MOBILNA APLIKACIJA ZA PLAG-SCAN S PRETRAŽIVANJEM STRANE LITERATURE

Antunović, Ivan

Sveučilište Jurja Dobrile u Puli
Fakultet Informatike u Puli

IVAN ANTUNOVIĆ

MOBILNA APLIKACIJA ZA PLAG-SCAN S PRETRAŽIVANJEM STRANE LITERATURE

Diplomski rad

Pula, 2021. godine

Sveučilište Jurja Dobrile u Puli
Fakultet Informatike u Puli

IVAN ANTUNOVIĆ

MOBILNA APLIKACIJA ZA PLAG-SCAN S PRETRAŽIVANJEM STRANE LITERATURE

Diplomski rad

JMBAG: 0303068625, redovni student

Studijski smjer: Diplomski studij informatike

Predmet: Mobilne aplikacije

Znanstveno područje: Društvene znanosti
Znanstveno polje: Informacijske i komunikacijske znanosti Znanstvena grana: Informacijski sustavi i informatologija

Mentor: Doc. dr. sc. Siniša Sovilj

Pula, 20. kolovoza 2021. godine

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Ivan Antunović, kandidat za magistra informatike ovime izjavljujem da je ovaj Diplomski rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Diplomskog rada nije napisan na nedozvoljeni način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

U Puli, 30.08.2021

IZJAVA O KORIŠTENJU AUTORSKOG DJELA

Ja, Ivan Antunovic dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj diplomski rad pod nazivom

Mobilna aplikacija za plag-scan s pretraživanjem strane literature

koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, 30.08.2021

Potpis

Sadržaj

UVOD

Upotreba literature,znanstvenih članaka i radova smatra se jednim od najvažnijih resursa za uspješno učenje,istraživanje svijeta oko nas, a posebice za današnje studiranje. Preko takvoga materijala mnogi ljudi dosežu za novim informacijama, donose se samo zaključke i na temelju istih kasnije te zaključke provjeravaju i zapisuju te tako rade novu literaturu i podlogu za učenje budućih generacija. Zasigurno za proučavanje literature, učenje i donošenje novih zaključaka potrebno je mnogo vremena, svaku novu spoznaju ili proizvod koji smo napravili iz vlastitog učenja i iskustva smatramo kao osobnu pobjedu koja nas gura dalje.

Internet je odavno sveprisutan u učenju i izlaganju radova, kao što smo mogli vidjeti i tokom ove pandemije. Preko njega se dolazi puno lakše do literature nego li što je to bilo prije, što je ujedno i najvažnija odrednica njega u učenju (uz naravno popratne alate i sadržaje). Kao što znamo, nažalost internet je i poprište nelegalnih radnji, ali i alat istih.

Kao što je već gore navedeno, puno truda je utkano u učenje i postizanje uspjeha, svatko bi za svoj uspjeh morao biti sam zaslužan, bilo to upornim učenjem ili snalažljivošću. Ovo drugo moralo bi biti u granicama legalnog, pri tome se 'legalno' prvenstveno odnosi na izbjegavanje krađi znanja i varanja. Nije nepoznato da se u današnje vrijeme, što zbog načina života, ali i dostupnosti vrijednog materijala, događa mnogo krađa znanja tj. plagiranja znanstvenih, ali i umjetničkih radova.

Način provjere i suzbijanje takvih nelegalnih aktova naziva se plag-scan (od engleske skraćenice 'scanning' – skeniranje,iščitavanje). Najpoznatiji alat je vjerovano web-servis https://www.plagscan.com koji se trenutno koristi na skoro svakoj obrazovnoj ustanovi. Sprječavanje plagiranja jako je važna borba, ne samo u obrazovnim ustanovama već i u vremenu nakon što čovjek izađe iz istih. Tako naprimjer, ukoliko odlazite iz jedne kompanije u drugu, ne bi ste smjeli pričati o procesu rada, rješenjima i strukturi sustava iz prethodne kompanije. Prvenstveno, to

je nekorektno ponašanje, a uostalom, ukoliko potpišete ugovor, što zasigurno hoćete ukoliko se radi o unikatnim rješenjima ili sustavu, moguće vas je tužiti i kazniti novčano,ali i zatvorski za takvu vrstu akta.

Kreativnost pri takvim radnjama često seže i izvan granica domišljatosti onoga tko je bio zadužen za obranu sustava. Tako na primjer, biti ćete tuženi za plagijat ukoliko se on dokaže u više od 5% rada, no nitko vas neće moći optužiti ukoliko dio za koji se sumnja na plagiranje, nije isti kao točan dio dokumenta iz kojeg se sumnja da ste ga preuzeli, što uvelike olakšava stvar za prikrivanje krađe. Još jedan od trendova je i krađa iz strane literature te prevođenje iste, što je naknadno, još teže dokazati jer treba više vremena i poznavanje stranih jezika i njihove gramatike. Takvi slučajevi i moguća rješenja prezentirani su u ovome diplomskom radu.

1. Načini plagiranja

Da bi se nešto proglasilo plagijatom moraju postavit osnovne za to. Struktura teksta, upotrebljene riječi i određeni znakovi na identičnim mjestima (zarezi, točke,upitnici, razmak između riječi itd.) Razlog plagijata je često neznanje o području o kojem netko radi znanstvenu analizu i dokumentira ju, ali i obična lijenost. Neke od vrsta pokušaja plagiranja su:

1. Potpuno kopiranje – ovdje osoba koja plagira uopće ne vodi računa o tome ili ne zna da bi ga netko mogao uhvatiti u njegovoj radnji. On/ona uzima dio teksta, potpuno ga kopira i predočuje to kao svoj rad. Ovakvu vrstu je najlakše uhvatiti.

2. Mijenjanje nekoliko riječi u dokumentu – ovdje se radi o tome da osoba uzima dio teksta te koristi zamjenske riječi za svaku riječ gdje je to moguće.

3. Mijenjanje poredaka riječi – kada osoba uzme tekst, promjeni redoslijed rečenica ili redoslijed riječi u jednoj rečenici.

Za svaki od navedenih primjera postoje pravila i granice za koje nema nekog univerzalnog mjerila, no poznato je otprilike u kojim postotcima je dio teksta za koji se sumnja dozvoljen imati isti/sličan poredak riječi ili dio rečenice kako ga se ne bi proglasilo plagijatom. Naravno, ne može se tvrdi uvijek sto postotno da je neki rad plagijat, moguće da 2 autora / znanstvenika imaju isto viđenje i isti stil pisanja pa se može pronaći i pokoji redoslijed riječi isti kod jednoga i drugoga. Također bi se trebalo pripaziti i kod nekih terminologija koje se isto pišu svugdje u svijetu kao što su poruke upozorenja, problema ili obavijesti npr. rečenica 'Molimo držite razmak' koja se može naći na autocesti, ulazima u trgovine u jeku pandemije i slično jer tada to ne možemo nazvati pokušajem plagiranja i takve rečenice kasnije moramo ručno isključiti pošto ih sustav neće moći prepoznati.

Domišljatost ponekad nema granica, pa se tako trenutno sve više koristi gore napomenuta krađa iz stranih literature ili iz članaka pronađenih na internetu na stranom tekstu. Ovdje trenutno prednjače novinari i studenti koji najvjerovatnije zbog lijenosti, uzmu tekst koji pronađu, prevedu pomoću 'Google prevoditelja' ili nekog drugog servisa za prevođenje te predstave kao svoj. Čitatelj/profesor neće vjerovatno pomisliti da to nije njihov rad, servis za plagiranje neće otkriti da se radi o plagijatu jer će uspoređivati tekstove koji imaju iste riječi i krajnji rezultat će biti uspješna krađa dokumenta/rada.

2. Načini rada servisa za otkrivanje plagijata

Prepoznavanje plagijata ponekad može biti teško, gore navedeni načini su jedni lakši za otkriti od spomenutog prevođenja teksta ili ubacivanja cijelih rečenica unutar jednog djela teksta koji je prvo kopiran, a za time izmijenjen. U takvim slučajevima zbog prava koji autor dokumenta kojeg pregledavamo ima, ne može ga se optužiti za plagijat, osim ako nije dokumentacija patenta koji već postoji,a da pri tome autor i dalje pušta svoj proizvod (uz dokumentaciju) u opticaj na tržište, no s time se bave neke druge institucije dok im alati i udruge protiv nelegalnih aktiva samo pomažu ili rade na ukazivanju istih. Proces otkrivanja plagijata trebao bi izgledati ovako:

1. Priloži se dokument koji se treba pregledati
2. Pripremi se literatura s popisa literature koju je autor koristio
3. Ukoliko autor ne navodi literaturu, priprema se literatura srodna radu koju je proizveo autor
4. Kroz pripremljenu literaturu, krećemo se kroz dio po dio te uspoređujemo dio po dio autorove dokumentacije
5. Označavamo sumnjive ili očigledno kopirane dijelove
6. Zbrajamo označene dijelove i zbroj podijelimo s količinom dokumenta što nazivamo rezultat
7. Uz određene mjere zaključujemo da li je dokument proglašen legalnim, sumnjivim ili očitim plagijatom. Treba napomenuti da svaka ustanova/država/ alat ima svoje granice u postotcima kada je to plagijat. U ovome radu riječ je o odnosima : <1.5% →nije plagijat

   >1.5% i <5% →mogući plagijat

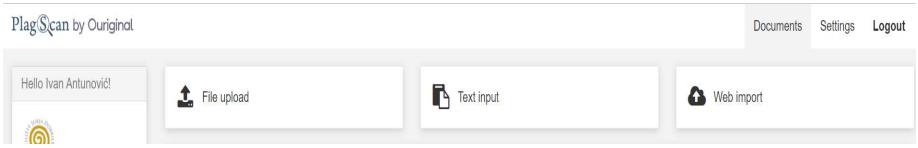
   >5% → očigledni plagijat

Ukoliko nam je dokument sumnjiv, potrebno je ručno pregledati još jednom označene dijelove i tražiti po dodatnoj literaturu slične dijelove onome djelu teksta koji je označen kao sumnjiv.

Što se tiče servisa, već smo u uvodu rekli da je trenutno najpopularniji, a vjerovatno i najbolji 'plag-scan' koji je online servis i kojeg možemo upotrijebiti na web stranici www.plagscan.com.

Upute:

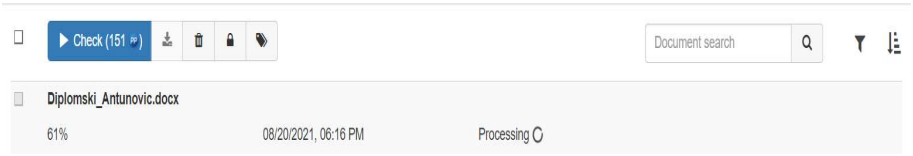
1. Učitamo file koji može biti vrste pdf, word dokument ili tekstualna datoteka (.txt). Ukoliko nemamo file, možemo jednostavno kopirati tekst i postaviti ga unutar prozora 'text input' ili postaviti URL kako bi plag-scenirali mjesto na web stranici.



Slika 1. Način predaje dokumenta za pregled

Izvor: slika zaslona napravljena od autora na stranici www.plagscan.com

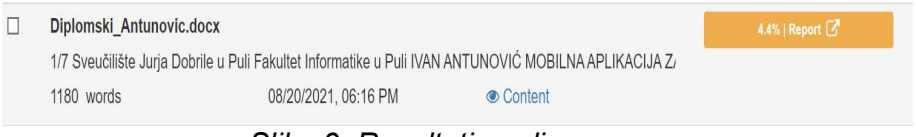
2. Nakon toga, pričekati da servis analizira tekst i usporedi sa bazom znanja i pretraživanjem na internetu.



Slika 2. Trajanje analize dokumenta

Izvor: slika zaslona napravljena od autora na stranici www.plagscan.com

Kada analiza bude gotova, dobit ćemo rezultate uspoređivanja teksta, u ovom slučaju rezultat je 4.4% zbog toga što servis analizira cijeli tekst,a naslovna stranica teksta i izjave o čestitosti imaju isti iste rečenice i riječi za svakog studenta.



Slika 3. Rezultati analize

Izvor: slika zaslona napravljena od autora na stranici www.plagscan.com

3. Prednosti i mane ovakvog servisa

Ovakav servis je jednostavan za korištenje, daje većinom točne rezultate tj. rezultati mu prvenstveno ovise o snalažljivosti osobe koja radi plagijat i trudi se da ju ne otkriju. No i u većini slučajeva je tada vrlo sposoban odrediti da li se treba sumnjati na plagijat ili ne i da li je očigledno da je plagijat. Tu nam pomaže i detaljan report na kojeg dolazimo klikom na gumb 'Report' koji se nalazi u segmentu dokumenta koji smo analizirali (na slici 3). Pri ulasku u report, s lijeva strane će nam se stvoriti lista linkova i nazivi literatura ili web stranica s kojih je vađen tekst za usporedbu s dokumentom kojeg analiziramo dok će se na središnjem djelu pronaći cjeloviti dokument s oznakama na djelu teksta za koji se sumnja da je plagiran. Tu dolazimo do prve mane ovog sustava. Kao što vidimo dolje na slici 4, servis proglašava za sumnjiv dio teksta čak i naziv sveučilišta jer se pojavljuje u mnogim dokumentima prije toga kao i grad i riječ 'godine'. Zbog toga se na svaki ovaj report postavlja poruka odricanja od odgovornosti od strane servisa. Također, korisniku može izgledati da je jedna od mogućih mana sustava i preveliko čekanje na analizi i najmanjih dokumenata, no gledajući s tehničke strane, takvo trajanje je normalno; što zbog brzine interneta i količine podataka, što zbog toga što se analiziraju veliki dokumenti gdje se skoro svaka riječ i struktura najmanjih dijelova teksta detaljno provjeravaju.

