Razvoj mobilne aplikacije za učenje tehničkog vokabulara engleskog jezika prepoznavanjem objekata

Landeka, Matej

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Stručni studij

RAZVOJ MOBILNE APLIKACIJE ZA UČENJE TEHNIČKOG VOKABULARA ENGLESKOG JEZIKA PREPOZNAVANJEM OBJEKATA

Završni rad

Matej Landeka

Osijek, 2020.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1S: Obrazac za imenovanje Povjerenstva za završni ispit na preddiplomskom stručnom studiju

Osijek, 04.09.2020.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za završni ispit na preddiplomskom stručnom studiju

Ime i prezime studenta:	Matej Landeka
Studij, smjer:	Preddiplomski stručni studij Elektrotehnika, smjer Informatika
Mat. br. studenta, godina upisa:	AI4560, 23.09.2019.
OIB studenta:	92701974924
Mentor:	Yvonne Liermann-Zeljak
Sumentor:	Prof.dr.sc. Goran Martinović
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Izv. prof. dr. sc. Krešimir Nenadić
Član Povjerenstva 1:	Yvonne Liermann-Zeljak
Član Povjerenstva 2:	Izv. prof. dr. sc. Ivica Lukić
Naslov završnog rada:	Razvoj mobilne aplikacije za učenje tehničkog vokabulara engleskog jezika prepoznavanjem objekata
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rada	Zadatak završnog rada je razviti mobilnu aplikaciju za učenje tehničkog vokabulara engleskog jezika prepoznavanjem objekata s fotografija slikanih mobilnim uređajima pri tome uzimajući u obzir funkcionalnosti postojećih sličnih aplikacija. Korištenjem odgovarajućeg algoritma te programskih jezika i tehnologija, aplikaciju je potrebno programski ostvariti i ispitati njezinu funkcionalnost. Sumentor: prof. dr. sc. Goran Martinović
Prijedlog ocjene pismenog dijela ispita (završnog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	04.09.2020.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 28.09.2020.

Ime i prezime studenta:	Matej Landeka
Studij:	Preddiplomski stručni studij Elektrotehnika, smjer Informatika
Mat. br. studenta, godina upisa:	AI4560, 23.09.2019.
Turnitin podudaranje [%]:	5

Ovom izjavom izjavljujem da je rad pod nazivom**: Razvoj mobilne aplikacije za učenje tehničkog vokabulara engleskog jezika prepoznavanjem objekata**

izrađen pod vodstvom mentora Yvonne Liermann-Zeljak

i sumentora Prof.dr.sc. Goran Martinović

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.

Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Cilj ovog rada je napraviti mobilnu aplikaciju za Android telefone koja će pomagati pri učenju tehničkog vokabulara engleskog jezika u području elektrotehnike i računarstva, a uz to aplikacija treba imati mogućnost prepoznavanja objekata sa slike.

Strojno učenje je primjena umjetne inteligencije (eng. *artificial intelligence*) pri kojoj se sustav podučava kroz niz iskustava. Računalima se daje niz podataka iz kojih uče i primjenjuju ih bez eksplicitnog programiranja. Cilj je omogućiti računalu da što je više moguće autonomno i bez ljudske pomoći uči i djeluje prema naučenim podacima. Postoji nekoliko metoda strojnog učenja: nadgledano učenje gdje se računalu daju označeni podaci kako bi kasnije samo moglo iz naučenog dati oznake neoznačenim podacima (ova metoda se koristi u radu); nenadgledano učenje kod kojeg podaci nisu označeni; polu-nadgledano učenje gdje je dio podataka označen, a ostatak nije, obično su takvi sustavi precizniji; pojačano učenje gdje se koristi metoda pokušaja i pogreške kako bi sustav naučio optimalni način rada u pojedinim situacijama. [1]

Aplikacija treba koristiti metode strojnog učenja kako bi prepoznala objekte s fotografije ili učitane slike. Objekti koje prepoznaje pripadaju području elektrotehnike ili računarstva. Za prepoznate objekte aplikacija treba dati engleski i hrvatski naziv. Osim prepoznavanja navedenih objekata, aplikacija treba služiti i za učenje istih pojmova na engleskom jeziku. Aplikacija će se pokretati na jednom od najčešće korištenih operacijskih sustava za mobilne telefone - Android. Aplikacija će biti napisana na programskom jeziku Java.

U drugom poglavlju analizirat će se nekoliko postojećih aplikacija za učenje engleskog jezika, dok će se u trećem poglavlju analizirati potrebni alati za razvoj mobilne aplikacije i za strojno učenje. Nadalje, u četvrtom poglavlju bit će objašnjene sve funkcionalnosti aplikacije, nakon čega će u petom poglavlju biti prikazano na koji način aplikacija radi. Konačno, u šestom poglavlju prikazane su upute kako koristiti aplikaciju i sve njene funkcionalnosti popraćene snimkama zaslona.

1.1 Zadatak rada

Zadatak završnog rada je razviti mobilnu aplikaciju za učenje tehničkog vokabulara engleskog jezika prepoznavanjem objekata s fotografija slikanih mobilnim uređajima pri tome uzimajući u obzir funkcionalnosti postojećih sličnih aplikacija. Korištenjem odgovarajućeg algoritma te programskih jezika i tehnologija, aplikaciju je potrebno programski ostvariti i ispitati njezinu funkcionalnost.

2. ISTRAŽIVANJE POSTOJEĆIH RJEŠENJA I STANJA U PODRUČJU

U ovom poglavlju daje se prikaz postojećih rješenja u području aplikacija za učenje stranih jezika i usporedba mogućnosti aplikacije s postojećim rješenjima.

2.1 Prikaz postojećih rješenja

Postojeće aplikacije za učenje stranih jezika koje će biti analizirane su: Duolingo, Drops, Memrise i Mondly.

2.1.1 Duolingo

Aplikacija Duolingo nudi učenje preko trideset i pet jezika, preko kratkih lekcija s provjerama napravljenim da budu poput igre. Pomaže učenju vokabulara, gramatike, čitanja, pisanja, slušanja i pričanja stranog jezika. Duolingo je *freemium* aplikacija, dakle osnovna je verzija besplatna, a plaća se za napredne funkcionalnosti.[2]

Aplikacija omogućuje korisniku odabir jezika koji želi učiti. Jezici su podijeljeni u niz lekcija za učenje. Korisnik prolaskom kroz jednostavnije lekcije otključava sve složenije lekcije. Za svaku lekciju korisnik dobiva bodove, to jest iskustvo (eng. *experience*) i može se uspoređivati s ostalim korisnicima. Lekcije se sastoje od čitanja i slušanja, a provjere znanja sadrže različita pitanja u obliku igre.

2.1.2 Drops

Mobilna aplikacija Drops osim odabira jezika omogućava i različito učenje za različite razine predznanja jezika. U svojoj besplatnoj verziji korisnik može učiti pet minuta na dan. Kada se pojavi nova riječ, korisnik sam odlučuje želi li ju učiti ili ne. Riječi su podijeljene u različite kategorije kao na primjer hrana, životinje, brojevi, poslovi, znanost itd. Sve riječi koje je korisnik do sada učio spremljene su u kolekcije tako da su dostupne za ponavljanje. Svaka riječ dolazi sa svojom ilustracijom u obliku obrazovne kartice (eng. *flashcard*), a vježbanje riječi se postiže raznim zadacima, na primjer spajanjem odgovarajućih riječi s ilustracijama, pitanjima točno-netočno, slaganjem riječi itd.

2.1.3 Memrise

Aplikacija Memrise također nudi različite razine učenja ovisno o prethodnom znanju korisnika. Kreće s jednostavnijim frazama, a zatim dodaje sve složenije fraze. Svaka nova riječ ili fraza je popraćena videom na kojemu izvorni govornici jezika izgovaraju tu riječ ili rečenicu. Korisnik ima uvid u riječi koje je do sada učio i koliko dobro ih je svladao. Vježba se obavlja slušanjem i čitanjem te pisanjem ili prevođenjem zadanih riječi.

2.1.4 Mondly

Još jedna *freemium* aplikacija koja nudi lekcije popraćene slikom i zvukom. Vježba novih riječi se obavlja različitim provjerama pisanja, prevođenjem ili uparivanjem slike s riječi, slično ostalim prethodno analiziranim aplikacijama. Korisnik ima uvid u broj riječi i fraza koje je do sada naučio. Rješavanjem lekcija korisnik skuplja bodove i smješta se na ljestvicu s ostalim korisnicima koji uče isti jezik kako bi se usporedio s drugima i bio potaknut na daljnju vježbu.

2.1.5 Osvrt na postojeća rješenja

Analizirane aplikacije neke su od najpopularnijih aplikacija za učenje stranih jezika. Obično koriste obrazovne kartice kako bi uvele nove riječi u korisnikov vokabular i sve su napravljene tako da korisnici vježbaju kroz neku vrstu igre.

Prednosti aplikacija Duolingo i Drops su što, iako imaju premium opciju, nema zaključanih lekcija, dok aplikacije Memrise i Mondly besplatno nude samo nekoliko osnovnih lekcija. Međutim, Mondly daje dnevno jednu besplatnu lekciju. Prednost Memrise nad ostalim aplikacijama jest što ima različite snimke izvornih govornika. Duolingo i Memrise nude vježbanje govora jezika, dok Drops i Mondly nemaju tu opciju. Aplikacija Drops jedina nema tablicu bodovanja. Prednost aplikacija Drops i Memrise jest što korisnici imaju uvid u kolekciju fraza s kojima su se susreli i koliko su dobro naučene. U tablici 2.1 nalazi se usporedba nekih od najvažnijih elemenata aplikacija, funkcionalnosti koje pojedine aplikacije imaju označene plusom ('+').

Tablica 2.1 Tablica funkcionalnosti postojećih aplikacija

	Prijevod	Pisanje	Čitanje	Slušanje	Govor	Sve lekcije besplatne	Tablica bodova	Kolekcija
Duolingo	+	+	+	+	+	+	+	
Drops	+	+	+	+		+		+
Memrise	+	+	+	+	+		+	+
Mondly	+	+	+	+			+	

2.2 Prikaz stanja u području

S obzirom na to da je cilj ove aplikacije učenje stručnog engleskog vokabulara, to jest učenje novih riječi i fraza, ova aplikacija koristit će neka od svojstava koja imaju navedene postojeće aplikacije.

Novi pojmovi bit će prikazani slikom, a kvizovi će služiti za vježbu prijevoda s engleskog na hrvatski ili s hrvatskog na engleski, za prepoznavanje pojmova ili preko slike ili čitanjem. Odgovori na pitanja tražit će se ili unosom teksta tako da korisnik vježba pisanje ili odabirom točnog odgovora između nekoliko ponuđenih.

Aplikacija neće imati zvučne datoteke, a time ni mogućnost vježbanja slušanja i govora. Aplikacija također neće imati tablicu bodovanja za razliku od postojećih aplikacija. Iako aplikacija neće imati pregled kolekcije, to jest uvid u fraze s kojima su se već susreli, dostupan je rječnik u kojemu će se nalaziti sve riječi iz aplikacije. Korisnik će moći vježbati riječi iz predefiniranih setova ili nasumično iz cijelog rječnika.

Funkcionalnost koju će ova aplikacija imati, a koju ostale postojeće aplikacije nemaju jest detekcija objekata. Određene riječi iz rječnika aplikacija će moći prepoznati na slikama te će korisniku dati njihov naziv i prijevod.

Glavni dio aplikacije činit će edukacijske kartice, to jest prikazi novih pojmova pomoću slika i riječi te različiti tipovi kvizova pomoću kojih će korisnici učiti tehnički engleski vokabular. Kvizovima će se vježbati prijevod, čitanje i pisanje. Kako korisniku ne bi bilo previše riječi odjednom, svaka lekcija obuhvaća po pet riječi. Ako korisnika zanima pojedina riječ uvijek će ju moći pronaći u rječniku, bilo da traži prijevod s hrvatskog na engleski ili s engleskog na hrvatski.

3. POTREBNE PROGRAMSKE TEHNOLOGIJE OKOLINE I JEZICI

U ovom poglavlju dan je opis programskih jezika, razvojnog okruženja i tehnologija koje su korištene za izradu aplikacije.

3.1 Operacijski sustav Android

Android je operacijski sustav baziran na modificiranoj verziji Linuxa. Razvijen je od tvrtke Google i Open Handset Alliance. Komercijalno se koristi od 2008. godine. Android sustav je besplatan i *open source*. Koristi se na mobitelima, tabletima, pametnim satovima i narukvicama, televizorima, kamerama i drugim uređajima. Koristi ga preko dvije i pol milijarde uređaja. Trenutno najnovija verzija Android operacijskog sustava je Android 10.[3,4]

Android je jedan od dva najčešće korištena operacijska sustava za mobitele pa će ova aplikacija biti napravljena za Android platformu.

3.2 Razvojna okolina Android Studio

Za razvoj aplikacije odabrano je razvojno okruženje Android Studio. Android Studio nudi mogućnost razvoja aplikacija za bilo koji Android uređaj, dolazi s emulatorom i alatima za testiranje aplikacija, ima integraciju s GitHub-om i ugrađenu podršku za Google Cloud platformu. Nudi pregled strukture projekta, *debugger*, *logcat* i druge alate. Android Studio formatira kod i upozorava na greške.[5]

Najčešće korišteni programski jezici su Java i Kotlin.

