Министерство образования Республики Беларусь
Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра программного обеспечения информационных технологий
Дисциплина: Разработка программного обеспечения для мобильных платформ

ОТЧЕТ

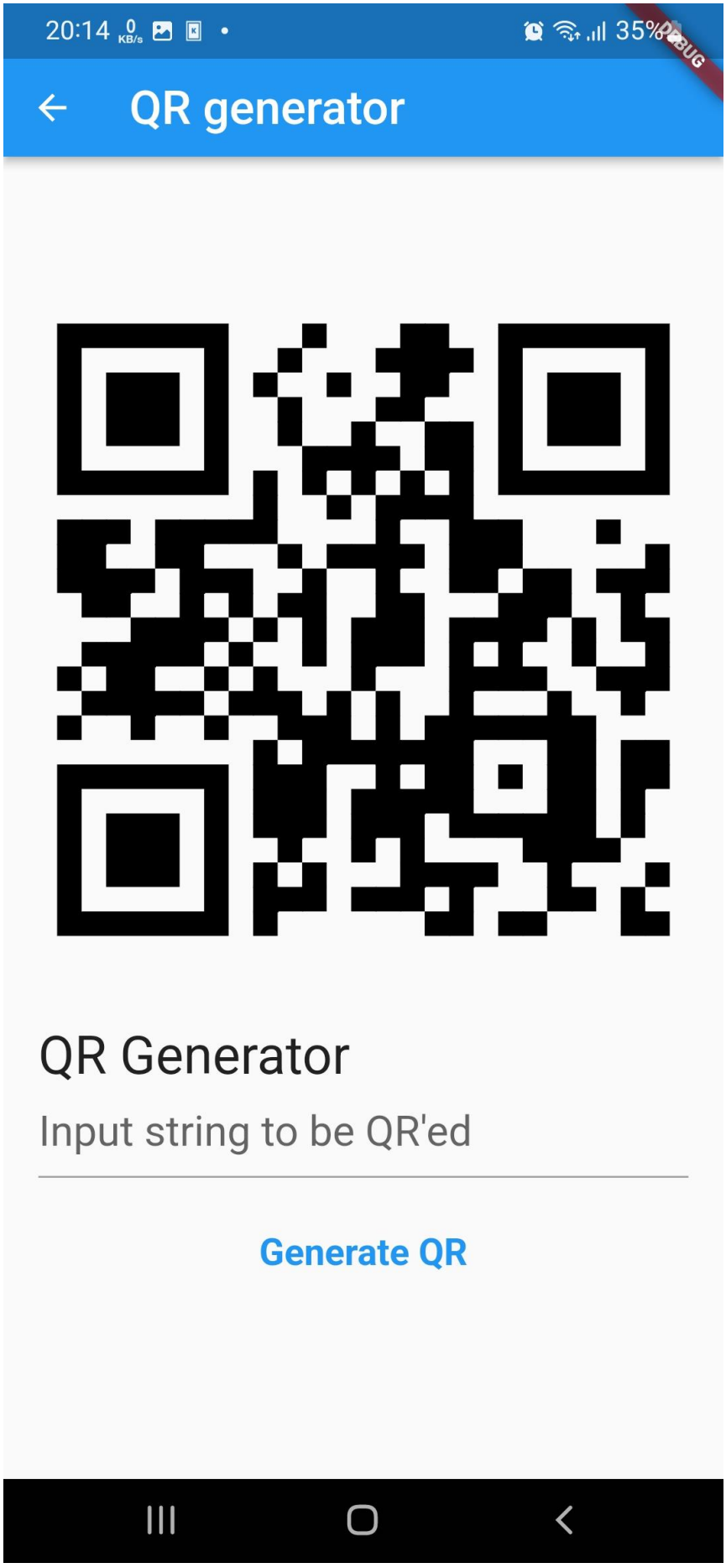
По лабораторной работе №3

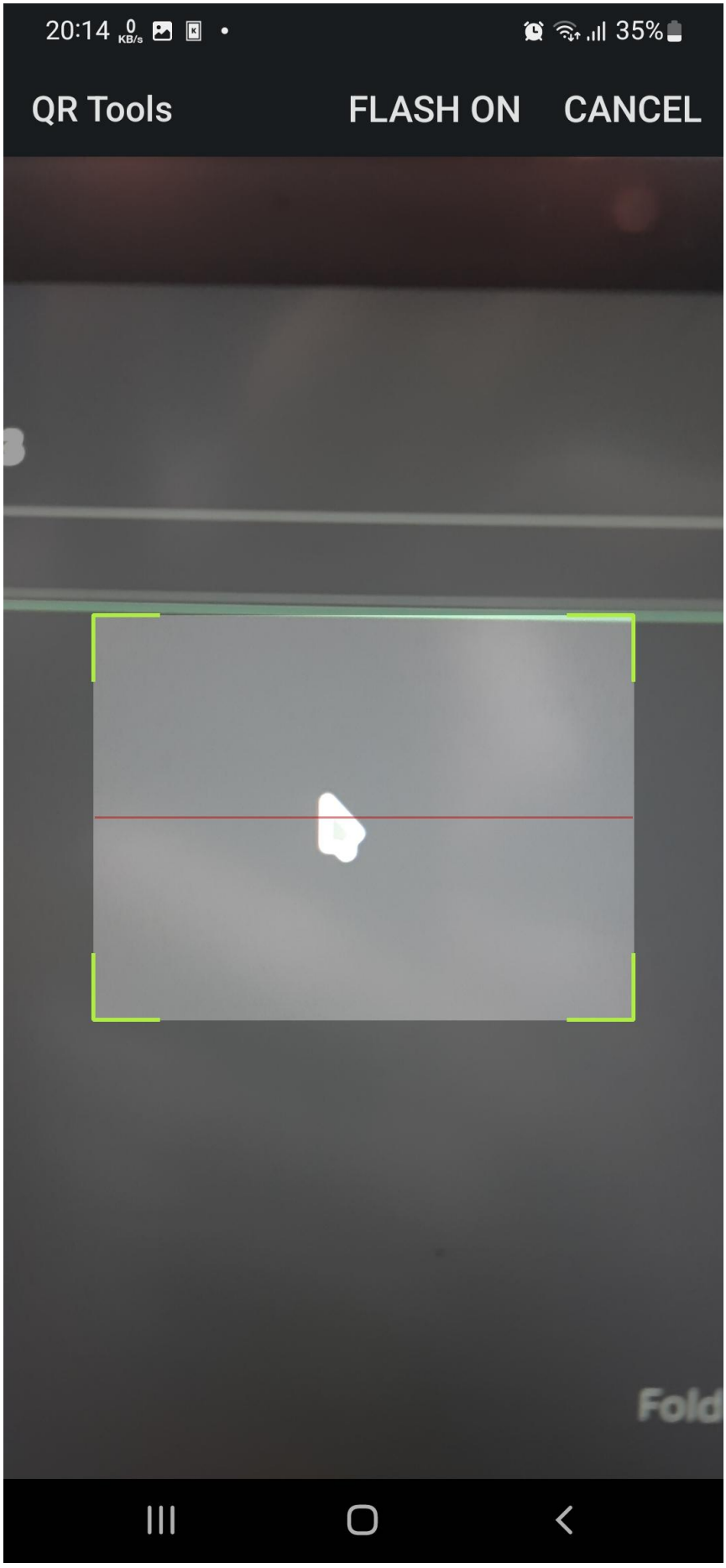
Тема работы: «QR-считыватель/ генератор»