Prednosti:

1. Lijepo dizajnirano korisničko sučelje
2. Izvrstan UX
3. Većinom precizne i točne analize
4. Ukazivanje na moguća web mjesta i literaturu koji bi mogli biti pravi izvor
5. Ukazivanje na moguće dijelove teksta koje su plagirani

Sveučilište Jurja Dobrile u Puli
Fakultet Informatike u Puli

IVAN ANTUNOVIĆ

MOBILNA APLIKACIJA ZA PLAG-SCAN S PRETRAŽIVANJEM STRANE LITERATURE

Diplomski rad

Slika 4. Dokument s naznačenim sumnjivim dijelovima

Izvor: slika zaslona napravljena od autora na stranici www.plagscan.com

Također,sa lijeve strane sučelja možemo vidjeti i mali prozor sa sadržanim linkovima adresa web mjesta i nazive web stranica ili pisane literature(knjige, znanstveni članci, ostali radovi), a koji bi mogli biti ili na koje se sumnja da su originalni izvor djelova teksta.

Slika 5. Prozor s linkovima

Izvor: slika zaslona napravljena od autora na stranici www.plagscan.com

4. Prijevod teksta kao nova značajka

U ovom diplomskom radu obrađena je ideja nove značajke jednog ovakvog servisa ,a to je da se tekst dokumenta koji se pregledava prevodi na odabrani jezik ukoliko za to postoji sumnja da se plagiranje dogodilo na sljedeći način:

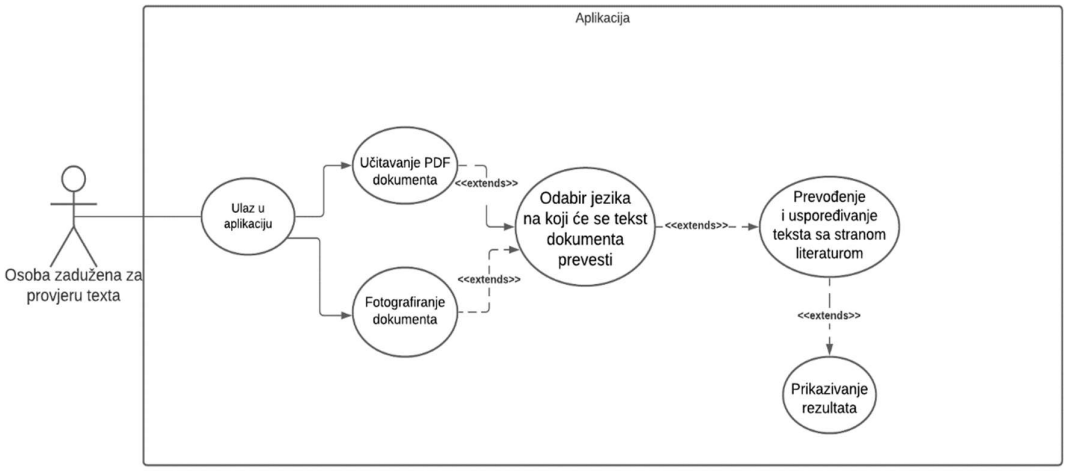
1. Autor plagijata je odabrao/pronašao stranu literaturu.
2. Dijelove literature je preveo te stavio u svoj rad.
3. Nakon toga je pokušao izmijeniti redoslijed riječi ili umetnuti nove u rečenicu.

Opisani proces, ukoliko ne postoji već prevedena literatura i ako nije dostupna servisu, je nemoguće ocijeniti kao plagijat jer za to servis ne nudi nikakve dokaze. Često za ovakvu soluciju profesori i ljudi koji se bave borbom protiv krivotvorenja nisu čuli ili za tako nešto nisu imali rješenje. Ovakav problem je moguće ublažiti ako bi se dokument koji analiziramo prevodio na određeni jezik i rečenicu po rečenicu uspoređivao i tražio da li se koji dio teksta poklapa sa literaturom tj. uz prijevod dokumenta moguće bi bilo napraviti osnovni plag scan jer on radi po principu da traži identičan tekst, a koji je jezik u pitanju tada nije važno. Ovo za sobom također vuče jednu manu, a to je da se ukoliko dođe do prevođenja, prijevod mora odraditi preko jednog od dostupnih API-a tj. servisa za prevođenje kao što je 'Googlov Mlkit Translator' ili 'Google prevoditelj'. Oni nisu u mogućnosti pregledati i prevesti cijeli tekst od jednom već dijele tekst na segmente,a zatim jedan po jedan prevode i u obliku odgovora vraćaju sklopljen i preveden tekst. To znači da rad servisa produžuje kada govorimo o vremenu izvođenja.

Također jedna od mana je i ta da se prijevod može razlikovati kada se radi o kontekstu rečenice, djelomično točnom prijevodu kada su u pitanju jezici koji za jednu stvar imaju više istoznačnica – tada kroz prijevod dobijemo drugu riječ u rečenici, mijenja se kontekst, a s time i smanjuje raspoznavanje plagijata što kasnije može bitno utjecati na ukupni postotak pronađenog plagiranog teksta.

5. Ideja rješenja

Jedno od mogućih rješenja je razrađeno u ovome radu u vidu mobilne Android aplikacije koja može na 2 načina uzeti dokument: učitavanjem sa lokalnog mobilnog uređaja ili fotografiranjem. Nakon toga putem aplikacije odabirete jedan od ponuđenih jezika za prijevod. Tekst se prevodi, analizira i aplikacija daje rezultate pretrage. Detaljniji prikaz odnosa funkcija i redoslijeda prikazuje sljedeći dijagram.

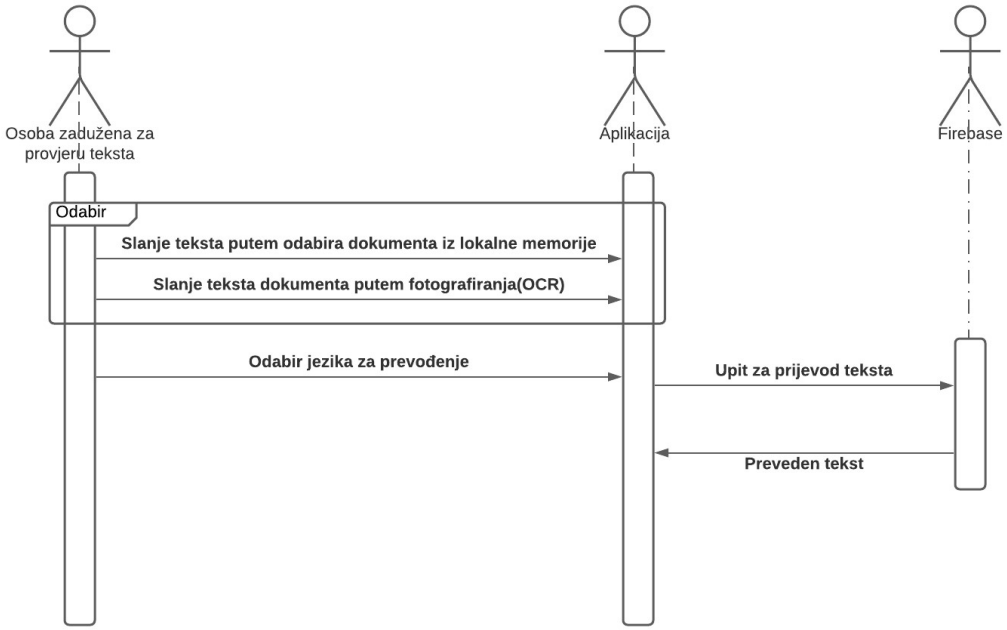


Slika 6. Use case dijagram

Izvor:autor

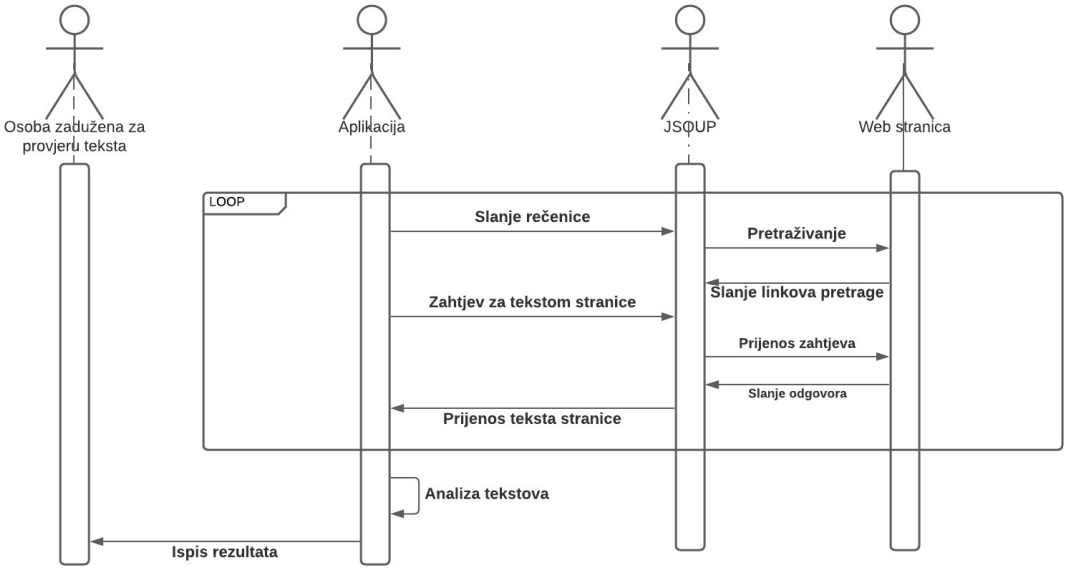
Kako bi se sve što je više pravilno izvršavalo i kako bi funkcije uspješno bile izvršene potrebno je:

1. Poslati upit servisima za prevođenje tj. u ovom slučaju 'ML KIT Translatoru' i putem JSOUP-a (Java servis za potraživanje HTML elemenata i izvršavanja upita na mrežu) napraviti pretragu s određenim dijelovima teksta, u ovom slučaju rečenicama.
2. Nakon toga krećemo sa upitima za pretraživanje na pretraživaču 'Google' i to rečenicu po rečenicu kako bi dobili što više relevantnih linkova.
3. Pomoću linkova šaljemo još jedan upit putem JSOUP-a te izvlačimo tekstove sa web stranica čije su adrese bili navedeni linkovi.



Slika 7. Sequence dijagram 1.dio

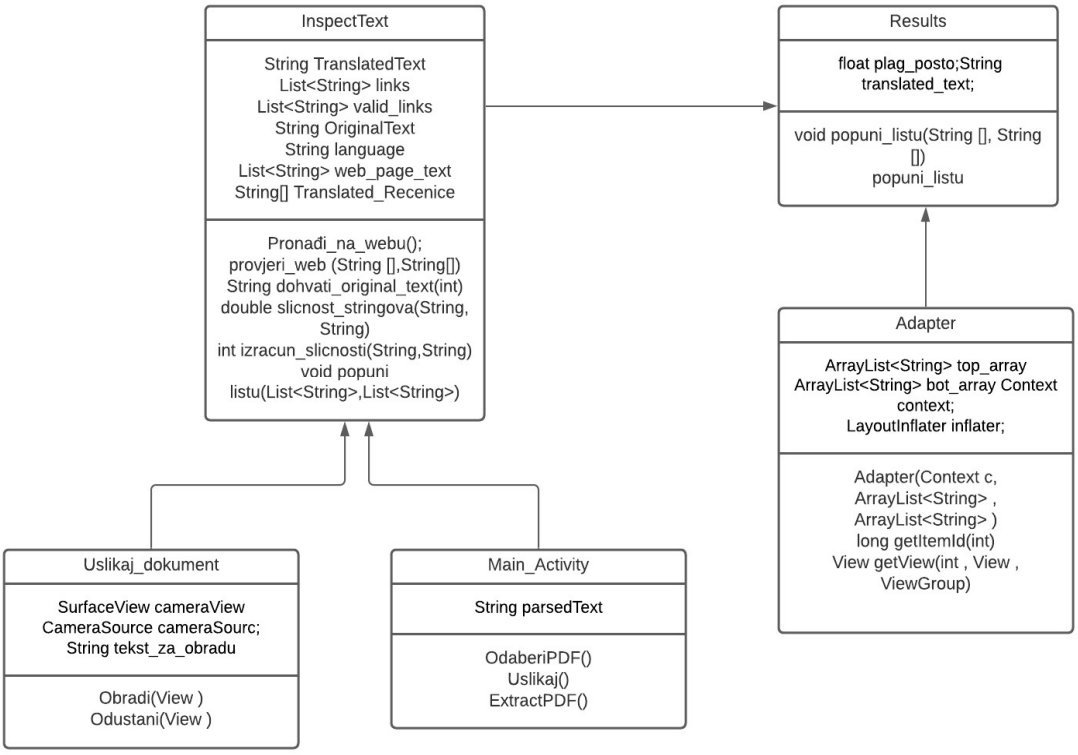
Izvor: autor



Slika 8. Sequence dijagram 2.dio

Izvor: autor

Kada je u pitanju klasni dijagram, prikazan na slici 9, on predočava klase i njihove međusobne odnose. Sve se zapravo vrti oko klasa 'Results' i 'InspectText'. 'Results' dobiva rezultate iz klase 'InspectText', a redoslijed prikaza istih i obrađenu listu od klase 'Adapter' koja osim za preslagivanje rezultata, služi i za pravljenja rasporeda u prikazivanju istih. 'InspectText' dobiva preveden tekst za obradu u obliku stringa, kojeg kasnije obrađuje i analizira sa sadržajima pronađenim na internetu. Takav preveden tekst može dobiti od 2 izvora: iz klase 'Uslikaj dokument' ili iz prvobitne, početne, 'Main_Activity'.



