Software Tools 4

Assignment 7

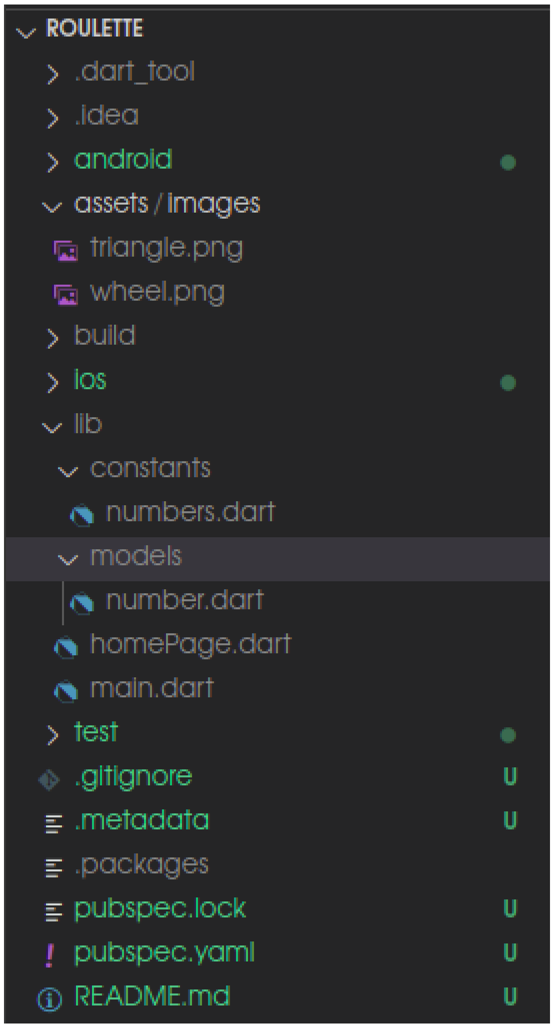
Krunal Rank
U18CO081

Develop a Roulette Game for Android.

Answer:

Tech Stack used :
Dart
Flutter SDK

Project Directory Structure:



```
ROULETTE
  > .dart_tool
  > .idea
  > android
  v assets / images
      triangle.png
      wheel.png
  > build
  > ios
  v lib
    v constants
        numbers.dart
    v models
        number.dart
      homePage.dart
      main.dart
  > test
    .gitignore        U
    .metadata         U
    .packages
    pubspec.lock      U
    pubspec.yaml      U
    README.md         U
```

Code:

./lib/main.dart:

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

import 'homePage.dart';

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

class MyApp extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     title: 'Casino Roulette',
     theme: ThemeData(
       primaryColor: Color(0xff3f51b5),
       accentColor: Color(0xff3f51b5),
       visualDensity: VisualDensity.adaptivePlatformDensity,
     ),
     home: HomePage(title: 'Casino Roulette'),
   );
 }
}
```

./lib/homePage.dart:

```dart
import 'dart:math';

import 'package:confetti/confetti.dart';
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:roulette/constants/numbers.dart';
import 'package:roulette/models/number.dart';

class HomePage extends StatefulWidget {
 HomePage({Key key, this.title}) : super(key: key);

 final String title;

 @override
 _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<HomePage>
   with SingleTickerProviderStateMixin {
 AnimationController _controller;
 ConfettiController _confettiController;
 double rotatedDegree = 0.0;
 bool wheelRotated = false;
 bool wheelRotating = false;
 double startValue = 0;
 double endValue = Random().nextDouble() + Random().nextInt(100);
 Number num = numbers[0];

 int gameState = 0;

 int investedAmount = 0;
 String investedAmountString = "";
 String choice = "Even Numbers";
 int earnings = 0;
 int roundEarning = 0;
 int totalTrials = 0;
 double profit = 0;
 int trialNumber = 0;
 int trialsLeft = 0;
 int fixedNumber = 0;
 String fixedNumberString = "";
 int amountUsed = 0;
```

```
@override
void initState() {
  _controller = AnimationController(
    duration: const Duration(milliseconds: 5000),
    vsync: this,
  );
  _confettiController = new ConfettiController(
    duration: new Duration(seconds: 4),
  );
  super.initState();
}