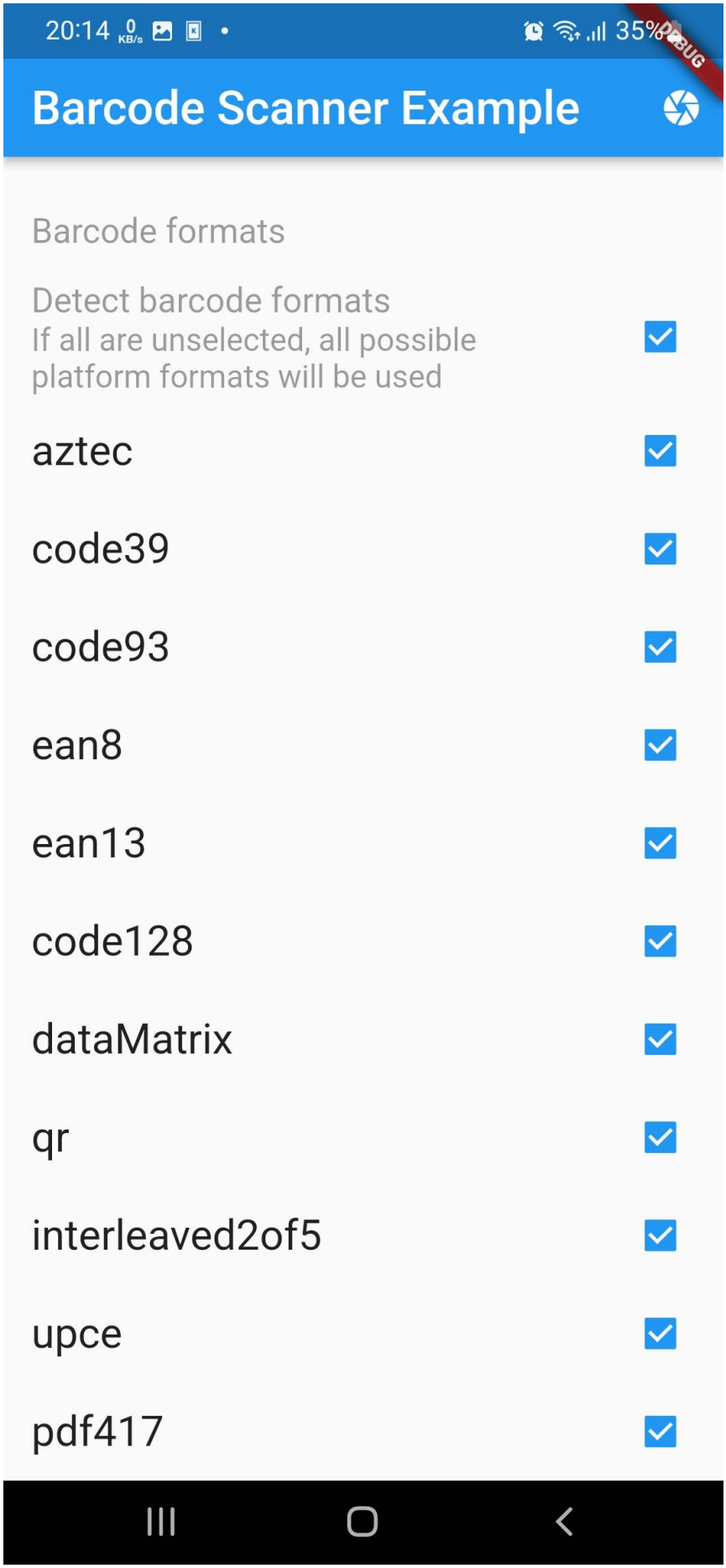
Выполнил:
студент: гр. 051006 Шуляк А.В.