Slika 9. Klasni dijagram

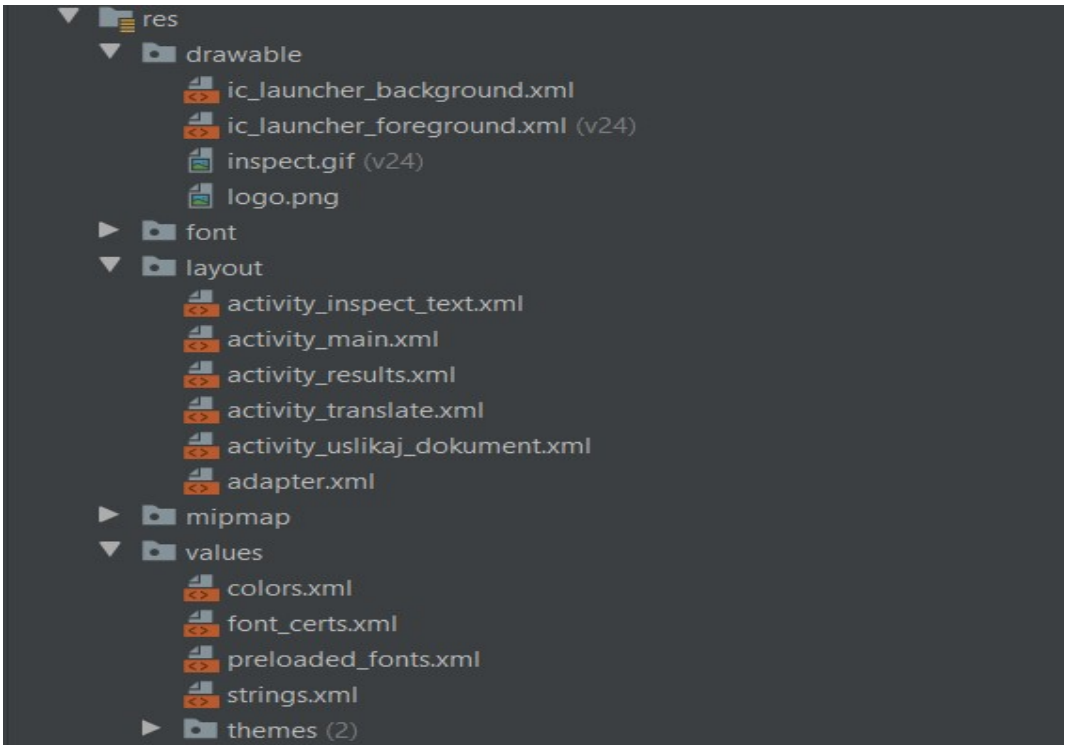
Izvor: autor

6. Postavljanje okruženja

6.1 Mape za izradu GUI-a

Za ovakvu ideju korišten je 'Android studio', tj. integrirano razvojno okruženje u koje je moguće pisati kod u jeziku 'Java' ili 'Kotlin'. Koncept izrade aplikacije u takvom okruženju temelji se na jednoj ili više aktivnosti tj. 'Activity' formi gdje u klasama svake forme piše se programski kod unutar 'Jave', dok dizajn i obrada vizualnih elemenata se može pisati unutar 'res/layout' mape koja obrađuje kod napisan u XML jeziku.

Unutar dizajn segmenata također postoje i druge mape poput 'drawable' gdje smještamo slike koje želimo umetnuti u okvire poput 'ImageView' ili tome slično. Također postoje mape poput 'values' gdje možemo postaviti konstantne varijable koje kasnije koristimo u 'layoutu', 'mipmap' direktorij nam služi za postavljanje ikone prečaca koja je vidljiva na mobilnom uređaju s Android operativnim sustavom.

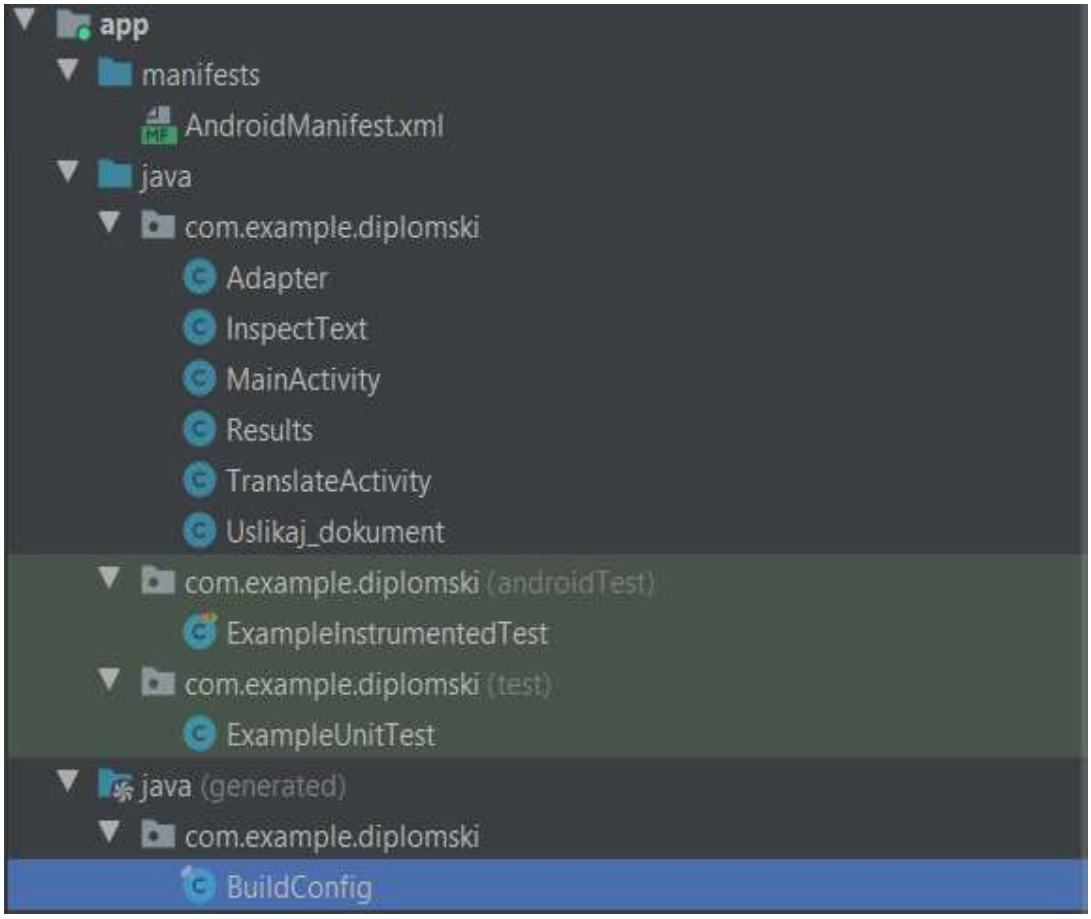


Slika 10. mape i XML datoteke aplikacije

Izvor: autor

6.2 Mape iz kojih se izvršava programski kod

U direktoriju 'Java/ime_aplikacije' nalaze se datoteke sa 'Java' kodom koje kada se kompajliraju, ukoliko nema grešaka koje narušavaju samo izvršavanje sustava pri njegovom početnom prvobitnom dizanju, daju funkcionalnost samoj aplikaciji. Pored tog direktorija nalazima i direktorije s imenom paketa aplikacije koje su služile okruženju za testiranje rada aplikacije tj. provjeru valjanosti koda i sintakse . Ovakve testove okruženje radi samostalno pri gradnji procesa aplikacije kako bi se uvjerilo da nema grešaka koji bi ozbiljnije naštetili mobilnom uređaju i da izbjegne sukob pravilnog rada sustava s radom koji je zadan programskim kodom.Unutar 'java(generated) pronaći ćemo datoteku 'BuildConfig' tj. konfiguraciju pokretanja i izrađene aplikacije.



Slika 11. Datoteke s programskim kodom

Izvor: autor

Unutar datoteke manifest.xml nalaze se sve dopuštenje radnje aplikacije prema mobilnom uređaju na kojem se ista izvršava. Također tu se nalazi i sav važniji popis datoteka i puteva do istih koje aplikacije koristi kao generalni izgled (koje datoteke se uzimaju za sliku početne ikonice, 'meta-data' odnosno način rada i obrade podataka unutar aplikacije, samo ime paketa koji sadrži sve direktorije u kojima se nalaze programski kodovi i ostalo za valjan rad aplikacije.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.diplomski">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.CAMERA" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Diplomski">
        <activity android:name=".Uslikaj_dokument"></activity>
        <activity android:name=".Results" />
        <activity android:name=".InspectText" />
        <activity android:name=".TranslateActivity" />
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" /
            </intent-filter>
        </activity>

        <meta-data
            android:name="preloaded_fonts"
            android:resource="@array/preloaded_fonts" />
    </application>

</manifest>
```

Slika 11. XML kod unutar Manifesta

Izvor: autor

6.3 Gradle skripte

Skripte koje se koriste za uspješno kompajliranje koda i za uvoz određenih paketa iz drugih okruženja nazivaju se Gradle skripte. Razlikujemo dvije vrste takvih skripti : one koje se odnose na razini cijelog 'Projecta' i one koje se odnose samo na razini jednog modula aplikacije.

Na razini projekta skripta pod nazivom 'build.gradle' je zapravo lista poziva u kojoj se nalaze paketi pozvani iz drugih okruženja na razini jednog projekta. U ovom slučaju to su 'google', 'Maven' i 'Jcenter'. Drugi segment je 'dependencies' tj. način i verzije na koji će okruženje gledati na 'build scriptu' i segment 'allproject' gdje se nalaze zbrojeni i izlistani pozivi za uvoz ostalih biblioteka iz drugih pomoćnih alata/okruženja svih projekata povezanih s trenutnim.

```
// Top-level build file where you can add configuration options common to all sub-projects/modules.
buildscript {
    repositories {
        google()
        jcenter()
        mavenCentral()
    }
    dependencies {

        classpath "com.android.tools.build:gradle:4.1.1"
        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        google()
        jcenter()
        mavenCentral()

    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

Slika 12. Kod 'build.gradle' skripte na razini projekta

Izvor: autor

Druga 'build.gradle' skripta je ona na razini jednog modula aplikacije. Ona sadrži segmente u kojima se nalaze pozivi pomoćnih alata, nazivi verzija koji se koriste za kompajliranje koda, ali i kao poziv određene verzije vanjskog okruženja. Pod segmentom 'plugins' nalaze se svi plug-inovi,kao što i samo ime kaže, no u ovom slučaju ih nema te se tada nalazi samo osnovni plug-in tj. android paket. Kada bi oni postojali, pozivali bi se na način da se upiše **id** 'id_plug-ina'.

Segment 'android' nam govori u kojoj trenutno verziji 'Android studio' okruženja radimo, koje verzije se koriste za 'build',a koje za 'compile', a uz atribut 'useLibrary' odabiremo početnu biblioteku s početnim pozivima već gotovih funkcija iz okruženja. Kao i kod skripte koja se temelji na razini cijelog projekta, i ovdje postoje 'dependencies' gdje su uvežene određene datoteke iz određenih verzija drugih okruženja i programskih rješenja.

```
dependencies {

    implementation 'androidx.appcompat:appcompat:1.3.0'
    implementation 'com.google.android.material:material:1.3.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    implementation 'com.google.mlkit:translate:16.1.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
    implementation 'com.itextpdf:itextg:5.5.10'
    implementation 'org.jsoup:jsoup:1.10.3'
    implementation 'pl.droidsonroids.gif:android-gif-drawable:1.2.20'
    compile 'com.google.android.gms:play-services-vision:20.1.3'
}
```

Slika 13. 'Dependencies' ovog projekta

Izvor: autor

7. Izrada rješenja

7.1 Početna klasa i početni zaslon

Da bi aplikacija radila uredno i da bi se izbjeglo urušavanje rada iste, potrebno je prvo konfigurirati Manifest.xml i 'gradle' skripte. Nakon toga se može početi pisati programski kod, poštujući svu sintaksu jezika u kojem pišemo, u ovom slučaju 'Jave', ali također se mora poštovati i pozivi za uvoz biblioteka iz stranih i osnovnih dijelova okruženja. Takvi pozivi se unutar jedne klase rade iznad same deklaracije klase. Iznad poziva nalazi se naziv paketa za kojeg uključujemo rad klase. Primjer kao na slici 14.

```
package com.example.diplomski;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

import androidx.appcompat.app.AppCompatActivity;

import com.itextpdf.text.pdf.PdfReader;
import com.itextpdf.text.pdf.parser.PdfTextExtractor;

import java.io.FileNotFoundException;
import java.io.InputStream;
```

Slika 14. Uvoz biblioteka i naziv paketa

Izvor: autor

Nakon toga 'Android' okruženje samo postavlja kostur glavne funkcije koja počinje s deklaracijom naziva klase ispod koje se mogu napisati globalne varijable ili konstante,a koje će biti dostupne iz svih lokalnih funkcija jedne klase. Lokalne funkcije pišu se izvan definicije glavne klase, ali se iz nje mogu pozivati. Također, jako je važno znati da se varijable deklarirane unutar glavne funkcije ne mogu pozivati u lokalnim

funkcijama, kao i to da se pozivi elemenata iz .xml datoteka neće moći raditi prije linije koda *setContentView(R.layout.activity_main);* zbog toga što tek od te linije koda klasa zna kojoj formi odgovara. Naravno, klasa može biti napisana za jednu formu,a kasnije poštujući sva pravila sintakse i postojanja/imenovanja elemenata unutar formi, odgovarati nekoj drugoj formi tako što se u pozivu funkcije *setContentView* postavi parametar 'R.layout.naziv_forme'.