3.3 Programski jezik Java

Java je objektno orijentirani programski jezik koji je razvila tvrtka Sun Microsystems. Razvoj jezika je počeo 1991., a objavljen je 1995. godine. Java omogućuje da se jednom napisani program može pokretati na svim platformama koje imaju Java Virtual Machine bez specifičnih preinaka za pojedini sustav. Sintaksa jezika Java slična je sintaksi C i C++ jezika. Za razliku od C++, Java ne dopušta preopterećivanje operatora, a ima posebnu vrstu komentara koja se zove Javadoc i služi za pisanje dokumentacije klasa i metoda, Javadoc komentari počinju s '/**' i završavaju s '*/'. S obzirom na to da se radi o objektno orijentiranom jeziku, kod se piše u klasama i svi su podaci, osim primitivnih tipova, objekti.[6]

3.4 Platforma TensorFlow

TensorFlow je *open source* platforma za stvaranje i treniranje modela strojnog učenja (eng. m*achine learning*). Trenutno je to jedna od najpopularnijih platformi za strojno učenje i duboko učenje (eng. *deep learning*). Razvio ju je Google Brain Team i objavio 2015. godine. Koriste ga znanstvenici i programeri. Može raditi s različitim jezicima, na primjer Java, C++ i Python. TensorFlow kao ulaz dobiva višedimenzijsko polje podataka koje se zove tenzor i na njemu vrši niz operacija zbog čega je i nazvan TensorFlow. TensorFlow se može pokretati na Windows, MacOS i Linux računalima, na Android i iOS mobitelima i na „oblaku" (eng. *cloud*). [7,8]

TensorFlow Lite je niz alata koji omogućuju pokretanje TensorFlow modela na mobilnim uređajima i *IoT (Internet of Things)* uređajima. Aplikacija će koristiti TensorFlow Lite tip modela kako bi klasificirala slike.

3.5 Platforma Firebase AutoML

Za treniranje modela aplikacija će koristiti automatizirano strojno učenje u sklopu platforme Firebase. Firebase je platforma tvrtke Google za razvoj, poboljšanje i proširenje mobilnih aplikacija. Neki od alata koje nudi Firebase su autentikacija, baze podataka, analitika, *push* poruke i pohrana datoteka. U ovom radu koristi se ML Kit iz Firebase platforme, alat koji služi za strojno učenje. On omogućuje treniranje TensorFlow Lite modela koji služe za prepoznavanje različitih objekata snimljenih kamerom. [9, 10]

AutoML Kit omogućuje jednostavno treniranje modela. Potrebno je dati niz slika - više slika znači da će model biti bolje utreniran za prepoznavanje objekata na slikama. Nakon učitavanja slika potrebno je pokrenuti treniranje modela. Nakon što je trening završen, model se može preuzeti i koristiti u mobilnoj aplikaciji. Besplatna Firebase verzija omogućuje učitavanje do tisuću slika i treniranje modela do tri sata.

4. PRIKAZ FUNKCIONALNOSTI I POSTUPAKA KOJI APLIKACIJA OBAVLJA

U ovom poglavlju objašnjene su funkcionalnosti koje aplikacija ima, prikazan je i objašnjen dijagram toka te *Mock-up* aplikacije.

4.1 Funkcionalnost mobilne aplikacije

Cilj je stvoriti aplikaciju koja ima sljedeće funkcionalnosti:

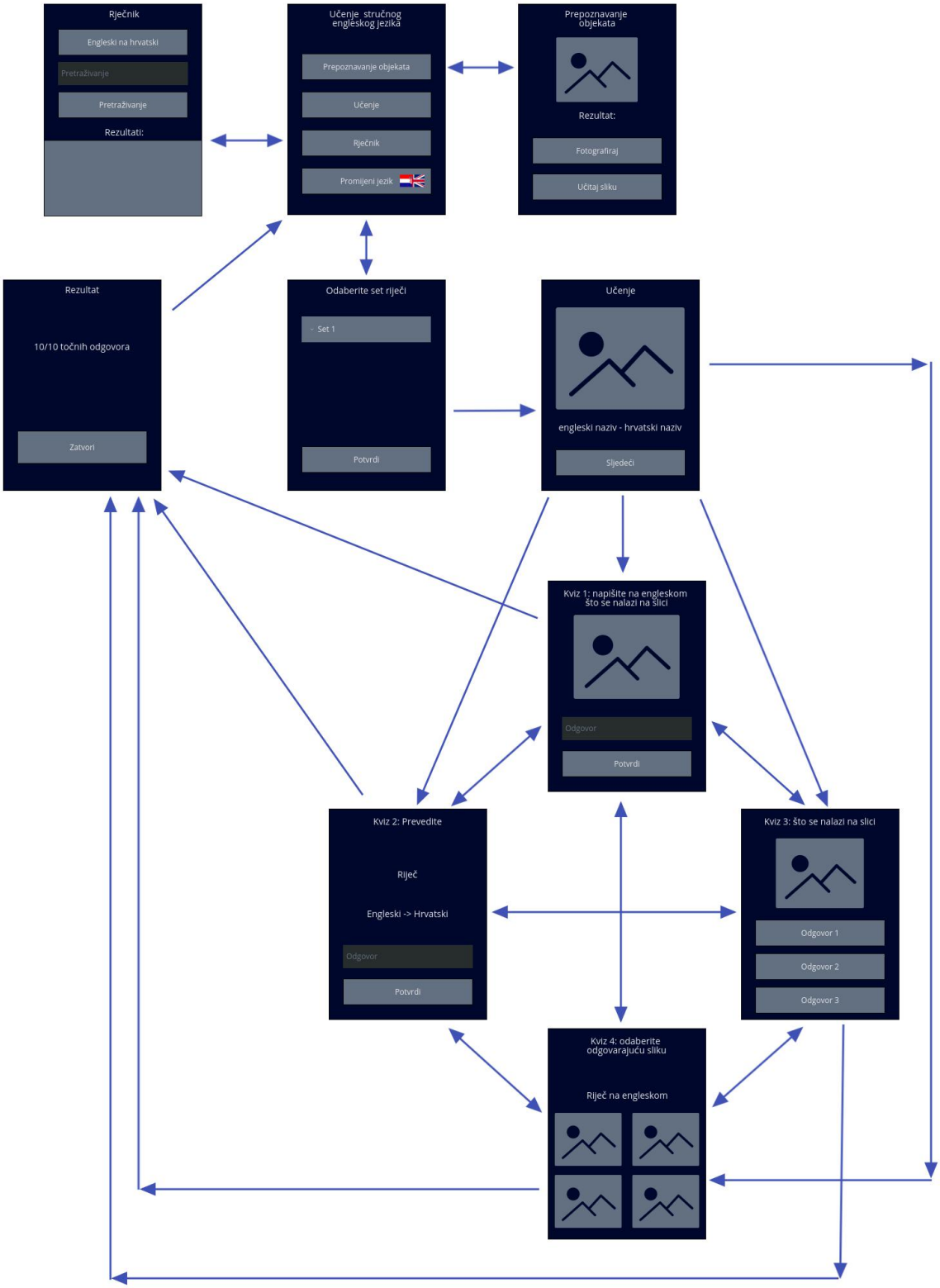
- Fotografiranje i prepoznavanje objekta s fotografije
- Prepoznavanje objekta sa slike učitane iz galerije
- Prikaz niza edukacijskih kartica o pojmovima iz elektrotehnike ili računarstva u obliku slike i engleskog i hrvatskog naziva
- Postaviti niz pitanja za vježbu pojmova u obliku kviza
- Pružiti povratnu informaciju o broju točnih odgovora u kvizu
- Pretraživanje pojmova u rječniku s engleskog na hrvatski i s hrvatskog na engleski
- Promjena jezika sučelja na hrvatski ili engleski

4.2 Mock-up aplikacije

Mock-up je prikaz dizajna aplikacije. Na njemu se nalaze svi planirani prikazi zaslona koje će imati aplikacija kao i izbor boja, tipografije, ikona i raspored. Cilj *mock-upa* je biti što sličniji krajnjem izgledu aplikacije kako bi se na temelju njega mogle otkloniti moguće greške u dizajnu. Zasloni su povezani strelicama koje označavaju iz kojeg se zaslona može prijeći u drugi zaslon.[11]

Na slici 4.1 prikazana je *mock-up* aplikacija. Otvaranjem aplikacije pokazuje se prvi zaslon s glavnim izbornikom, čiji je naslov „Učenje stručnog engleskog jezika”. Ako na glavnom izborniku korisnik odabere gumb „Prepoznavanje objekata”, otvara se odgovarajući zaslon. Zatvaranjem istog zaslona korisnik se vraća u glavni izbornik. Na isti način, ako korisnik

odabere gumb „Rječnik” u glavnom izborniku, otvara se novi zaslon. Korisnik se može vratiti na glavni izbornik zatvaranjem rječnika. Odabirom gumba „Učenje” otvara se zaslon s naslovom „Odaberite set riječi”. Korisnik može zatvoriti taj zaslon i vratiti se na glavni izbornik ili odabrati gumb „Potvrdi” koji otvara novi zaslon s aktivnosti „Učenje”. Iz zaslona „Učenje” nasumično se može otvoriti jedan od zaslona s kvizom: kviz jedan, dva, tri ili četiri. Iz bilo kojeg zaslona s kvizom može se nasumično otvoriti jedan od preostala tri zaslona s kvizom. Prolaskom kroz dva zaslona s kvizom otvara se zaslon „Rezultat”. Na tom se zaslonu nalazi gumb „Zatvori” kojim se zatvara trenutni zaslon i vraća se na glavni izbornik.



Slika 4.1 *Mock-up* aplikacije

4.3 Dijagram toka načina rada aplikacije

Na slici 4.2 prikazan je dijagram toka aplikacije.

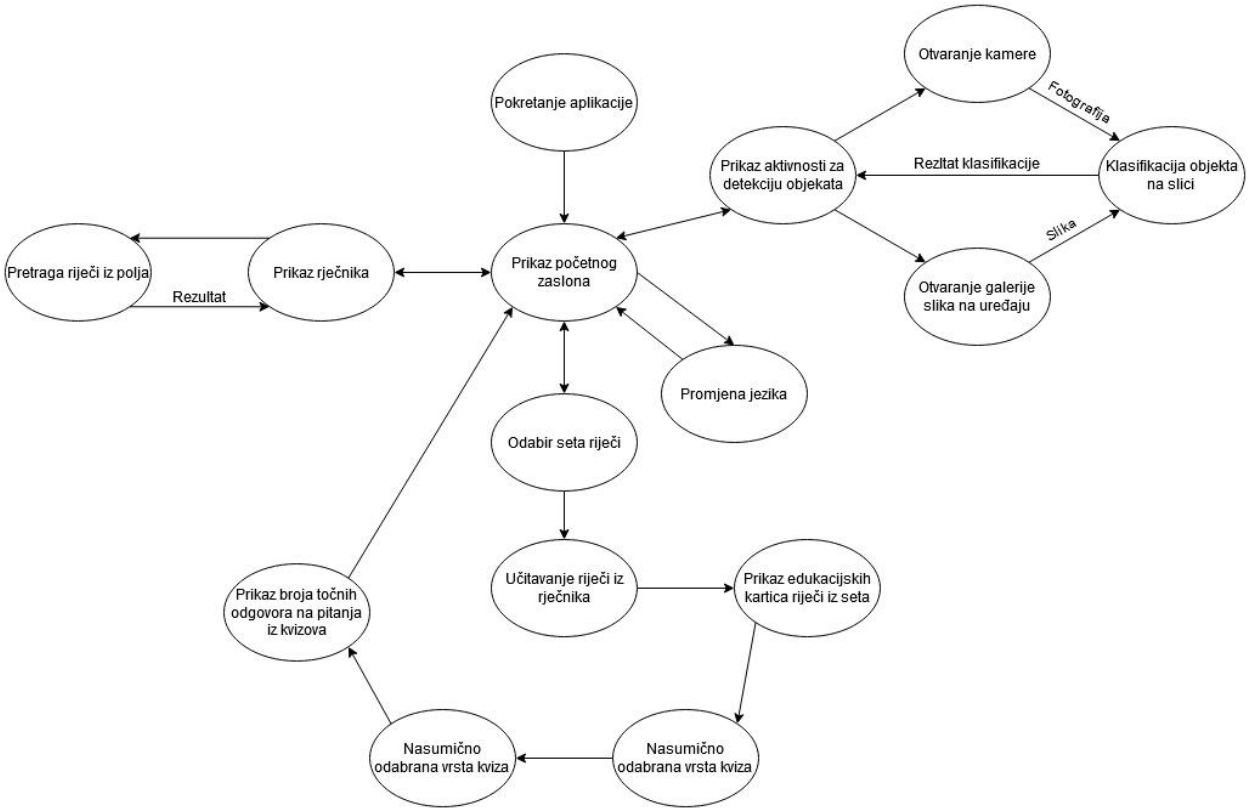
Pokretanjem aplikacije otvara se sučelje na kojemu su prikazane funkcionalnosti programa.

Odabirom promjene jezika, jezik sučelja aplikacije mijenja se s hrvatskog na engleski ili obrnuto i sučelje se ponovno učitava.

Kod funkcionalnosti rječnika, korisnik u tekstualno polje unosi riječ koju želi pronaći. Tekst se iz polja uspoređuje s riječima u rječniku i u prikaz rječnika ispisuju se svi rezultati pretrage, to jest sve riječi jednake traženom tekstu.

