

**SVEUČILIŠTE U SPLITU
FAKULTET ELEKTROTEHNIKE, STROJARSTVA I
BRODOGRADNJE**

ZAVRŠNI RAD

**MOBILNA APLIKACIJA ZA OPTIČKO
PREPOZNAVANJE ZNAKOVA**

Igor Ujević

Split, srpanj 2019.



SVEUČILIŠTE U SPLITU
FAKULTET ELEKTROTEHNIKE, STROJARSTVA I
BRODOGRADNJE



Preddiplomski sveučilišni studij: **Računarstvo**

Smjer/Usmjerenje: /

Oznaka programa: 120

Akadska godina: 2018./2019.

Ime i prezime: **Igor Ujević**

Broj indeksa: 229-2016

ZADATAK ZAVRŠNOG RADA

Naslov: **Mobilna aplikacija za optičko prepoznavanje znakova**

Zadatak: Objasniti postupak izrade mobilne aplikacije za Android OS, te dati pregled relevantnih tehnologija. Opisati zahtjeve aplikacije koja bi primjenjivala algoritam optičkog prepoznavanja znakova na ekstrakciju znakova sa slike dohvaćene mobilnim uređajem. Opisati postupak implementacije, te testirati aplikaciju.

Datum obrane: 15.07.2019.

Mentor:

doc. dr. sc. Ana Kuzmanić Skelin

IZJAVA

Ovom izjavom potvrđujem da sam završni rad s naslovom „Mobilna aplikacija za optičko prepoznavanje znakova“ pod mentorstvom prof. dr. sc. Ane Kuzmanić Skelin pisao samostalno, primijenivši znanja i vještine stečene tijekom studiranja na Fakultetu elektrotehnike, strojarstva i brodogradnje, kao i metodologiju znanstveno-istraživačkog rada, te uz korištenje literature koja je navedena u radu. Spoznaje, stavove, zaključke, teorije i zakonitosti drugih autora koje sam izravno ili parafrazirajući naveo/la u završnom radu citirao/la sam i povezao/la s korištenim bibliografskim jedinicama.

Handwritten signature of Igor Ujević in black ink.

SADRŽAJ

1. Uvod.....	1
2. Android OS	2
2.1 Arhitektura Androida.....	2
2.2 Upravljanje memorijom.....	4
3. Softverske komponente i tehnologije	6
3.1 Android Studio	6
3.2 Gradle	7
3.3 XML u Androidu	8
3.3.1 Različite XML datoteke korištene u Androidu	9
3.4 Java (programski jezik)	15
3.4.1 Principi	16
3.4.2 Java paketi	18
3.4.3 Funkcija za slikati.....	18
3.4.4 Funkcija skeniranja	19
3.4.5 Funkcija za spremanje	20
4. Optičko prepoznavanje znakova (OCR tehnologija)	21
4.1 OCR metode	21
4.2 OCR za izrađenu aplikaciju	22
5. Zahtjevi i sučelje aplikacije „Text scanner“.....	23
5.1 Funkcionalni zahtjevi	23
5.2 Ne-funkcionalni zahtjevi	24
5.3 Sučelje aplikacije „Text Scanner“	24
6. Testiranje uspješnosti skeniranja.....	26
7. Zaključak.....	30
Literatura	31
Popis oznaka i kratica.....	32
Sažetak	33
Summary	34
Prilog	35

1. Uvod

Android je u svom relativno kratkom postojanju prešao s tržišnog sudionika na dominantnu snagu, te je danas vodeći mobilni operacijski sustav na tržištu. Donosi mnogo veći spektar mogućnosti od iOS-a i Windows-a.

OCR tehnologija se razvijala desetljećima te je tokom svog razvoja postigla značajan napredak. Danas postoje velike mogućnosti primjene te tehnologije, od skeniranja i digitaliziranja papirnatih dokumenata, pa sve do automatizacije unosa podataka, indeksiranja dokumenata za tražilice, automatskog prepoznavanja registarskih oznaka, kao i pomoć slijepim i slabovidnim osobama.

Optičko prepoznavanje znakova ne samo da smanjuje troškove ručnog rada uz brže rješenje za unos računala, već i osigurava korisnicima bolje korisničko iskustvo.

U ovom završnom radu OCR tehnologija je primijenjena za izradu Android aplikacije pod nazivom „Text Scanner“. Korisnici bi trebali moći slikani tekst skenirati, greške preurediti te na kraju spremiti tekstualni dokument na svoj uređaj.

2. Android OS

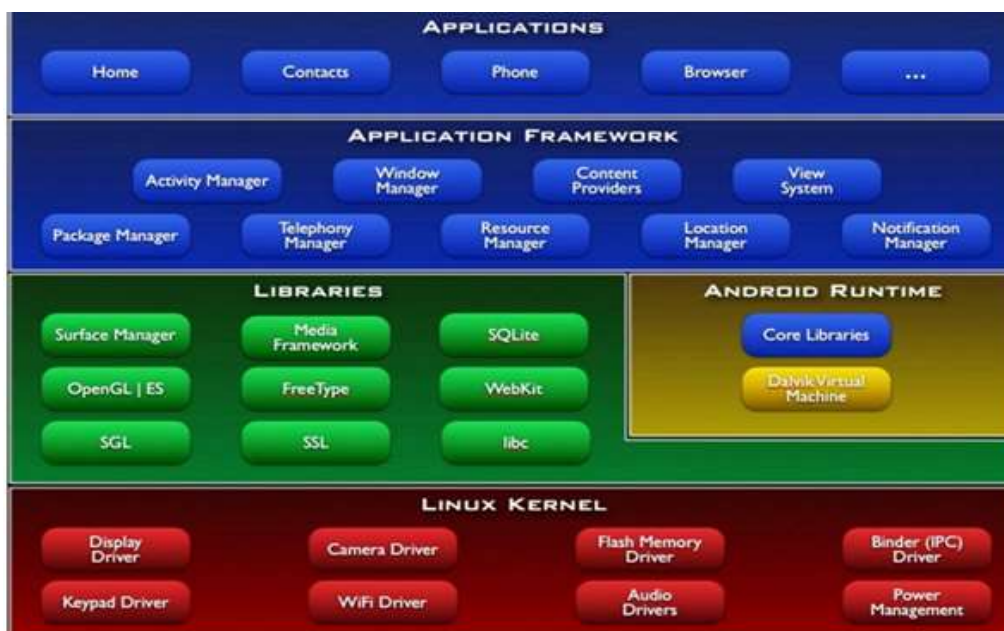
Android je otvoreni operacijski sustav kojeg su osnovali Andy Rubin, Rich Miner, Nick Sears i Chris White u Palo Altu u Kaliforniji u listopadu 2003. godine. Sustav je temeljen na Linuxu i dizajniran je prvenstveno za mobilne uređaje osjetljive na dodir, kao što su pametni telefoni i tablet računala. Operativni sustav razvio se mnogo u posljednjih 15 godina počevši od crno-bijelih telefona do najnovijih pametnih telefona ili tableta te je jedan od najčešće korištenih mobilnih OS današnjice.

Android je moćan operativni sustav i podržava veliki broj aplikacija na pametnim telefonima. Hardver koji podržava android softver temelji se na ARM arhitekturi. Sam Android je *open source* operativni sustav što znači da je besplatan i svatko ga može koristiti. Android ima milijune dostupnih aplikacija koje vam mogu pomoći da upravljate svojim životom na jedan ili drugi način, a na tržištu je dostupan s niskim troškovima zbog čega je također vrlo popularan, [0].

2.1 Arhitektura Androida

Android je operativni sustav i hrpa softverskih komponenti koja je podijeljena u pet dijelova i četiri glavna sloja:

- Linux jezgra
- Biblioteke
- ART (vrijeme izvođenja za Android)
- Aplikacijski radni okvir



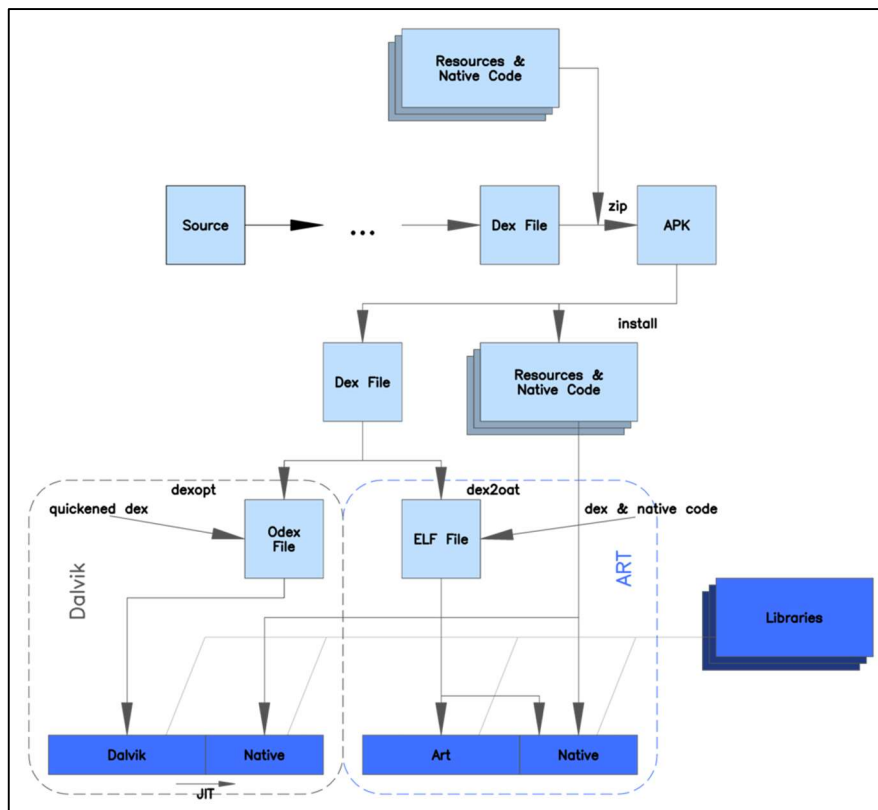
Slika 2.1 Arhitektura Androida

Android koristi snažnu Linux jezgru i podržava širok raspon upravljačkih programa za hardver. Jezgra je srce operativnog sustava koji upravlja ulaznim i izlaznim zahtjevima iz softvera. Pruža osnovne funkcionalnosti sustava kao što su upravljanje procesima, upravljanje memorijom, upravljanje uređajima kao što su kamera, tipkovnica, zaslon itd. Linux je dobar u umrežavanju i nije ga potrebno povezati s perifernim hardverom. Sama jezgra ne stupa u izravnu interakciju s korisnikom, već komunicira s ljuskom i drugim programima, kao i s hardverskim uređajima na sustavu.