@override
void dispose() {
  _controller.dispose();
  super.dispose();
}

void _playRoulette() async {
  if (choice == "Fixed Number") {
    try {
      int val = int.parse(fixedNumberString);
      if (val < 0 || val > 36)
        throw 'Fixed Number not between 0 and 36 (Inclusive)';
      setState(() {
        fixedNumber = val;
      });
    } catch (error) {
      await showDialog<bool>(
          context: context,
          child: _alertDialog(
              message:
                  'Your Fixed Number should be between 0 and 36 (Inclusive).',
              title: 'Invalid Fixed Number'));
      return;
    }
  }
  if (fixedNumber < 0 || fixedNumber > 36) {}
  if (gameState != 1) {
    await showDialog<bool>(
        context: context,
        child: _alertDialog(
            message:
                'The Game has been reset! Please try again later after some time!',
            title: 'Something Went Wrong'));
      return;
    }
    final response = await showDialog<bool>(
      context: context,
      child: new AlertDialog(
        contentPadding: const EdgeInsets.all(16.0),
        title: Text('Roll Confirmation',
            style: TextStyle(color: Theme.of(context).primaryColor)),
        content: new Row(
          children: <Widget>[
            new Expanded(
                child: Text(
                    'Are you sure you want to roll? (This will deduct your Invested
Amount by INR 100.)',
                    style: TextStyle(color: Theme.of(context).primaryColor)))
          ],
        ),
        actions: <Widget>[
          new FlatButton(
              child: const Text('NO'),
              color: Theme.of(context).primaryColor,
              onPressed: () {
                Navigator.of(context).pop(false);
              }),
          new FlatButton(
              child: const Text('YES'),
              color: Theme.of(context).primaryColor,
              onPressed: () {
                Navigator.of(context).pop(true);
              })
        ],
      ),
    );
    if (response == null || !response) return;

    setState(() {
      startValue = endValue;
      endValue = endValue + Random().nextDouble() + Random().nextInt(10);
      wheelRotating = true;
    });
    _controller.reset();
    await _controller.forward();
    final eachSector = 1.0 / 37.0;
final deg = endValue - endValue.floor() + eachSector / 2.0;
dynamic idx = deg / eachSector;
idx = idx.floor() % 37;

setState(() {
  trialsLeft = trialsLeft - 1;
  trialNumber = trialNumber + 1;
  amountUsed = amountUsed + 100;
});
Number num1 = numbers[idx];
int val = num1.value;

if (choice == "Even Numbers" && val % 2 == 0) {
  setState(() {
    roundEarning = 100;
    earnings = earnings + roundEarning;
  });
} else if (choice == "Odd Numbers" && val % 2 == 1) {
  setState(() {
    roundEarning = 100;
    earnings = earnings + roundEarning;
  });
} else if (choice == "Prime Numbers" &&
    [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31].contains(val)) {
  setState(() {
    roundEarning = 500;
    earnings = earnings + roundEarning;
  });
} else if (choice == "Fixed Number" && fixedNumber == val) {
  setState(() {
    roundEarning = 5000;
    earnings = earnings + roundEarning;
  });
} else {
  setState(() {
    roundEarning = 0;
    earnings = earnings + roundEarning;
  });
}
setState(() {
  wheelRotated = true;
  wheelRotating = false;
  num = numbers[idx];
});
```

```dart
  if (roundEarning != 0) {
    _confettiController.play();
  }
  if (trialsLeft == 0) {
    double p = ((earnings - amountUsed) / (amountUsed)) * 100;
    setState(() {
      gameState = 2;
      profit = p;
    });
  }
}

void _startGame() async {
  final response = await showDialog<bool>(
    context: context,
    child: new AlertDialog(
      contentPadding: const EdgeInsets.all(16.0),
      title: Text('Start Roulette Game',
          style: TextStyle(color: Theme.of(context).primaryColor)),
      content: new Row(
        children: <Widget>[
          new Expanded(
            child: new TextField(
              autofocus: true,
              keyboardType: TextInputType.phone,
              decoration: new InputDecoration(
                  labelText: 'Investment (in INR)', hintText: 'Eg: 500'),
              onChanged: (e) {
                setState(() {
                  investedAmountString = e;
                });
              },
            ),
          )
        ],
      ),
      actions: <Widget>[
        new FlatButton(
            child: const Text('CANCEL'),
            color: Theme.of(context).primaryColor,
            onPressed: () {
              Navigator.of(context).pop(false);
            }),
        new FlatButton(
            child: const Text('START'),
            color: Theme.of(context).primaryColor,
            onPressed: () {
              Navigator.of(context).pop(true);
            })
      ],
    ),
  );

