SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 465

AUTOMATSKA PROVJERA DIKTATA KORIŠTENJEM GOOGLE ML KITA

Marta Dulibić

Zagreb, lipanj 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 465

AUTOMATSKA PROVJERA DIKTATA KORIŠTENJEM GOOGLE ML KITA

Marta Dulibić

Zagreb, lipanj 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 11. ožujka 2022.

ZAVRŠNI ZADATAK br. 465

Pristupnica:	**Marta Dulibić (0036525495)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	izv. prof. dr. sc. Boris Milašinović

Zadatak: **Automatska provjera diktata korištenjem Google ML Kita**

Opis zadatka:

Diktat kao transkripcija izgovorenog teksta čest je način provjere pravopisa u školama te se upotrebljava za provjeru pravopisa hrvatskog jezika, ali i stranih jezika koje su učenici upisali. Svrha ovog rada je izraditi mobilnu aplikaciju za platformu Android koja će omogućiti samostalno vježbanje diktata u različitim jezicima koristeći automatsko prepoznavanje napisanog teksta. Zvučni i tekstualni zapisi tekstova koji će se koristiti za diktate bit će pohranjeni na platformi Firebase. Učenik svoje vježbanje započinje skeniranje koda koji jednoznačno određuje tekst koji se diktira, nakon čega se u mobilnoj aplikaciji reproducira zvučni zapis teksta. Ovisno o postavkama, omogućiti ponavljanje i privremeno zaustavljanje reprodukcije. Učenik tekst može pisati na papiru, pa u aplikaciji mora biti omogućeno slikanje napisanog teksta, ili ga može pisati pomoću olovke na za to predviđenim mobilnim uređajima (tabletima). Prepoznati tekst se mora moći usporediti s diktiranim tekstom te se učeniku trebaju prikazati pogreške. Za skeniranje koda, prepoznavanje teksta iz slika te prepoznavanje digitalno pisanog teksta koristiti alate iz Googleovog razvojnog kompleta ML Kit.

Rok za predaju rada: 10. lipnja 2022.

Sadržaj

Uvod

Od početka obrazovanja kroz osnovnoškolske klupe sve do slušanja kolegija na fakultetu, veliki se naglasak, neovisno o stručnom usmjerenju učenika ili studenta, stavlja na važnost pravopisne pismenosti, poznavanja gramatike te mogućnosti dobrog pismenog izražavanja pojedinca. Pismeno izražavanje je sveprisutno i najčešći alat u ispitivanju znanja u obrazovnom sustavu. Kroz osnovnu i srednju školu najčešće je to u obliku ispita, eseja i diktata, dok se kroz fakultetsko obrazovanje pojavljuje u obliku seminara i znanstvenih radova.

Diktat je vrsta pismene vježbe koja se primjenjuje u svim fazama školovanja (najčešće ipak u osnovnoj školi) te je mnogi metodičari hrvatskog jezika smatraju najboljom vježbom za stjecanje pravopisnih navika i uvježbavanja osnova gramatike. [1] Provođenjem diktata kroz nastavu, učinkovito se može pratiti, provjeravati i ocjenjivati sposobnost učenikova pisanog izražavanja kao i poznavanje pravopisnih i gramatičkih pravila. [2]

Mnogim učenicima, pisanja diktata je vrlo stresna i neugodna situacija zbog straha od nepoznatog teksta te pretjeranog razmišljanja o uspjehu ispravnog zapisivanja diktiranog teksta korištenjem pravopisnih i gramatičkih pravila.