Na vrhu Linuxa jezgre nalazi se skup biblioteka uključujući web-preglednik otvorenog koda kao što je webkit, library libc. Sve se to koristi kako bi se moglo reproducirati i snimati zvuk ili videozapis. SQLite je baza podataka koja je korisna za pohranu i dijeljenje aplikacijskih podataka. SSL knjižnice su odgovorne za internetsku sigurnost itd.

Android Runtime (ART) je aplikacijsko okruženje koje koristi Android sustav. Zamjenjujući Dalvik, procesni virtualni stroj koji je izvorno koristio za Android, ART izvodi prijevod bajtnog koda aplikacije u izvorne upute koje kasnije izvršava okruženje uređaja.

Dalvik Virtual Machine je procesni virtualni stroj u operativnom sustavu Android. To je softver koji pokreće aplikacije na Android uređajima. Dalvik VM koristi značajke jezgre Linuxa kao što je upravljanje memorijom i višedretvenost na programskom jeziku Java. Dalvik VM omogućuje svakoj Android aplikaciji pokretanje vlastitog procesa. [0]



Slika 2.2 Usporedba Dalvik i ART arhitekture

Radni sloj aplikacijskog okvira (eng. *framework*) pruža mnoge usluge na višoj razini aplikacijama kao što su Windows manager, sustav za pregled, paketni menadžer, upravitelj resursa itd. Programeri aplikacija mogu koristiti te usluge u svojoj aplikaciji.

Programeri imaju puni pristup istim API-jevima koje koriste osnovne aplikacije. Arhitektura aplikacije je dizajnirana kako bi se pojednostavilo ponovno korištenje komponenti. Svaka aplikacija može objaviti svoje sposobnosti (eng. *capabilities*), a svaka druga aplikacija tada može koristiti te iste sposobnosti (podložno sigurnosnim ograničenjima koje je nametnuo *framework*). [0]

2.2 Upravljanje memorijom

Budući da su uređaji s Androidom obično pokretani baterijom, Android je dizajniran za upravljanje procesima kako bi se potrošnja energije održala na minimumu. Kada aplikacija nije u uporabi, sustav obustavlja rad tako da, iako je dostupan za neposrednu upotrebu, a ne zatvoren, ne koristi energiju baterije ili CPU resurse.

Android automatski upravlja aplikacijama pohranjenim u memoriji: kada je memorija niska, sustav će početi nevidljivo i automatski zatvoriti neaktivne procese, počevši s onima koji su neaktivni najdulje vrijeme.

3. Softverske komponente i tehnologije

U ovom projektu su korištene sljedeće tehnologije:

- Android Studio
- Android SDK
- Java (objektno-orijentirani programski jezik)
- Gradle
- XML

Od softverskih komponenti koristi se Android operacijski sistem. Aplikacija može raditi na svim postojećim verzijama od 4.0.x (Ice Cream Sandwich) pa nadalje (zadnja je 9.0 Pie). Od hardverskih komponenti potreban je osobni Android pametni mobitel.

3.1 Android Studio

Temelji se na IntelliJ IDEA. Osim IntelliJ-ovog moćnog uređivača koda i alata za razvojne programere, Android Studio nudi još više značajki koje povećavaju produktivnost pri izradi aplikacija za Android, kao što su:

- Fleksibilni build sustav temeljen na Gradleu
- Brz i bogat emulator
- Jedinstveno okruženje u kojem možemo razviti sve Android uređaje
- Instant Run kako bi mogli automatski „gurnuti“ izmijene u pokrenutu aplikaciju bez izrade novog APK-a
- Predlošci koda i integracija GitHub-a pomažu u izgradnju uobičajenih značajki aplikacije te integriranje uzoraka koda
- Alati za testiranje
- Lint alati za hvatanje performansi, upotrebljivosti, kompatibilnosti verzija i drugih problema
- Podrška za C++ i NDK
- Ugrađena podrška za Google Cloud Platform

Trenutno se koristi (svibanj 2019. godina) verzija 3.4.1., [0].

Tablica 3-1 Kriterij za Android *Studio* 3.x koji treba zadovoljavati je sljedeći:

Tablica 3-1 Kriterij za Android Studio 3.x

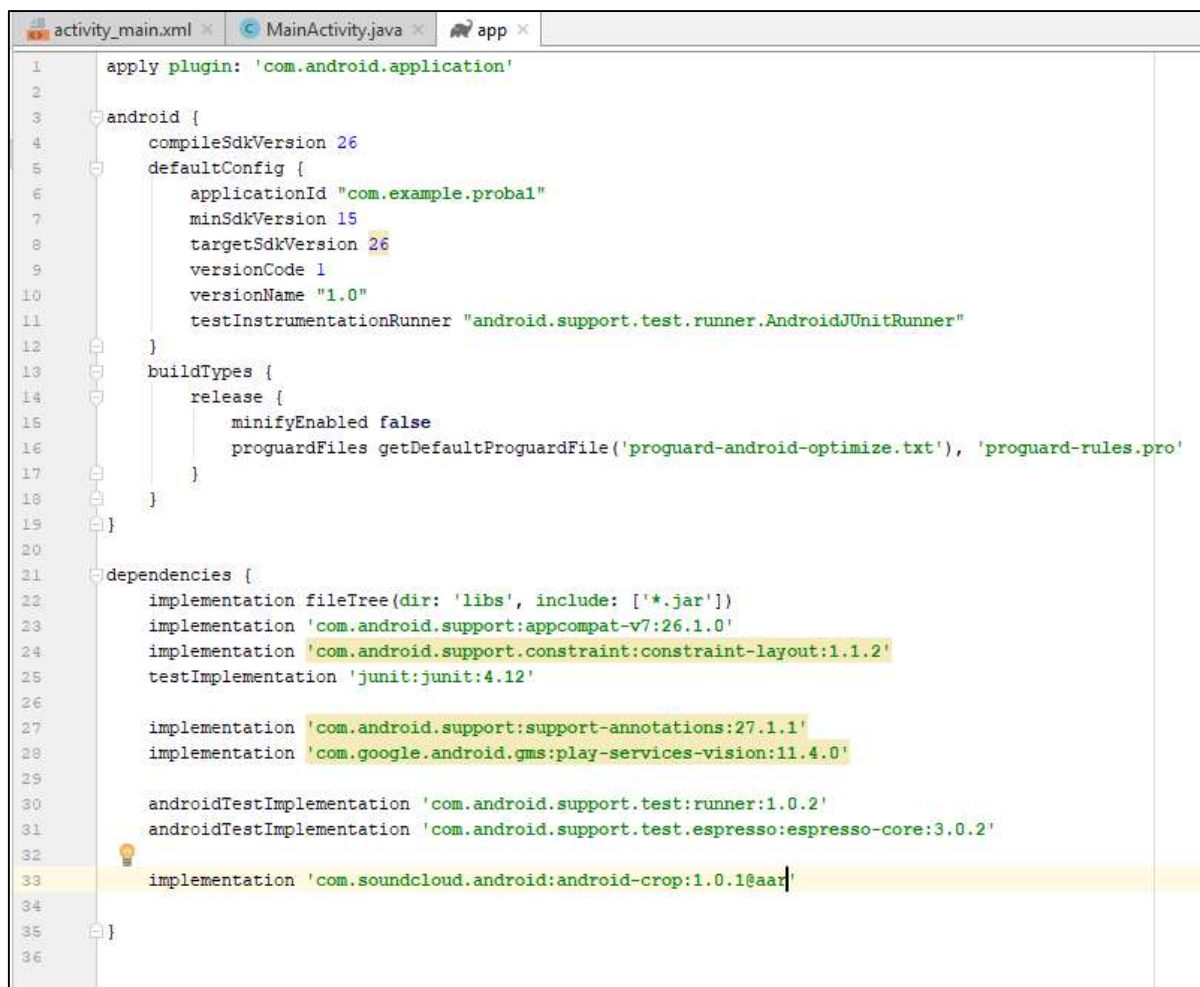
KRITERIJA	OPIS
OS verzija	Microsoft Windows 7/8/10, Mac OS X 10.10 GNOME ili KDE desktop Linux
RAM	3 GB RAM minimum, 8 GB RAM preporuka, 1 GB za Android Emulator
Veličina diska	6 GB minimum, 16 GB preporuka
Java verzija	Java Development Kit (JDK) 8
Rezolucija ekrana	1280×800

3.2 Gradle

Sustav za „build“ Android Studija temelji se na Gradleu.