  if (response == null || !response) return;

  try {
    int val = int.parse(investedAmountString);
    if (val % 100 != 0) throw 'Amount not Integral multiple of INR 100';
    if (val == 0) throw 'Amount equal to 0';
    if (val < 0) throw 'Negative Amount';

    setState(() {
      investedAmount = val;
      totalTrials = val ~/ 100;
      trialNumber = 1;
      trialsLeft = totalTrials - trialNumber + 1;
      earnings = 0;
      roundEarning = 0;
      amountUsed = 0;
      gameState = 1;
      profit = 0;
      fixedNumber = 0;
      choice = 'Even Numbers';
      wheelRotated = false;
    });
  } catch (error) {
    await showDialog<bool>(
      context: context,
      child: _alertDialog(
          message:
              'Please enter valid Investment Amount in multiples of INR 100.',
          title: 'Invalid Investment Amount'),
    );
  }
}

void _endGame() async {
  final response = await showDialog<bool>(
    context: context,
    child: new AlertDialog(
      contentPadding: const EdgeInsets.all(16.0),
      title: Text('End Game',
          style: TextStyle(color: Theme.of(context).primaryColor)),
      content: new Row(
        children: <Widget>[
          new Expanded(
              child: Text('Are you sure you want to end the Game?',
                  style: TextStyle(color: Theme.of(context).primaryColor)))
        ],
      ),
      actions: <Widget>[
        new FlatButton(
            child: const Text('NO'),
            color: Theme.of(context).primaryColor,
            onPressed: () {
              Navigator.of(context).pop(false);
            }),
        new FlatButton(
            child: const Text('YES'),
            color: Theme.of(context).primaryColor,
            onPressed: () {
              Navigator.of(context).pop(true);
            })
      ],
    ),
  );
  if (response == null || !response) return;
  double p = ((earnings - amountUsed) / (amountUsed)) * 100;
  setState(() {
    gameState = 2;
    profit = p;
  });
}