Pozivanja grafičkih elemenata odvija se tako da deklariramo pomoćnu varijablu iste vrste kao i grafičkog elementa kojeg pozivamo, deklariramo naziv pomoćne varijable, a zatim pozovemo grafički element putem funkcije *findViewById(R.id.naziv_grafičkog_elementa)*. Na slici 15 se nalazi primjer.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Button button = (Button) findViewById(R.id.button);

    button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) { odaber_Pdf(); }
    });
}
```

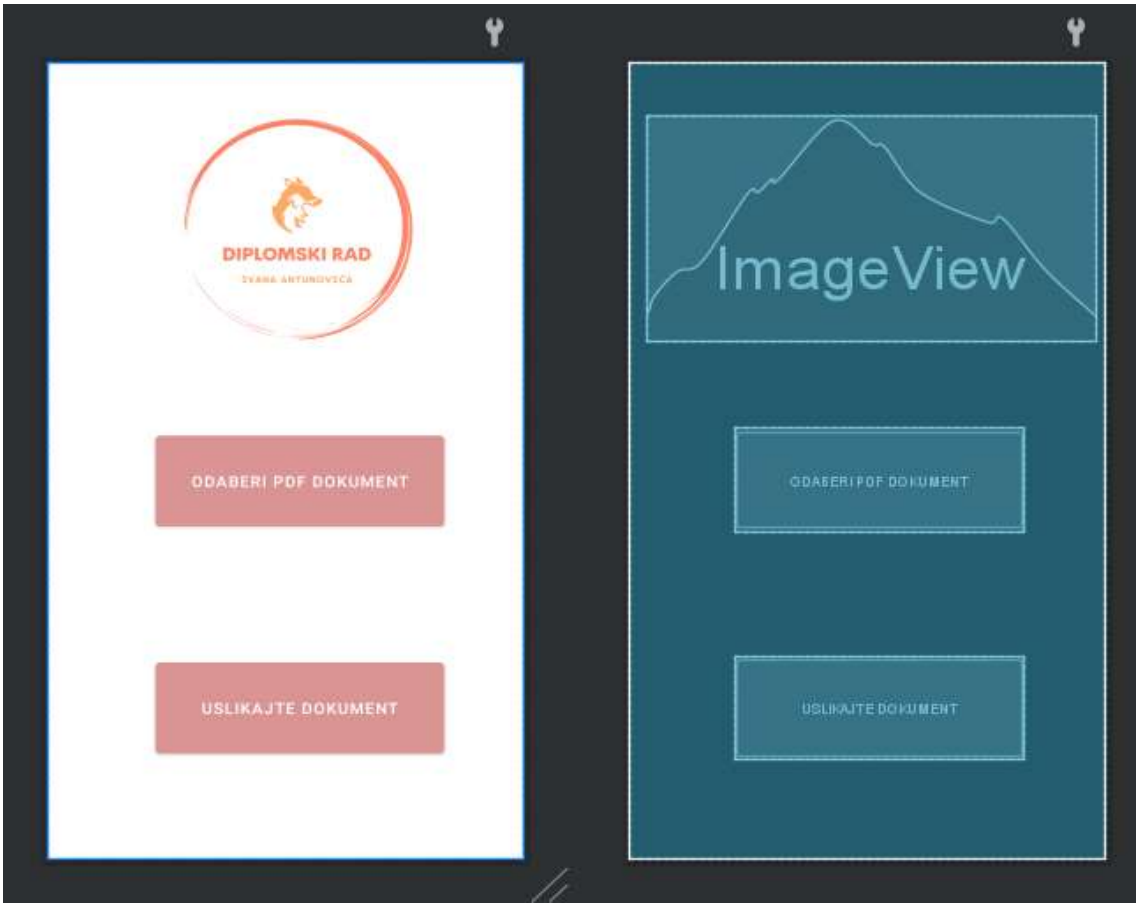
Slika 15. Deklariranje pomoćne varijable za kontrolu grafičkog elementa

Izvor: autor

Na slici 15 je primjer kako se to radi s 'Buttonom' odnosno s 'gumbom' na grafičkom sučelju. Takav element ima i opciju da mu se dodjeli 'Listener' odnosno funkcija koja će sa svakim klikom na 'gumb' pozvati funkciju 'onClick'. Ista solucija je moguća uz drugačiju sintaksu, tj. moguće je deklarirati i naziv funkcije u .xml datoteci unutar bloka za grafički element, a potom u Java datoteci napraviti definiciju funkcije. Takav primjer je vidljiv na slici 17.

Početni zaslon aplikacije ima dva 'gumba'. Oba vode ka funkcionalnosti za skupljanje polaznog teksta iz dokumenta kojeg obrađujemo. Prvi gumb služi za to da iz lokalnog repozitorija mobitela uzmemo .pdf datoteku, a iz nje se onda iščitava tekst te prosljeđuje sljedećoj aktivnosti/klasi koja će ga prevesti. Drugi gumb vodi na aktivnost gdje postoji mogućnost, ukoliko to korisnik dopusti (aplikacija će pri prvom korištenju ove funkcionalnosti upitati za dopuštenje) da fotografira dokument. Dolje na slici je prikazano s lijeve strane izgled kakav bi se trebao pojaviti u slučaju pravilnog pozicioniranja pri pokretanju aplikacije. Na lijevoj strani su prikazani elementi grafičkog prikaza. Pozicioniranje je moguće putem tri načina :

1. 'Drag and drop '
2. S lijeva strane nalaze se postavke atribute te tamo upisati visinu, širinu i pravilo pozicioniranja u odnosu na okvir ili na drugi element
3. U XML datoteci kao na slici 17



Slika 16. Grafički elementi Main_activity forme

Izvor : autor

Izgled grafičkog sučelja može se kao što je navedeno prije, uređivati i preko xml koda. Tako npr. postavke poput 'andoird:id' daje ime jednom elementu,'android:text' postavlja tekst na element. Uz to, može ih se centrirati ili udaljavat od drugog elementa ili od ruba zaslona pomoću 'contraintEnd', 'constraintStart' itd.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <Button
        android:id="@+id/button"
        android:layout_width="250dp"
        android:layout_height="95dp"
        android:layout_marginTop="336dp"
        android:text="Odaberi PDF dokument"
        app:backgroundTint="#DA9494"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.571"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/uslikaj_doc"
        android:layout_width="250dp"
        android:layout_height="95dp"
        android:layout_marginBottom="92dp"
        android:onClick="Uslikaj"
        android:text="Uslikajte dokument"
        app:backgroundTint="#DA9494"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.571"
        app:layout_constraintStart_toStartOf="parent" />
```

Slika 17. dio xml koda u formi 'MainActivity'

Izvor: autor

Kad se uspješno postave sve grafički elementi, deklariraju se i povežu unutar klase odgovorne za formu, kreće se radili logika i definirati način obrade podataka i redoslijed izvršavanja procesa. Ako se klikne na gumb 'Odaberi pdf' dokument, pokreće se 'onClick' listener koji zatim pokreće funkciju 'Odaberi_pdf' unutar koje navodimo novi 'Intent' (hr. namjera) koji obavještava Android sustav da se dogodio ili se sprema dogoditi događaj kao što je ulazak u internu memoriju mobitela.

Postavimo kategoriju 'Intenta' na 'Openable' kako bi moglo ući u memoriju, deklarira se namjera s 'ACTION_OPEN_DOCUMENT' argumentom, postavi se tip dokumenta koji se traži pomoću argumenta (“application/pdf“) unutar funkcije .setType. Nakon toga, korisnik će kliknuti na jedan od mogućih dokumenata te će aplikacija to registrirati. Kako bi povodom događaja registriranja jednog takvog događaja aplikacija nastavila činiti nešto, mora se postaviti i 'event activity' tj. funkcionalnost koja se okida kada se dogodi događaj koji je gore naveden.

```
private void odaberi_Pdf(){
    Intent intent = new Intent(Intent.ACTION_OPEN_DOCUMENT);
    intent.addCategory(Intent.CATEGORY_OPENABLE);
    intent.setType("application/pdf");
    startActivityForResult(intent,CHOOSE_PDF_FROM_DEVICE);

}
```

Slika 18. definicija funkcije 'odaberi_Pdf'

Izvor: autor

Nakon toga, ukoliko unutar funkcije 'startActivityForResult' se provjerava da li je korisnik dao dopuštenje za otvaranje dokumenta i da li otvaranje uspješno izvršeno, kreće se prema vađenju teksta iz .pdf datoteteke koju smo odabrali. Put do datoteke ostaje zapisan te se postavlja u varijablu tipa 'InputStream' koja služi za čitanje objekata koji nisu definirani unutar funkcije/klase.

Nakon toga kreiramo instancu objekta klase 'PdfReader', klase iz biblioteke ' itextpdf.text.pdf.PdfReader' koja dolazi iz okruženja 'itext'. Pdf datoteku predajemo kroz pdfReader instancu u kojoj se nalazi svaka stranica i svaki redak zasebno iščitani,a koji se vade i pridodaju stringu koji će biti poslan drugoj aktivnosti zaduženoj za prevođenje.

Iščitavanje teksta često zna sadržavati previše praznog prostora što kasnije usporava prijevod. Zbog toga se mora učiniti i čišćenje teksta pomoću 'replace' funkcije. Primjer koda na slici 19.

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if ( requestCode == CHOOSE_PDF_FROM_DEVICE && resultCode == RESULT_OK ){
        ExtractPDFText(data.getData());
    }
}
public InputStream inputStream;
private void ExtractPDFText(Uri uri){

    try {
        inputStream=MainActivity.this.getContentResolver().openInputStream(uri);
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }
    try{
        String parsedText="";
        PdfReader reader = new PdfReader(inputStream);
        int n = reader.getNumberOfPages();
        for (int i = 0; i <n ; i++) {
            parsedText   = parsedText+PdfTextExtractor.getTextFromPage(reader,  pageNumber: i+1).trim()+"\n";
        }
        reader.close();
        Intent intent = new Intent( packageContext: MainActivity.this, TranslateActivity.class);
        Bundle b = new Bundle();
        parsedText= parsedText.replace( target: "\br",  replacement: " ").replace( target: "\n",  replacement: " ")
                              .replace( target: "\r",  replacement: "");
        b.putString("text", parsedText);
        intent.putExtras(b);
        startActivity(intent);
        finish();
    } catch (Exception e) {
        System.out.println(e);
    }
}
```

Slika 19. Kod za iščitavanje pdf datoteke u string

Izvor: autor

Nakon što se tekst očistio, potrebno je poslati ga sljedećoj aktivnosti. Pošto druga aktivnost ne zna da postoji tekst iz prethodne, to joj se mora navesti kada se radi deklaracija za prelazak u drugu aktivnost.

Definira 'intent' u čije argumente postavljamo ime trenutne klase , dok za drugi argument po se postavlja klasa u koju se želi doći. Kako bi druga klasa dobila tekst, moramo mu dati ključ pod kojim će ga ona naći kada aplikacija bude koristila njezinu formu. S naredbom '.putExtras' postavlja se 'Bundle'(hr.paket) u kojeg se opet postavlja string s naredbom '.putString'. Za kraj pokreće se nova aktivnost s funkcijom 'startActivity' gdje kao argument se postavlja 'intent'

7.2 Fotografiranje dokumenta

Kako bi se mogućnost fotografiranja s mobitelom mogla uopće ostvariti najvažnije je za to postaviti određene postavke: unutar manifesta napraviti dozvolu pomoću linije koda *uses-permission android:name="android.permission.CAMERA"* , zatim napraviti odgovarajući podlogu odnosno grafički element 'Surface_view' i za isti podesiti postavke unutar 'layout' datoteke pisane u xml-u. Primjer xml koda nalazi se na slici 20, a vizualni design kao rezultat koda na slici 21. Forma sadrži naziv 'Uslikaj_dokument'

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

    tools:context=".Uslikaj_dokument"
    >


    <SurfaceView
        android:id="@+id/surface_view"

        android:layout_width="412dp"
        android:layout_height="664dp"
        android:layout_alignParentStart="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentRight="true"
        android:layout_marginStart="0dp"
        android:layout_marginLeft="0dp"
        android:layout_marginTop="0dp"
        android:layout_marginRight="-2dp" />
```