Gradle je automatizirani sustav za „build“ te je također i open-source. Temelji se na konceptima Apache Ant i Apache Maven i uvodi jezik temeljen na domeni (DSL) temeljen na Groovy umjesto XML obrasca koji koristi Apache Maven za deklariranje konfiguracije projekta.

Gradle spaja dobre dijelove oba alata i nadograđuje ih na DSL i druga poboljšanja. Gradle ne koristi XML. Umjesto toga, ima vlastiti DSL baziran na Groovy (jedan od JVM jezika). Kao rezultat, Gradle skripte za skeniranje imaju tendenciju da budu mnogo kraće i jasnije od onih koje su napisane za Ant ili Maven. Za Gradle se može reći da je sljedeći evolucijski korak u JVM (Java Virtual Machine) alatima za „build“. Taj alat se oslanja na naučene lekcije alata kao što su Ant i Maven i donosi njihove najbolje ideje na sljedeću razinu. Budući da je Gradle „JVM native“, omogućuje nam pisanje prilagođene logike na jeziku koji nam najviše odgovara, bilo da je u pitanju Java ili Groovy. [0]

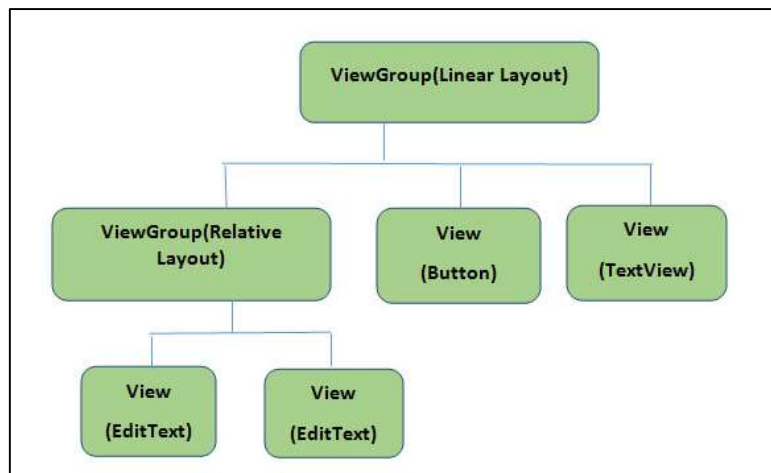


Slika 3.1 App build.gradle

3.3 XML u Androidu

XML je skraćenica za *Extensible Markup Language*. XML je markup jezik sličan HTML-u koji se koristi za opisivanje podataka. XML oznake nisu unaprijed definirane u XML-u. Moramo definirati vlastite oznake. XML je sam po sebi dobro čitljiv i od ljudi i od stroja. Također je skalabilna i jednostavan za razvoj. U Androidu koristimo XML za osmišljavanje rasporeda aplikacije koje razvijamo.

Cijeli koncept korisničkog sučelja za Android definiran je pomoću hijerarhije objekata *View* i *ViewGroup*. *ViewGroup* je nevidljivi spremnik koji organizira podređene prikaze. Jedna grupa za pregledavanje može imati drugu grupu *ViewGroup* kao podređeni element kao što je prikazano na donjoj slici. [0]

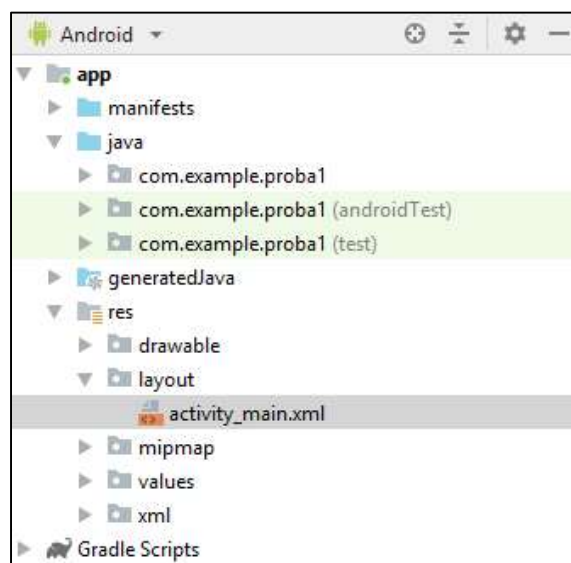


Slika 3.2 Koncept XML-a u Androidu

3.3.1 Različite XML datoteke korištene u Androidu

U Androidu postoji nekoliko XML datoteka koje se koriste u nekoliko različitih svrha. U nastavku ćemo definirati neke od njih.

Layout XML datoteke koristi se za definiranje stvarnog korisničkog sučelja (korisničkog sučelja) izrađene aplikacije. Sadrži sve elemente (prikaze) ili alate koje želimo koristiti u aplikaciji. Poput TextViewa, gumba i drugih elemenata korisničkog sučelja. Pronaći ćemo ovu datoteku unutar mape *res* i unutar nje postoji još jedna mapa pod nazivom *layout* gdje možemo pronaći sve layout datoteke.



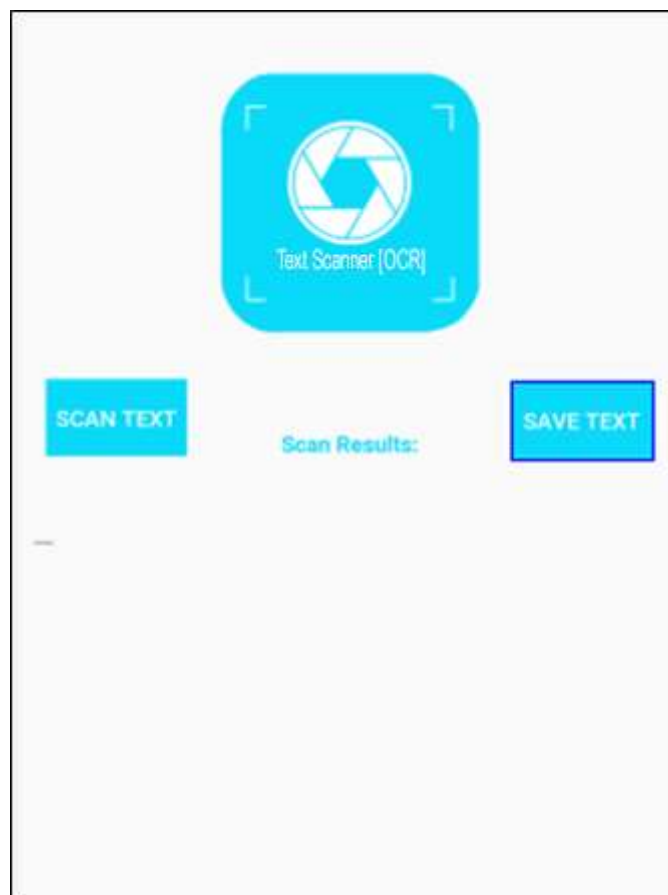
Slika 3.3 Lokacija layout XML datoteke

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <android.support.constraint.ConstraintLayout
3      android:id="@+id/activity_main"
4      xmlns:android="http://schemas.android.com/apk/res/android"
5      xmlns:app="http://schemas.android.com/apk/res-auto"
6      xmlns:tools="http://schemas.android.com/tools"
7      android:layout_width="match_parent"
8      android:layout_height="match_parent"
9  >
10
11      <ImageView
12          android:id="@+id/imageView"
13          android:layout_width="200dp"
14          android:layout_height="200dp"
15          android:layout_marginTop="16dp"
16          app:layout_constraintLeft_toLeftOf="parent"
17          app:layout_constraintRight_toRightOf="parent"
18          app:layout_constraintTop_toTopOf="parent"
19          app:srcCompat="@drawable/logo"
20          tools:layout_constraintLeft_creator="1"
21          tools:layout_constraintRight_creator="1"/>
22
23      <Button
24          android:id="@+id/button"
25          android:layout_width="wrap_content"
26          android:layout_height="wrap_content"
27          android:layout_marginTop="8dp"
28          android:layout_marginBottom="8dp"
29          android:text="SCAN TEXT"
30          android:textColor="#FFFF"
31          app:layout_constraintBottom_toTopOf="@+id/scrollView2"
32          app:layout_constraintHorizontal_bias="0.051"
33          app:layout_constraintLeft_toLeftOf="parent"
34          app:layout_constraintRight_toRightOf="parent"
35          app:layout_constraintTop_toBottomOf="@+id/imageView"
36          tools:layout_constraintLeft_creator="1"
37          tools:layout_constraintRight_creator="1"/>

