„Intelligence Camera”

Politechnika Śląska

Krystian Barczak, Krzysztof Dragon,

Marta Lewandowska, Szymon Babula

Wydział Matematyki Stosowanej

Kierunek Informatyka

VI semestr, PAM - Grupa 2C

Spis treści:

Część I – Narzędzia oraz źródła 3

1. Wykorzystane narzędzia 3
2. Wykorzystane źródła 3

Część II – Zagadnienia techniczne 3

1. Pliki odpowiedzialne za funkcjonalaność 3
2. Opis kluczowych funkcji 4
3. Wygląd aplikacji 7

Część I – Narzędzia oraz źródła

1. Wykorzystane narzędzia

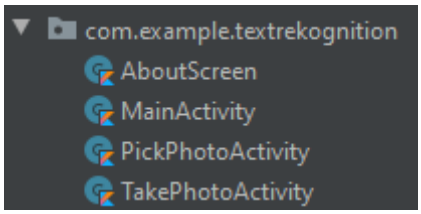
Projekt stworzony został za pomocą programu Android Studio z wykorzystaniem języka Kotlin. Pewna część grafiki wektorowej została wykonana w programie Inkscape Project oraz pobrana ze strony internetowej wpisanych w źródłach na licencji do użytku darmowego.

2. Wykorzystane źródła

- https://stackoverflow.com/
- https://www.pexels.com/pl-pl
- https://firebase.google.com/
- https://iconmonstr.com/

Część II – Zagadnienia techniczne

1. Pliki odpowiedzialne za funkcjonalaność



AboutScreen – Jest to aktywność, która służy do wyświetlenia informacji o aplikacji takich jak autorzy oraz do czego służy dana aplikacja.

MainActivity – Jest to aktywność, którą użytkownik aplikacji widzi, jako pierwszą po włączeniu. Na niej znajduje się informacja, aby po kliknięciu przejść dalej, a w dolnej części, w jakim celu została stworzona.

PickPhotoActivity – Aktywność, która umożliwia użytkownikowi wybrać zdjęcie ze swojego urządzenia w celu wyszukania tekstu zawartego w zdjęciu.

TakePhotoActivity – Aktywność podobna do opisanej poprzednio, lecz z takim wyjątkiem, aby przeskanować zdjęcie trzeba zrobić je wbudowanym w urządzenie aparatem.

2. Opis kluczowych funkcji

Activity Main

Głównym elementem jest przycisk, btnNext jest nasłuchiwany i odpowiada on za przejście między ekranem początkowym do aktywności odpowiedzialnej za detekcję tekstu z zdjęcia zrobionym wbudowanym w urządzenie aparatem. Za to przejście odpowiedzialna jest funkcja, takePhotoActivity().

```
class MainActivity : AppCompatActivity() {
        override fun onCreate(savedInstanceState: Bundle?) {
            super.onCreate(savedInstanceState)
            setContentView(R.layout.activity_main)

            val btnNext:Button = findViewById(R.id.button_next)
            btnNext.setOnClickListener { it: View!
                takePhotoActivity();
            }
        }

    override fun onCreateOptionsMenu(menu: Menu?): Boolean {
        super.onCreateOptionsMenu(menu)
        menuInflater.inflate(R.menu.menu,menu)
        return true
    }
    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        when(item.itemId){
            R.id.take_photo -> takePhotoActivity();
            R.id.choose_photo -> pickPhotoActivity();
            R.id.about -> aboutActivity();
        }
        return super.onOptionsItemSelected(item)
    }

    private fun takePhotoActivity(){
        val intent = Intent( packageContext: this,TakePhotoActivity::class.java)
        ContextCompat.startActivity( context: this, intent,  options: null)
    }

    private fun pickPhotoActivity(){
        val intent = Intent( packageContext: this,PickPhotoActivity::class.java)
        ContextCompat.startActivity( context: this, intent,  options: null)
    }

    private fun aboutActivity(){
        val intent = Intent( packageContext: this,AboutScreen::class.java)
        ContextCompat.startActivity( context: this, intent,  options: null)
    }
}
```