Kada je pokrenuta funkcionalnost za detekciju objekata, detekcija se može obaviti na dva načina. Prvi način je pomoću kamere: korisniku se otvara kamera na uređaju, korisnik snimi fotografiju, ona se učitava u aplikaciju i klasificira se, rezultat klasifikacije se ispisuje unutar aktivnosti za detekciju objekata. Drugi način detekcije obavlja se klasifikacijom prethodno snimljene slike. Prvo se otvara galerija slika na korisnikovu uređaju, zatim aplikacija učitava sliku koju je korisnik odabrao i klasificira ju. Na kraju se ispisuje rezultat klasifikacije kao i za snimljenu fotografiju.

Funkcionalnost učenje počinje odabirom seta riječi. Nakon što je set odabran, učitava se niz odgovarajućih riječi iz rječnika i slijedi njihov prikaz. Nakon što su prikazane sve riječi iz seta pokreće se kviz kojim se provjerava korisnikovo znanje. Odabir vrste kviza je nasumičan. Kada je kviz završen, ponovno se nasumično pokreće jedna od preostalih vrsta kvizova. Nakon što je i drugi kviz završen, na novom se zaslonu ispisuje broj točnih odgovora koje je korisnik ponudio u prethodna dva kviza. Zatvaranjem tog zaslona korisnik se vraća na početni zaslon.



Slika 4.2 Dijagram toka aplikacije

4.4 Prepoznavanje objekata

Korisnik na početnom zaslonu odabire gumb „Prepoznavanje objekata" koji ga vodi na novi zaslon. Na tom zaslonu se nalaze dva gumba: „Fotografiraj" i „Učitaj sliku". Odabirom gumba „Fotografiraj" otvara se kamera. Kada korisnik snimi fotografiju, ona bude prikazana na ekranu i ispod nje bude ispisan rezultat klasifikacije: oznaka i prijevod na hrvatski te postotak sigurnosti klasifikacije. Ako model nije uspio klasificirati fotografiju, ispod slike se ispisuje "*No result*". Odabirom gumba „Učitaj sliku", korisnik može odabrati sliku iz svoje galerije na uređaju, klasifikacija se dalje odvija jednako kao i za fotografije. Kako bi korisnik mogao koristiti ove funkcionalnosti, aplikaciji mora dati dopuštenje za pristup kameri i datotekama.

4.5 Prikaz skupa edukacijskih kartica

Korisnik na početnom zaslonu odabire gumb „Učenje". Otvara se novi zaslon na kojem je padajući izbornik. Korisnik iz padajućeg izbornika može odabrati set pojmova. Svaki set sadrži

pet pojmova. U padajućem izborniku nalazi se i opcija „nasumično" koja će nasumično odabrati pet pojmova iz rječnika. Kada je korisnik odabrao željeni set treba pritisnuti gumb „Potvrdi". Korisniku se zatim prikazuju edukacijske kartice pojmova iz seta, na njima se nalazi slika, engleski naziv i prijevod na hrvatski. Nakon toga pokreće se kviz za vježbanje prikazanih pojmova.

4.6 Kviz

Postoje četiri vrste kviza za vježbu. Prva vrsta pokazuje sliku korisniku i traži od korisnika da navede njen engleski naziv. Druga vrsta nasumično ispisuje hrvatski ili engleski naziv pojma i traži od korisnika da ga prevede na engleski ili hrvatski. Treća vrsta pokazuje sliku i tri odgovora, korisnik treba odabrati odgovor koji pripada slici. Četvrta vrsta kviza ispisuje pojam na engleskom i nudi četiri slike, korisnik treba odabrati sliku koja pripada pojmu. Svaki put kada korisnik pokrene učenje, nasumično se odabiru dvije vrste kviza, svaki s pet pitanja o pojmovima s kartica. Kada korisnik završi kviz ispisuje se točan i ukupan broj odgovora.

4.7 Rječnik

Korisnik na početnom zaslonu odabire rječnik. Otvara se zaslon s poljem za unos teksta. Korisnik upisuje željeni tekst i odabire gumb „Pretraži". Dohvaćaju se sve riječi iz datoteke koja sadrži rječnik koje počinju s tekstom u polju i upisuju se u listu na donjem dijelu ekrana, dohvaćaju se i prijevodi riječi iz rječnik datoteke za drugi jezik. Tekst u polju nije osjetljiv na velika i mala slova. Korisnik može pretraživati engleske ili hrvatske riječi, a koji rječnik želi pretraživati mijenja pritiskom na gumb „Engleski na hrvatski" ili „Hrvatski na engleski" na vrhu ekrana.

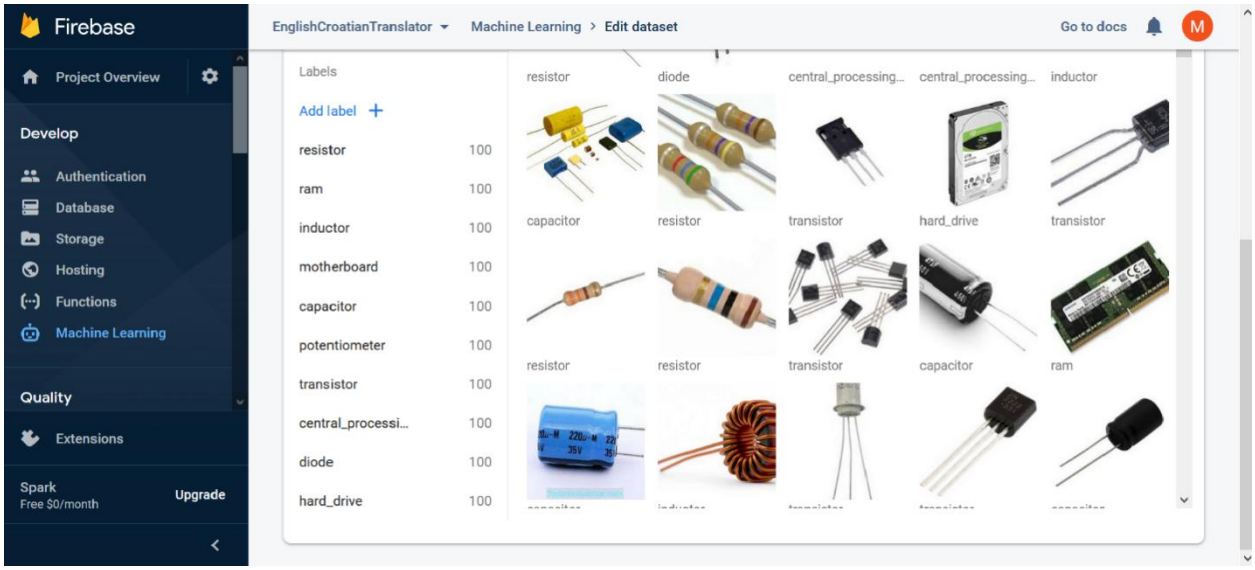
5. PROGRAMSKO RJEŠENJE

U ovom će poglavlju biti prikazano treniranje modela strojnog učenja i programsko rješenje aplikacije popraćeno slikama koda i njihovim objašnjenjem.

5.1 Treniranje modela strojnog učenja

Na platformi Firebase potrebno je napraviti novi projekt. Na upravljačkoj ploči projekta nalazi se alat za strojno učenje i unutar njega potrebno je odabrati AutoML. AutoML omogućuje treniranje vlastitih modela strojnog učenja sa željenim oznakama i slikama koje služe kao podaci za trening. [12]

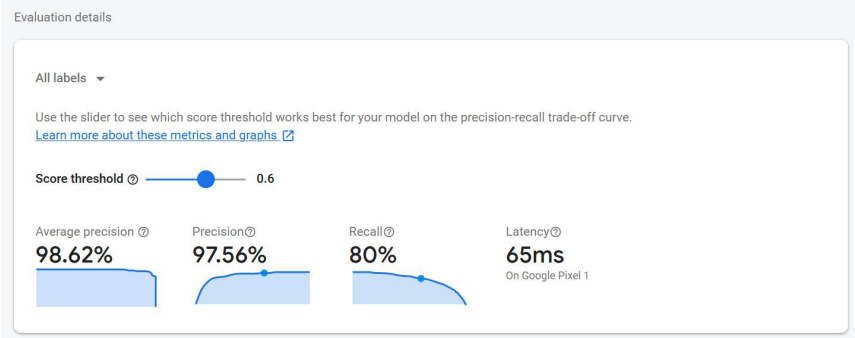
Potrebno je učitati slike s oznakama u set podataka koji će se koristiti za treniranje modela. Za treniranje modela korišteno je deset različitih oznaka, po sto slika za svaku oznaku što je prikazano na slici 5.1.



Slika 5.1 Prikaz slika i oznaka u setu podataka na Firebase upravljačkoj ploči

Nakon što su željene slike učitane i obilježene, potrebno je pokrenuti treniranje modela. Postoje tri opcije s različitim brzinama i preciznostima prepoznavanja. Kada je treniranje modela gotovo može se vidjeti njegova prosječna preciznost (Slika 5.2) kao i preciznost za klasifikaciju pojedinih oznaka (Slika 5.3). Srednja preciznost modela korištenog u ovoj aplikaciji je 98.62%,

dok je pri minimalnoj sigurnosti klasifikacije 0.6 preciznost (eng. *precision*) 97.56% . Drugim riječima, što je preciznost veća bit će manje krivih pozitivnih klasifikacija, to jest model će rjeđe dodijeliti krivu oznaku objektu, *recall* modela je 80%, što je veća *recall* vrijednost, model će imati manje krivih negativnih klasifikacija, to jest model rjeđe neće dodijeliti pravu oznaku objektu zbog male sigurnosti. Kod klasifikacije pojedinih oznaka najčešće dolazi do greške kod klasifikacije električne zavojnice (oznaka *inductor*) i kondenzatora (oznaka *capacitor*) koji su točno klasificirani u 90% slučajeva i kod klasifikacije diode (oznaka *diode*) koja je točna u 70% slučajeva. Klasifikacija ostalih oznaka je približno 100%.



Slika 5.2 Prosječna preciznost modela

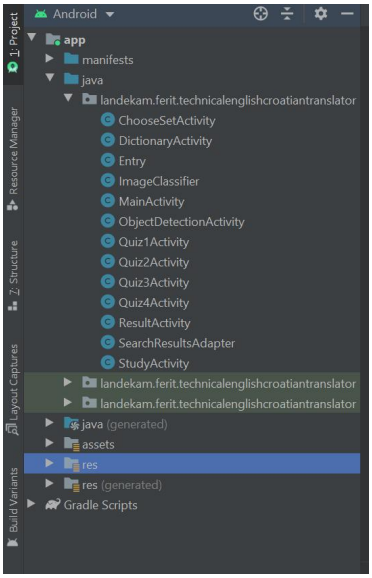
Slika 5.3 Preciznost modela prema pojedinim oznakama

Metoda koja se koristi za treniranje modela u Firebase AutoML Vision zove se nadgledano strojno učenje. Ono uključuje obuku računala tako da raspoznaje obrasce iz označenih podataka koji su mu dani u setu podataka, što više različitih podataka omogućuje bolje utrenirani model. Korištenjem nadziranog učenja model se može osposobiti za prepoznavanje obrazaca i sadržaja u željenim kategorijama. [13]

Trenirani model može se koristiti u Android aplikaciji. Kako bi se koristio potrebno je preuzeti .zip datoteku u kojoj se nalaze *dict.txt* datoteka u kojoj su zapisane oznake modela, *manifest.json* datoteka i *model.tflite* datoteka, to jest sami model. Sve tri datoteke potrebno je staviti u *assets* mapu u projektu aplikacije.

5.1 Struktura aplikacije

Aplikacija ima trinaest Java klasa, od toga deset aktivnosti, odnosno deset različitih zaslona sa svojim korisničkim sučeljem, jednim adapterom koji pomaže pri ispisu podataka i još dvije zasebne klase. Osim Java klasa, aplikacija ima i trinaest *xml (EXtensible Markup Language)* datoteka kojima se definira izgled sučelja. U mapi *assets* smješteni su model i njegov rječnik, zatim *croDict.txt* datoteka u kojoj su hrvatski prijevodi engleskog rječnika te mapa *images* u kojoj se nalaze slike koje se koriste za edukacijske kartice i kvizove. Definirane su i različite *strings* datoteke za hrvatski i engleski jezik kako bi sučelje aplikacije moglo biti na oba jezika. Java klase izlistane su na slici 5.4.



Slika 5.4 Java klase aplikacije

5.2 Implementacija Firebase biblioteke i TensorFlow Lite u aplikaciju

Potrebno je preuzeti datoteku *google-services.json* s Firebasea i staviti ju u mapu *app* u projektu. Zatim je potrebno u *build.gradle* projektu unutar *dependencies* dodati **“***classpath 'com.google.gms:google-services:4.3.3'* **“**, a u *build.gradle* aplikacije dodati **“***apply plugin: 'com.google.gms.google-services'* **“**, zatim **“***aaptOptions {noCompress “tflite”}* **“** te unutar *dependencies* dodati **“***implementation 'com.google.firebase:firebase-ml-vision:20.0.0'* **“** i **“***implementation 'com.google.firebase:firebase-ml-vision-automl:16.0.0'* **“**.

5.3 Početni zaslon

ActivityMain je aktivnost koja se prva pokreće s pokretanjem aplikacije te je glavni izbornik aplikacije.