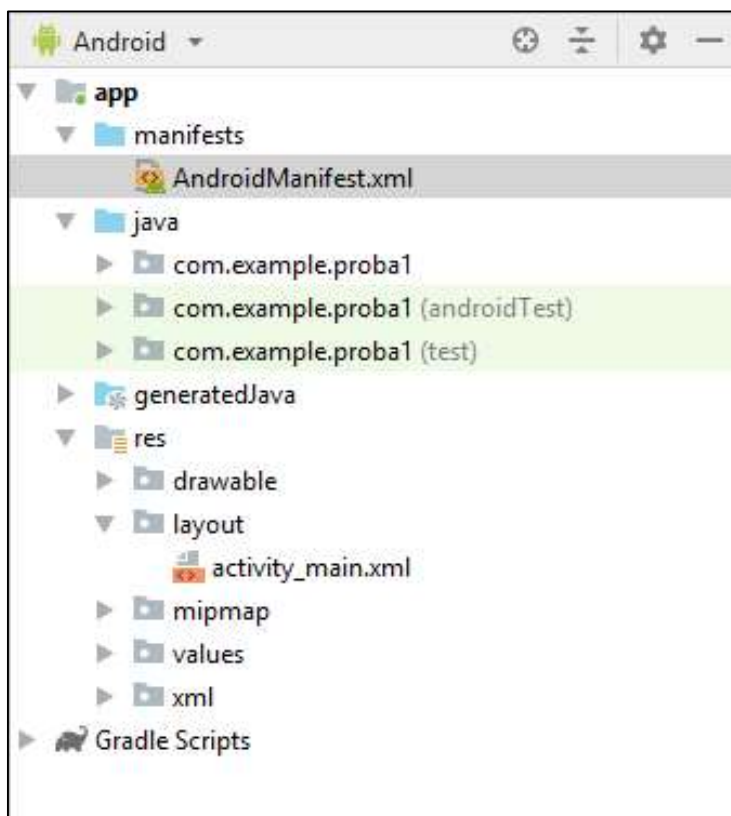
```

Slika 3.4 Izgled layout XML datoteke 1



Slika 3.5 Izgled layout XML datoteke 2

Manifest XML datoteka koristi se za definiranje svih komponenti aplikacije. Uključuje nazive paketa izrađene aplikacija, aktivnosti, prijemnike, usluge i dozvole koje aplikacija treba. Primjerice, pretpostavimo da trebamo koristiti internet u aplikaciji, onda moramo definirati internetsko dopuštenje u ovoj datoteci. Nalazi se unutar mape *app*, te potom unutar mape *manifests*.

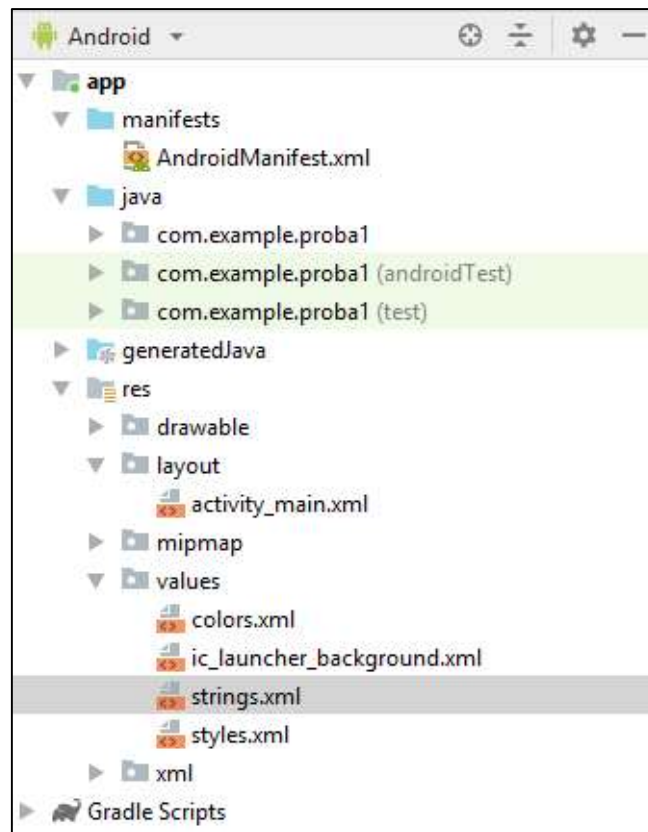


Slika 3.6 Lokacija Manifest.xml datoteke

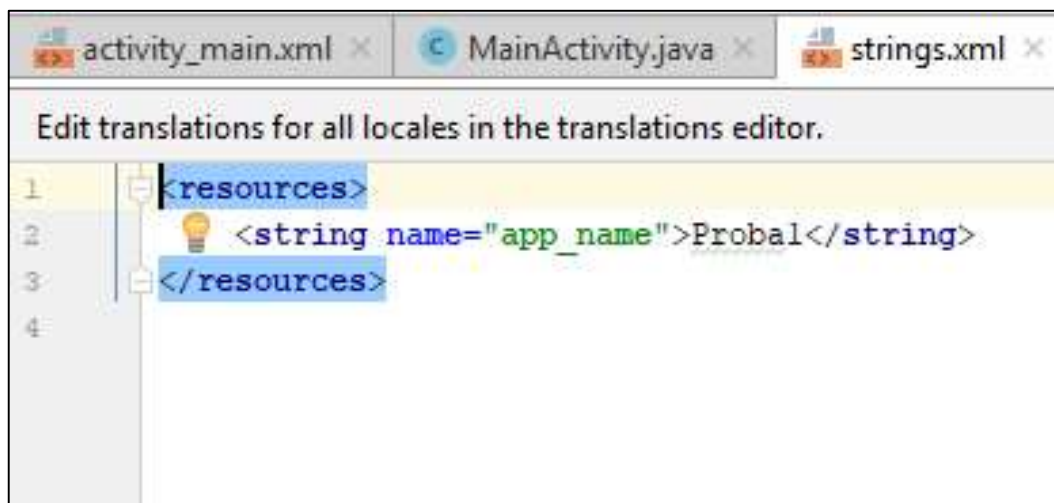


Slika 3.7 Izgled Manifest.xml datoteke

Strings XML datoteka koristi se za zamjenu *hard-coded* kodova s jednim nizom. Definiramo sve nizove u ovoj XML datoteci, a zatim im pristupamo u aplikaciji (Activity ili u Layout XML datotekama) iz ove datoteke. Ova datoteka povećava mogućnost ponovnog korištenja koda. Lokacija u Android studiju:

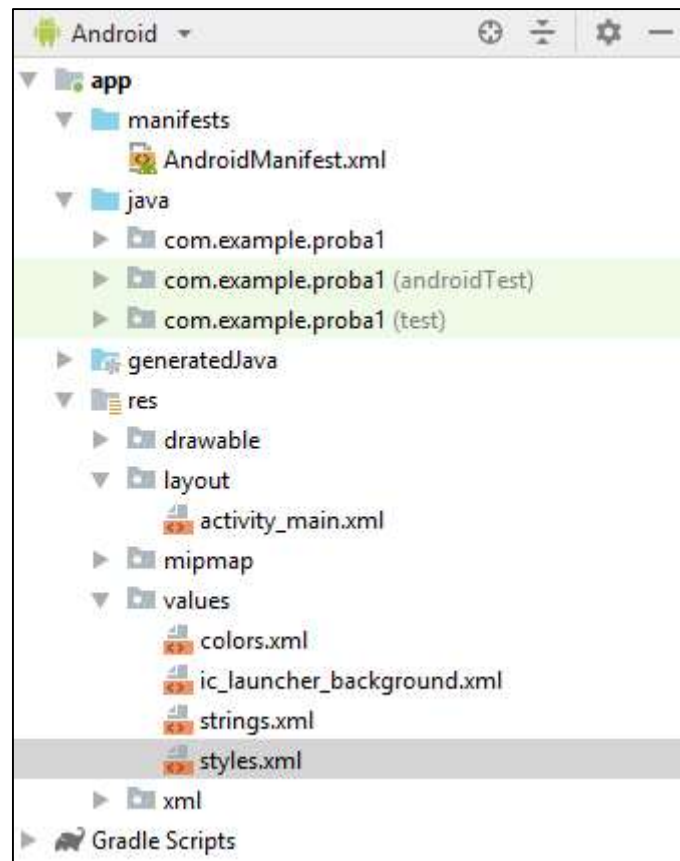


Slika 3.8 Lokacija Strings.xml

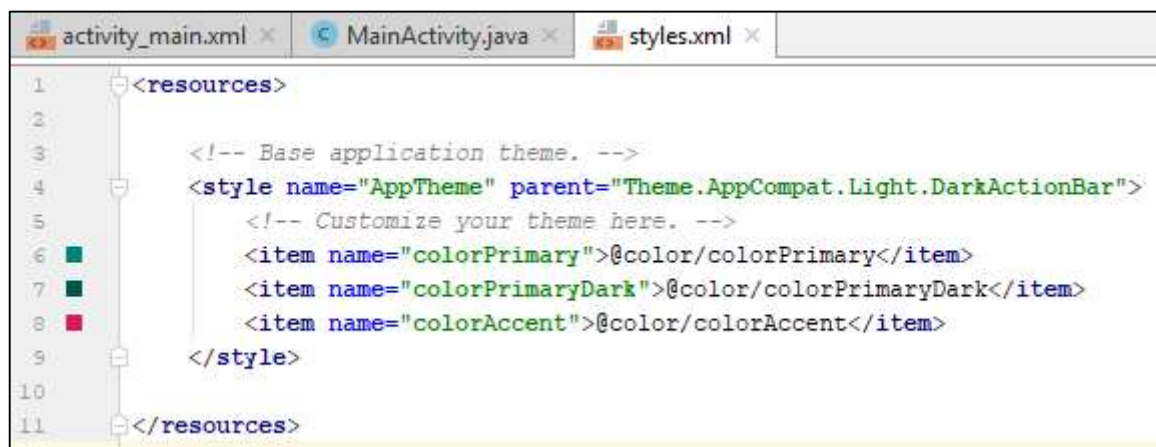


Slika 3.9 Izgled strings.xml

Styles XML datoteka služi za definiranje različitih stilova i traži UI (korisničko sučelje) aplikacije. Definiramo prilagođene teme i stilove u ovoj datoteci. Lokacija u Android studiju:



Slika 3.10 Lokacija styles.xml



Slika 3.11 Izgled styles.xml

3.4 Java (programski jezik)

Java je programski jezik opće namjene koji je zasnovan na klasi, objektno orijentiran (iako nije čisti OO jezik, jer sadrži primitivne tipove), te je osmišljen tako da ima što je moguće manje ovisnosti o implementaciji.

Programski jezik opće namjene znači da je Java programski jezik koji se široko primjenjuje u različitim područjima, ali nedostaje specijalizirana značajka za određenu domenu. To je u suprotnosti s specifičnim jezikom domene (DSL) koji je specijaliziran za određenu domenu aplikacije.

Zasnovan na klasi sporazumijeva da je OO jezik u kojem se nasljedstvo javlja putem definiranja klase.

Namjera programskog jezika Java je omogućiti razvojnim programerima „pisanje jednom, pokretanje bilo gdje“ (WORA - write once, run anywhere), što znači da se Java kod koji je preveden može izvoditi na svim platformama koje podržavaju Java bez potrebe za ponovnim računalnim prevođenjem koda. Java aplikacije se obično računalno prevode u bajtni kod koji se može izvoditi na bilo kojem Java virtualnom stroju (JVM) bez obzira na temeljnu arhitekturu računala.

Sintaksa Java je slična C i C++, ali ima manje objekata nižih razina nego bilo koji od njih. Od 2018. godine programski jezik Java je jedan od najpopularnijih programskih jezika u uporabi prema GitHub-u.

Java programski jezik je izvorno razvio James Gosling iz tvrtke Sun Microsystems (koji je od tada kupio Oracle), a 1995. godine je taj programski jezik objavljen kao ključna komponenta platforme Sun Microsystems. Izvorne i referentne implementacije Java kompilatora, virtualnih strojeva i knjižnica klasa izvorno je izdala tvrtka Sun pod vlasničkim licencama. Od svibnja 2007. godine, u skladu sa specifikacijama Java Community Process, Sun je licencirao većinu svojih Java tehnologija pod GNU Općom javnom licencom. U međuvremenu, drugi su razvili alternativne implementacije tih Sun tehnologija, kao što su „GNU Compiler for Java“ (računalni prevoditelj za bajt-kod), GNU Classpath (standardne knjižnice) i IcedTea-Web (plugin za preglednike za applete). [0]

3.4.1 Principi

Bilo je pet glavnih ciljeva u stvaranju jezika Java:

- Mora biti "jednostavna, objektno orijentirana i poznata"
- Mora biti "robustan i siguran"
- Mora biti "neutralan u arhitekturi i prenosiv"
- Mora se izvršavati s "visokim performansama"
- Mora biti "interpretirana, navođena i dinamična"

Primarne karakteristike Java programskog jezika uključuju *jednostavan* jezik koji se može programirati bez opsežnog programerskog treninga dok je prilagođen trenutnim softverskim praksama.