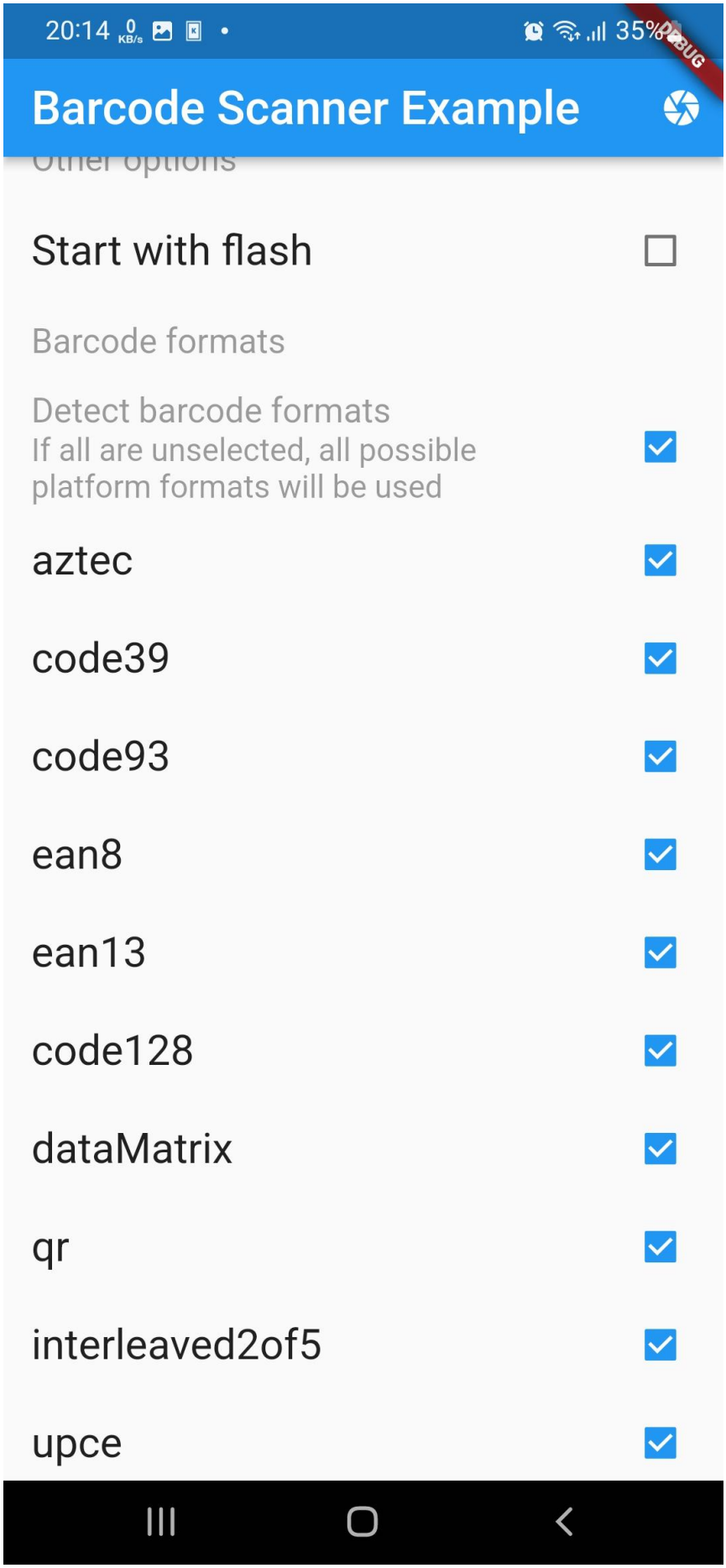
Проверил: Коловайтис Н. А.

