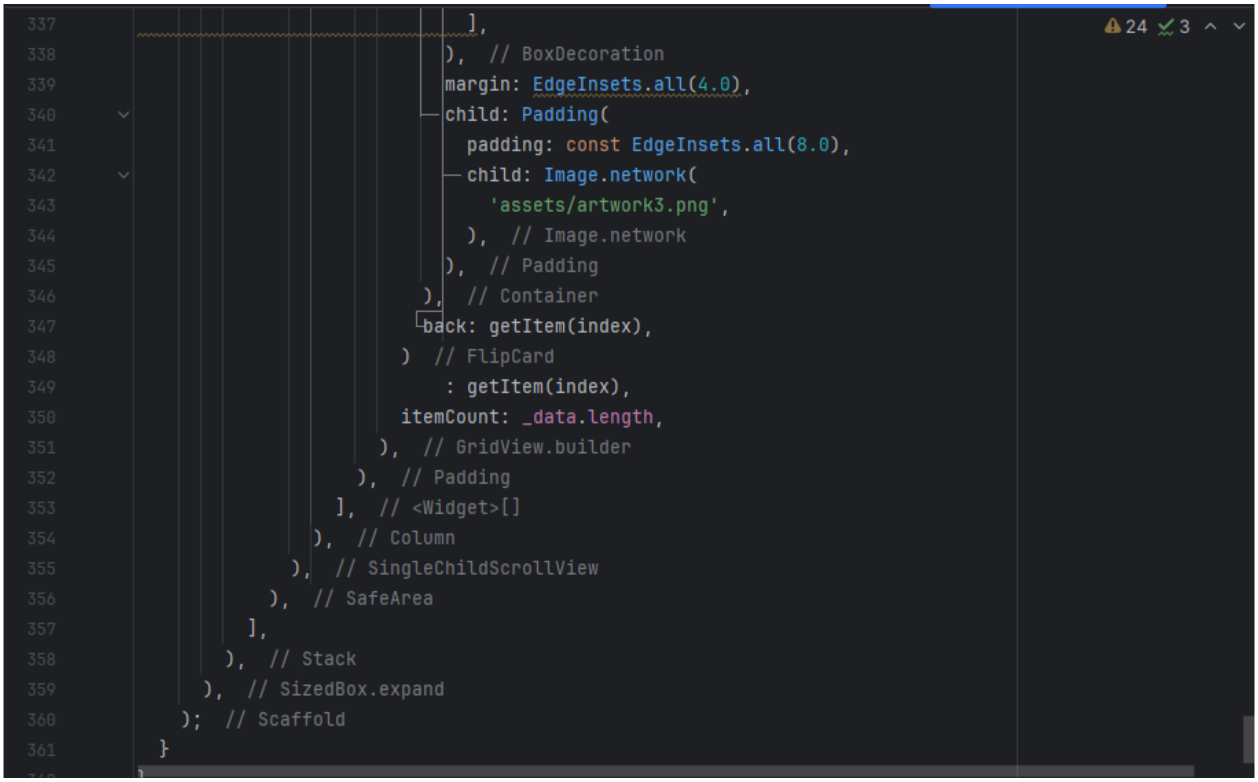
```
                        "Back to Home",
                        style: TextStyle(
                          color: Colors.white,
                          fontSize: 17,
                          fontWeight: FontWeight.w500,
                        ), // TextStyle
                      ), // Text
                    ), // Container
                  ), // GestureDetector
                ], // <Widget>[]
              ), // Column
            ) // Center
                : SafeArea(
              child: SingleChildScrollView(
                child: Column(
                  children: <Widget>[
                    Padding(
                      padding: const EdgeInsets.all(16.0),
                      child: _time > 0
                          ? Text(
                        '$_time',
                        style: Theme.of(context)
                            .textTheme
                            .headlineMedium
                            ?.copyWith(color: Colors.white), // Ensure text is visible on background
```



```
        ?.copyWith(color: Colors.white), // Ensure text is visible
  )  // Text
      : Text(
    'Attempts left: $_remainingAttempts',
    style: Theme.of(context)
        .textTheme
        .headlineMedium
        ?.copyWith(color: Colors.white), // Ensure text is visible on background
  ),  // Text
),  // Padding
Padding(
  padding: const EdgeInsets.all(4.0),
  child: IconButton(
    icon: Icon(
      _isPlaying ? Icons.volume_up : Icons.volume_off,
      size: 48.0,
      color: Colors.blue,
    ),  // Icon
    onPressed: _toggleSound, // Toggle sound when pressed
  ),  // IconButton
),  // Padding
Padding(
  padding: const EdgeInsets.all(4.0),
  child: GridView.builder(
    shrinkWrap: true,
    physics: NeverScrollableScrollPhysics(),
```



```
    shrinkWrap: true,
    physics: NeverScrollableScrollPhysics(),
    gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
      crossAxisCount: level == Level.Easy ? 3 : level == Level.Medium ? 4 : 4,
      crossAxisSpacing: 4.0,
      mainAxisSpacing: 4.0,
    ),  // SliverGridDelegateWithFixedCrossAxisCount
    itemBuilder: (context, index) => _start
        ? FlipCard(
      key: _cardStateKeys[index],
      onFlip: () => _handleCardFlip(index),
      flipOnTouch: !_wait && _cardFlips[index],
      direction: FlipDirection.HORIZONTAL,
      front: Container(
        decoration: BoxDecoration(
          color: Colors.cyan,
          borderRadius: BorderRadius.circular(5),
          boxShadow: [
            BoxShadow(
              color: Colors.black45,
              blurRadius: 3,
              spreadRadius: 0.8,
              offset: Offset(2.0, 1),
            ),  // BoxShadow
          ],
        ),  // BoxDecoration
```



```
                              ],
                            ),  // BoxDecoration
                            margin: EdgeInsets.all(4.0),
                            child: Padding(
                              padding: const EdgeInsets.all(8.0),
                              child: Image.network(
                                'assets/artwork3.png',
                              ),  // Image.network
                            ),  // Padding
                          ),  // Container
                          back: getItem(index),
                        )  // FlipCard
                            : getItem(index),
                        itemCount: _data.length,
                      ),  // GridView.builder
                    ),  // Padding
                  ],  // <Widget>[]
                ),  // Column
              ),  // SingleChildScrollView
            ),  // SafeArea
          ],
        ),  // Stack
      ),  // SizedBox.expand
    );  // Scaffold
  }
```