Programski jezik Java dizajniran je da bude objektno orijentiran od temelja. Objektna tehnologija konačno pronalazi svoj put u široj upotrebi nakon tridesetogodišnjeg razdoblja razvoja. Potrebe distribuiranih sustava utemeljenih na klijentu i poslužitelju podudaraju se s enkapsuliranim paradigmama koje objekt prolazi kroz objektni softver. Za funkcioniranje u sve složenijim, mrežnim okruženjima, programski sustavi moraju usvojiti objektno orijentirane koncepte. Java tehnologija osigurava čistu i učinkovitu objektnu razvojnu platformu.

Programeri koji koriste Java programski jezik mogu pristupiti postojećim bibliotekama testiranih objekata. Iako je C++ odbačen kao jezik implementacije, Java programski jezik se održava tako da izgleda kao C++ koliko god je to moguće, što rezultira time da je to poznati programski jezik dok uklanja nepotrebne složenosti C++-a.

Programski jezik Java dizajniran je za stvaranje visoko *pouzdanog* softvera. Omogućuje opsežnu provjeru vremena kompajliranja, nakon čega slijedi druga razina provjere vremena izvršavanja.

Model upravljanja memorijom je vrlo jednostavan: objekti se stvaraju s *new* operatorom. Nema eksplicitnih tipova podataka pokazivača definiranih od strane programera, nema aritmetike pokazivača i automatskog prikupljanja „smeća“. Java kod možemo razviti s povjerenjem da će sustav brzo pronaći mnogo pogrešaka i da veliki problemi neće uspavati sve dok se proizvodni kod ne dostavi.

Java tehnologija je dizajnirana za rad u distribuiranim okruženjima, što znači da je *sigurnost* od najveće važnosti. Sa sigurnosnim značajkama dizajniranim u jezičnom i izvršnom sustavu, Java tehnologija nam omogućuje da izrađujemo aplikacije koje se ne mogu napasti izvana.

Java tehnologija je dizajnirana za podršku aplikacijama koje će biti razmještene u heterogena mrežna okruženja. U takvim okruženjima, aplikacije moraju biti sposobne za izvršavanje na različitim hardverskim arhitekturama.

Kako bi zadovoljio raznolikost operacijskih okruženja, *Java Compiler* generira *bajt*-kodove – namijenjen učinkovitom prijenosu koda na više hardverskih i softverskih platformi. Java tehnologije rješava i problem binarne distribucije i problem verzije; isti bajtni kodovi Java programskog jezika će se izvoditi na bilo kojoj platformi.

Izvedba je uvijek važna. Java platforma postiže vrhunske performanse usvajanjem sheme pomoću koje tumač može raditi punom brzinom bez potrebe za provjerom okoline u kojoj se izvodi. *Automatic garbage collector* osigurava visoku vjerojatnost da je dovoljno memorije dostupno kada je to potrebno, što dovodi do bolje izvedbe. Aplikacije koje zahtijevaju velike količine računalne moći mogu se dizajnirati tako da se dijelovi koji se temelje na računanju mogu ponovno pisati u izvornom strojnom kodu prema potrebi i povezati se s Java platformom. Općenito, korisnici smatraju da interaktivne aplikacije brzo reagiraju iako su interpretirane.

Java interpreter može izvršiti Java bytecodes izravno na bilo koji stroj u koji je prenesen *interpreter* i *run-time* sustav. Faza povezivanja programa je jednostavna, inkrementalna i lagana. Platforma Java podržava višedretvenost na razini jezika s dodatkom sofisticiranih primitive za sinkronizaciju: jezična knjižnica osigurava *Thread* klasu, a sustav za vrijeme izvođenja osigurava nadzor i stanje zaključavanja. Dok je Java Compiler strog u statičkoj provjeri vremena kompajliranja, jezik i vrijeme izvršavanja su dinamični u svojim fazama povezivanja. Klase se povezuju samo po potrebi. Novi kodni moduli mogu se povezati na zahtjev iz različitih izvora, čak i iz izvora preko mreže. [0]

3.4.2 Java paketi

Java paket je spremnik koji grupira srodne vrste (Java klase, sučelja, enumeracije i napomene). Ugrađeni paketi postojeći java paketi koji dolaze zajedno s JDK-om (Java Development Kit), kao što su java.lang, java.util, java.io itd. Paketi koje smo uvezli i koji su bitni za rad aplikacije možemo vidjeti na sljedećoj slici.



Slika 3.12 Java paketi aplikacije Text Scanner

3.4.3 Funkcija za slikati

Na sljedećoj slici može se vidjeti izgled funkcije koja omogućava korištenje mobilnog fotoaparata i kako bi slikali željeni tekst.