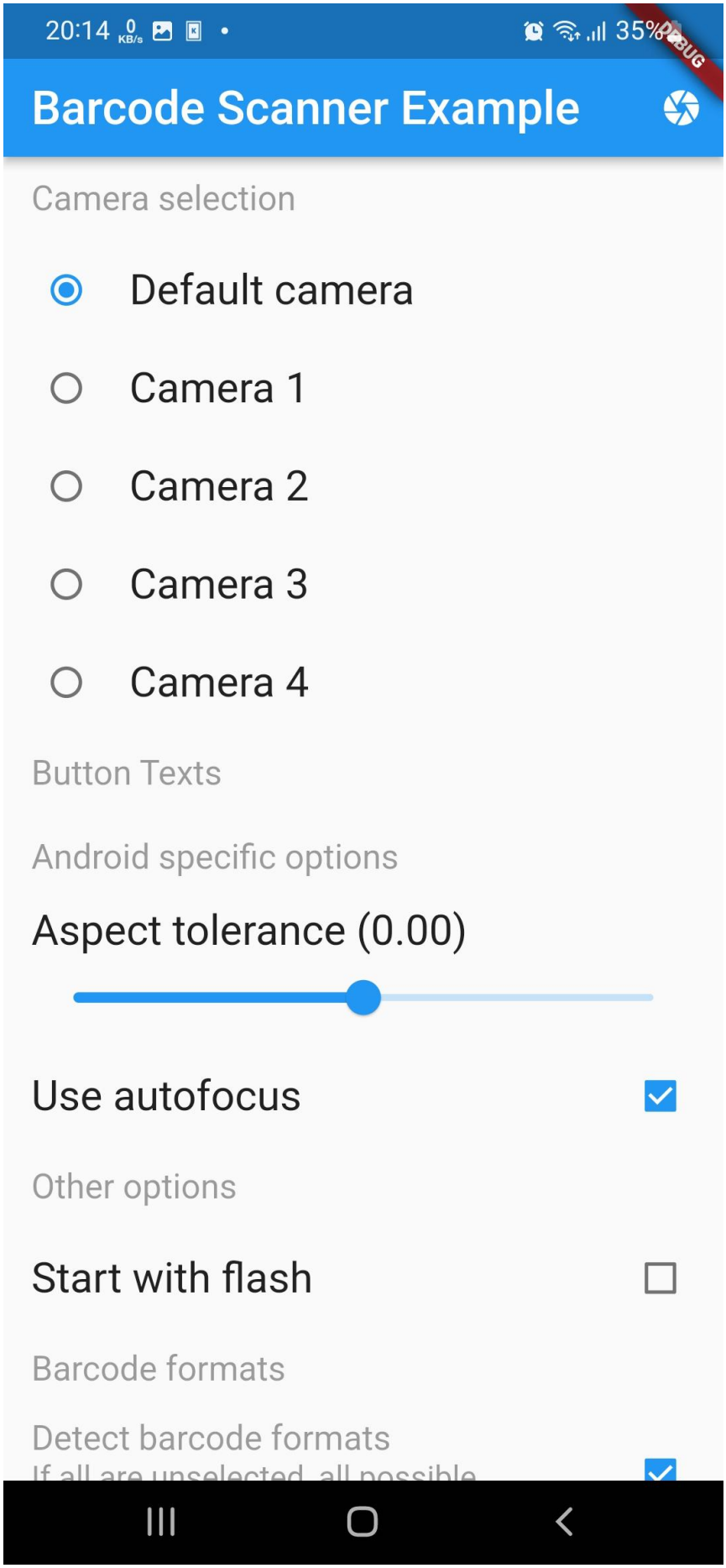
Минск 2023

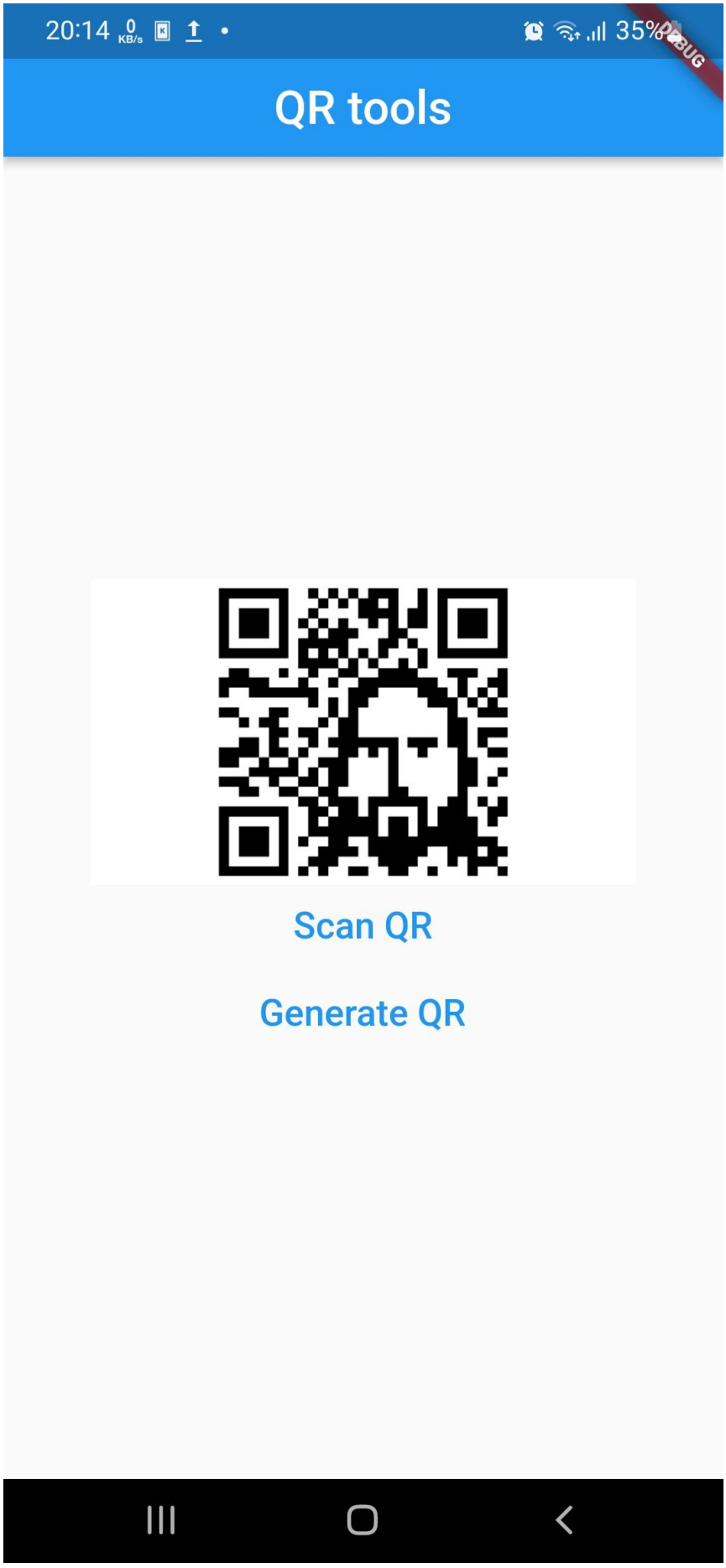

20:14
35%
DEBUG
QR generator
QR Generator
Input string to be QR'ed
Generate QR


20:14
35%
QR Tools
FLASH ON
CANCEL


20:14
35%
DEBUG
Barcode Scanner Example
Barcode formats
Detect barcode formats
If all are unselected, all possible platform formats will be used
aztec
code39
code93
ean8
ean13
code128
dataMatrix
qr
interleaved2of5
upce
pdf417


20:14
35%
DEBUG
Barcode Scanner Example
Other options
Start with flash
Barcode formats
Detect barcode formats
If all are unselected, all possible platform formats will be used
aztec
code39
code93
ean8
ean13
code128
dataMatrix
qr
interleaved2of5
upce


20:14
35%
DEBUG
Barcode Scanner Example
Camera selection
Default camera
Camera 1
Camera 2
Camera 3
Camera 4
Button Texts
Android specific options
Aspect tolerance (0.00)
Use autofocus
Other options
Start with flash
Barcode formats
Detect barcode formats


20:14
35%
DEBUG
QR tools
Scan QR
Generate QR

Исходный код:
main.dart:

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Home(),
    );
  }
}
```

home.dart:

```
import 'package:flutter/material.dart';
import 'package:qr_tools/generate.dart';
import 'package:qr_tools/scan.dart';

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

  @override
  State<Home> createState() => _HomeState();
}

class _HomeState extends State<Home> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
          title: const Text("QR tools"),
          centerTitle: true,
        ),
        body: Container(
            padding: const EdgeInsets.all(50.0),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              crossAxisAlignment: CrossAxisAlignment.stretch,
              children: <Widget>[
                const Image(
                    image: NetworkImage(
                        "http://www.rocketfarmstudios.com/wp-
content/uploads/2015/08/wallpaper_qr_code_jesus_by_existcze-d5krc2g.jpg")),
                textButton("Scan QR", ScanPage()),
                const SizedBox(
                  height: 2.0,
```

```
            ),
            textButton("Generate QR", GeneratePage()),
          ],
        )));
  }