Jezik se mijenja pomoću metode *setAppLocale* koja prima kod jezika u obliku *stringa*: **“**en**”** za engleski i **“**hr**”** za hrvatski. Metoda je prikazana na slici 5.5. Prvo dohvaća potrebne resurse i, ovisno o verziji Android sustava na mobitelu, postavlja kod jezika te ažurira korisničko sučelje.

```
private void setAppLocale(String localeCode){
    Resources resources = getResources();
    DisplayMetrics dm = resources.getDisplayMetrics();
    Configuration config = resources.getConfiguration();
    if (Build.VERSION.SDK_INT>=Build.VERSION_CODES.JELLY_BEAN_MR1){
        config.setLocale(new Locale(localeCode.toLowerCase()));
    } else {
        config.locale = new Locale(localeCode.toLowerCase());
    }
    resources.updateConfiguration(config, dm);
    setContentView(R.layout.activity_main);
    InitializeUI();
}
```

Slika 5.5 Metoda *setAppLocale*

Metoda InitializeUI, koja je prikazana na slici 5.6, služi za povezivanje elemenata Java koda s elementima *xml layout* sučelja i postavljanje *OnClickListenera*. Sve aktivnosti imaju istoimenu metodu prilagođenu za svoje potrebe. Kod *ActivityMain* su prvo svi gumbi povezani sa željenim elementima pomoću metode *findViewById* i njihovih identifikatora. Aktivnosti koje će se

događati pritiskom na pojedini gumb definirane su unutar *setOnClickListener* metode, tako gumbi *buttonObjectDetection*, *buttonDictionary* i *buttonStudy* pokreću nove aktivnosti dok *buttonChangeLanguage* pokreće već navedenu metodu *setAppLocale*.

```
private void InitializeUI() {
    buttonObjectDetection = findViewById(R.id.buttonObjectDetection);
    buttonDictionary = findViewById(R.id.buttonDictionary);
    buttonChangeLanguage = findViewById(R.id.buttonChangeLanguage);
    buttonStudy = findViewById(R.id.buttonStudy);


    buttonObjectDetection.setOnClickListener(v -> StartObjectDetectionActivity());

    buttonDictionary.setOnClickListener(v -> StartDictionaryActivity());

    buttonChangeLanguage.setOnClickListener(v -> {
        String locale = getResources().getString(R.string.locale);
        if(locale.equals("hr")){
            setAppLocale("en");
        }
        else {
            setAppLocale("hr");
        }
    });

    buttonStudy.setOnClickListener(v -> StartStudyActivity());
}
```

Slika 5.6 Metoda *InitializeUI* u *ActivityMain*

5.4 Korisnička dopuštenja

ObjectDetectionActivity je aktivnost koja služi za detekciju objekata, to jest za klasifikaciju slika. Kako bi funkcionirala, ova aktivnost traži dopuštenje od korisnika za pristup kameri i datotečnom sustavu. Dopuštenja koja aplikacija traži potrebno je prvo navesti u *AndroidManifest* kako je prikazano na slici 5.7.

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

Slika 5.7 Dopuštenja za kameru i pristup datotekama u *AndroidManifest.xml*

Upit korisniku za dopuštenje dan je na klik buttonTakePhoto kako je prikazano na slici 5.8. Aplikacija prvo provjerava verziju Android sustava u liniji koda *if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.M)*. Starije verzije (22 i niže) ne traže dopuštenje od korisnika tijekom rada nego prilikom instalacije pa u tom slučaju aplikacija samo nastavlja s radom. U slučaju da je verzija sustava dvadeset i tri ili novija, aplikacija provjerava je li korisnik već prethodno dao dopuštenje, ako jest - nastavlja s radom, ako nije - traži dopuštenje. Prvo se u polje stringova unose željena dopuštenja i zatim se poziva metoda *requestPermissions*.

```
buttonTakePhoto.setOnClickListener(v -> {
    if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.M){
        if(checkSelfPermission(Manifest.permission.CAMERA) == PackageManager.PERMISSION_DENIED
                || checkSelfPermission(Manifest.permission.WRITE_EXTERNAL_STORAGE) == PackageManager.PERMISSION_DENIED){
            String[] permission = {Manifest.permission.CAMERA, Manifest.permission.WRITE_EXTERNAL_STORAGE};
            requestPermissions(permission, PERMISSION_CODE);
        }
        else{
            OpenCamera();
        }
    }
    else {
        OpenCamera();
    }
});
```

Slika 5.8 Traženje dopuštenja za pristup kameri i datotečnom sustavu

Nakon korisnikovog odgovora na zahtjev pristupu, pokreće se metoda *onRequestPermissionResult* prikazanu na slici 5.9. Metoda prvo provjerava je li tražen pristup i kameri i datotečnom sustavu ili samo datotečnom sustavu. Nakon toga provjerava je li korisnik dozvolio pristup u liniji *if(grantResults > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED).* Ako ima dozvolu, aplikacija nastavlja s radom, a ako ne, ispisuje se *toast* poruka „Nema dopuštenja**”**.

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    switch (requestCode){
        case PERMISSION_CODE:{
            if(grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED){
                OpenCamera();
                break;
            }
            else {
                Toast.makeText( context: this,"Nema dopuštenje", Toast.LENGTH_SHORT).show();
                break;
            }
        }
        case PERMISSION_CODE2:{
            if(grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                LoadImage();
                break;
            }
            else {
                Toast.makeText( context: this,"Nema dopuštenje", Toast.LENGTH_SHORT).show();
                break;
            }
        }
    }
}

}
```

Slika 5.9 Metoda *onRequestPermissionResult*

5.5 Detekcija objekata

Ako korisnik dozvoli, aplikacija će pokrenuti drugu aplikaciju - kameru ili preglednik datoteka ovisno o potrebi te od njih čekati rezultat, sliku koju će tada aplikacija prikazati na ekranu i klasificirati. Metode koje to rade nalaze se na slici 5.10. Metoda *OpenCamera* daje ime novoj fotografiji i služi za otvaranje kamere, a metoda *LoadImage* služi za otvaranje preglednika datoteka i postavljena je tako da prikazuje samo jpeg i png format slika. Metoda *onActivityResult* poziva se nakon navedenih metoda i postavlja uslikanu ili odabranu sliku na ekran te poziva metodu *classifyImage* kako bi se klasificirala slika.

```
private void OpenCamera() {
    ContentValues contentValues = new ContentValues();
    contentValues.put(MediaStore.Images.Media.TITLE, "New Picture");
    imageUri = getContentResolver().insert(MediaStore.Images.Media.EXTERNAL_CONTENT_URI, contentValues);
    Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    cameraIntent.putExtra(MediaStore.EXTRA_OUTPUT, imageUri);
    startActivityForResult(cameraIntent, IMAGE_CAPTURE_CODE);
}

private void LoadImage() {
    Intent intent = new  Intent(Intent.ACTION_PICK);
    intent.setType("image/*");

    String[] mimeTypes = {"image/jpeg", "image/png"};
    intent.putExtra(Intent.EXTRA_MIME_TYPES, mimeTypes);

    startActivityForResult(intent, IMAGE_LOAD_CODE);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if(requestCode == IMAGE_CAPTURE_CODE){
        if (resultCode == RESULT_OK) {
            imageViewCaptured.setImageURI(imageUri);
            classifyImage(((BitmapDrawable)imageViewCaptured.getDrawable()).getBitmap());
        }
    }
    if(requestCode == IMAGE_LOAD_CODE){
        if (resultCode == RESULT_OK) {
            imageUri = data.getData();
            imageViewCaptured.setImageURI(imageUri);
            classifyImage(((BitmapDrawable)imageViewCaptured.getDrawable()).getBitmap());
        }
    }
}
```

Slika 5.10 Metode za učitavanje slike

Za klasifikaciju slike koristi se posebna klasa nazvana *ImageClassifier* koja prima *bitmapu* slike, komunicira s modelom i iščitava podatke iz rječnika te vraća rezultate klasifikacije. Unutar konstruktora prikazanog na slici 5.11, klasi se daje referenca na model strojnog učenja koji će se koristiti pomoću *setAssetFilePath* metode. Osim toga, klasa je postavljena tako da vraća rezultat samo ako je preciznost klasifikacije veća od 0.6. U konstruktoru se korištenjem *BufferedReader* iščitava rječnik u kojemu se nalaze oznake za klasifikaciju i spremaju se u listu stringova. Vrijednosti engleskog rječnika spremaju se u listu *englishDictionary*, a vrijednosti hrvatskog rječnika spremaju se u listu *croatianDictionary*.

```
public ImageClassifier(Context context) throws FirebaseMLException {

    FirebaseModelManager.getInstance()
            .registerLocalModel(
                    new FirebaseLocalModel.Builder(LOCAL_MODEL_NAME)
                            .setAssetFilePath(LOCAL_MODEL_PATH)
                            .build()
            );

    FirebaseVisionOnDeviceAutoMLImageLabelerOptions options = new FirebaseVisionOnDeviceAutoMLImageLabelerOptions.Builder()
            .setConfidenceThreshold(CONFIDENCE_THRESHOLD)
            .setLocalModelName(LOCAL_MODEL_NAME)
            .build();

    labeler = FirebaseVision.getInstance().getOnDeviceAutoMLImageLabeler(options);

    BufferedReader reader;
    try {
        reader = new BufferedReader(new InputStreamReader(context.getAssets().open( fileName: "automl/dict.txt")));
        String line;
        while((line = reader.readLine()) != null){
            englishDictionary.add(line);
        }
        reader.close();
    } catch (IOException e) {
        e.printStackTrace();
    }

    try {
        reader = new BufferedReader(new InputStreamReader(context.getAssets().open( fileName: "croDict.txt")));
        String line;
        while((line = reader.readLine()) != null){
            croatianDictionary.add(line);
        }
        reader.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Slika 5.11 Konstruktor klase *ImageClassifier*

Klasifikacija se odvija pomoću metode *classifyFrame* (Slika 5.12) koja prima *bitmapu* koju prvo pretvara u oblik koji može klasificirati koristeći *FirebaseVisionImage.fromBitmap*, zatim poziva metodu *labeler.processImage* kojom se vrši klasifikacija. Ako uspješno klasificira sliku, poziva metodu *printLabel* koja sastavlja *string* rezultat koji će se prikazati na ekranu. Metoda *printLabel* uspoređuje dodijeljenu oznaku s vrijednostima iz engleskog rječnika i dohvaća poziciju oznake, zatim dohvaća vrijednost iz hrvatskog rječnika na istoj poziciji zato što se tamo nalazi željeni prijevod riječi. Osim toga, ispisuje i postotak sigurnosti u klasifikaciju slike.

```
public Task<String> classifyFrame(Bitmap bitmap)  {
    if (labeler == null) {
        Log.e(TAG,  msg: "Image classifier has not been initialized; Skipped.");
        Exception e = new IllegalStateException("Uninitialized Classifier.");

        TaskCompletionSource completionSource = new TaskCompletionSource<String>();
        completionSource.setException(e);
        return completionSource.getTask();
    }

    long startTime = SystemClock.uptimeMillis();
    FirebaseVisionImage image = FirebaseVisionImage.fromBitmap(bitmap);

    return labeler.processImage(image).continueWith(task -> {
        long endTime = SystemClock.uptimeMillis();
        Log.d(TAG,  msg: "Time to run model inference: " + (endTime - startTime));

        List<FirebaseVisionImageLabel> labelProbList = task.getResult();


        String textToShow = "";
        if (labelProbList.isEmpty()){
            textToShow += "No Result";
        }
        else{
            textToShow += printLabel(labelProbList);
        }
        return textToShow;
    });

}

private String printLabel(List<FirebaseVisionImageLabel> labels) {
    String topLabel = labels.get(0).getText();
    int position = englishDictionary.indexOf(topLabel);
    topLabel += " - " + croatianDictionary.get(position);
    topLabel = topLabel.replace( target: "_",  replacement: " ");
    topLabel += "\n" + String.format("%.2f", (labels.get(0).getConfidence() * 100))  + "%";
    return topLabel;
}
```

Slika 5.12 Metode *classifyFrame* i *printLabel* u klasi *ImageClassifier*

5.6 Učenje pomoću edukacijskih kartica

Prvo se otvara aktivnost *ChooseSetActivity* koja ima padajući izbornik iz kojeg se može izabrati set fraza. Sami padajući izbornik koristi još dvije dodatne *xml* datoteke: *spinner_choice_item* i *spinner_choice_dropdown_item* kojima se određuje izgled padajućeg izbornika. Svakom setu je dodijeljen broj koji se koristi u aktivnosti *StudyActivity*. Kako bi se podaci slali iz jedne aktivnosti u drugu, potrebno je za *Intent* kojim se nova aktivnost poziva koristiti metodu *putExtra* kao što je prikazano na slici 5.13.

```
private void StartStudyActivity() {
    int set = spinnerChoice.getSelectedItemPosition();
    Intent intent = new Intent( packageContext: this, StudyActivity.class);
    intent.putExtra( name: "SET", set);
    startActivity(intent);
    finish();
}
```

Slika 5.13 Metoda *StartStudyActivity* kojom se pokreće *StudyActivity* i zatvara *ChooseSetActivity*

StudyActivity pri inicijalizaciji koristi broj seta i preko njega pohranjuje podatke o pojmovima u listu *Entry* objekata. *Entry* klasa (Slika 5.14) napravljena za držanje podataka o pojmovima: sadrži hrvatski i engleski naziv pojma te njegovu poziciju rječniku, *Entry* također implementira sučelje *Parcelable* kako bi se objekti mogli slati iz jedne aktivnosti u drugu. Klasa ima dva konstruktora, jedan uobičajeni koji prima dva stringa i jednu *integer* vrijednost koje sprema kao vrijednosti svojih atributa, a drugi konstruktor prima parcelu i služi za stvaranje Entry objekta preko podataka poslanih iz jedne aktivnosti u drugu korištenjem metoda *putParcellableArrayListExtra* i *getParcellableArrayList*.

```
public class Entry implements Parcelable{
    public String English, Croatian;
    public int Position;

    public Entry(String english, String croatian, int position){
        English = english;
        Croatian = croatian;
        Position = position;
    }


    protected Entry(Parcel in) {
        English = in.readString();
        Croatian = in.readString();
        Position = in.readInt();
    }

    public static final Creator<Entry> CREATOR = new Creator<Entry>() {
        @Override
        public Entry createFromParcel(Parcel in) {
            return new Entry(in);
        }

        @Override
        public Entry[] newArray(int size) {
            return new Entry[size];
        }
    };

    @Override
    public int describeContents() { return 0; }

    @Override
    public void writeToParcel(Parcel dest, int flags) {
        dest.writeString(English);
        dest.writeString(Croatian);
        dest.writeInt(Position);
    }
}
```