Slika 20. XML kod potreban za SurfaceView

Izvor: Autor

Slika 21. Design forme 'Uslikaj_dokument'

Izvor: Autor

Nakon podešvanje vizualne forme, potrebno je uvesti sve biblioteke Googlovog OCR API-a koje se pozivaju sljedećim linijama koda:

import com.google.android.gms.vision.CameraSource;

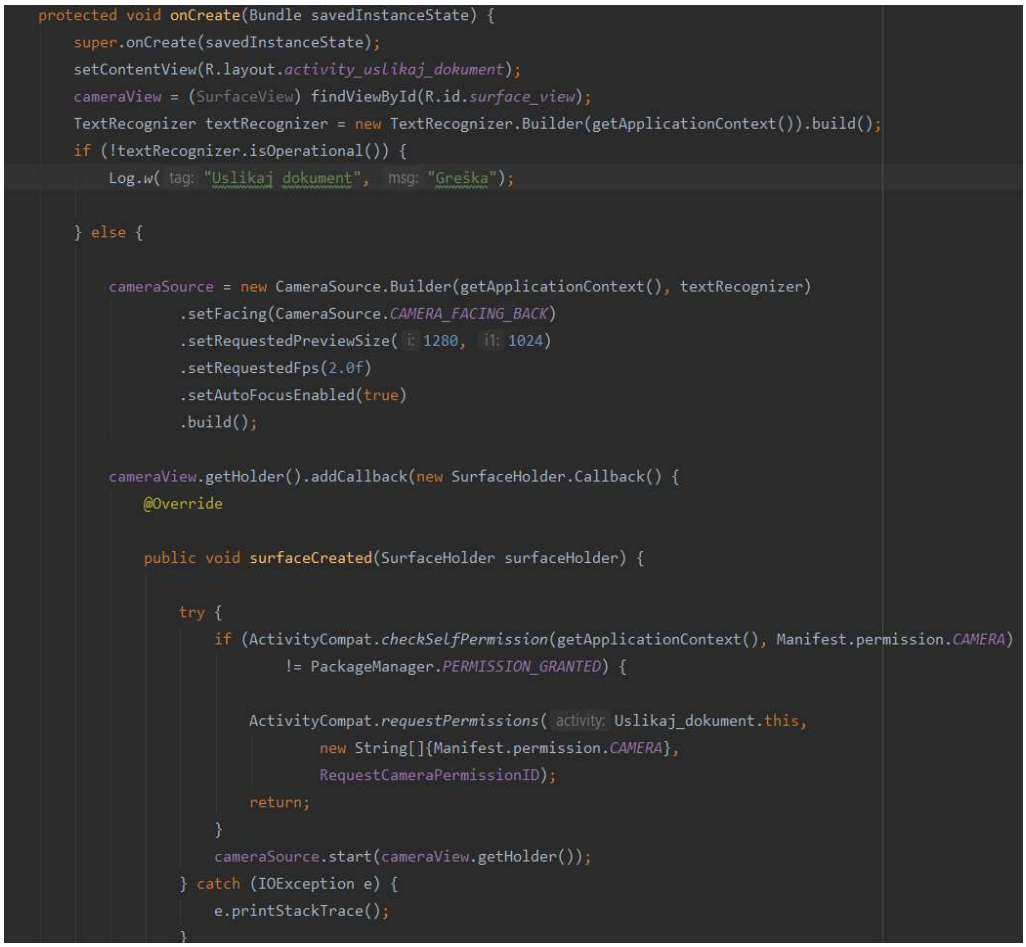
import com.google.android.gms.vision.Detector;

import com.google.android.gms.vision.text.TextBlock;

import com.google.android.gms.vision.text.TextRecognizer;

Kada su importane sve potrebene biblioteke, iz klase 'TextRecognizer' pravimo novu instancu koja počinje s radom čim klasa aktivnosti prepozna svoju odgovarajuću formu. Pri prvom pokretanju ove forme, aplikacija će tražiti dopuštenje za pristup rada kamere. Ukoliko ne dobije dopuštenje,ista će se vratiti u formu prije. Kada dobije dopuštenje, pozicionirat će se na kameru sa zadnje strane mobitela te započeti novu dretvu koja će sinkrono radu aplikacije prikupljati sve što prepozna kao tekst i njega spremati u string. Pri samom gašenju forme ili prebacivanjem na novu, dretva će se uništit. Gašenje forme potiče jedna od funkcija 'Odustani' ili 'Obradi'. Funkcija odustani vraća nas na početni zaslon, dok funkcija 'Obradi' pokreće rad sljedeće aktivnosti koja nam služi za prevođenje. Stvara se novi 'intent' koji u sebi kao dodatan paket nosi tekst koji je uspio zabilježiti za vrijeme aktivnog trajanja dretve. Na slici 22. vidljiv je

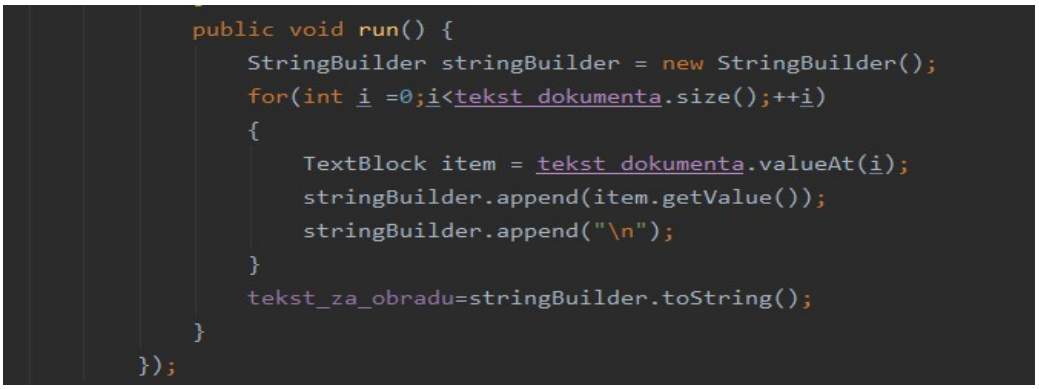
kod pokretanja i traženja dopuštenja za kameru, dok je na slici 23. vidljivo skupljanje teksta prepoznatog od strane OCR-a i njegovo prebacivanje u sljedeću aktivnost.



```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_uslikaj_dokument);
    cameraView = (SurfaceView) findViewById(R.id.surface_view);
    TextRecognizer textRecognizer = new TextRecognizer.Builder(getApplicationContext()).build();
    if (!textRecognizer.isOperational()) {
        Log.w( tag: "Uslikaj dokument",  msg: "Greška");

    } else {

        cameraSource = new CameraSource.Builder(getApplicationContext(), textRecognizer)
                .setFacing(CameraSource.CAMERA_FACING_BACK)
                .setRequestedPreviewSize( i: 1280,  i1: 1024)
                .setRequestedFps(2.0f)
                .setAutoFocusEnabled(true)
                .build();

        cameraView.getHolder().addCallback(new SurfaceHolder.Callback() {
            @Override

            public void surfaceCreated(SurfaceHolder surfaceHolder) {

                try {
                    if (ActivityCompat.checkSelfPermission(getApplicationContext(), Manifest.permission.CAMERA)
                            != PackageManager.PERMISSION_GRANTED) {

                        ActivityCompat.requestPermissions( activity: Uslikaj_dokument.this,
                                new String[]{Manifest.permission.CAMERA},
                                RequestCameraPermissionID);
                        return;
                    }
                    cameraSource.start(cameraView.getHolder());
                } catch (IOException e) {
                    e.printStackTrace();
                }
```

Slika 21. Pokretanje OCR-a i traženja dopuštenja za rad kamere

Izvor: autor



```
            public void run() {
                StringBuilder stringBuilder = new StringBuilder();
                for(int i =0;i<tekst_dokumenta.size();++i)
                {
                    TextBlock item = tekst_dokumenta.valueAt(i);
                    stringBuilder.append(item.getValue());
                    stringBuilder.append("\n");
                }
                tekst_za_obradu=stringBuilder.toString();
            }
        });
```

Slika 22. Prijenos teksta iz OCR-a u string

Izvor: autor

7.3 Prevođenje teksta dokumenta

Prevođenje dokumenta sa hrvatskog jezika na neki drugi jezik odvija se na način da korisnik odabere jedan od ponuđenih jezika, program prihvati opciju, pozove ML KIT API za prevođenje, prenese mu request, a zatim u responseu dobije preveden tekst na odabrani jezik. Nakon toga aplikacija prenosi prevedeni i originalni tekst na sljedeću aktivnost. Dok prijevod traje, zbog estetskih i UX razloga, postavljen je grafički element 'ProgressBar' odnosno animacija kružnog procesa koja je sakrivena dok zato ne dođe vrijeme. Biranje jezika na koji će se prevodit, u vizualnom smislu riješeno je elementom 'RadioButtona' odnosno grupom opcija gdje je potrebno odabrati jednu opciju i pritisnuti gumb 'Prevedi'. Kod xml koda nalazi se na slikama 23 i 24, a poredak grafičkih elemenata na slici 25.

```
<ProgressBar
    android:id="@+id/progressBar1"
    style="?android:attr/progressBarStyle"
    android:layout_width="402dp"
    android:layout_height="736dp"
    android:background="#FAFAFA"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.478"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<RadioGroup
    android:id="@+id/GrupaJezika"
    android:layout_width="353dp"
    android:layout_height="246dp"
    android:layout_marginTop="28dp"
    android:fadingEdge="vertical"
    android:foregroundGravity="center_vertical"
    android:gravity="center"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.724"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">

    <RadioButton
        android:id="@+id/radioButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Engleski" />
```

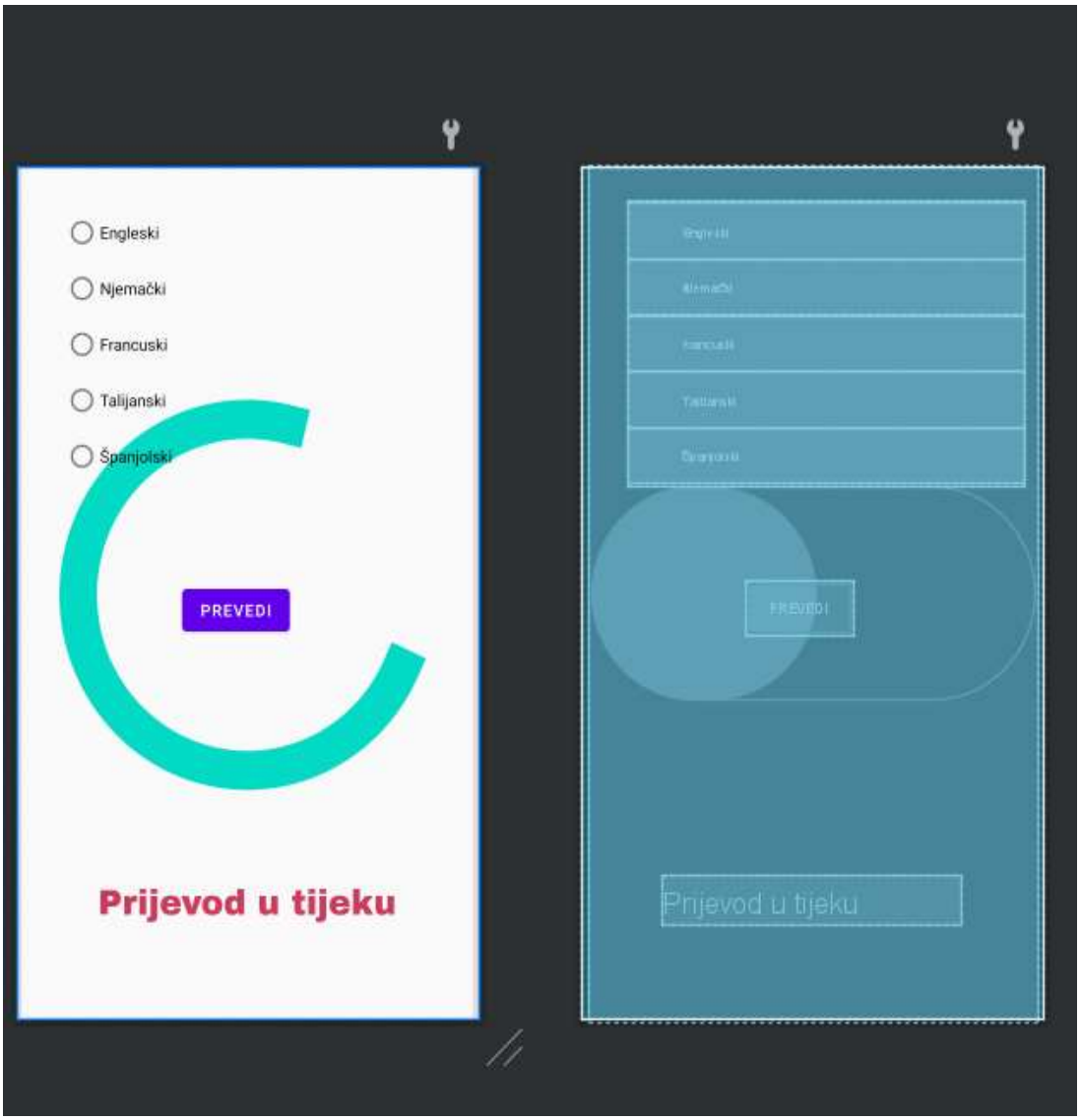
Slika 23. XML datoteka 'Translate_activitya' 1.dio

Izvor: autor

```
    <RadioButton
        android:id="@+id/radioButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Engleski" />

    <RadioButton
        android:id="@+id/radioButton2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Njemački" />

    <RadioButton
        android:id="@+id/radioButton3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Francuski" />

    <RadioButton
        android:id="@+id/radioButton4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Talijanski" />

    <RadioButton
        android:id="@+id/radioButton6"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Španjolski" />
</RadioGroup>
```

Slika 24. XML datoteka 'Translate_activitya' 2.dio

Izvor: autor



Slika 25. Vizualni elementi 'Translate_activitya'

Izvor: autor

Kada se preko vizualnog djela odabere jezik i pritisne gumb 'Preved' pokrene se funkcija 'Jezik_odabran' gdje se pregledava koji 'RadioButton' je aktiviran, pokrene se funkcija koja tada pokreće novu dretvu za prevođenje. Preko 'ML KIT' API-a šaljemo request u kojem je potrebno naznačiti s kojeg jezika se prevodi na koji. Podfunkcija za izvorni jezik naziva se ' setSourceLanguage', za jezik na koji treba biti preveden setTargetLanguage. Obje primaju string kao argument, točnije, skraćenicu jezika u obliku 2 slova(npr. za hrvatski je HR, za njemački je DE itd.)