@override
Widget build(BuildContext context) {
  final screenHeight = MediaQuery.of(context).size.height;
  final screenWidth = MediaQuery.of(context).size.width;
  return Scaffold(
    appBar: AppBar(
      title: Text(widget.title),
    ),
    body: ConfettiWidget(
        blastDirectionality: BlastDirectionality.explosive,
        confettiController: _confettiController,
        particleDrag: 0.05,
        emissionFrequency: 0.05,
        numberOfParticles: 25,
        gravity: 0.05,
        shouldLoop: false,
        colors: [
          Colors.green,
          Colors.red,
          Colors.yellow,
          Colors.blue,
        ],
        child: Center(
            child: ListView(
                scrollDirection: Axis.vertical,
                shrinkWrap: true,
                children: [
              Padding(
                padding: EdgeInsets.all(screenHeight * 0.01),
                child: Image(
                  image: AssetImage('assets/images/triangle.png'),
                  height: screenHeight * 0.05,
                ),
              ),
              Padding(
                padding: EdgeInsets.all(screenHeight * 0.01),
                child: RotationTransition(
                  turns: Tween(begin: startValue, end: endValue)
                      .animate(_controller),
                  child: Image(
                    image: AssetImage('assets/images/wheel.png'),
                    height: screenHeight * 0.4,
                  ),
                ),
              ),
              gameState == 1 || gameState == 2
                  ? Padding(
                      padding: EdgeInsets.all(screenHeight * 0.01),
                      child: Center(
                          child: Text(
                        wheelRotated
                            ? 'Number is ${num.value} !'
                    : 'Rotate the Wheel',
                style: TextStyle(
                    color: wheelRotated
                        ? num.color == 'Red'
                            ? Colors.red
                            : num.color == 'Green'
                                ? Colors.green
                                : Colors.black
                        : Theme.of(context).primaryColor,
                    fontWeight: FontWeight.bold,
                    fontSize: 28),
              )))
          : Container(width: 0, height: 0),
      wheelRotated
          ? Padding(
              padding: EdgeInsets.all(screenHeight * 0.01),
              child: Center(
                  child: Text(
                roundEarning == 100
                    ? 'Congratulations! You have won INR 100!'
                    : roundEarning == 500
                        ? 'Jackpot! You have won INR 500!'
                        : roundEarning == 5000
                            ? 'BingPot!!!! You have won INR 5000!'
                            : 'Sorry! Try again Next time!',
                textAlign: TextAlign.center,
                style: TextStyle(
                    color: Theme.of(context).primaryColor,
                    fontWeight: FontWeight.bold,
                    fontSize: 28),
              )))
          : Container(width: 0, height: 0),
      gameState == 1 && !wheelRotating
          ? Padding(
              padding: EdgeInsets.all(screenHeight * 0.01),
              child: DropdownButtonHideUnderline(
                  child: ButtonTheme(
                      alignedDropdown: true,
                      child: DropdownButton<String>(
                        hint: Text('Choice'),
                        style: TextStyle(
                            color: Theme.of(context).primaryColor,
                            fontSize: 18),
                        items: <String>[
          'Even Numbers',
          'Odd Numbers',
          'Prime Numbers',
          'Fixed Number'
         ].map((String value) {
          return new DropdownMenuItem<String>(
           value: value,
           child: new Text(value),
          );
         }).toList(),
         value: choice,
         onChanged: (e) {
          setState(() {
           choice = e;
          });
         },
        ))))
  : (Container(width: 0, height: 0)),
gameState == 1 && choice == 'Fixed Number' && !wheelRotating
  ? Padding(
    padding: EdgeInsets.all(screenHeight * 0.01),
    child: Center(
     child: new TextFormField(
      autofocus: true,
      keyboardType: TextInputType.phone,
      initialValue: fixedNumber.toString(),
      decoration: new InputDecoration(
        labelText: 'Number', hintText: 'Eg: 15'),
      onChanged: (e) {
       setState(() {
        fixedNumberString = e;
       });
      },
     ),
    ))
  : (Container(width: 0, height: 0)),
Padding(
  padding: EdgeInsets.all(screenHeight * 0.01),
  child: Center(
    child: Wrap(
   alignment: WrapAlignment.spaceEvenly,
   direction: Axis.horizontal,
   children: [
    gameState == 1
    ? _statistics(
        screenHeight: screenHeight,
        screenWidth: screenWidth,
        aMessage:
            'This is going to be your Trial No. ' +
                trialNumber.toString() +
                '.',
        aTitle: 'Trial Number',
        chipText: trialNumber.toString())
    : (Container(width: 0, height: 0)),
gameState == 1
    ? _statistics(
        screenHeight: screenHeight,
        screenWidth: screenWidth,
        aMessage: 'You have earned an earning of INR ' +
            roundEarning.toString() +
            '.',
        aTitle: 'Earnings',
        chipText: 'INR ' + roundEarning.toString())
    : (Container(width: 0, height: 0)),
gameState == 1
    ? _statistics(
        screenHeight: screenHeight,
        screenWidth: screenWidth,
        aMessage: 'You have a total of ' +
            trialsLeft.toString() +
            'Trials left.',
        aTitle: 'Trials Left',
        chipText: trialsLeft.toString())
    : (Container(width: 0, height: 0)),
gameState == 1
    ? _statistics(
        screenHeight: screenHeight,
        screenWidth: screenWidth,
        aMessage: 'You have a total of INR' +
            (trialsLeft * 100).toString() +
            'left.',
        aTitle: 'Remaining Amount',
        chipText:
            'INR ' + (trialsLeft * 100).toString())
    : (Container(width: 0, height: 0)),
gameState == 2
    ? _statistics(
        screenHeight: screenHeight,
                            screenWidth: screenWidth,
                            aMessage: 'You have invested a total of INR ' +
                                (amountUsed).toString() +
                                '.',
                            aTitle: 'Invested Amount',
                            chipText: 'INR ' + (amountUsed).toString())
                        : (Container(width: 0, height: 0)),
                    gameState == 2
                        ? _statistics(
                            screenHeight: screenHeight,
                            screenWidth: screenWidth,
                            aMessage:
                                'You have earned a total earning of INR ' +
                                    earnings.toString() +
                                    '.',
                            aTitle: 'Earnings',
                            chipText: 'INR ' + earnings.toString())
                        : (Container(width: 0, height: 0)),
                    gameState == 2
                        ? _statistics(
                            screenHeight: screenHeight,
                            screenWidth: screenWidth,
                            aMessage:
                                'Your Profit/Loss is estimated to be ' +
                                    profit.toString() +
                                    '%.',
                            aTitle: 'Earnings',
                            chipText: profit.toString() + ' %')
                        : (Container(width: 0, height: 0)),
                  ],
                ))),
            Container(height: 200)
          ]))),
  floatingActionButton: wheelRotating
      ? null
      : Row(children: [
          Spacer(),
          gameState == 0 || gameState == 2
              ? _startButton()
              : Container(height: 0, width: 0),
          gameState == 1 ? (_endButton()) : Container(height: 0, width: 0),
          gameState == 1 ? (_playButton()) : Container(height: 0, width: 0),
        ]), // This trailing comma makes auto-formatting nicer for build methods.
);
```