Slika 5.14 Klasa *Entry*

Čitanje riječi iz tekstualnih dokumenata *dict.txt* i *croDict.txt*, to jest rječnika, odvija se u metodi *InitializeDictionaries*, a spremanje pojmova u *Entry* listu odvija se u metodi *InitializeEntries* (Slika 5.15). *InitializeDirectories* pomoću klasa *BufferedReader* i *InputStreamReader* pristupa tekstualnim datotekama i korištenjem metode *readLine* čita datoteke red po red i dodaje ih u listu *englishDictionary* ili *croatianDictionary*. Metoda *InitializeEntries,* ako je vrijednost varijable *set* postavljena na nulu (nasumično), koristi metodu *ThreadLocalRandom.current().nextInt()* kako bi generirala nasumične cjelobrojne vrijednosti i spremila ih u listu *randomNumbers*. Kada dobije pet različitih nasumičnih brojeva, u listu *entries* sprema vrijednosti iz rječnika na istim pozicijama. Ako je vrijednost varijable *set* jedan, bit će spremljeno prvih pet pojmova iz rječnika, ako je vrijednost dva - drugih pet pojmova itd. To je definirano *for* petljom: *for(int i = (set - 1) * 5; i < (set – 1) * 5 + 5; i++) { entries.add(new Entry(englishDictionary.get(i).replace(„_”, „ “), croatianDictionary.get(i), i)); }*

```
private void InitializeDictionaries() {
    BufferedReader reader;
    try {
        reader = new BufferedReader(new InputStreamReader(getAssets().open( fileName: "automl/dict.txt")));
        String line;
        while((line = reader.readLine()) != null){
            englishDictionary.add(line);
        }
        reader.close();
    } catch (IOException e) {
        e.printStackTrace();
    }

    try {
        reader = new BufferedReader(new InputStreamReader(getAssets().open( fileName: "croDict.txt")));
        String line;
        while((line = reader.readLine()) != null){
            croatianDictionary.add(line);
        }
        reader.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
    InitializeEntries();
}
private void InitializeEntries() {
    if(set == 0){
        ArrayList<Integer> randomNumbers = new ArrayList<>();
        int randomNumber;
        while (randomNumbers.size() < 5){
            randomNumber = ThreadLocalRandom.current().nextInt( origin: 0, englishDictionary.size());
            if(randomNumbers.indexOf(randomNumber) == -1){
                randomNumbers.add(randomNumber);
            }
        }
        for (int number: randomNumbers) {
            entries.add(new Entry(englishDictionary.get(number).replace( target: "_",  replacement: " "), croatianDictionary.get(number), number));
        }
    }
    else {
        for(int i = (set - 1) * 5; i < (set - 1) * 5 + 5; i++){
            entries.add(new Entry(englishDictionary.get(i).replace( target: "_",  replacement: " "), croatianDictionary.get(i), i));
        }
```

Slika 5.15 Metode *InitializeDictionaries* i *InitializeEntries*

Klikom na gumb **„sljedeći”,** aktivnost pokazuje jednu po jednu od svih pet učitanih kartica. Slike se nalaze u mapi *images* unutar mape *assets* i označene su brojevima koji odgovaraju poziciji pojmova koje predstavljaju u rječniku. Nakon prikaza kartica poziva se *Collections.shuffle(entries)* kako bi se izmijenio redoslijed kojim će biti postavljana pitanja u kvizu, a unutar metode *StartQuizActivity*(Slika 5.16) nasumično započinje jedna od četiri kviz aktivnosti. Kao i u prethodnom slučaju, nasumični broj se generira korištenjem *ThreadLocalRandom.current().nextInt()*. Novoj aktivnosti šalje se lista *entries*, broj točnih odgovora postavljen na nulu i *boolean* varijabla postavljena na *true* koja označava da je to prvi kviz. Metoda *startActivity(intent)* započinje novu aktivnost, a metoda *finish()* zatvara trenutnu aktivnost.

```
        else {
            Collections.shuffle(entries);
            StartQuizActivity();
        }
    });
}

private void StartQuizActivity() {
    int randomNumber = ThreadLocalRandom.current().nextInt( origin: 0, bound: 5);
    Intent intent;
    if (randomNumber == 0){
        intent = new Intent( packageContext: this, Quiz1Activity.class);
    }
    else if (randomNumber == 1){
        intent = new Intent( packageContext: this, Quiz2Activity.class);
    }
    else if (randomNumber == 2){
        intent = new Intent( packageContext: this, Quiz3Activity.class);
    }
    else {
        intent = new Intent( packageContext: this, Quiz4Activity.class);
    }
    intent.putParcelableArrayListExtra( name: "ENTRIES", (ArrayList<? extends Parcelable>) entries);
    intent.putExtra( name: "CORRECT_ANSWERS", value: 0);
    intent.putExtra( name: "FIRST_QUIZ", value: true);

    startActivity(intent);
    finish();
}
```

Slika 5.16 Metoda *StartQuizActivity*

5.7 Kviz

Kviz je predstavljen s četiri aktivnosti (*QuizActivity1*, *QuizActivity2*, *QuizActivity3* i *QuizActivity4*) s različitim načinima postavljanja pitanja. Sve četiri aktivnosti iz prethodne aktivnosti primaju listu *Entry* objekata nazvanu *entries*, cjelobrojnu vrijednost koja označava

broj do sada točno odgovorenih pitanja i *boolean* vrijednost koja označava radi li se o prvom ili drugom kvizu. Ako joj je vrijednost *true,* znači da je to prvi kviz i da će na kraju pozvati jedan od preostala tri kviza, ako je vrijednost *false*, znači da će na kraju pozvati *ResultActivity* gdje će biti ispisan rezultat kviza, to jest broj točnih odgovora od ukupno deset postavljenih pitanja.

U aktivnosti *Quiz1Activity* prikazuje se slika pojma i od korisnika se traži da upiše engleski naziv, program provjerava jednakost upisanog teksta i engleskog naziva i preko *toast* poruke na ekran ispisuje „Točno" ili „Netočno" (Slika 5.17). Kviz redom prikazuje sve pojmove koji se nalaze u *entries* listi i na kraju listu izmiješa pomoću *Collections.shuffle(entries)* prije nego pozove novu aktivnost. Jednakost stringova provjerava se pomoću metode *equals*. Ako je odgovor točan, osim ispisivanja *toast* poruke uvećava se i vrijednost varijable *correctAnswers*. Aktivnost sa svakim odgovorom uvećava vrijednost varijable *counter*, to jest brojača i kada je njegova vrijednost pet znači da su postavljena sva pitanja. U tom se slučaju mijenja redoslijed liste *entries* kako redoslijed pitanja ne bi bio isti u sljedećem kvizu te se pokreće sljedeća aktivnost metodom *StartNextActivity* (Slika 15.18). *StartNextActivity* provjerava vrijednost *firstQuiz* i ako je *true* generira nasumičan broj između nula i tri i ovisno o tome pokreće jedan od preostalih kvizova. Ako je vrijednost *false,* pokreće se aktivnost *ResultActivity*.

```
buttonConfirm.setOnClickListener(v -> {
    String answer = editTextAnswer.getText().toString().toLowerCase().trim();
    if(answer.equals(entries.get(counter).English)){
        correctAnswers++;
        Toast.makeText( context: Quiz1Activity.this, R.string.correct, Toast.LENGTH_SHORT).show();
    }
    else {
        Toast.makeText( context: Quiz1Activity.this, R.string.incorrect, Toast.LENGTH_SHORT).show();
    }
    counter++;
    if (counter<5){
        editTextAnswer.setText("");
        try {
            InputStream inputStream = getAssets().open( fileName: "images/" + entries.get(counter).Position + ".jpg");
            Drawable drawable = Drawable.createFromStream(inputStream, srcName: null);
            imageViewItem.setImageDrawable(drawable);
            inputStream.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    else {
        Collections.shuffle(entries);
        StartNextActivity();
    }
});
```

Slika 5.17 Provjera točnosti odgovora u *ActivityQuiz1*

```
private void StartNextActivity() {
    Intent intent;
    if (firstQuiz){
        int randomNumber = ThreadLocalRandom.current().nextInt( origin: 0,  bound: 3);
        if (randomNumber == 0){
            intent = new Intent( packageContext: this, Quiz2Activity.class);
        }
        else if (randomNumber == 1){
            intent = new Intent( packageContext: this, Quiz3Activity.class);
        }
        else {
            intent = new Intent( packageContext: this, Quiz4Activity.class);
        }
        intent.putParcelableArrayListExtra( name: "ENTRIES", (ArrayList<? extends Parcelable>) entries);
        intent.putExtra( name: "FIRST_QUIZ",  value: false);
    }
    else {
        intent = new Intent( packageContext: this, ResultActivity.class);
    }
    intent.putExtra( name: "CORRECT_ANSWERS", correctAnswers);
    startActivity(intent);
    finish();
}
```

Slika 5.18 *StartNextActivity* metoda - poziv sljedeće aktivnosti u *QuizActivity1*

QuizActivity2 nasumično ispisuje ili hrvatski ili engleski naziv pojma i traži od korisnika da unese prijevod. Točnost odgovora se provjerava na isti način kao i u prvom kvizu. *QuizActivity3* prikazuje sliku pojma i nudi tri odgovora korisniku, korisnik treba pritisnuti na gumb s odgovorom, tekst gumba se uspoređuje s engleskim nazivom i utvrđuje se točnost odgovora i tako za svih pet pojmova. *QuizActivity4* ispisuje engleski naziv pojma i nudi korisniku četiri slike kao odgovore na koje može kliknuti, svaka slika ima spremljenu vrijednost pozicije riječi koju predstavlja u varijablama *image1Index*, *image2Index*, *image3Index* ili *image4Index*. Ovisno o odgovoru, jedna od vrijednosti se uspoređuje s indeksom točnog odgovora i korisniku se na ekran ispisuje je li odgovorio točno ili netočno (Slika 5.19). Poziv sljedeće aktivnosti, to jest metoda *StartNextActivity* jednaka je kod svih kviz aktivnosti.

```
imageViewAnswer1.setOnClickListener(v -> {
    if(image1Index == entries.get(counter).Position){
        correctAnswers++;
        Toast.makeText( context: Quiz4Activity.this, "Točno", Toast.LENGTH_SHORT).show();
    }
    else {
        Toast.makeText( context: Quiz4Activity.this, "Netočno", Toast.LENGTH_SHORT).show();
    }
    counter++;
    if (counter < 5){
        SetUpQuestion();
    }
    else {
        Collections.shuffle(entries);
        StartNextActivity();
    }
});
```

Slika 5.19 Provjera točnosti odgovora u aktivnosti QuizActivity4

Na kraju kviza otvara se *ResultActivity* (Slika 5.20) koja ispisuje broj točnih odgovora iz prethodne dvije aktivnost. Izlaskom iz ove aktivnosti korisnik se vraća na *ActivityMain* - glavni izbornik.

```
public class ResultActivity extends AppCompatActivity {

    private int correctAnswers;
    private TextView textViewResult;
    private Button buttonClose;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_result);
        Bundle data = getIntent().getExtras();
        correctAnswers = data.getInt( key: "CORRECT_ANSWERS");
        InitializeUI();
    }

    private void InitializeUI() {
        textViewResult = findViewById(R.id.textViewResult);
        buttonClose = findViewById(R.id.buttonClose);

        textViewResult.setText(correctAnswers + "/10\n" + "točnih odgovora");

        buttonClose.setOnClickListener(v-> finish());
    }
}
```

Slika 5.20 *ResultActivity* aktivnost

5.8 Rječnik

DictionaryActivity koristi gumb *buttonLanguage* za postavljanje pretraživanja hrvatskog ili engleskog jezika - mijenja *boolean* varijablu *englishToCroatian* na *true* ili *false* (Slika 5.21).

```
buttonLanguage.setOnClickListener(v -> {
    if(englishToCroatian){
        englishToCroatian = false;
        buttonLanguage.setText("Hrvatski na engleski");
    }
    else {
        englishToCroatian = true;
        buttonLanguage.setText("Engleski na hrvatski");
    }
});
```