Također moguće je unutar argumenta postaviti varijablu ' TranslateLanguage.ime_jezika' koja vraća skračenicu. Kada su odabrana oba jezika, pokreće se dretva koja šalje request, nakon toga preuzima se paket od oba jezika ukoliko na mobilnom uređaju ne postoje već svi znakovi neka od toga dva jezika.

Ukoliko je sve prošlo u redu, pokreće se 'event-funkcija' pod nazivom 'OnSuccesListener' gdje se pomoću parametra 'Object o' ,gdje je 'Object' klasa, a 'o' njezina instanca, dolazi do teksta na način da se kompletni 'o' prebaci u string, tj. linijom koda *o.ToString().* Nakon toga unutar paketa se stavlja prevedeni i original tekst,paket se stavlja u 'intent' i poziva se prijenos na sljedeću aktivnost. Dio koda nalazi se na slici 26.

```
new OnSuccessListener() {
    @Override
    public void onSuccess(Object o) {
        Translator.translate(text)
                .addOnSuccessListener(
                        new OnSuccessListener() {
                            @Override
                            public void onSuccess(Object o) {
                                TranslatedText = o.toString();
                                Translator.close();
                                Bundle b = new Bundle();
                                b.putString("jezik",jezik);
                                b.putString("translated",TranslatedText);
                                b.putString("original",OriginalText);
                                Intent intent = new Intent( packageContext: TranslateActivity.this, InspectText.class);
                                intent.putExtras(b);
                                startActivity(intent);

                            }
                        })
```

Slika 26. Funkcija 'onSuccess'

Izvor: autor

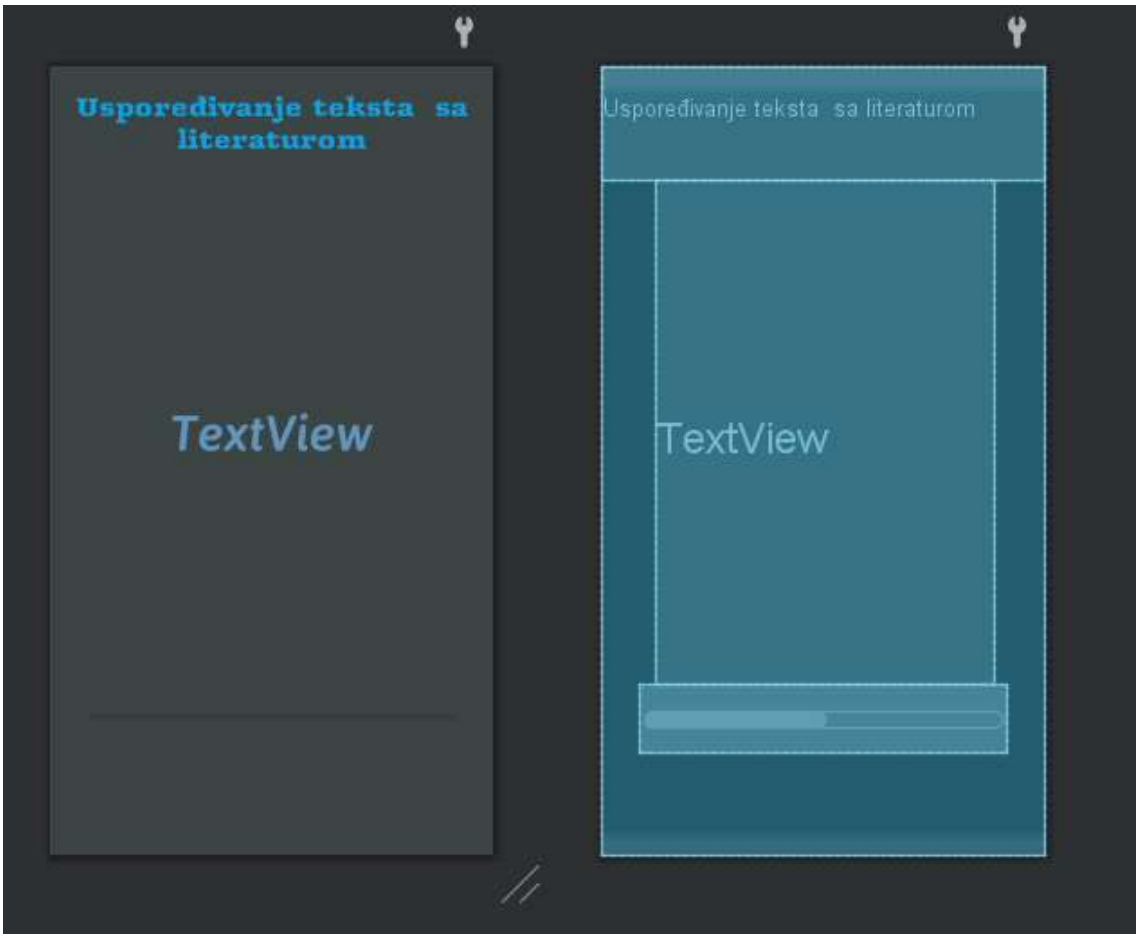
7.4 Analiziranje teksta

Tekstovi koji su došli putem odabiranja dokumenata iz lokalne memorije uređaja ili analizom teksta prenose se na sljedeću aktivnost pod imenom 'Inspect_text'. Unutar nje od vizualnih elemenata se nalaze samo 'TextVieowi' i 'ProgressBar'. 'TextView' koji se nalazi u gornjem djelu na sredini se mijenja se tokom izvođenja ove forme već ostaje isti. Donji 'TextView' služi za prikazivanje postotka obrađenosti analize dokumenta, što ujedno radi i 'ProgressBar'. XML kod vidljiv je na slici 27, dok su njihove pozicije na formi vidljive na slici 28.

```
<TextView
    android:id="@+id/textView2"
    android:layout_width="match_parent"
    android:layout_height="106dp"
    android:layout_gravity="center"
    android:fontFamily="@font/arbutus"
    android:gravity="center"
    android:text="Usporedivanje teksta  sa literaturom"
    android:textColor="#0FA2E4"
    android:textSize="24sp" />


<TextView
    android:id="@+id/textResult"
    android:layout_width="313dp"
    android:layout_height="466dp"
    android:layout_gravity="center"
    android:fontFamily="@font/asap_medium_italic"
    android:gravity="center"
    android:text="TextView"
    android:textColor="#6096BF"
    android:textSize="46sp" />

<ProgressBar
    android:id="@+id/progressBar"
    style="@style/Widget.AppCompat.ProgressBar.Horizontal"
    android:layout_width="340dp"
    android:layout_height="63dp"
    android:layout_gravity="center" />
```

Slika 28. XML kod forme 'inspect_text'

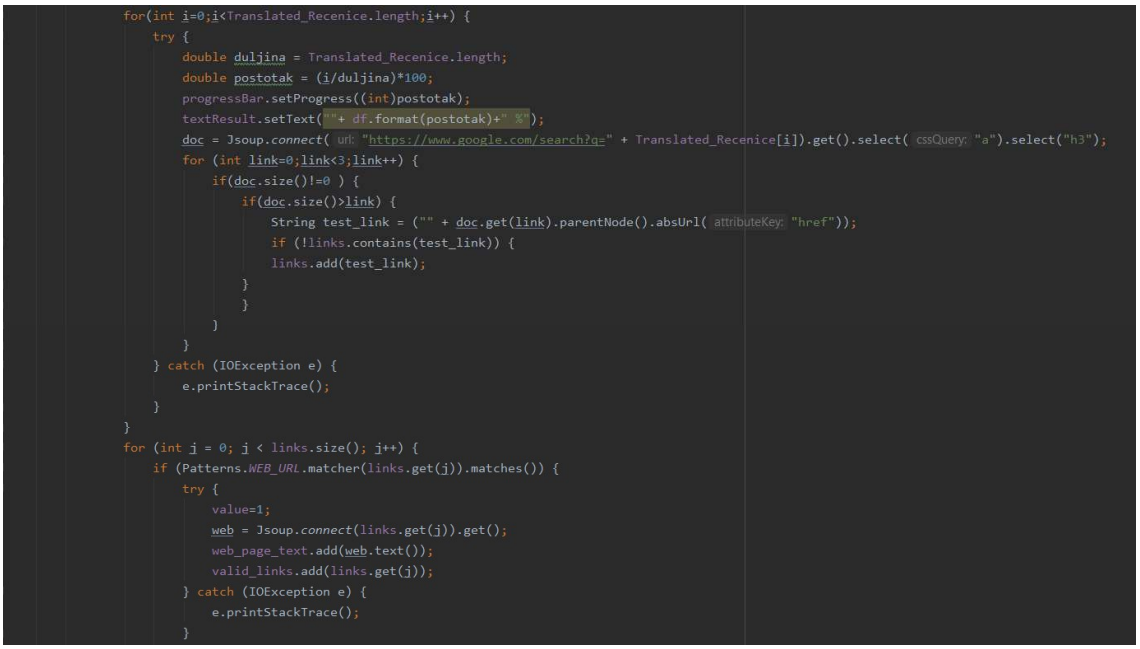
Izvor : autor



Slika 29. forma 'Inspect text'

Izvor: autor

Kada je u pitanju programiranje logike i kontrola vizualnih elemenata, klasa 'InspectText.java' zauzima najveće značenje za cijelokupni program. Također se u ovoj klasi nalaze i funkcije za koje treba najviše vremena kako bi se izvršile, pa čak i do 15 minuta za dokumente koji otprilike imaju 80 stranica, no toliko otprilike treba i ostalim servisima koji se bave plag scanom. Razlog tomu je što aplikacija kod svakog pokušaja iščitavanja teksta sa web stranice mora ponovno slati request i čekati određeno vrijeme za response. Način na koji se pomoću 'Jsoupa' iščitavaju linkovi može se vidjeti na slici 30.



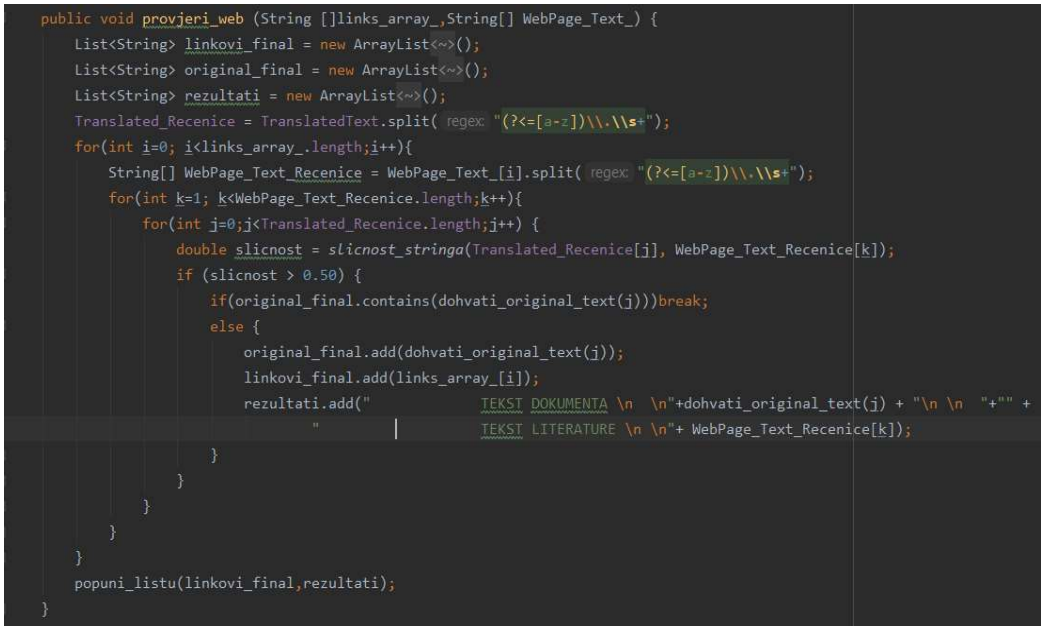
Slika 30. Preuzimanje teksta s web stranica pomoću 'Jsoupa'

Izvor : autor

Nakon što je funkcija za preuzimanje teksta s web stranica završena, analize tekstova se odvijaju na sljedeći način:

1. Usporedno analiziraj, rečenicu po rečenicu prevedenog teksta sa tekstom svih web stranica. Ukoliko je podudaranje veće od 50%, napravi string gdje ćeš smjestiti rezultat u obliku dijela teksta iz originalnog djela sa djelom teksta koji se preveden podudara tekstu web stranice i zapamti link gdje je pronađeno.
2. Prebaci sve rezultate, linkove koji su povezani s rezultatima i broj rezultata na sljedeću aktivnost.
3. Za vrijeme rada aktivnosti, pravovremeno osvježuj 'ProgressBar' i 'TextView' za to zadužen.