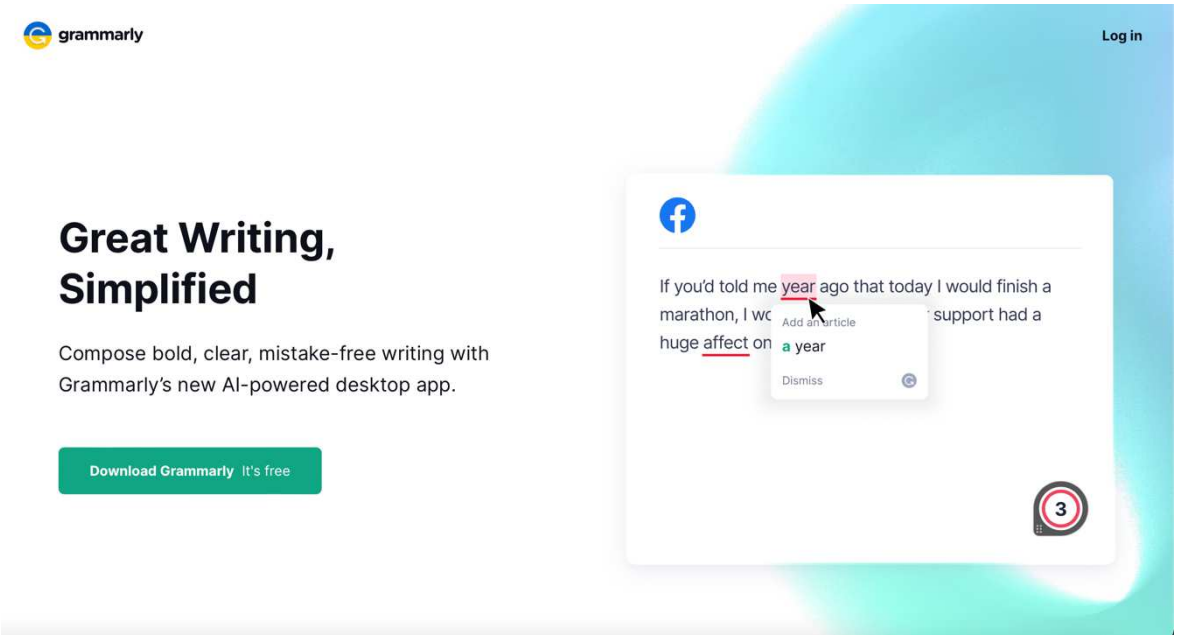
My Spelling Buddy je mobilna aplikacija namijenjena upravo učenicima te svima onima koji žele poboljšati i uvježbati svoje vještine pisanja. Glavni cilj aplikacije je da služi kao pomoćni obrazovni alat pri uvježbavanju pisanja diktata te poboljšavanju pravopisnih i gramatičkih znanja korisnika. Zbog jednostavnog i intuitivnog sučelja, namijenjena je za korištenje svim dobnim skupinama te podržava uvježbavanje i ostalih jezika osim hrvatskog.

Kroz iduća poglavlja, detaljno je objašnjena izrada same aplikacije kao i korištene tehnologije te biblioteke. U prvom poglavlju su predstavljena postojeća programska rješenja koja se bave sličnim problemskim područjem kao i naša aplikacija. Drugo poglavlje sadrži specifikaciju programske potpore u kojem su raspisani korisnički i funkcionalni zahtjevi kao i model dijaloga aplikacije. Nadalje, u trećem poglavlju prikazana je arhitektura rješenja te su navedene i objašnjene korištene tehnologije i biblioteke. Prikaz programskog rješenja predstavljen je u četvrtom poglavlju, dok se u posljednjem poglavlju mogu pronaći upute za korištenje i instalaciju aplikacije.

1. Pregled sličnih rješenja

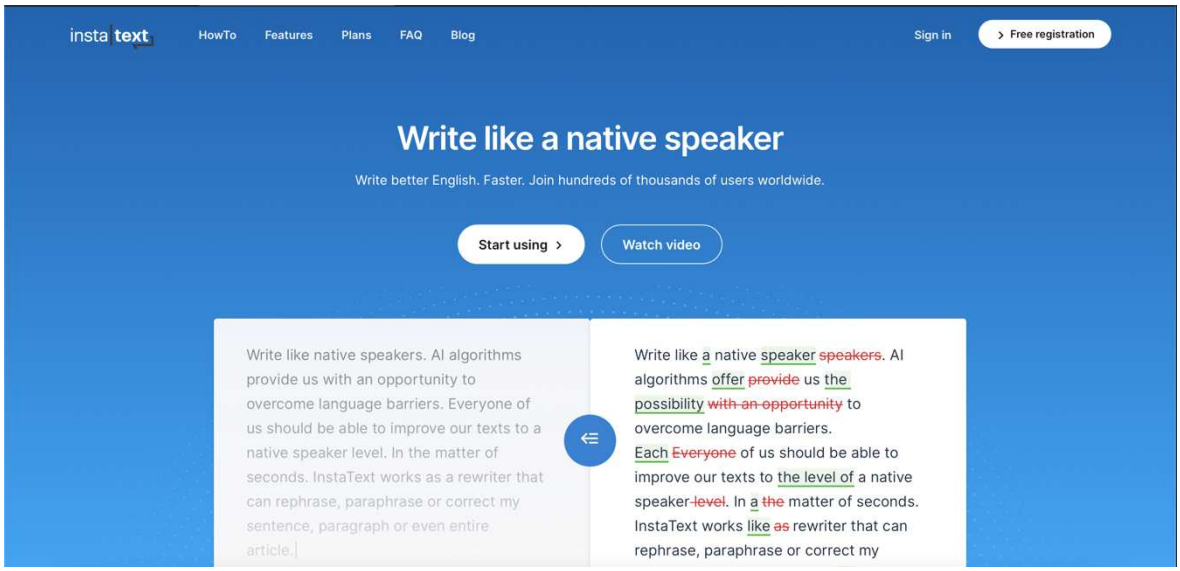
Na tržištu trenutno, već postoje programska rješenja koja pomažu pri pisanju tekstova i poboljšavanju pravopisa i gramatike korisnika.