Slika 5.21 *buttonLanguage*

Pritiskom na gumb **„**pretraži**”** pokreće se metoda *SearchWords* (Slika 5.22), uzima se tekst iz polja koje je korisnik upisao, na njemu se izvrši metoda *toLowerCase()* kako na pretraživanje ne bi utjecala velika i mala slova te metoda *trim()* kako bi se eliminirali dodatni, nepotrebni razmaci. Zatim ga se uspoređuje sa svim riječima u rječniku koristeći metodu *startsWith()* te se time dohvaćaju sve riječi koje počinju jednako kao traženi tekst i ispisuje ih se u listu *listViewSearchResults*. Na kraju je potrebno pozvati funkciju *searchResultsAdapter.notifyDataSetChanged()* kako bi rezultati bili ispisani na ekranu.

```
private void SearchWords() {
    String searchText = editTextSearch.getText().toString().toLowerCase().trim();
    results.clear();
    searchResultsAdapter.notifyDataSetChanged();
    String result;
    if(englishToCroatian){
        for (String entry: englishDictionary) {
            if(entry.startsWith(searchText)){
                result = entry;
                int position = englishDictionary.indexOf(entry);
                result += " - " + croatianDictionary.get(position);
                result = result.replace( target: "_",  replacement: " ");
                results.add(result);
            }
        }
    }
    else {
        for (String entry: croatianDictionary) {
            if(entry.startsWith(searchText)){
                result = entry;
                int position = croatianDictionary.indexOf(entry);
                result += " - " + englishDictionary.get(position);
                result = result.replace( target: "_",  replacement: " ");
                results.add(result);
            }
        }
    }
    if(results.isEmpty()){
        Toast.makeText( context: DictionaryActivity.this, text: "No results were found.",Toast.LENGTH_SHORT).show();
    }
    searchResultsAdapter.notifyDataSetChanged();
}
```

Slika 5.22 *SearchWords* metoda u *DictionaryActivity*

Za ispis riječi i prijevoda *ListView* koristi se posebna *xml layout* datoteka - *search_results_layout.xml* i *adapter* klasa - *SearchResultsAdapter* koja nasljeđuje apstraktnu klasu *BaseAdapte*r (Slika 5.23). Klasa *SearchResultsAdapter* ima listu string vrijednosti koja služi za spremanje rezultata pretraživanja i *LayoutInflater* koji se koristi za povezivanje vrijednosti rezultata sa željenim elementima *search_results_layout* rasporeda u metodi *getView()* kako bi se omogućilo njihovo ispisivanje.

```java
public class SearchResultsAdapter extends BaseAdapter {
    private Activity Context;
    private ArrayList<String> Results;
    private static LayoutInflater inflater = null;

    public SearchResultsAdapter(Activity context, ArrayList<String> results){
        Context = context;
        Results = results;
        inflater = (LayoutInflater) Context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    }

    @Override
    public int getCount() { return Results.size(); }

    @Override
    public Object getItem(int position) { return Results.get(position); }

    @Override
    public long getItemId(int position) { return position; }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View itemView = convertView;
        itemView = (itemView==null) ? inflater.inflate(R.layout.search_results_layout, root: null):itemView;
        TextView textViewResult = itemView.findViewById(R.id.textViewResult);
        String selectedResult = Results.get(position);
        textViewResult.setText(selectedResult);
        return itemView;
    }
}
```

Slika 5.23 Klasa *SearchResultsAdapter*

6. PRIKAZ NAČINA RADA I ISPITIVANJE APLIKACIJE

U ovom poglavlju prikazan je način rada aplikacije popraćen prikazima zaslona te ispitivanje točnosti aplikacije.

6.1 Prikaz načina rada aplikacije

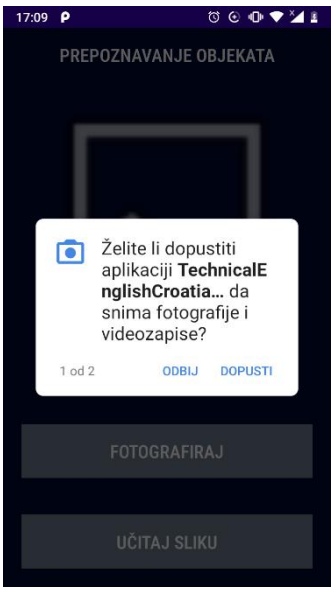
Kada korisnik pokrene aplikaciju otvara se početni zaslon (Slika 6.1). Na početnom zaslonu nalaze se četiri gumba: „prepoznavanje objekata”, „učenje”, „rječnik” i „promijeni jezik”.

Slika 6.1 Početni zaslon

Kada korisnik klikne na gumb „prepoznavanje objekata”, otvara se novi, istoimeni zaslon (Slika 6.2). Na tom se zaslonu pokazuje ikona za sliku na mjestu gdje će biti učitana slika koju korisnik želi klasificirati i dva gumba.

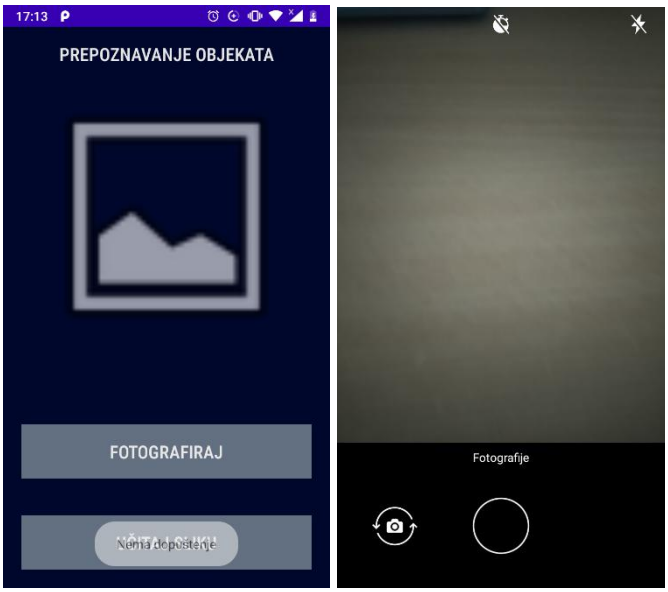
Slika 6.2 Zaslon za prepoznavanje objekata

Ako korisnik prvi puta u aplikaciji koristi mogućnost detekcije objekata, kada klikne na gumb **„**fotografiraj**”** aplikacija će tražiti dopuštenje za pristup kameri i datotekama (Slika 6.3). Kako bi koristio funkcionalnosti aplikacije, korisnik treba kliknuti „dopusti**”** za sve upite.



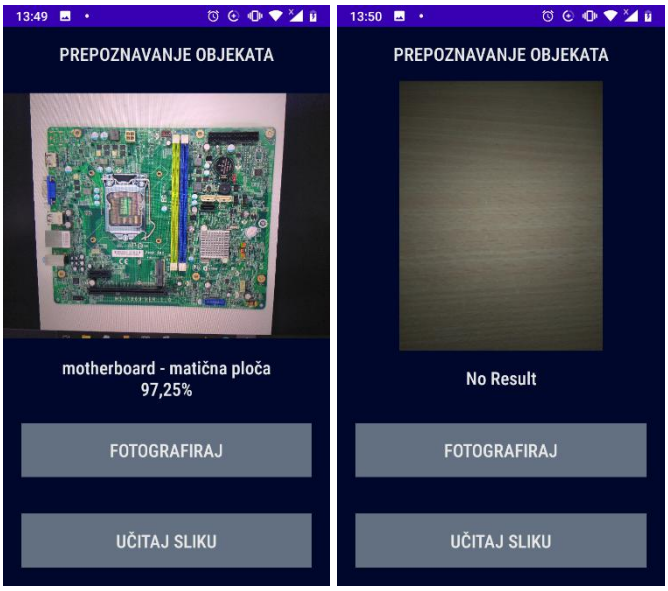
Slika 6.3 Upit za dozvolu pristupu

Ako korisnik ne dozvoli aplikaciji pristup, kamera se neće otvoriti i na ekran će biti ispisana *toast* poruka „Nema dopuštenje" (Slika 6.4, lijevo). Ako korisnik dozvoli pristup aplikaciji, otvara se kamera (Slika 6.4, desno).



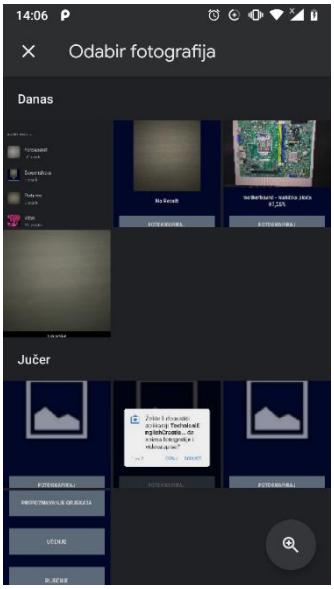
Slika 6.4 Lijevo - aplikacija nema pristup kameri ili datotekama; desno - kamera otvorena u aplikaciji

Kada korisnik snimi fotografiju, ona se prikazuje na zaslonu te se ispod nje ispisuje rezultat klasifikacije u obliku „engleski naziv – hrvatski naziv" te postotak sigurnosti u klasifikaciju sveden na dvije decimale kao što je prikazano na slici 6.5 lijevo. Ako objekt na slici nije moguće klasificirati, ispod slike će se ispisati "*No Result*" (Slika 6.5, desno). Objekt neće biti klasificiran ako je sigurnost klasifikacije manja od 0.6, to jest 60%.



Slika 6.5 Lijevo - uspješno klasificirana slika; desno - neuspješno klasificirana slika

Kada korisnik klikne na gumb **"**učitaj sliku**"**, na zaslonu „Prepoznavanje objekata**"** otvara se galerija slika na uređaju i prikazuju se sve slike formata .jpeg i .png (Slika 6.6). Korisnik tada iz galerije može odabrati sliku koju želi klasificirati. Aplikacija će tražiti pristup datotekama ako nije prethodno dozvoljen. Kada korisnik učita sliku rezultat će se ispisati jednako kao i za fotografiju, odnosno ispisat će se „No Result**"** ako klasifikacija nije uspješna.



Slika 6.6 Odabir fotografija u aplikaciji

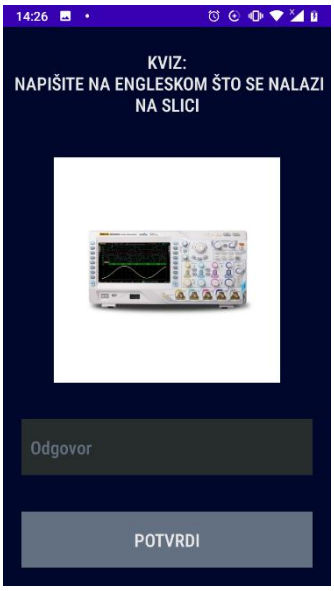
Klikom na gumb **“**učenje**”** u glavnom izborniku otvara se zaslon za odabir seta pojmova (Slika 6.7). U gornjem dijelu ekrana nalazi se padajući izbornik kojemu je prvotno zadana opcija **„**nasumično**”**. U padajućem izborniku nalazi se još šest opcija – od „Set 1**”** do „Set 6**”**. Svaki set sadrži po pet riječi, a opcija „nasumično**”** isto tako uzima nasumično pet riječi iz rječnika. Nakon što korisnik odabere željeni set, treba kliknuti na gumb **„**potvrdi**”** na dnu ekrana kako bi nastavio.

Slika 6.7 Zaslon za odabir seta pojmova

Nakon što korisnik potvrdi odabir, otvara se novi zaslon na kojemu se nalazi edukacijska kartica koju čine slika, engleski i hrvatski naziv pojma (Slika 6.8). Na dnu ekrana nalazi se gumb „sljedeći**”**, a klikom na taj gumb prikazuje se sljedeći pojam.

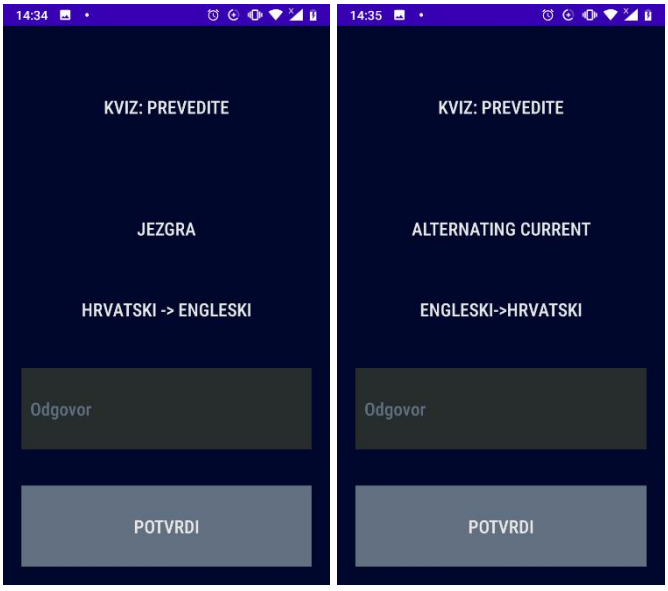
Slika 6.8 Edukacijska kartica

Nakon prikaza svih pet kartica iz seta otvara se kviz. Postoje četiri različita tipa kviza i za svaki prikazani set otvorit će se dva kviza jedan za drugim. Kvizovi se mogu pojaviti u bilo kojoj kombinaciji i bilo kojim redoslijedom. Prvi tip kviza (Slika 6.9) prikazuje sliku i traži od korisnika da upiše njezin engleski naziv u tekstualno polje „Odgovor" nakon čega korisnik treba kliknuti na gumb „potvrdi" da bi nastavio dalje. Kviz, ovisno o točnosti odgovora, ispisuje *toast* poruku „točno" ili „netočno" na ekran. Svi kvizovi postavljaju po jedno pitanje za svaki pojam koji se nalazi u edukacijskim karticama. Nakon što je korisnik odgovorio na pet pitanja, otvara se nova aktivnost, to jest novi kviz ili se prikazuje konačni rezultat.