```
}

Widget _statistics(
    {double screenHeight,
    double screenWidth,
    String aMessage,
    String aTitle,
    String chipText}) {
  return Container(
      margin: EdgeInsets.all(screenWidth * 0.01),
      child: ElevatedButton(
          onPressed: () {
            showDialog<bool>(
              context: context,
              child: _alertDialog(message: aMessage, title: aTitle),
            );
          },
          style: ButtonStyle(
              backgroundColor: MaterialStateProperty.all<Color>(
                  Theme.of(context).primaryColor),
              shape: MaterialStateProperty.all<RoundedRectangleBorder>(
                  RoundedRectangleBorder(
                      borderRadius: BorderRadius.circular(18.0),
                      side: BorderSide(
                          color: Theme.of(context).primaryColor)))),
          child: Text(chipText, style: TextStyle(fontSize: 18))));
}

Widget _startButton() {
  return Padding(
      padding: EdgeInsets.all(10),
      child: FloatingActionButton(
        onPressed: _startGame,
        tooltip: 'Start Game',
        child: Icon(Icons.arrow_forward_ios),
      ));
}

Widget _endButton() {
  return Padding(
      padding: EdgeInsets.all(10),
      child: FloatingActionButton(
        onPressed: _endGame,
        tooltip: 'End Game',
        child: Icon(Icons.stop),
      ));
 }

 Widget _playButton() {
   return Padding(
       padding: EdgeInsets.all(10),
       child: FloatingActionButton(
         onPressed: _playRoulette,
         tooltip: 'Play',
         child: Icon(Icons.play_arrow),
       ));
 }

 Widget _alertDialog({String message, String title}) {
   return AlertDialog(
     contentPadding: const EdgeInsets.all(16.0),
     title:
         Text(title, style: TextStyle(color: Theme.of(context).primaryColor)),
     content: new Row(
       children: <Widget>[
         new Expanded(
             child: Text(message,
                 style: TextStyle(color: Theme.of(context).primaryColor)))
       ],
     ),
     actions: <Widget>[
       new FlatButton(
           child: const Text('OKAY'),
           color: Theme.of(context).primaryColor,
           onPressed: () {
             Navigator.pop(context);
           }),
     ],
   );
 }
}
```

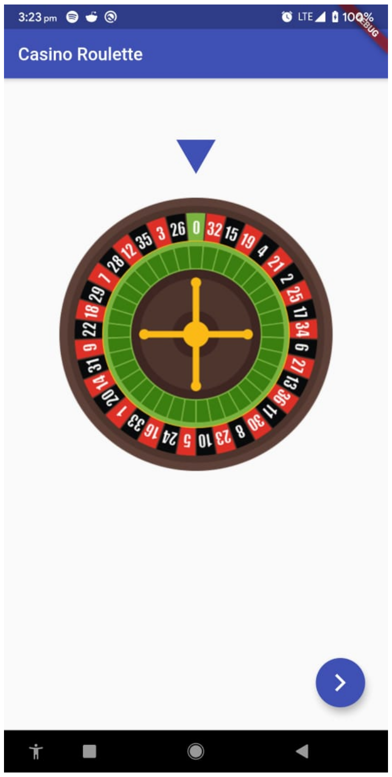
./lib/models/number.dart:

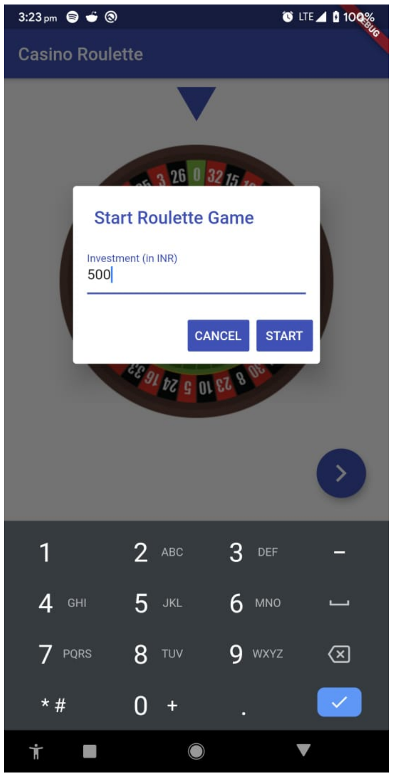
```dart
import 'package:flutter/material.dart';