```
private void takePicture() {  
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);  
    File photo = new File(Environment.getExternalStorageDirectory(), "picture.jpg");  
    Uri uri = FileProvider.getUriForFile(context, MainActivity.this,  
        authority: BuildConfig.APPLICATION_ID + ".provider", photo);  
    intent.putExtra(MediaStore.EXTRA_OUTPUT, uri);  
  
    startActivityForResult(intent, PHOTO_REQUEST);  
}
```

Slika 3.13 Funkcija za slikati

3.4.4 Funkcija skeniranja

Na sljedećim slikama prikazan je proces skeniranja.

```
//skeniranje
Button button = (Button) findViewById(R.id.button);
scanResults = (TextView) findViewById(R.id.results);
if (savedInstanceState != null) {
    imageUri = Uri.parse(savedInstanceState.getString(SAVED_INSTANCE_URI));
    scanResults.setText(savedInstanceState.getString(SAVED_INSTANCE_RESULT));
}
detector = new TextRecognizer.Builder(getApplicationContext()).build();
button.setOnClickListener((view) -> {
    ActivityCompat.requestPermissions(activity: MainActivity.this, new
        String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE}, REQUEST_WRITE_PERMISSION);
});
```

Slika 3.14 Funkcija skeniranja 1.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == PHOTO_REQUEST && resultCode == RESULT_OK) {
        launchMediaScanIntent();
        try {
            Bitmap bitmap = decodeBitmapUri(cbc: this, imageUri);
            if (detector.isOperational() && bitmap != null) {
                Frame frame = new Frame.Builder().setBitmap(bitmap).build();
                SparseArray<TextBlock> textBlocks = detector.detect(frame);
                String blocks = "";
                String lines = "";
                String words = "";
                for (int index = 0; index < textBlocks.size(); index++) {
                    //extract scanned text blocks here
                    TextBlock tBlock = textBlocks.valueAt(index);
                    blocks = blocks + tBlock.getValue() + "\n" + "\n";
                    for (Text line : tBlock.getComponents()) {
                        //extract scanned text lines here
                        lines = lines + line.getValue() + "\n";
                        for (Text element : line.getComponents()) {
                            //extract scanned text words here
                            words = words + element.getValue() + " ";
                        }
                    }
                }
                if (textBlocks.size() == 0) {
                    scanResults.setText("Scan Failed: Found nothing to scan");
                } else {
                    //scanResults.setText(scanResults.getText() + "Result: " + "\n");
                    scanResults.setText(scanResults.getText() + blocks + "\n");
                    //scanResults.setText(scanResults.getText() + "-----" + "\n");
                    //scanResults.setText(scanResults.getText() + "Lines: " + "\n");
                    //scanResults.setText(scanResults.getText() + lines + "\n");
                    //scanResults.setText(scanResults.getText() + "-----" + "\n");
                    //scanResults.setText(scanResults.getText() + "Words: " + "\n");
                    //scanResults.setText(scanResults.getText() + words + "\n");
                    //scanResults.setText(scanResults.getText() + "-----" + "\n");
                }
            } else {
                scanResults.setText("Could not set up the detector!");
            }
        } catch (Exception e) {
            Toast.makeText(context: this, text: "Failed to load Image", Toast.LENGTH_SHORT)
                .show();
            Log.e(LOG_TAG, e.toString());
        }
    }
}
```

Slika 3.15 Funkcija skeniranja 2.


```

private void launchMediaScanIntent() {
    Intent mediaScanIntent = new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE);
    mediaScanIntent.setData(imageUri);
    this.sendBroadcast(mediaScanIntent);
}

private Bitmap decodeBitmapUri(Context ctx, Uri uri) throws FileNotFoundException {
    int targetW = 600;
    int targetH = 600;
    BitmapFactory.Options bmOptions = new BitmapFactory.Options();
    bmOptions.inJustDecodeBounds = true;
    BitmapFactory.decodeStream(ctx.getContentResolver().openInputStream(uri), null, bmOptions);
    int photoW = bmOptions.outWidth;
    int photoH = bmOptions.outHeight;

    int scaleFactor = Math.min(photoW / targetW, photoH / targetH);
    bmOptions.inJustDecodeBounds = false;
    bmOptions.inSampleSize = scaleFactor;

    return BitmapFactory.decodeStream(ctx.getContentResolver().openInputStream(uri), null, bmOptions);
}

```

Slika 3.16 Funkcija skeniranja 3.

3.4.5 Funkcija za spremanje

Kod funkcije *Save* prvo kreiramo direktorij u koji ćemo spremati skenirani tekst. Potom kreiramo tekstualnu datoteku u koju ćemo spremiti, a ona se sastoji od imena „skenirano“ te slučajno generiranog broja. Zatim prikupljamo skenirani text i upisujemo ga u novonastalu tekstualnu datoteku. Na kraju je zatvaramo i brišemo spremljeni tekst iz EditText-a. Pri završetku spremanja se ispisuje poruka gdje je spremljena tekstualna datoteka.

```

public void save(View v) {
    //create file
    File dir = new File(Environment.getExternalStorageDirectory().getAbsolutePath(), DIR_FILE_NAME);
    if(!dir.exists()) {
        dir.mkdir();
    }

    Random rand = new Random();
    int n = rand.nextInt();

    String filename = "skenirano" + n + ".txt";

    File file = new File(dir, filename);
    String data = mEditText.getText().toString();
    try {
        FileOutputStream fos = new FileOutputStream(file);
        fos.write(data.getBytes());
        fos.close();
        mEditText.getText().clear();
        Toast.makeText(context, this, "Saved to" + getFilesDir() + "/" + DIR_FILE_NAME + "/" + filename, Toast.LENGTH_LONG).show();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

```

Slika 3.17 Funkcija spremanja skeniranog teksta u tekstualnu datoteku

4. Optičko prepoznavanje znakova (OCR tehnologija)

Optičko prepoznavanje znakova ili OCR tehnologija se koristi za pretvaranje gotovo bilo koje vrste slika koje sadrže pisani tekst (rukopisni ili tiskani) u strojno čitljive tekstualne podatke. Većina OCR sustava koristi kombinaciju hardvera i softvera koja prepoznaje znakove slova.

Tehnologija OCR postala je popularna početkom devedesetih godina, a od tada je tehnologija doživjela nekoliko poboljšanja. Vjerojatno je najpoznatiji slučaj uporabe OCR-a pretvaranje tiskanih papirnatih dokumenata u tekstualno dokumentirane dokumente. Nakon što je skenirani papirni dokument prošao kroz OCR obradu, tekst dokumenta može se uređivati pomoću programa za obradu teksta.

4.1 OCR metode

Postoje dvije osnovne metode koje se koriste za OCR:

- Uspoređivanje matrica (eng. Matrix matching)
- Pronalaženje ključnih obilježja (eng. Feature extraction)

Jednostavnija i učestalija metoda je uspoređivanje matrica. Ta metoda uspoređuje što skener vidi kao znak, slovo, sa popisom slovnih matrica ili predložaka. Kada skenirana slika odgovara jednoj od zadanih matrica unutar postavljenog stupnja sličnosti, računalo joj dodjeljuje kôd jednog od ASCII znakova.

Pronalaženje ključnih obilježja je OCR metoda ovisi o „računalnoj inteligenciji” postavljenoj od strane proizvođača. Računalo traži osnovne oblike kao što su: otvorene površine, zatvoreni oblici, dijagonalne linije itd.

4.2 OCR za izrađenu aplikaciju

Izrađena aplikacija „Text Scanner“ kako bi koristila OCR unosi (eng. *import*) različite Google API-je za Android.

```
import com.google.android.gms.vision.Frame;  
import com.google.android.gms.vision.text.Text;  
import com.google.android.gms.vision.text.TextBlock;  
import com.google.android.gms.vision.text.TextRecognizer;
```

Aplikacijsko programsko sučelje (eng. *application programming interface, API*) je skup određenih pravila i specifikacija koje programeri slijede tako da se mogu služiti uslugama ili resursima operacijskog sustava ili nekog drugog složenog programa kao standardne biblioteke rutina (funkcija, procedura, metoda), struktura podataka, objekata i protokola.

Korištenjem API-ja omogućava programerima koristiti rad drugih programera štedeći vrijeme i trud koji je potreban da se napiše neki složeni program, pri čemu svi programeri koriste iste standarde. Napretkom u operacijskim sustavima, osobito napretkom u grafičkom korisničkom sučelju API je nezaobilazan u stvaranju novih aplikacija. Umjesto da se programi pišu novi iz temelja, programeri nastavljaju na radu drugih. [0]

Mobilni uređaj preuzima izvorne knjižnice, a obično se to izvršava prije nego što se aplikacija prvi put pokrene.

Public final class TextRecognizer proširuje detektor TextBlock. Pronalazi i prepoznaje tekst u okviru (eng. *frame*). Odnosno, OCR algoritam pokušava zaključiti raspored teksta i organizirati svaki odlomak u TextBlock instance. Ako se otkrije bilo koji tekst, TextBlock će vratiti i najmanju instancu. Možemo zamisliti da je TextBlock poveznica ka OCR motoru.

5. Zahtjevi i sučelje aplikacije „Text Scanner“

U ovom poglavlju izložene su opće informacije i karakteristike projekta. Osnovna namjena „Text Scanner“ aplikacije je olakšati skeniranje teksta, prepravljanje pogrešaka kod skeniranog teksta i spremanje istoga u tekstualnu datoteku.

Korisničko sučelje nudi sljedeće opcije:

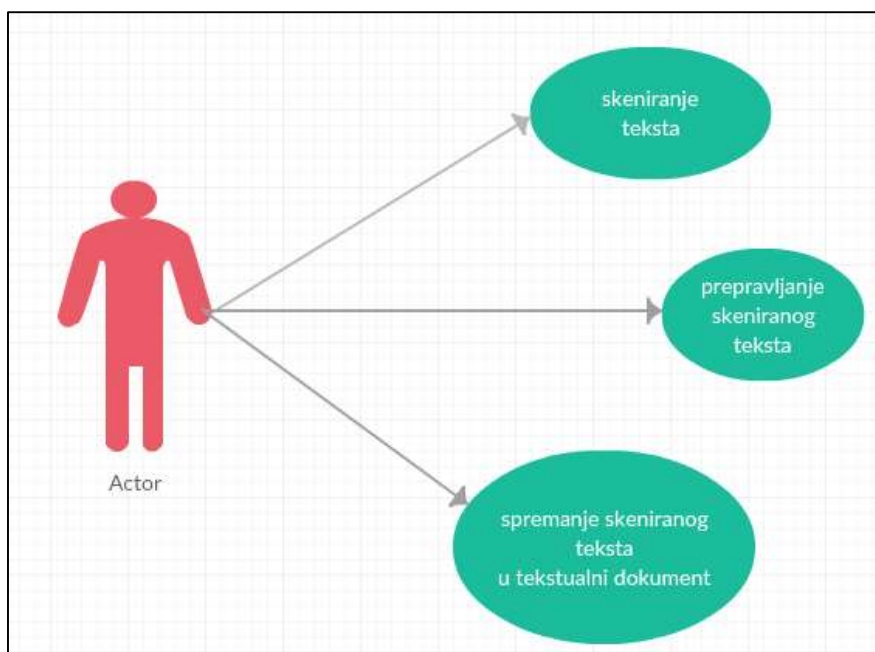
- Skeniranje teksta
- Prepravljanje teksta
- Spremanje skeniranog teksta u .txt dokument

Mobilna aplikacija „Text Scanner“ će se koristiti na android mobilnim uređajima. Potrebno je imati minimalnu verziju androida 4.0. Također, biti će testirana za rad na sljedećim uređajima:

- Huawei
- Xiaomi
- Samsung
- Ostali mobilni android uređaji

5.1 Funkcionalni zahtjevi

Use-caseovi nam prikazuju funkcionalne zahtjeve koje projekt teži postići tijekom procesa razvoja aplikacije.