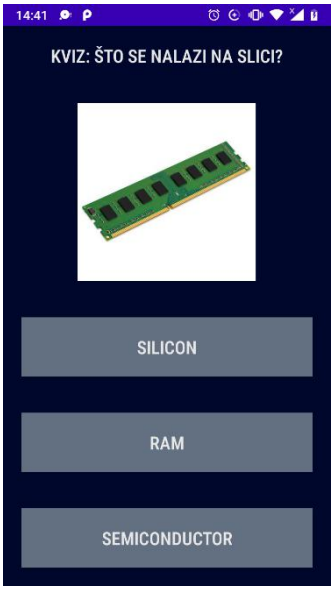
Slika 6.9 Prvi tip kviza

Drugi tip kviza (Slika 6.10) ispisuje hrvatski naziv i od korisnika traži engleski prijevod ili ispisuje engleski naziv i traži hrvatski prijevod. Korisnik odgovor mora upisati u tekstualno polje „Odgovor" i kliknuti gumb „potvrdi" kako bi provjerio točnost svoga odgovora. Kviz, ovisno o točnosti odgovora, ispisuje *toast* poruku „točno" ili „netočno" na ekran i postavlja sljedeće pitanje ili otvara novu aktivnost.



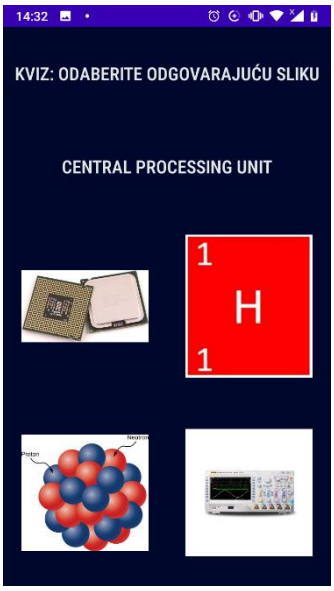
Slika 6.10 Drugi tip kviza

Treći tip kviza (Slika 6.11) prikazuje sliku i na tri gumba nudi različite odgovore. Od korisnika se traži da odabere točan engleski naziv. Svi ponuđeni odgovori su pojmovi koji su se pojavili u setu. Kviz nasumično prikazuje slike svih pet pojmova koji su bili prikazani korisniku, a točnost odgovora ispisuje se *toast* porukom kao i kod ostalih kvizova.



Slika 6.11 Treći tip kviza

Četvrti tip (Slika 6.12) kviza ispisuje engleski naziv pojma i nudi četiri različite slike te traži od korisnika da klikne na sliku koja predstavlja napisani pojam. Kao i kod prethodnih kvizova, postavlja se pet pitanja nasumično o pojmovima koji su prethodno bili prikazani, a točnost odgovora ispisuje se toast porukom „točno” ili „netočno”.



Slika 6.12 Četvrti tip kviza

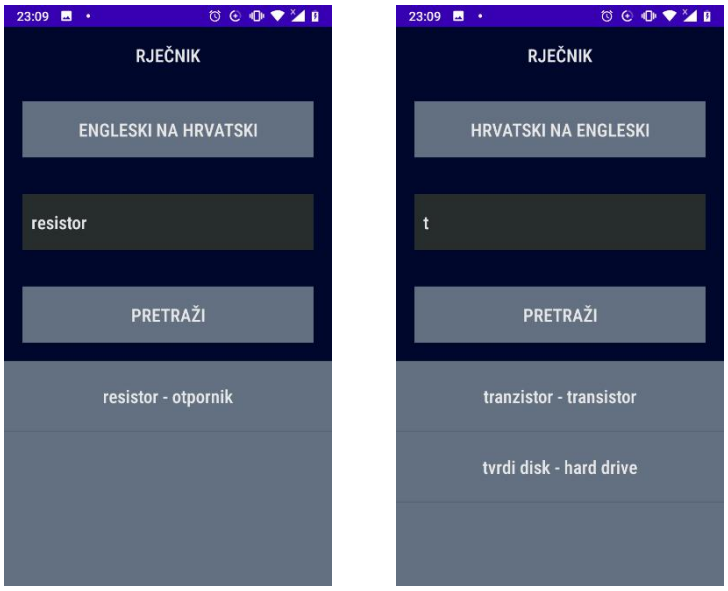
Nakon kviza korisniku se na zaslon ispisuje rezultat s ukupnim brojem točnih odgovora (Slika 6.13). Na ekranu se samo nalazi tekst „X/10 točnih odgovora" gdje je 'X' broj točnih odgovora u prethodna dva kviza i gumb „zatvori" kojim se korisnik vraća na glavni izbornik. Korisnik se može vratiti na glavni izbornik i klikom na tipku natrag na svom uređaju.

Slika 6.13 Zaslon s rezultatom kviza

Ako korisnik u glavnom izborniku klikne na gumb „rječnik”, otvara se novi zaslon u kojemu može pretraživati rječnik (Slika 6.14).

Slika 6.14 Zaslon s rječnikom

Korisnik klikom na gumb na vrhu ekrana može odabrati želi li pretraživati preko engleskih (Slika 6.15, lijevo) ili hrvatskih izraza (Slika 6.15, desno). Početna postavka je prijevod s engleskog na hrvatski te je na gumbu napisano „engleski na hrvatski”, a klikom na njega naziv se mijenja u „hrvatski na engleski”, što znači da će se kao rezultat pretraživanja davati prijevodi s hrvatskog na engleski jezik. Korisnik zatim treba upisati željeni izraz ili početni dio izraza u tekstualno polje „Pretraži” i kliknuti na gumb „pretraži”. Rezultati se ispisuju u listu na dnu ekrana. Ako nije pronađen niti jedan rezultat ispisuje se *toast* poruka „Nema rezultata”.



Slika 6.15 Pretraživanje rječnika

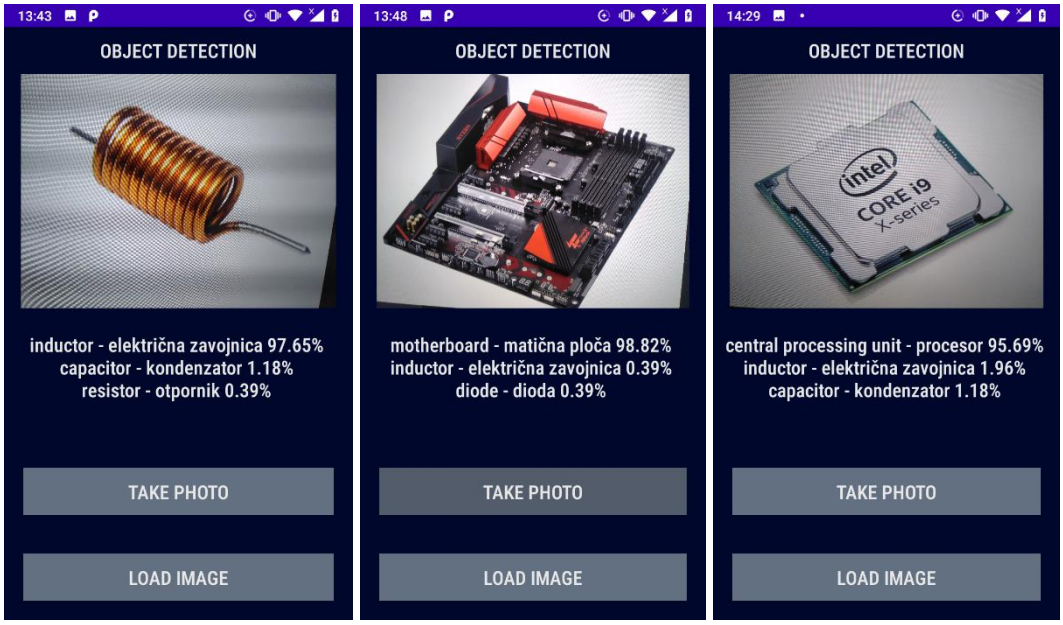
Klikom na gumb **„**promijeni jezik**”** u glavnom izborniku, korisnik može promijeniti jezik sučelja aplikacije. Prikaz glavnog izbornika na engleskom jeziku nalazi se na slici 6.16.

Slika 6.16 Glavni izbornik na engleskom jeziku

6.2 Prikaz ispitivanja aplikacije

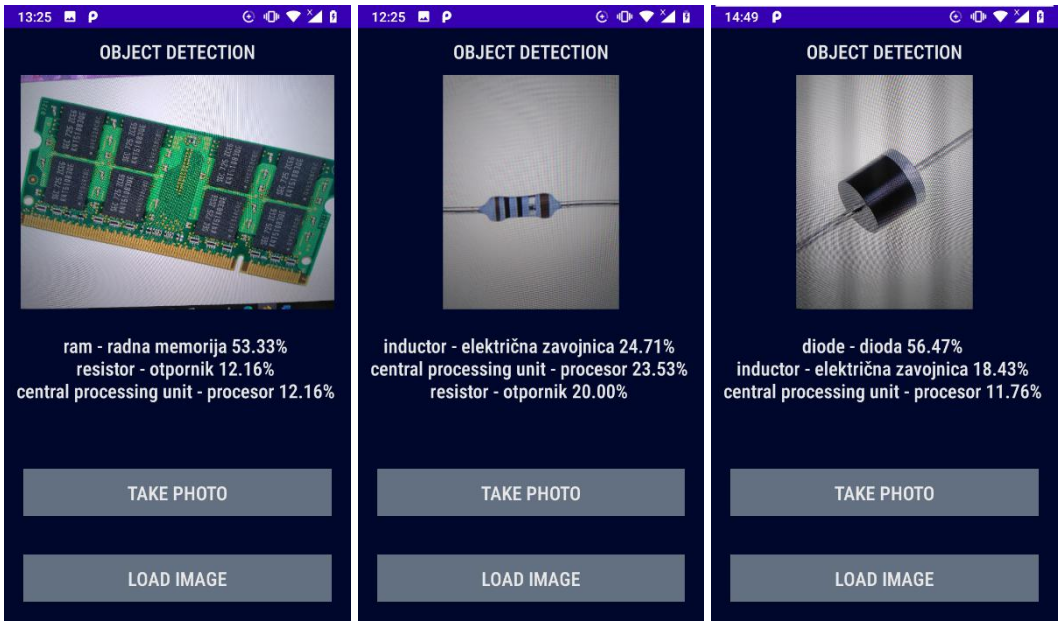
U ovom potpoglavlju ispitivat će se točnost aplikacije pri klasifikaciji objekata. Za svaku od deset klasa objekata bit će izabrane četiri slike koje će biti uslikane kamerom mobilnog uređaja po dva puta. Ispitivat će se je li klasifikacija uspješna i točna. Aplikacija će biti modificirana tako da prikazuje prva tri rezultata s najvećim sigurnostima klasifikacije kako bi se bolje analizirala točnost modela strojnog učenja korištenog u aplikaciji.

Najveća točnost postignuta je pri klasifikaciji električne zavojnice, matične ploče i procesora (prosječno oko 90%). To je najvjerojatnije zato što su ovi objekti najjasnije definirani, to jest imaju jedinstvene oblike i detalje koji se lako raspoznaju. Primjeri klasifikacije ovih objekata nalaze se na slici 6.17.



Slika 6.17 Primjeri prepoznavanja električne zavojnice(lijevo), matične ploče(sredina) i procesora(desno)

Nešto manja preciznost (prosječno oko 70%) postignuta je pri klasifikaciji otpornika, radne memorije i diode, gdje su određene slike klasificirane velikom vjerojatnošću dok su druge nešto nižom vjerojatnošću. Primjeri prepoznavanja ovih objekata s manjom vjerojatnošću nalaze se na slici 6.18. Do slabijeg prepoznavanja dolazi zato što je set podataka kojima je treniran model strojnog učenja malen pa se ne nalazi velik broj različitih vrsta otpornika, dioda i radne memorije da bi svi bili lako raspoznatljivi.