PickPhotoActivity

Img_pick_btn – Guzik ten odpowiedzialny jest za sprawdzenie uprawnień do pamięci urządzenia, których znajdują się zdjęcia oraz za uruchomienie funkcji dającej możliwość wybrania zdjęcia pamięci. W razie niepowodzenia przy sprawdzeniu uprawnień wyświetlony zostanie popup z żądaniem o przyznanie brakujących uprawnień w celu poprawnego działania aplikacji.

```
img_pick_btn.setOnClickListener { it: View!
    //check runtime permission
    if (VERSION.SDK_INT >= VERSION_CODES.M){
        if (checkSelfPermission(Manifest.permission.READ_EXTERNAL_STORAGE) ==
            PackageManager.PERMISSION_DENIED){
            //permission denied
            val permissions = arrayOf(Manifest.permission.READ_EXTERNAL_STORAGE);
            //show popup to request runtime permission
            requestPermissions(permissions, PERMISSION_CODE);
        }
        else{
            //permission already granted
            pickImageFromGallery();
        }
    }
    else{
        //system OS is < Marshmallow
        pickImageFromGallery();
    }
}
```

detectTextFromImage – funkcja sprawdzająca czy na zdjęciu znajduje się tekst, który później wyświetlany jest za pomocą funkcji displayTextFromImage na ekranie w odpowiednim miejscu.

W razie nie znalezienia tekstu wyświetlony zostanie komunikat w postaci Toast na ekranie urządzenia.

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if (resultCode == Activity.RESULT_OK && requestCode == IMAGE_PICK_CODE){
        image_view.setImageURI(data?.data)
        imageBitmap = BitmapFactory.decodeStream(contentResolver.openInputStream(data?.data as Uri));
    }
    detectTextFromImage();
}
private fun detectTextFromImage() {
    text_display.text = ""
    val image = FirebaseVisionImage.fromBitmap(imageBitmap)
    val detector = FirebaseVision.getInstance().onDeviceTextRecognizer
    val result = detector.processImage(image)
    result.addOnSuccessListener { firebaseVisionText ->

        displayTextFromImage(firebaseVisionText)
    }
        .addOnFailureListener { it: Exception
            Toast.makeText( context: this, text: "No text to detect", Toast.LENGTH_SHORT).show()
        }