Slika 5.1 Prikaz Use-case dijagrama

5.2 Ne-funkcionalni zahtjevi

Zadaća ove aplikacije je korisniku omogućiti brzo i jednostavno skeniranje teksta, mogućnost jednostavnog prepravljanja skeniranog teksta te spremanje u tekstualni dokument. Da bi takvo nešto bilo omogućeno potrebno je kreirati pregledno i jednostavno sučelje.

Korisničko sučelje mora sadržavati:

- Dugme za skeniranje
- Prostor za prepravljanje teksta
- Dugme za spremanje teksta

Od aplikacije se očekuje da ima brz odziv na korisničke zahtjeve.

Aplikacija traži korisnikovu dozvolu pristupa pohrani i korištenju fotoaparata, a sigurnost i privatnost korisnikovih podataka je zajamčena jer mobilna aplikacija ne zahtjeva nikakav dodatni pristup sadržaju mobilnog uređaja.

5.3 Sučelje aplikacije „Text Scanner“

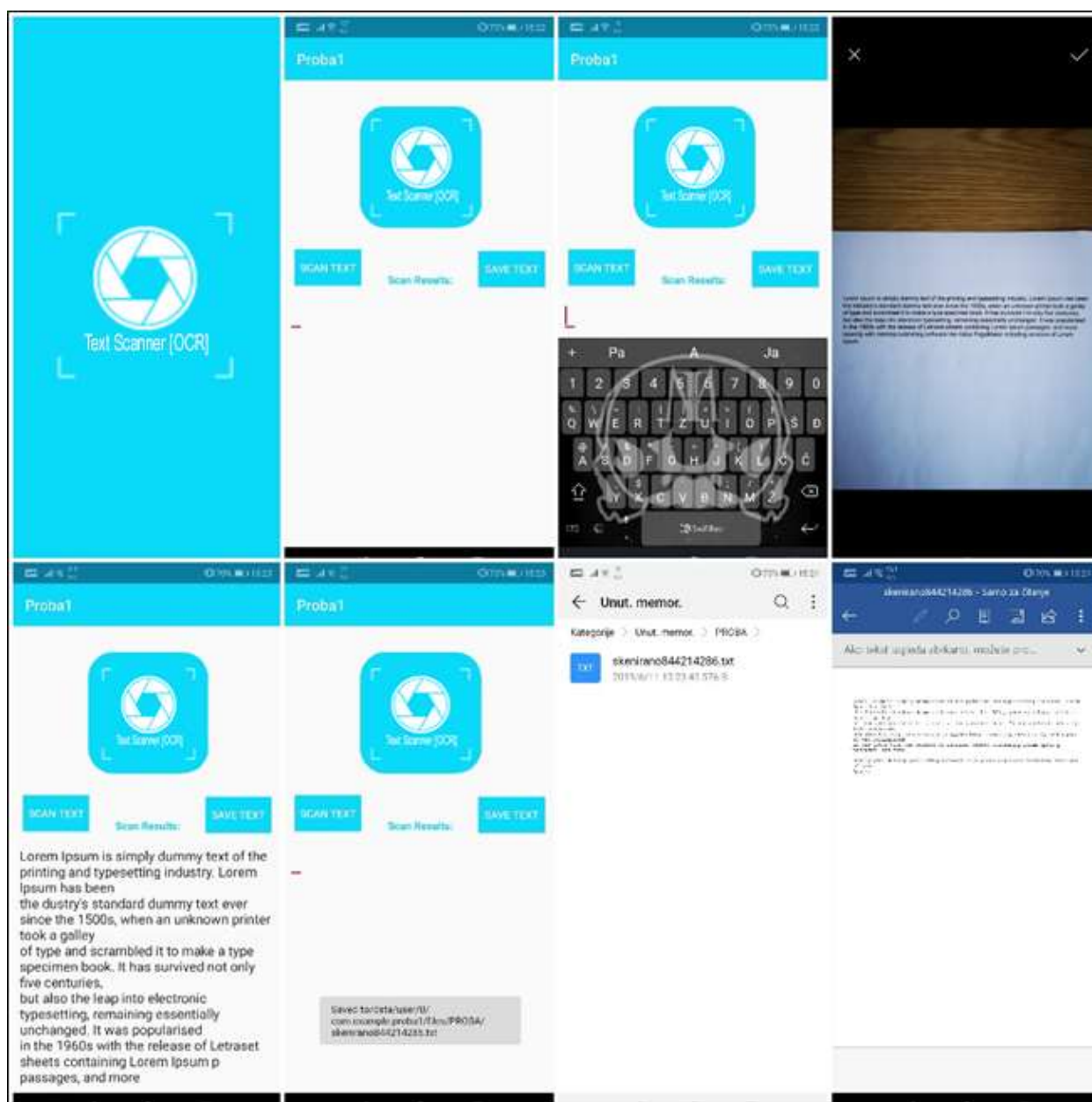
Dizajn sučelja definira interakcije sustava s vanjskim entitetima. Razlikujemo korisničko sučelje i sistemsko sučelje. Korisničko sučelje je dio sustava preko kojeg korisnik komunicira sa sustavom. Sistemsko sučelje je dio sustava preko kojeg sustav komunicira s drugim sustavima.

Korisničko sučelje je mjesto susreta odnosno dodira između korisnika (osobe) i sustava.

Korisničko sučelje može se promatrati kroz tri osnovna dijela:

- mehanizam navigacije (*navigation mechanism*) određuje na koji način korisnik postavlja zahtjeve prema sustavu tj. kako korisnik kaže sustavu što da radi
- mehanizam ulaza (*input mechanism*) definira način na koji će korisnik unijeti informaciju u sustav
- mehanizam izlaza (*output mechanism*) definira način na koji sustav isporučuje informaciju korisnicima

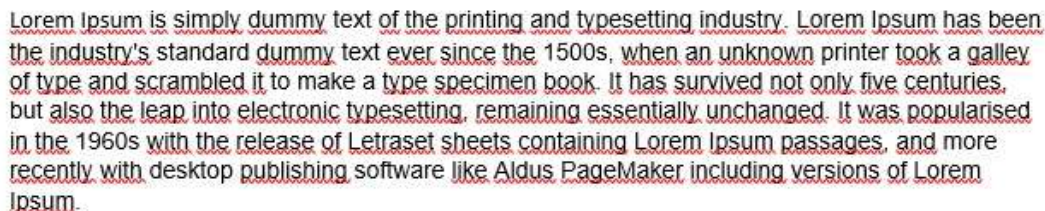
Na sljedećoj slici možemo vidjeti izgled sučelja izrađene aplikacije.



Slika 5.2 Izgled sučelja aplikacije, mjesto spremanja i izgled spremljenog dokumenta

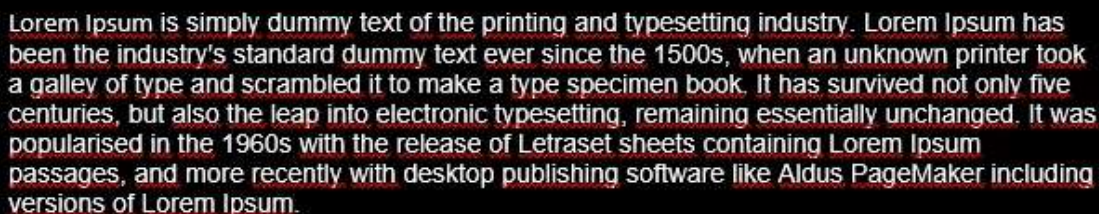
6. Testiranje uspješnosti skeniranja

Glavni cilj testiranja je demonstracija da aplikacija zadovoljava uvjete korištenja. Prilikom testiranja korištena su sljedeća 4 uzorka teksta:



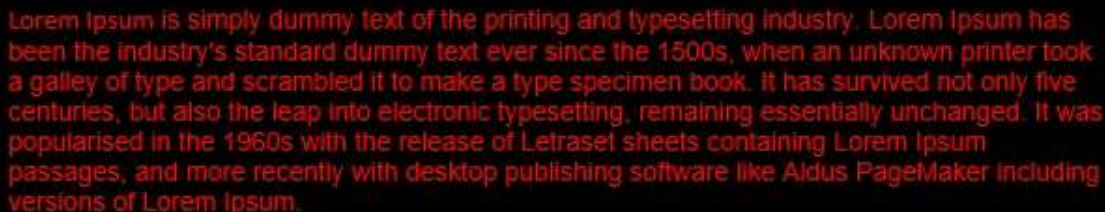
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Slika 6.1 Testni uzorak 1