  Widget textButton(String text, Widget widget) {
    return TextButton(
      onPressed: () {
        Navigator.of(context)
            .push(MaterialPageRoute(builder: ((context) => widget)));
      },
      child: Text(text),
    );
  }
}
```

scan.dart:

```
import 'dart:async';
import 'dart:developer';
import 'dart:io' show Platform;

import 'package:barcode_scan2/barcode_scan2.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';

import 'package:url_launcher/url_launcher.dart';

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

  @override
  State<ScanPage> createState() => _ScanPageState();
}

class _ScanPageState extends State<ScanPage> {
  ScanResult? scanResult;

  final _flashOnController = TextEditingController(text: 'Flash on');
  final _flashOffController = TextEditingController(text: 'Flash off');
  final _cancelController = TextEditingController(text: 'Cancel');

  var _aspectTolerance = 0.00;
  var _numberOfCameras = 0;
  var _selectedCamera = -1;
  var _useAutoFocus = true;
  var _autoEnableFlash = false;

  static final _possibleFormats = BarcodeFormat.values.toList()
    ..removeWhere((e) => e == BarcodeFormat.unknown);

  List<BarcodeFormat> selectedFormats = [..._possibleFormats];

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

    Future.delayed(Duration.zero, () async {
```

```
      _numberOfCameras = await BarcodeScanner.numberOfCameras;
      setState(() {});
    });
  }

  bool _isUrl(String text) {
    final urlRegex = RegExp(
        r'^((?:http|https):\/\/)?([a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*\.[a-zA-
Z]{2,})(?::\d{1,5})?(\/[^\s]*)?$');
    var res = urlRegex.hasMatch(text);
    log("url is $res");
    return res;
  }

  _launchUrl(String url) async {
    var uri = Uri.parse(url);
    if (_isUrl(url)) {
      await launchUrl(uri, mode: LaunchMode.externalApplication);
    }
  }

  void _copyToClipboard(String text) {
    Clipboard.setData(ClipboardData(text: text));
    ScaffoldMessenger.of(context).showSnackBar(const SnackBar(
      content: Text('Copied to clipboard'),
    ));
  }

  @override
  Widget build(BuildContext context) {
    final scanResult = this.scanResult;
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: const Text('Barcode Scanner Example'),
          actions: [
            IconButton(
              icon: const Icon(Icons.camera),
              tooltip: 'Scan',
              onPressed: _scan,
            )
          ],
        ),
        body: ListView(
          shrinkWrap: true,
          children: <Widget>[
            if (scanResult != null)
              Card(
                child: Column(
                  children: <Widget>[
                    ListTile(
                      title: const Text('Result Type'),
                      subtitle: Text(scanResult.type.toString()),
                    ),
                    ListTile(
                      title: const Text('Raw Content'),
                      // subtitle: Text(scanResult.rawContent),
                      subtitle: TextButton(
                        onPressed: () {
                          log("log: ${scanResult.rawContent}");
```

```
            _copyToClipboard(scanResult.rawContent);
            _launchUrl(scanResult.rawContent);
          },
          child: Text(scanResult.rawContent),
        ),
      ),
      ListTile(
        title: const Text('Format'),
        subtitle: Text(scanResult.format.toString()),
      ),
      ListTile(
        title: const Text('Format note'),
        subtitle: Text(scanResult.formatNote),
      ),
    ],
  ),
 ),
const ListTile(
 title: Text('Camera selection'),
 dense: true,
 enabled: false,
),
RadioListTile(
 onChanged: (v) => setState(() => _selectedCamera = -1),
 value: -1,
 title: const Text('Default camera'),
 groupValue: _selectedCamera,
),
...List.generate(
 _numberOfCameras,
 (i) => RadioListTile(
  onChanged: (v) => setState(() => _selectedCamera = i),
  value: i,
  title: Text('Camera ${i + 1}'),
  groupValue: _selectedCamera,
 ),
),
const ListTile(
 title: Text('Button Texts'),
 dense: true,
 enabled: false,
),
if (Platform.isAndroid) ...[
 const ListTile(
  title: Text('Android specific options'),
  dense: true,
  enabled: false,
 ),
 ListTile(
  title: Text(
   'Aspect tolerance (${_aspectTolerance.toStringAsFixed(2)})',
  ),
  subtitle: Slider(
   min: -1,
   value: _aspectTolerance,
   onChanged: (value) {
    setState(() {
     _aspectTolerance = value;
    });
   },
```