}

private fun displayTextFromImage(resultText: FirebaseVisionText) {
    if (resultText.textBlocks.size == 0) {
        text_display.text = "No Text Found"
        return
    }
    for (block in resultText.textBlocks) {
        val blockText = block.text
        text_display.append(blockText + "\n")
    }
}
```

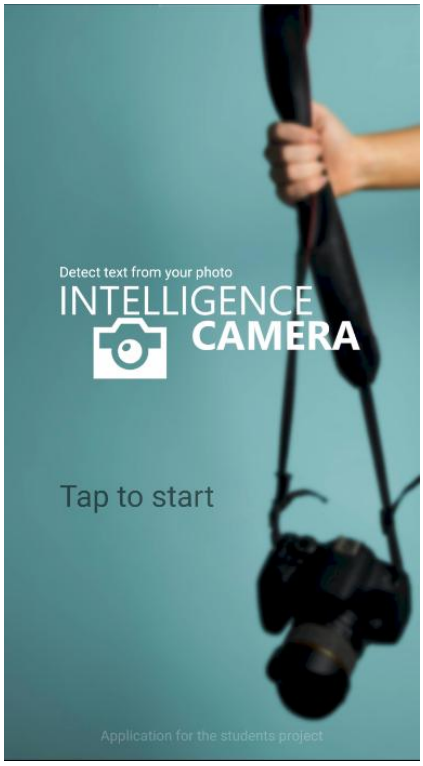
TakePhotoActivity

Podobnie jak w poprzedniej aktywności tutaj tekst sprawdzany jest dokładnie w taki sam sposób, lecz wyjątkiem jest sposób ładowania zdjęcia, jakim jest zrobienie zdjęcia aparatem i przesłaniem go do aplikacji

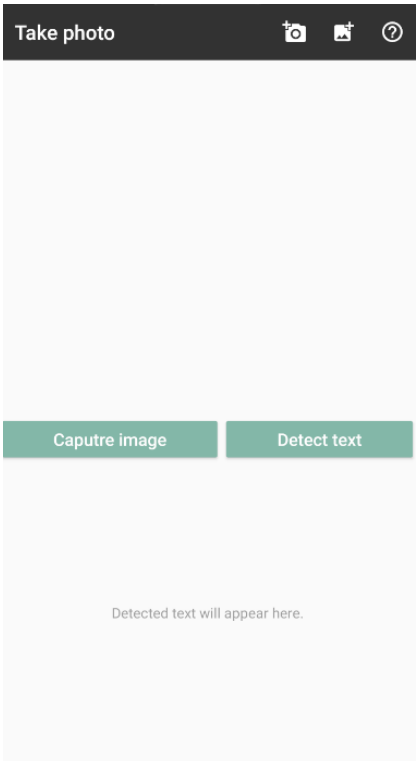
```
private fun dispatchTakePictureIntent() {
    Intent(MediaStore.ACTION_IMAGE_CAPTURE).also { takePictureIntent ->
        takePictureIntent.resolveActivity(packageManager)?.also { it: ComponentName
            startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE)
        }
    }
}

override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK) {
        imageBitmap = data?.extras?.get("data") as Bitmap
        imageView.setImageBitmap(imageBitmap)
    }
}

private fun detectTextFromImage() {
    if(this::imageBitmap.isInitialized) {
        text_display.text = ""
        val image = FirebaseVisionImage.fromBitmap(imageBitmap!!)
        val detector = FirebaseVision.getInstance().onDeviceTextRecognizer
        val result = detector.processImage(image)
        result.addOnSuccessListener { firebaseVisionText ->
            displayTextFromImage(firebaseVisionText)

        }
            .addOnFailureListener { it: Exception
                Toast.makeText( context: this,  text: "No text to detect", Toast.LENGTH_SHORT).show()
            }
    } else {
        Toast.makeText( context: this,  text: "First take a photo", Toast.LENGTH_SHORT).show()
    }
}

private fun displayTextFromImage(resultText: FirebaseVisionText) {
    if (resultText.textBlocks.size == 0) {
        textView.setText("No Text Found")
        return
    }
    for (block in resultText.textBlocks) {
        val blockText = block.text
        textView.append(blockText + "\n")
    }
}
```

3. Wygląd aplikacji

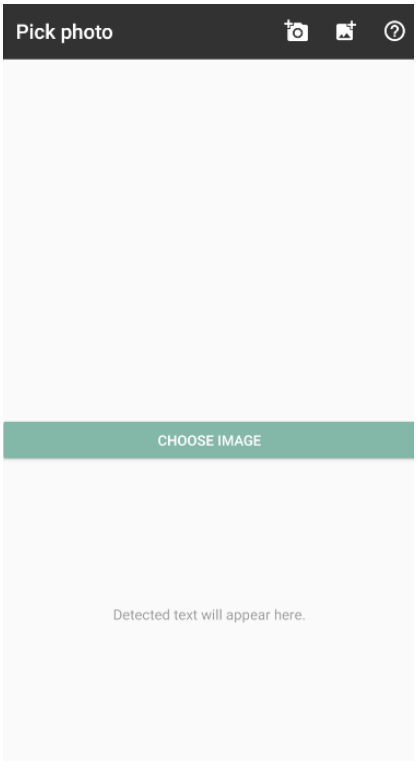
Ekran główny



Aktywność rozpoznająca po zrobieniu zdjęcia



Aktywność po wybraniu zdjęcia z pamięci



Aktywność o aplikacji