class Number {
 int value;
 String color;
 Number({@required this.value, @required this.color});
}
```

./lib/constants/numbers.dart:

```dart
import 'package:roulette/models/number.dart';

List<Number> numbers = [
 Number(color: 'Green', value: 0),
 Number(color: 'Black', value: 26),
 Number(color: 'Red', value: 3),
 Number(color: 'Black', value: 35),
 Number(color: 'Red', value: 12),
 Number(color: 'Black', value: 28),
 Number(color: 'Red', value: 7),
 Number(color: 'Black', value: 29),
 Number(color: 'Red', value: 18),
 Number(color: 'Black', value: 22),
 Number(color: 'Red', value: 9),
 Number(color: 'Black', value: 31),
 Number(color: 'Red', value: 14),
 Number(color: 'Black', value: 20),
 Number(color: 'Red', value: 1),
 Number(color: 'Black', value: 33),
 Number(color: 'Red', value: 16),
 Number(color: 'Black', value: 24),
 Number(color: 'Red', value: 5),
 Number(color: 'Black', value: 10),
 Number(color: 'Red', value: 23),
 Number(color: 'Black', value: 8),
 Number(color: 'Red', value: 30),
 Number(color: 'Black', value: 11),
 Number(color: 'Red', value: 36),
 Number(color: 'Black', value: 13),
 Number(color: 'Red', value: 27),
 Number(color: 'Black', value: 6),
 Number(color: 'Red', value: 34),
 Number(color: 'Black', value: 17),
 Number(color: 'Red', value: 25),
 Number(color: 'Black', value: 2),
 Number(color: 'Red', value: 21),
 Number(color: 'Black', value: 4),
 Number(color: 'Red', value: 19),
 Number(color: 'Black', value: 15),
 Number(color: 'Red', value: 32),
];
```

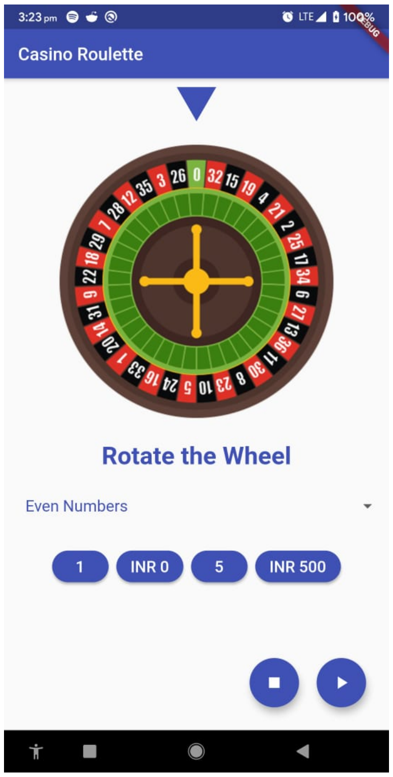
Screenshots:



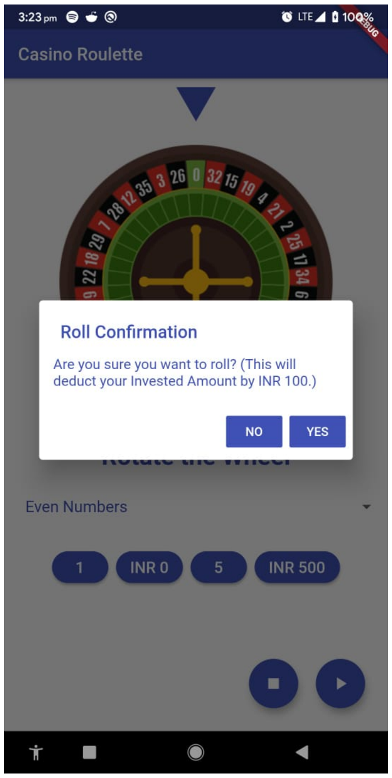


Start Roulette Game

Investment (in INR)
500

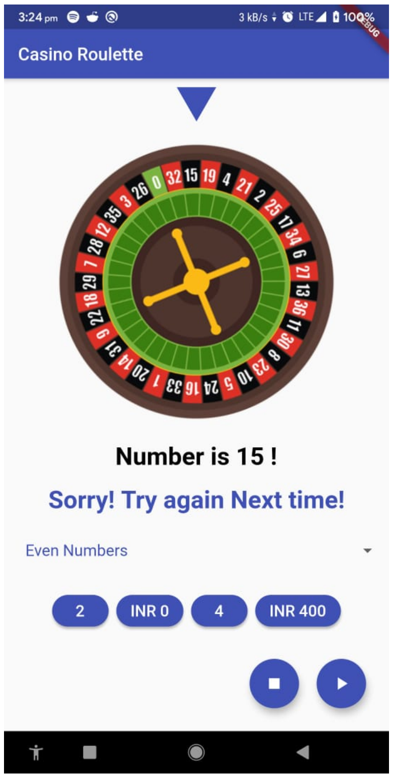


Rotate the Wheel



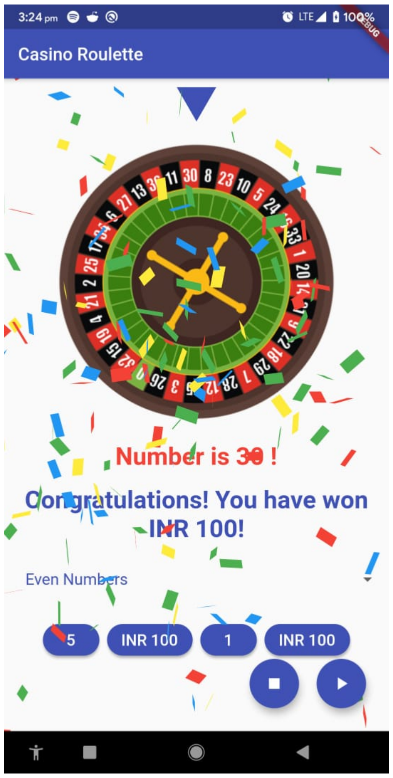
Roll Confirmation

Are you sure you want to roll? (This will deduct your Invested Amount by INR 100.)



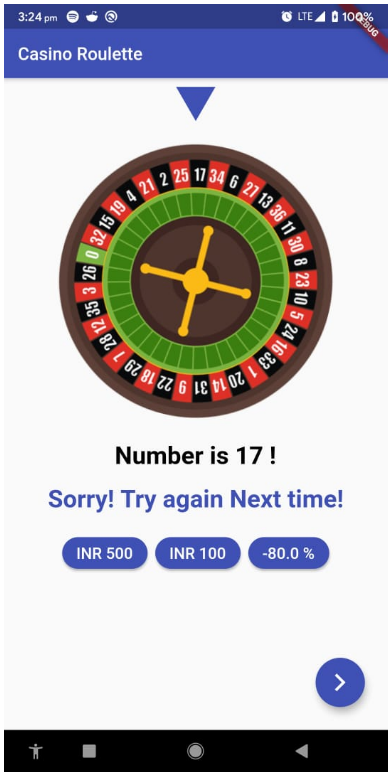
Number is 15 !

Sorry! Try again Next time!



Number is 30 !

Congratulations! You have won INR 100!



Number is 17 !

Sorry! Try again Next time!