```
    ),
  ),
  CheckboxListTile(
    title: const Text('Use autofocus'),
    value: _useAutoFocus,
    onChanged: (checked) {
      setState(() {
        _useAutoFocus = checked!;
      });
    },
  ),
],
const ListTile(
  title: Text('Other options'),
  dense: true,
  enabled: false,
),
CheckboxListTile(
  title: const Text('Start with flash'),
  value: _autoEnableFlash,
  onChanged: (checked) {
    setState(() {
      _autoEnableFlash = checked!;
    });
  },
),
const ListTile(
  title: Text('Barcode formats'),
  dense: true,
  enabled: false,
),
ListTile(
  trailing: Checkbox(
    tristate: true,
    materialTapTargetSize: MaterialTapTargetSize.shrinkWrap,
    value: selectedFormats.length == _possibleFormats.length
        ? true
        : selectedFormats.isEmpty
            ? false
            : null,
    onChanged: (checked) {
      setState(() {
        selectedFormats = [
          if (checked ?? false) ..._possibleFormats,
        ];
      });
    },
  ),
  dense: true,
  enabled: false,
  title: const Text('Detect barcode formats'),
  subtitle: const Text(
    'If all are unselected, all possible '
    'platform formats will be used',
  ),
),
..._possibleFormats.map(
  (format) => CheckboxListTile(
    value: selectedFormats.contains(format),
    onChanged: (i) {
```

```
                setState(
                  () => selectedFormats.contains(format)
                      ? selectedFormats.remove(format)
                      : selectedFormats.add(format),
                );
              },
              title: Text(format.toString()),
            ),
          ),
        ],
      ),
    ),
  );
 }

 Future<void> _scan() async {
  try {
    final result = await BarcodeScanner.scan(
      options: ScanOptions(
        strings: {
          'cancel': _cancelController.text,
          'flash_on': _flashOnController.text,
          'flash_off': _flashOffController.text,
        },
        restrictFormat: selectedFormats,
        useCamera: _selectedCamera,
        autoEnableFlash: _autoEnableFlash,
        android: AndroidOptions(
          aspectTolerance: _aspectTolerance,
          useAutoFocus: _useAutoFocus,
        ),
      ),
    );
    setState(() => scanResult = result);
  } on PlatformException catch (e) {
    setState(() {
      scanResult = ScanResult(
        type: ResultType.Error,
        rawContent: e.code == BarcodeScanner.cameraAccessDenied
            ? 'The user did not grant the camera permission!'
            : 'Unknown error: $e',
      );
    });
  }
 }
}
```

generate.dart:

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

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

 @override
 State<GeneratePage> createState() => _GeneratePageState();
}
```

```
class _GeneratePageState extends State<GeneratePage> {
  String qrData = "https://youtu.be/dQw4w9WgXcQ";
  final qrInputController = TextEditingController();

  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
          title: const Text("QR generator"),
          actions: const <Widget>[],
        ),
        body: Container(
          padding: const EdgeInsets.all(20.0),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.stretch,
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              QrImage(data: qrData),
              const SizedBox(height: 40.0),
              const Text(
                "QR Generator",
                style: TextStyle(fontSize: 20.0),
              ),
              TextField(
                controller: qrInputController,
                decoration:
                    const InputDecoration(hintText: "Input string to be QR'ed"),
              ),
              Padding(
                padding: const EdgeInsets.all(20.0),
                child: TextButton(
                  onPressed: () async {
                    if (qrInputController.text.isEmpty) {
                      setState(() {
                        qrData = "";
                      });
                    } else {
                      setState(() {
                        qrData = qrInputController.text;
                      });
                    }
                  },
                  child: const Text(
                    "Generate QR",
                    style: TextStyle(
                        color: Colors.blue, fontWeight: FontWeight.bold),
                  ),
                ),
              )
            ],
          ),
        ));
  }
}
```