Za kraj, rezultati se prenose na aktivnost 'Results' gdje će se unutar 'ListViewa' prikazuju pronađene sumnje i rezultat 'plag-scana'. Programski kod za analizu tekstova i prijenos teksta na sljedeću aktivnost vidljivi su na slikama 31. i 32.



```
public void provjeri_web (String []links_array_,String[] WebPage_Text_) {
    List<String> linkovi_final = new ArrayList<~>();
    List<String> original_final = new ArrayList<~>();
    List<String> rezultati = new ArrayList<~>();
    Translated_Recenice = TranslatedText.split( regex: "(?<=[a-z])\\.\\s+");
    for(int i=0; i<links_array_.length;i++){
        String[] WebPage_Text_Recenice = WebPage_Text_[i].split( regex: "(?<=[a-z])\\.\\s+");
        for(int k=1; k<WebPage_Text_Recenice.length;k++){
            for(int j=0;j<Translated_Recenice.length;j++) {
                double slicnost = slicnost_stringa(Translated_Recenice[j], WebPage_Text_Recenice[k]);
                if (slicnost > 0.50) {
                    if(original_final.contains(dohvati_original_text(j)))break;
                    else {
                        original_final.add(dohvati_original_text(j));
                        linkovi_final.add(links_array_[i]);
                        rezultati.add("            TEKST DOKUMENTA \n  \n"+dohvati_original_text(j) + "\n \n  "+"" +
                                "          TEKST LITERATURE \n \n"+ WebPage_Text_Recenice[k]);
                    }
                }
            }
        }
    }
    popuni_listu(linkovi_final,rezultati);
}
```

Slika 31. Analiza tekstova

Izvor: autor

```
private void popuni_listu(List<String> linkovi, List<String> rezultati){
        String[] linkovi_array = new String[rezultati.size()];
        String[] rezultati_array = new String[rezultati.size()];
    for(int i=0;i<rezultati.size();i++){
        linkovi_array[i]= linkovi.get(i);
        rezultati_array[i]= rezultati.get(i);
    }
    float rezultati_float = rezultati_array.length;
    float recenice_float = Translated_Recenice.length;
    float plag_posto = 100*(rezultati_float/recenice_float);
    Intent intent = new Intent(getBaseContext(), Results.class);
    intent.putExtra( name: "linkovi",linkovi_array);
    intent.putExtra( name: "rezultati",rezultati_array);
    intent.putExtra( name: "velicina",rezultati_array.length);
    intent.putExtra( name: "plag_posto",plag_posto);
    intent.putExtra( name: "translated_text",TranslatedText);
    startActivity(intent);
}
}
```

Slika 32. Prijenos rezultata na sljedeću aktivnost

Izvor: autor

7.5 Ispis rezultata

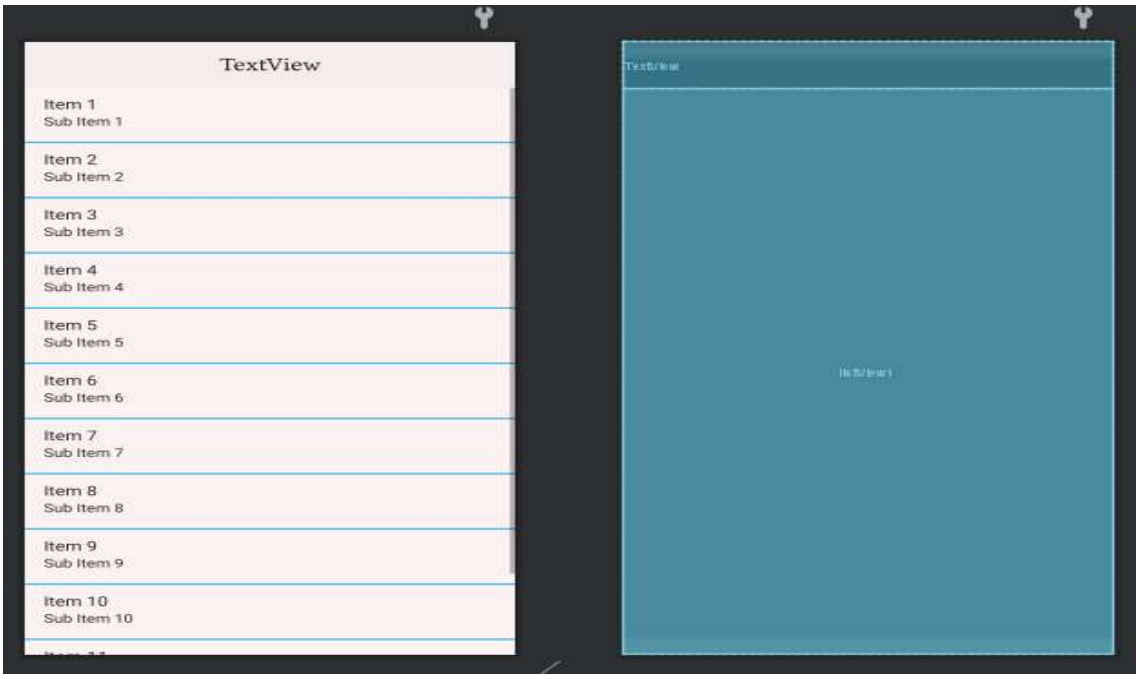
Nakon što su pronađeni rezultati, potrebno ih je ispisati vidljivo kao i mjesta na kojima su nađeni podudarajući dijelovi teksta. Za ispis koristi se vizualni element 'Listview' koji ima mogućnost ispisivati listu, čak ako se jedan njezin element sastoji od više podelemenata. Ukoliko nisu pronađene nikakve sumnje na rad, aplikacija će ispisati da nema rezultata i postotak plagijata će se voditi kao 0%. Kako bi se lista pravilno ispisala potrebno je napraviti adapter, tj. ukoliko imamo više elemenata unutar jednog elementa lista, moramo ih definirati, njihov tip i vizualni oblik u aktivnosti 'Adapter'. Unutar 'adapter.xml' registrirat će se samo dva 'TextViewa', a za njegovu klasu će se definirati dvije liste elemenata, 'top' i 'bot' (gornji i donji dio) koji ćemo smjestiti u 'Holder' za prikazivanje unutar 'ListViewa'. Unutar konstruktora 'Adaptera' prenijet ćemo liste rezultata i liste linkova.

```
public View getView(int position, View convertView, ViewGroup parent
{
    View v = convertView;
    Holder holder;
    if (v == null) {
        v = inflater.inflate(R.layout.adapter,  root: null);
        holder = new Holder();
        holder.top = (TextView)v.findViewById(R.id.top);
        holder.bot = (TextView) v.findViewById(R.id.bot);
        v.setTag(holder);
    } else {
        holder = (Holder) v.getTag();
    }


    holder.top.setText(top_array.get(position));
    holder.bot.setText(bot_array.get(position));
```

Slika 33. Postavljanje lista unutar holdera

Izvor: autor

Kod prikazivanja forme 'Results' u gornjem djelu stajati će koliki je postotak plagijata, ispod njega biti će prikazana lista elemenata i pripadajućih rezultata.



Slika 34. Vizualni elementi 'Results' aktivnosti

Izvor: autor

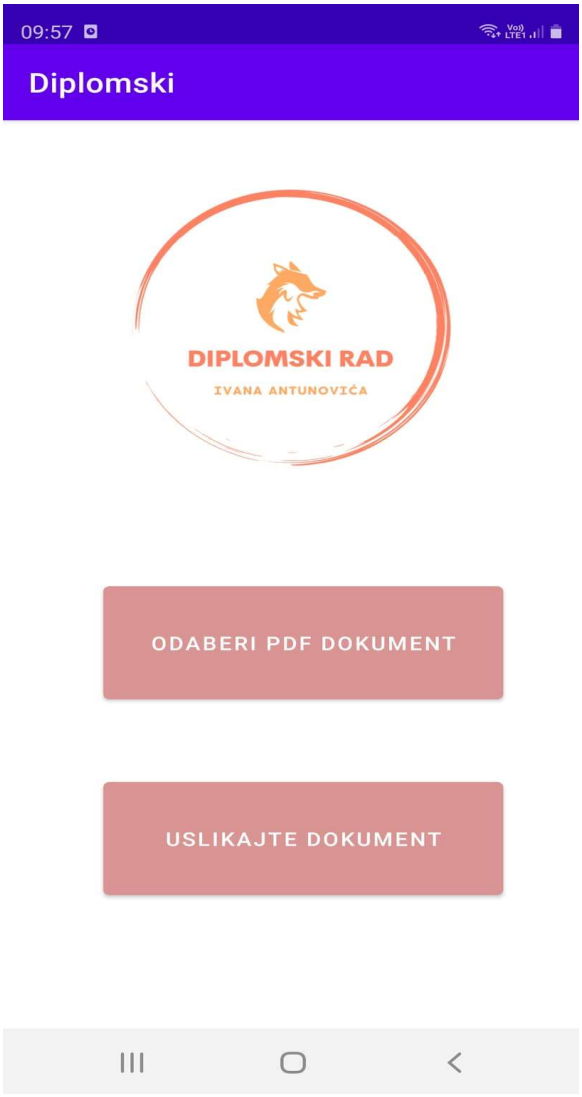
```
<TextView
    android:id="@+id/text_result"
    android:layout_width="match_parent"
    android:layout_height="55dp"
    android:layout_gravity="center"
    android:background="#F4EFEF"
    android:fontFamily="@font/petrona"
    android:gravity="center"
    android:includeFontPadding="true"
    android:text="TextView"
    android:textAppearance="@style/TextAppearance.AppCompat.Large" />

<ListView
    android:id="@+id/listView1"
    style="@style/Widget.AppCompat.ListView"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:background="#FBF2F2"
    android:cacheColorHint="#12B86B"
    android:divider="@android:color/holo_blue_light"
    android:dividerHeight="2.0sp"
    android:footerDividersEnabled="true"
    android:headerDividersEnabled="false" />
```

Slika 34. XML kod 'Results' aktivnosti

Izvor: autor

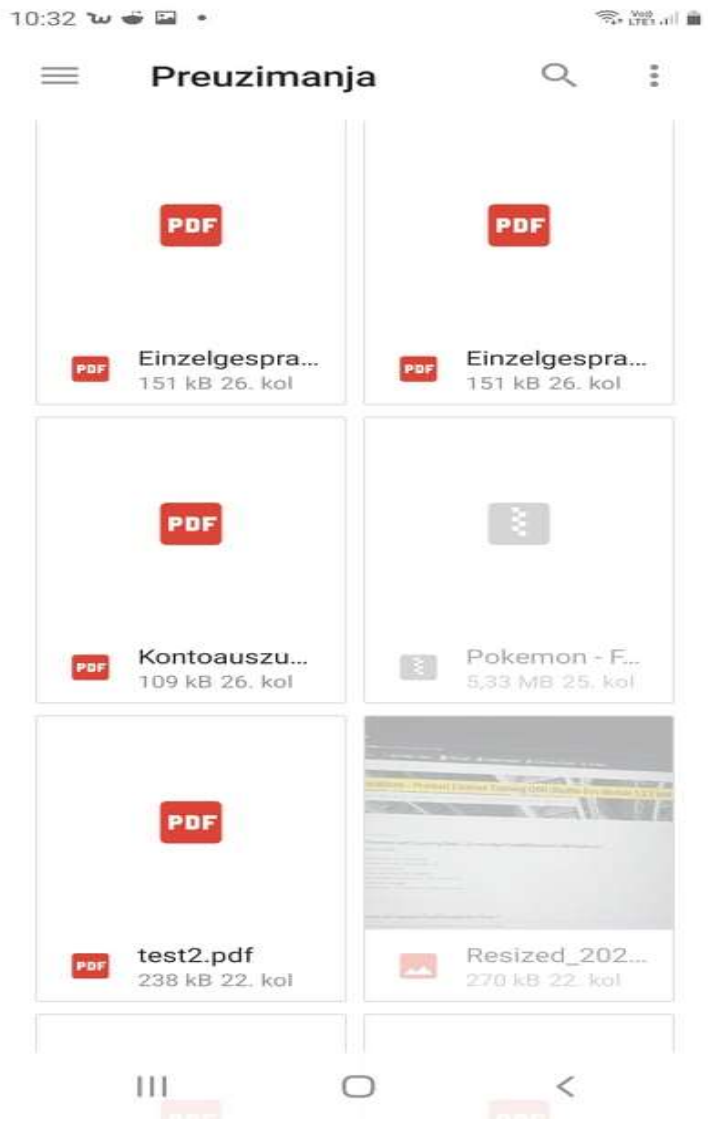
8. Korisničke upute

Rukovanje ovom aplikacijom je lako i svaka mogućnost biranja akcija se jasno predočuje pomoću teksta na okidačima istih (poput teksta na gumbovima, 'ProgressBara' ili opisnog teksta forme). Prvi zaslon predstavlja izbor mogućnosti unošenja teksta dokumenta koji treba biti analiziran: učiniti to iz interne memorije ili učiniti to putem fotografiranja jedne stranice dokumenta.