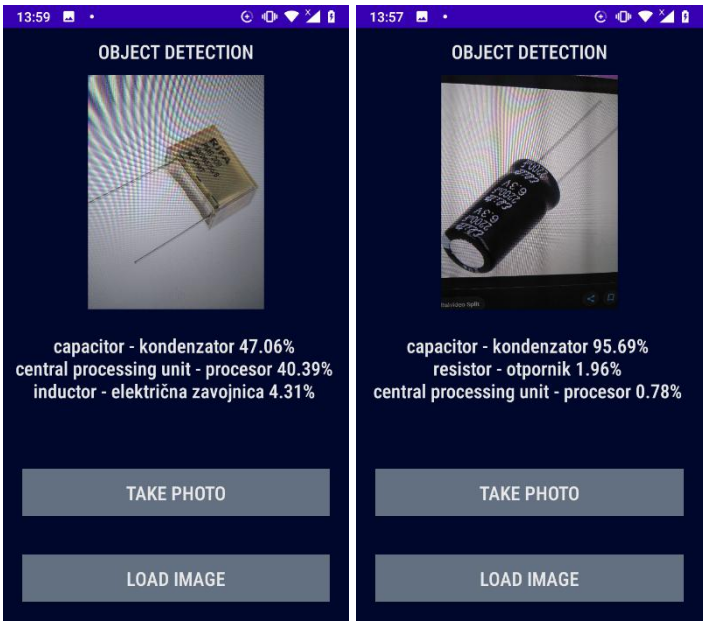


Slika 16.18 Primjeri prepoznavanja radne memorije(lijevo), otpornika(sredina) i diode(desno)

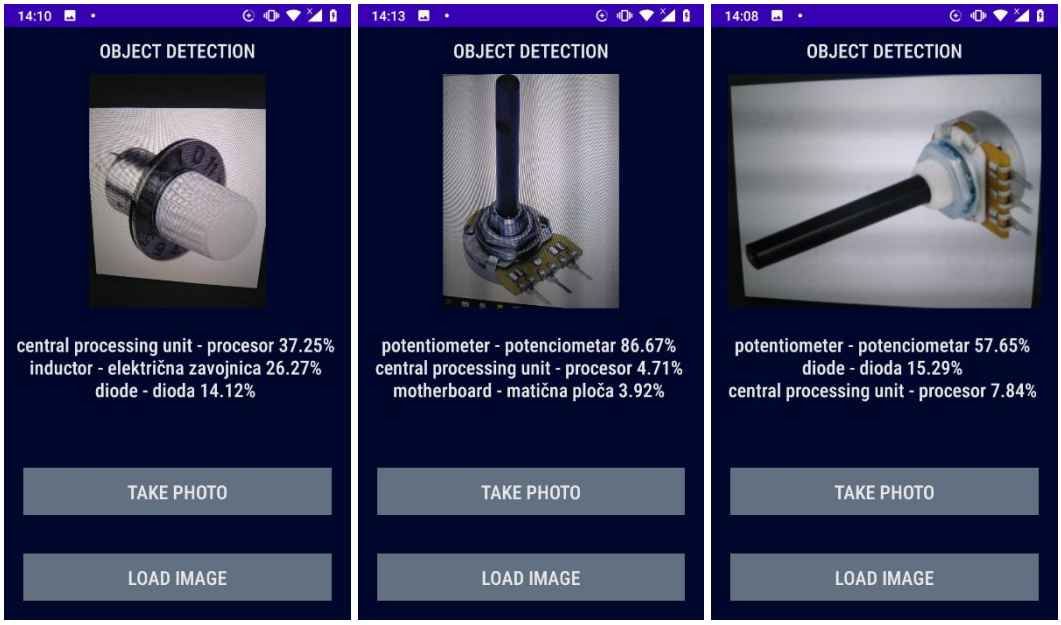
Pri klasifikaciji kondenzatora i potenciometra prosječna je preciznost još manja (oko 60%). Razlozi su slični kao i kod prethodnih klasifikacija. Set podataka ne pokriva sve različite oblike kondenzatora i potenciometra, zato model ne prepoznaje nove oblike koji su različiti od onih u setu podataka.

Na slici 16.19 lijevo, prikazan je kondenzator koji nije precizno klasificiran, već je zbog svojeg pravokutnog oblika podjednako klasificiran i kao procesor.

Na slici 16.20 prikazani su primjeri klasifikacije potenciometra. Lijevo se nalazi neuspješno klasificirani potenciometar. Ni jedan od prva tri rezultata njegove klasifikacije nije potenciometar, što je najvjerojatnije rezultat toga što se u setu podataka ne nalazi ni jedan primjer potenciometra sličnog njemu. U sredini se nalazi potenciometar klasificiran s velikom sigurnošću, a desno potenciometar sličan njemu, ali klasificiran s manjom vjerojatnošću zbog različitog položaja na kojemu se nalazi potenciometar na slici.



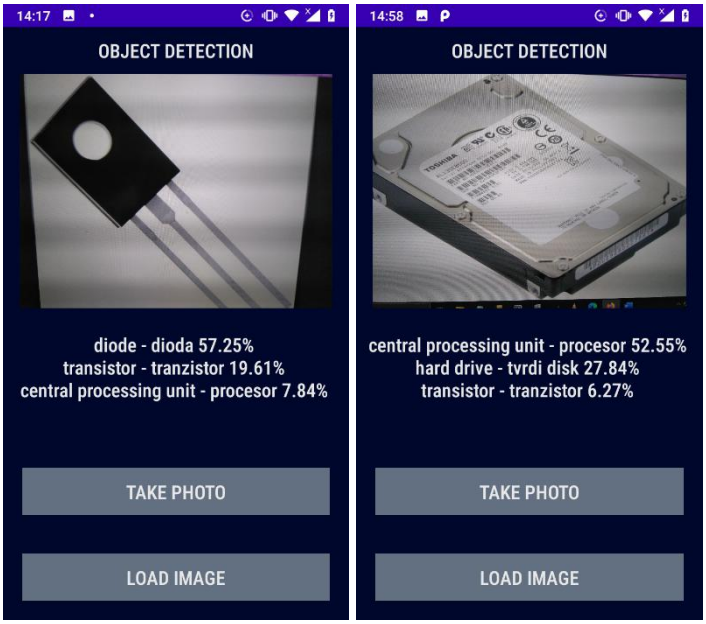
Slika 6.19 Primjeri prepoznavanja kondenzatora: lijevo – mala preciznost, desno – velika preciznost



Slika 6.20 Primjeri prepoznavanja potenciometra: lijevo – neuspješna klasifikacija, sredina – velika preciznost, desno – mala preciznost

Najmanja preciznost postignuta je pri klasifikaciji tranzistora i tvrdog diska (prosječno oko 40%). Naime tranzistor je često krivo klasificiran kao dioda zbog sličnog oblika, dok je iz istog razloga

tvrdi disk često prepoznat kao procesor. Primjeri klasifikacije ovih objekata nalaze se na slici 6.21.



Slika 6.21 Primjeri prepoznavanja tranzistora(lijevo) i tvrdog diska(desno)

U tablici 6.1 nalaze se podaci o točnostima klasifikacije testirani na nekoliko slika za svih deset objekata te njihova prosječna točnost. U slučaju kada se točna klasifikacija ne nalazi u prva tri rezultata s najvećim postotkom sigurnosti, u tablici je označena s 0%. U tablici je vidljivo da su već spomenute klase s najvećom prosječnom točnosti imale veliku sigurnost klasifikacije u svim testovima, one s nešto manjom točnosti su imale nekoliko testova gdje im je sigurnost mala, a klase s najmanjom prosječnom točnosti imale su malu sigurnost u većini slučajeva.

Tablica 6.1 Rezultati ispitivanja točnosti aplikacije

	1. slika, 1. klasifikacija	1. slika, 2. klasifikacija	2. slika, 1. klasifikacija	2. slika, 2. klasifikacija	3. slika, 1. klasifikacija	3. slika, 2. klasifikacija	4. slika, 1. klasifikacija	4. slika, 2. klasifikacija	Prosječna točnost
otpornik	98.04%	97.65%	86.27%	20.00%	45.10%	53.73%	98.82%	98.04%	74.71%
radna memorija	93.73%	80.00%	53.33%	39.22%	86.67%	78.43%	90.98%	57.65%	72.50%
električna zavojnica	99.22%	89.80%	99.22%	98.43%	72.94%	81.96%	97.25%	97.65%	92.06%
matična ploča	98.82%	98.43%	78.82%	79.22%	92.55%	96.47%	84.71%	72.94%	87.75%
kondenzator	84.31%	95.69%	81.18%	92.94%	47.06%	21.57%	19.22%	71.76%	64.22%
potenciometar	91.37%	86.67%	65.49%	57.65%	0%	0%	86.67%	85.88%	59.22%
tranzistor	27.84%	14.12%	19.61%	23.53%	97.25%	81.96%	21.18%	12.94%	37.30%
procesor	95.69%	96.86%	90.98%	98.04%	94.12%	94.51%	93.33%	57.65%	90.15%
dioda	50.98%	78.82%	86.67%	79.61%	75.69%	56.47%	50.20%	80.39%	69.85%
tvrdi disk	27.84%	7.06%	74.90%	77.65%	66.27%	34.90%	25.10%	51.76%	45.69%

7. ZAKLJUČAK

Svrha rada je programirati aplikaciju koja će služiti za učenje tehničkog engleskog jezika i koja će imati mogućnost prepoznavanja objekata sa slike. Analizirano je nekoliko postojećih aplikacija koje služe za učenje stranih jezika kako bi na njihovu primjeru moglo biti napravljeno rješenje za ovaj rad. Zatim su analizirane tehnologije potrebne za rad aplikacije - operacijski sustav, programski jezik i ostali alati koji su korišteni za strojno učenje. Nakon toga definirane su željene funkcionalnosti aplikacije i prikaz *mock-upa*, to jest plan izgleda aplikacije i dijagram toka. Slijedi objašnjenje pojedinih funkcionalnost nakon čega je dano programsko rješenje s isječcima koda i pojašnjenjima na koji način program radi. Konačno, prikazan je sam rad aplikacije s pridruženim snimkama zaslona i uputama za korištenje te je analizirana točnost aplikacije. Za većinu klasa objekata koje aplikacija može prepoznati preciznost je dobra ili odlična, jedino klasifikacija tranzistora i tvrdog diska ima točnost bitno manju od 60% (prosječno 37.3% za tranzistor i 45.69% za tvrdi disk). Analizom točnosti u stvarnim slučajevima vidljivo je da je preciznost zapravo manja nego što je pokazala prethodna analiza pomoću podataka iz seta kojim je model treniran.

Sve navedene funkcionalnosti su ostvarene, no postoje mogućnosti za njihovo proširenje. Aplikacija bi se unaprijedila povećanjem broja objekata koje utrenirani model strojnog učenja može prepoznati kao i mogućnošću dodavanja većeg broja engleskih riječi koje bi korisnici mogli naučiti koristeći aplikaciju.

LITERATURA

[1] What is Machine Learning? A definition, Expert System, 2020., dostupno na: https://expertsystem.com/machine-learning-definition/, [7. srpnja 2020.]

[2] Duolingo: Learn Languages Free, Google Play, 2020., dostupno na: https://play.google.com/store/apps/details?hl=en&id=com.duolingo&referrer=utm_source%3Dduolingo.com%26utm_medium%3Dduolingo_web%26utm_content%3Ddownload_button%26utm_campaign%3Dsplash, [27. lipnja 2020.]

[3] Android(operating system), Wikipedia, 2020., dostupno na: https://en.wikipedia.org/wiki/Android_(operating_system), [28. lipnja 2020.]

[4] Android, Android, dostupno na: https://www.android.com/, [28. lipnja 2020.]

[5] Meet Android Studio, Android, 2020., dostupno na: https://developer.android.com/studio/intro, [28. lipnja 2020.]

[6] Java (programski jezik), Wikipedia, 2020., dostupno na: https://hr.wikipedia.org/wiki/Java_(programski_jezik), [28. lipnja 2020.]

[7] TensorFlow, TensorFlow, dostupno na: https://www.tensorflow.org/, [28. lipnja 2020.]

[8] What is TensorFlow?, Guru99, dostupno na: https://www.guru99.com/what-is-tensorflow.html, [28. lipnja 2020.]

[9] FirebaseML Machine Learning, Firebase, dostupno na: https://firebase.google.com/products/ml/, [18. srpnja 2020.]

[10] D. Stevenson, What is Firebase? The complete story, abridged., Medium, 25. rujna 2018., dostupno na: https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0, [18. srpnja 2020.]

[11] J. Cao, What Is a Mockup: The Final Layer of UI Design, UXPin, 22. travnja 2020., dostupno na: https://www.uxpin.com/studio/blog/what-is-a-mockup-the-final-layer-of-ui-design/, [20. srpnja 2020.]

[12] AutoML Vision Edge, Firebase, 1. srpnja 2020., dostupno na: https://firebase.google.com/docs/ml/automl-image-labeling, [23. srpnja 2020.]

[13] AutoML Vision Beginner's Guide, Google Cloud, 22. lipnja 2020., dostupno na: https://cloud.google.com/vision/automl/docs/beginners-guide, [3. kolovoza 2020.]

SAŽETAK

Cilj ovog završnog rada je izrada mobilne aplikacije za Android telefone koja će pomagati studentima učenje tehničkog engleskog vokabulara. Aplikacija treba koristiti model strojnog učenja zato što joj je jedna od funkcionalnosti prepoznavanje objekata na slikama. U radu su opisane postojeće aplikacije koje služe za učenje stranih jezika i tehnologije potrebne za izradu Android aplikacije i treniranje modela. Aplikacija je napisana u Android Studio razvojnom okruženju na Java programskom jeziku i koristi TensorFlow Lite biblioteku za model strojnog učenja. Opisan je način na koji je model istreniran i implementiran u aplikaciju. Osim detekcije objekata, objašnjene su i druge funkcionalnosti potrebne za aplikaciju kao i detalji o njihovoj programskoj implementaciji. Aplikacija koristi kvizove kako bi korisnici mogli vježbati nove riječi iz tehničkog vokabulara. Aplikacija također ima i funkcionalnost rječnika kako bi korisnici mogli pretraživati naučene pojmove. Na kraju rada dane su upute za korištenje aplikacije i njen izgled, a analizirana je i točnost klasifikacije objekata kojom je prikazana preciznost i pouzdanost aplikacije pri detekciji objekata.

Ključne riječi: Android aplikacija, kviz, strani jezik, strojno učenje, TensorFlow

ABSTRACT

Title: Mobile application development for learning technical English vocabulary using object recognition

The goal of this paper is to develop a mobile application for Android phones that will help students learn technical English vocabulary. The application needs to use a machine learning model because one of its functionalities is object recognition in pictures. A few existing applications for foreign language learning as well as technologies needed for application development and model training are described in this paper. The application is written in Android Studio development environment using Java programming language. It uses TensorFlow Lite library for machine learning. The paper also describes how the model was trained and how it was implemented in the application. Besides object detection, other needed functionalities as well as details about their programming implementation are explained. The application uses quizzes so users can practice new technical vocabulary. Also, the application uses a dictionary functionality to let users search for the acquired terms. At the end of paper, user instructions and application design are given. Finally, the accuracy of object classification was analyzed to show the precision and reliability of the application when using object recognition.

Keywords: Android application, quiz, foreign language, machine learning, TensorFlow

PRILOZI

Projektna mapa s izvornim kodom nalazi se na priloženom optičkom disku.