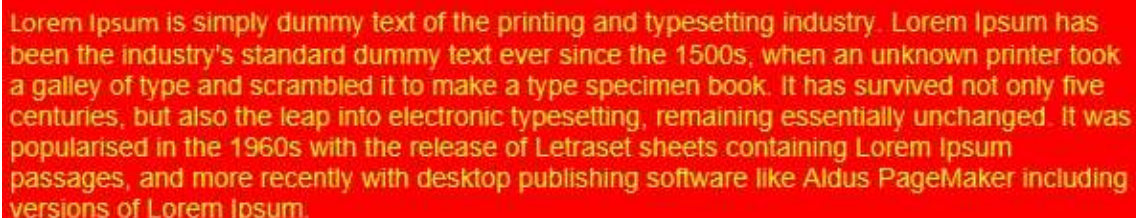
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Slika 6.2 Testni uzorak 2



Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Slika 6.3 Testni uzorak 3



Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Slika 6.4 Testni uzorak 4

Sva četiri testna uzorka predstavljaju isti tekst, ali različito uređen. Prvi uzorak predstavlja normalne uvjete, crni tekst na bijeloj podlozi. Drugi uzorak predstavlja negativ prvog uzorka, odnosno bijeli tekst na crnoj podlozi. Treći uzorak predstavlja boje koje mogu stajati skupa u paru, a za uzorak je uzet crveni tekst na crnoj podlozi. Zadnji, četvrti uzorak, predstavlja tekst na podlozi gdje postoji velika sličnost u boji. Za testiranje su uzete tople boje, odnosno žuti tekst i crvena podloga. Rezultati uspješnosti su sljedeći:

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Slika 6.5 Rezultat testnog uzorka 1

em Ipsum is simply dummy text of the printing and
been the industry's standard dummy text ever since the 1500s, when an u
a galley of type and scrambled it to make a type specimen book.It has su
centuries, but also the leap into electronic typesetting, remaining es
popularised in the 1960s with the rel
passages. and

se of Letraset sheets containing Lore
re recently with desktop publishing software like Aldus Pa

ons of Lorem Ipsum.

y. Lorem Ipsum has

n printer took
d not only five

It was

antiav Ichaine
Ipsum

vlkor icltoinc

Slika 6.6 Rezultat testnog uzorka 2

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. I It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Slika 6.7 Rezultat testnog uzorka 3

Lorem Ipsum is simply dummy text of the printing and typeseting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Slika 6.8 Rezultat testnog uzorka 4

Tablica 6-1 Tablica uspješnosti skeniranja testnih uzoraka

UZORCI	USPJEŠNOST
Uzorak 1	≈ 98,97%
Uzorak 2	≈ 56,325%
Uzorak 3	≈ 96%
Uzorak 4	≈ 99%

Iz testiranja skeniranja testnih uzoraka možemo zaključiti da je uspješnost skeniranja vrlo visoka u slučajevima kada se skenira uobičajeni akromatski tekst. Naspram tome, skeniranje negativa, odnosno teksta bijele boje na crnoj podlozi daje poprilično loše rezultate, što je i očekivano jer princip kojeg koristi algoritam prepoznavanja slova je zasnovan na prepoznavanju određenih malih blokova (slova) na bijeloj podlozi. Slova bijele boje na crnoj podlozi tretiraju se kao praznine, u ovisnosti o dužini riječi ili razmaku među riječima. Efekt ovakve karakteristike algoritma u pozadini stoga daje uspješnost prepoznavanja od svega 56%.

7. Zaključak

U ovom završnom radu izrađena je Android aplikacija pod nazivom „Text Scanner“ koja koristi OCR tehnologiju. Aplikacija omogućuje korisnicima da iz slike teksta dohvate digitalizirana slova, postojeće pogreške korigiraju opcijom uređivanja teksta te pohrane tekstualni dokument na svoj uređaj. Aplikacija je testirana na nekoliko uzoraka istog teksta, pri čemu je mijenjana boja pozadine, te boja slova. U trenutnoj izvedbi, rezultati prepoznavanja od iznad 95% postignuti su za sve testirane uzorke, osim u slučaju kada je skeniran bijeli tekst na crnoj podlozi, pri čemu je ostvarena točnost prepoznavanja od svega 56%. U budućem radu, aplikaciju bi se moglo poboljšati na način da se karakteristike algoritama za prepoznavanje teksta iz slike unaprijede ili implementiraju neki drugi algoritmi. Također, funkcionalnost aplikacije bi se mogla proširiti osiguravanjem pohrane skeniranog teksta u .pdf format.

Literatura

- [1] <https://www.elprocus.com/what-is-android-introduction-features-applications/>
- [2] https://en.wikipedia.org/wiki/Android_Runtime#cite_note-anandtech-8231-1
- [3] https://www.linuxtopia.org/online_books/android/devguide/guide/basics/android_what-is-android_application_framework.html
- [4] https://en.wikipedia.org/wiki/Android_Studio
- [5] <https://stackoverflow.com/questions/16754643/what-is-gradle-in-android-studio>
- [6] <https://abhiandroid.com/ui/xml>
- [7] [https://en.wikipedia.org/wiki/Java_\(programming_language\)](https://en.wikipedia.org/wiki/Java_(programming_language))
- [8] <https://www.oracle.com/technetwork/java/intro-141325.html>
- [9] <https://hr.wikipedia.org/wiki/API>

Popis oznaka i kratica

NDK – Native Development Kit

JVM – Java Virtual Machine

OS – operacijski sustav

ARM – Advanced RISC Machine

ART – Android Runtime

VM – Virtual Machine

APK – Android Package

RAM – Random Access Memory

XML – EXtensible Markup Language

UI – user interface

Sažetak

U ovom završnom radu izrađena je Android aplikacija pod nazivom „Text Scanner“ koja koristi OCR tehnologiju. Aplikacija omogućuje korisnicima da iz slike teksta dohvate digitalizirana slova, postojeće pogreške korigiraju opcijom uređivanja teksta te pohrane tekstualni dokument na svoj uređaj. Objasnjeno je postupak izrade mobilne aplikacije za Android OS, dan je pregled relevantnih tehnologija, te je opisana funkcionalnost aplikacije koja primjenjuje algoritam optičkog prepoznavanja znakova na ekstrakciju znakova sa slike dohvaćene mobilnim uređajem. Aplikacija je testirana na nekoliko uzoraka istog teksta, pri čemu je mijenjana boja pozadine, te boja slova.

Ključne riječi: Android OS, OCR, mobilna aplikacija, digitalizacija teksta

Summary

In this work an Android application „Text Scanner“ based on OCR technologies has been developed. Application enables user to acquire digitalized characters from text photographs, edit eventual scanning errors and save scanned text in text format. Process for mobile application development for Android OS has been described, basics of relevant technologies have been introduced and OCR-based mobile application functionalities has been proposed. Test results are given for several test scenarios.

Keywords: Android OS, OCR, mobile application, text digitalization

Prilog

<i>Slika 2.1 Arhitektura Androida</i>	<i>3</i>
<i>Slika 2.2 Usporedba Dalvik i ART arhitekture</i>	<i>4</i>
<i>Slika 3.1 App build.gradle.....</i>	<i>8</i>
<i>Slika 3.2 Koncept XML-a u Androidu</i>	<i>9</i>
<i>Slika 3.3 Lokacija layout XML datoteke</i>	<i>9</i>
<i>Slika 3.4 Izgled layout XML datoteke 1.....</i>	<i>10</i>
<i>Slika 3.5 Izgled layout XML datoteke 2.....</i>	<i>10</i>
<i>Slika 3.6 Lokacija Manifest.xml datoteke.....</i>	<i>11</i>
<i>Slika 3.7 Izgled Manifest.xml datoteke.....</i>	<i>12</i>
<i>Slika 3.8 Lokacija Strings.xml.....</i>	<i>13</i>
<i>Slika 3.9 Izgled strings.xml</i>	<i>13</i>
<i>Slika 3.10 Lokacija styles.xml</i>	<i>14</i>
<i>Slika 3.11 Izgled styles.xml</i>	<i>14</i>
<i>Slika 3.12 Java paketi aplikacije Text Scanner.....</i>	<i>18</i>
<i>Slika 3.13 Funkcija za slikati</i>	<i>18</i>
<i>Slika 3.14 Funkcija skeniranja 1.....</i>	<i>19</i>
<i>Slika 3.15 Funkcija skeniranja 2.....</i>	<i>19</i>
<i>Slika 3.16 Funkcija skeniranja 3.....</i>	<i>20</i>
<i>Slika 3.17 Funkcija spremanja skeniranog teksta u tekstualnu datoteku</i>	<i>20</i>
<i>Slika 5.1 Prikaz Use-case dijagrama</i>	<i>23</i>
<i>Slika 5.2 Izgled sučelja aplikacije, mjesto spremanja i izgled spremljenog dokumenta.....</i>	<i>25</i>
<i>Slika 6.1 Testni uzorak 1</i>	<i>26</i>
<i>Slika 6.2 Testni uzorak 2</i>	<i>26</i>
<i>Slika 6.3 Testni uzorak 3</i>	<i>26</i>
<i>Slika 6.4 Testni uzorak 4</i>	<i>26</i>
<i>Slika 6.5 Rezultat testnog uzorka 1</i>	<i>27</i>
<i>Slika 6.6 Rezultat testnog uzorka 2</i>	<i>27</i>
<i>Slika 6.7 Rezultat testnog uzorka 3</i>	<i>28</i>
<i>Slika 6.8 Rezultat testnog uzorka 4</i>	<i>28</i>
 <i>Tablica 3-1 Kriterij za Android Studio 3.x.....</i>	 <i>7</i>
<i>Tablica 7-1 Tablica uspješnosti skeniranja testnih uzoraka</i>	<i>28</i>