Jedna od takvih je Grammarly, desktopna aplikacija koja nudi prijedloge za poboljšanje teksta u stvarnom vremenu. Prijedlozi pomažu pri identificiranju i zamjenjivanju kompliciranih rečenica te provjeravanju točnost pravopisa, interpunkcije i gramatike. Za ostvarivanje svih tih funkcionalnosti, korišteni su inovativni pristupi s područja naprednog strojnog učenja i dubokog učenja. Također, postoji mogućnost integracije softvera u korisnikovu tipkovnicu.



Slika 1.2.1. Prikaz web stranice za desktopnu aplikaciju Grammarly (preuzeto s [3])

Uz Grammarly, aplikacija InstaText također služi kao alat za pisanje i uređivanje teksta. InstaText algoritam skenira tekst te poboljšava stil i izbor riječi, ispravlja gramatičke pogreške te dodatno obogaćuje tekst raznim prijedlozima kako bi ga učinio čitljivijim i razumljivijim.



Slika 1.2.2. Prikaz web stranice za desktopnu aplikaciju InstaText (preuzeto s [4])

2. Specifikacija programske potpore

Mobilna aplikacija My Spelling Buddy namijenjena je svima onima koji žele usavršiti svoje vještine pisanja te uz to poboljšati svoj pravopis i gramatiku. U sljedećim potpoglavljima slijedi pojašnjenje korisničkih i funkcionalnih zahtjeva koje aplikacija treba ispuniti.

2.1. Korisnički zahtjevi

Za korištenje aplikacije, korisniku nije potrebna nikakva dodatna prijava ili registracija te su svim korisnicima dostupne sve funkcionalnosti aplikacije. Korisnik može odabrati jezik aplikacije (hrvatski ili engleski).

Glavna funkcionalnost aplikacije je vježbanje diktata uz automatsko prepoznavanje teksta i prikaz pogreški. Aplikacija omogućuje reprodukciju diktata u obliku audiozapisa.

Kako bi se pristupilo vježbi, aplikacija omogućuje skeniranje QR koda koji jedinstveno određuje audiozapis i tekst koji se diktira te sadrži neke dodatne specifikacije kao što su mogućnost ubrzavanja ili usporavanja audiozapisa te privremeno zaustavljanje i ponavljanje reprodukcije.

Korisnik može birati hoće li pisati vježbu na papiru ili mobilnom uređaju (tablet). Ovisno o tome, u aplikaciji mora biti omogućeno slikanje napisanog teksta ili pisanje teksta na ekranu pomoću olovke na za to predviđenim mobilnim uređajima (tabletima). Također, moguća je ponovna predaja teksta prije finalne evaluacije rješenja vježbe.

Aplikacija nudi prikaz usporedbe napisanog i diktiranog teksta te prikaz pogreški.

2.2. Funkcionalni zahtjevi

Za pristup funkcionalnostima aplikacije nije potrebna nikakva vrsta autentifikacije, a sve postavke odabrane kroz aplikaciju su zapamćene sve do deinstalacije aplikacije s uređaja. Omogućen je odabir jezika aplikacije (hrvatski ili engleski) te se on ne treba nužno poklapati s generalnim postavkama jezika samog uređaja.

Omogućeno je skeniranje QR koda s kojeg aplikacija očitava i sprema audiozapis diktata te postavke pristupa vježbi kao što su mogućnost mijenjanja brzine reprodukcije, privremeno zaustavljanje i ponavljanje reprodukcije audiozapisa.