Slika 35. Početni zaslon

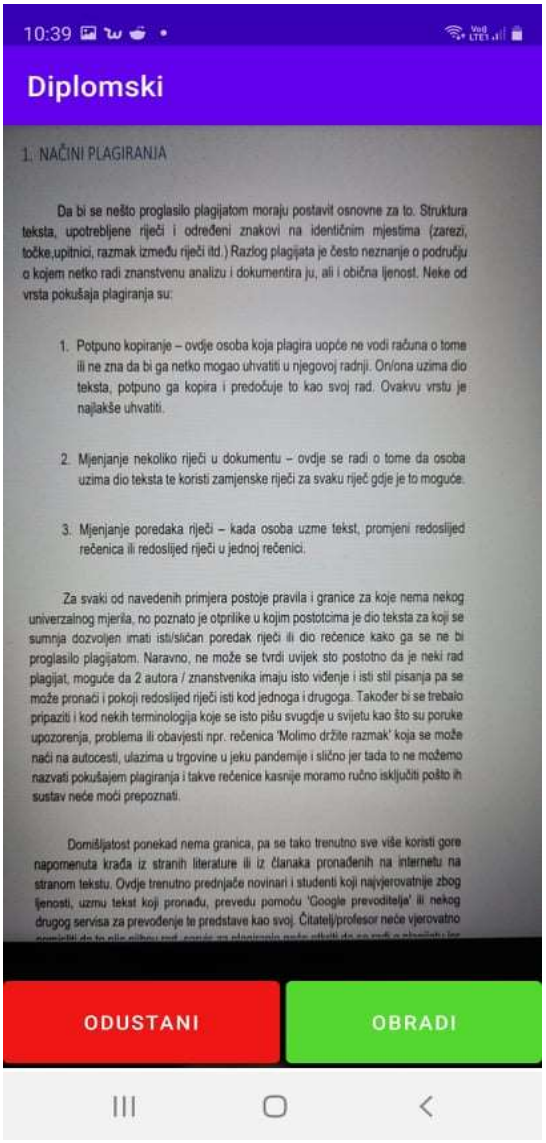
Kod odabira dokumenta, biti će moguće odabrati samo .pdf dokumente, dok će ostali biti osjenčani, sukladno s UX pravilima i na taj način će korisniku automatski dati doznanja da se ostale datoteke ne mogu prihvatiti kao izvor.



Slika 36. Odabir iz interne memorije

Izvor: autor

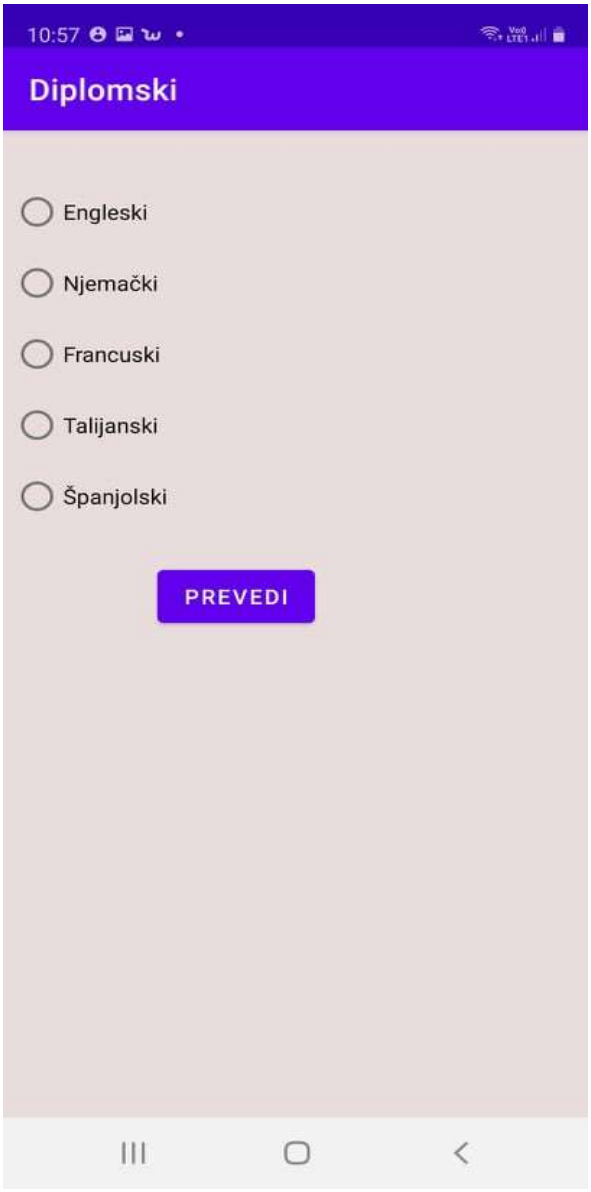
Za izradu analize putem fotografiranja, potrebno je dati dopuštenje korisnika aplikaciji koja nakon toga koristi prednju kameru. Postoje dva gumba : 'Odustani' i 'Obradi'. Na gumb 'Odustani' aplikacija vodi na početni zaslon, dok na gumb 'Obradi', aplikacija prelazi na aktivnost odabira prijevoda.



Slika 37. Fotografiranje dokumenta

Izvor: autor

Nakon što je odabran način uzimanja teksta, potrebno je odrediti na kojem jeziku se želi usporediti literatura. Trenutno je ponuđeno 5 jezika, vidljivo na slici 38. Poslije odabira jezika potrebno je pritisnuti gumb 'Prevedi'.



Slika 38. Odabir jezika

Izvor : autor

Nakon što se odabere jezik i krene se s prijevodom, prikazat će kružni 'ProgressBar' s obavijesti da je prijevod u toku. Nakon završetka prijevoda, krenut će se s analizom teksta i literature. Pri tome se pojavljuje obavijest o analizi, trenutni progres i 'ProgressBar' horizontalni.

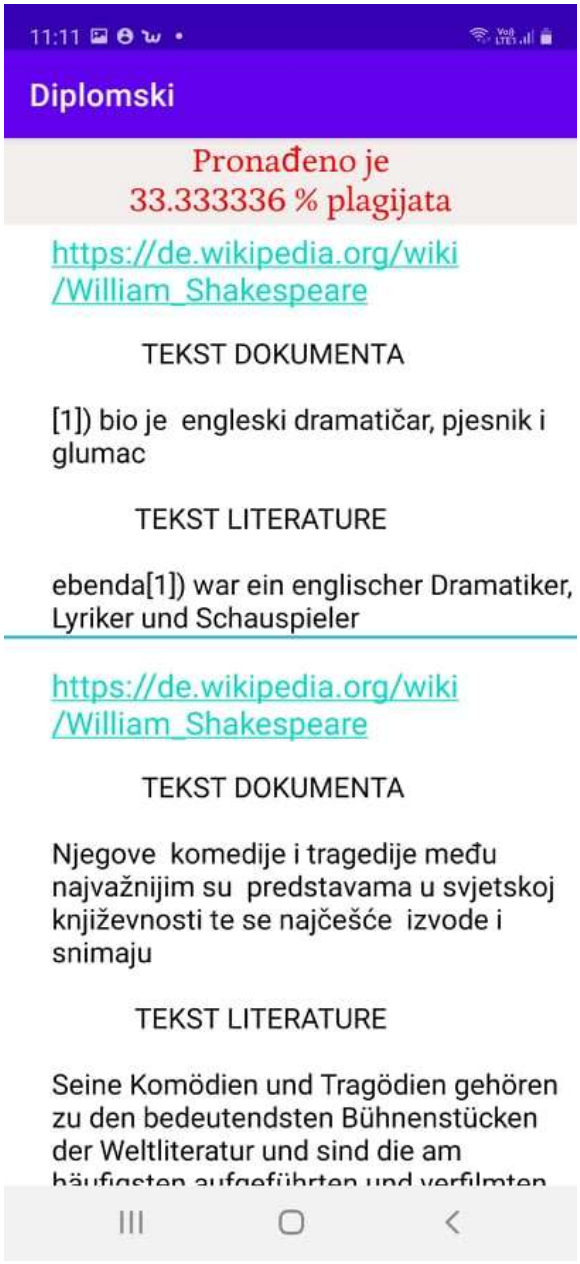
Slika 39. Progres prijevoda

Izvor : autor

Slika 40. Progres analize

Izvor : autor

Nakon izvršavanja svih gore navedenih aktivnosti, posljednja aktivnost je prikaz rezultata u aktivnosti 'Results' koja se sastoji od gornje teksta koji pokazuje koliki je postatak plagijata pronađen. Nakon njega vidljiva je lista sa svim sumnjivim dijelovima dokumenta, dijelovima strane literature iz koje se vjeruje da je kopiran tekst i link na kojima je literatura pronađena.



Slika 41. Pronađeni plagijat

Izvor: autor

Zaključak

Ovaj diplomski rad dao je jedan prijedlog rješenja nove značajke borbe protiv plagiranja radova. Iako rješenje izgleda jednostavno, da je potrebno samo prevesti tekst, ono zapravo i nije toliko lako.

Naime, kada bi jedan dokument pokušali prevesti na sve jezike svijeta, vremensko trajanje analize trajalo bi možda i nekoliko dana, uzimajući u obzir količinu literature koja bi se mogla pronaći za svaku rečenicu/dio teksta,zatim analiziranje dijelova teksta ili metodom 'rečenicu po rečenicu'. Također bilo bi uputno koliko je krajnji rezultat točan što zbog toga što možda autor rada nije plagirao, ali se negdje u nekoj literaturi na nekom jeziku nalazi rečenica koja je identična dokumentu kojeg pregledavamo. Zadnje navedenom problemu moguće je doskočiti na način da se pri 'internacionalnoj provjeri' rada poveća postotak koji je prijelazan prag u vrednovanju rada kao plagijata. Zbog toga, ovo rješenje ne izgleda kao valjano ,ali je na pragu nove generacije servisa 'plag-scana'.

Sigurno će jednog dana s poboljšanjem mreža, brzine protoka podataka i izradom novih tehnologije, ovo biti standardna značajka. Do tada, najbolje bi bilo da se ova značajka koristi isključivo kada se u literaturu dokumenta spominje literatura na stranom jeziku i da se tekst prevodeći na taj jezik, analizira sa stranom literaturom.

Kako i dalje postoje znanstveni radovi koji se predaju u obliku tiskanom na papiru, sigurno će se pojaviti i značajni doprinos 'OCR'-a koji će se uvrstiti u neke servise. Ova značajka mogla bi poslužiti i ljudima koji se bave problematikom novinarstva i prijenosa informacija jer često u vijestima/novinama se ne spominje izvor odakle je informacija došla, već je tekst preveden i prepisan i na taj način mogli bi se boriti za istinitost vijesti koje nam prenose, ali da i znamo tko je začetnik takvih vijesti.

Plagiranje je za svakog poštenog znanstvenika, bio on u nastajanju ili već s dosta godina iskustva ozbiljna prijetnja za razvoj znanosti u svijetu. S njom ne samo da se krade tuđi rad i trud, već se zaustavlja kritička misao, razvoj kreativnosti i sam razvoj znanosti, a samo stagniranje borbe protiv takvih radnji istu čine sve raširenijom.

Literatura

Internet izvori

a) Određivanje što je to plagijat

https://www.plagiarism.org/article/what-is-plagiarism

[pristupljeno zadnji puta 30.08.2021]

b) Jsoup dokumentacija

https://jsoup.org

[pristupljeno zadnji puta 30.08.2021]

c) Plag scan servis

https://www.plagscan.com

[pristupljeno zadnji puta 30.08.2021]

Popis slika

SAŽETAK

U ovom radu je teoretski je opisano što znači plagirati znanstveni rad, opasnosti takvih radnji, borba i nedostatci iste. Prikazana je ideja novih značajki, način rada kroz UML dijagrame i za kraj mobilna aplikacija napravljena pomoću okruženja unutar Android studia i Java programskog jezika sa dodatnim uvezenim programskim bibliotekama. Izrada mobilne aplikacije prikazana je korak po korak, aktivnost po aktivnost. Uz sve opcija koje prikazuju napredak, prikazani su i nedostatci zbog trenutnih poteškoća zbog manjeg obujma teksta koji se može poslati putem API-a.

Ključne riječi: *plagijat, mobilna aplikacija, literatura, prijevod, strani jezici, Jsoup, Android studio*

SUMMARY

This final thesis theoretically describes what it means to plagiarize scientific work, the dangers of such actions, the struggle and its shortcomings. The idea of new features, the way of working through UML diagrams and finally the mobile application made using the environment within Android studios and Java programming language with additional imported programming libraries are presented. The development of mobile applications is shown step by step, activity by activity. In addition to all the options that show progress, there are also disadvantages due to the current difficulties due to the smaller amount of text that can be sent via the API.

Keywords: *plagiarism, mobile application, literature, translation, foreign languages, Jsoup, Android studio*