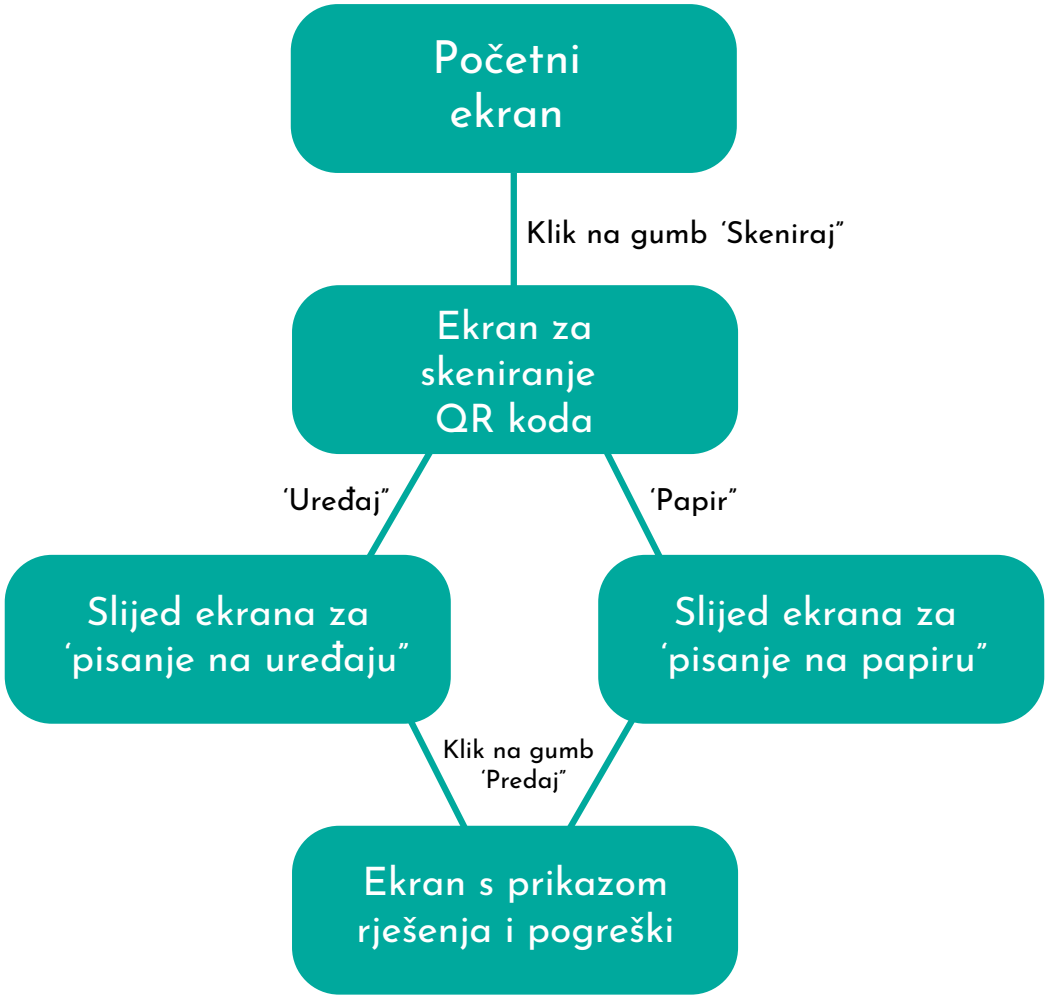
Aplikacija nudi odabir pristupa vježbi diktata pisanjem na papiru ili mobilnom uređaju (tabletu). Ako je odabrano pisanje na papiru, aplikacija nudi mogućnost reprodukcije audiozapisa nakon čega je korisniku omogućeno pohranjivanje slike teksta korištenjem kamere pripadnog uređaja. Ako je odabrano pisanje na uređaju, aplikacija otvara prikaz na kojem je omogućeno digitalno pisanje po zaslonu pomoću olovke kao i reprodukcija audiozapisa. Za razliku od odabira pisanja na papiru, pohranjivanje napisanog teksta se događa automatski nakon predaje rješenja.

Nakon predaja rješenja vježbe, aplikacija nudi prikaz usporedbe napisanog i diktiranog teksta te prikaz pogreški. Također, omogućena je i ponovna predaja teksta nakon prikaza rješenja. Ovisno o tome koji je način pisanja odabran, korisnik će biti vraćen na prikaz za digitalno pisanje (uređaj) ili na pohranu slike teksta (papir).

Odabir načina pisanja moguć je samo na početnom sučelju aplikacije te se ne može promijeniti tijekom pisanja određene vježbe.

3. Model dijaloga ekrana

Pokretanjem aplikacije otvara se početni ekran. Na početnom ekranu se nalazi padajući izbornik za izbor jezika aplikacije te načina pisanja vježbe diktata. Pritiskom na gumb „Skeniraj“ dolazi se na ekran za skeniranje QR koda. Nakon uspješnog skeniranje QR koda, ovisno o odabiru načina pisanja, dolazi se ili do slijeda ekrana vezanih za pisanje na papiru ili mobilnom uređaju (tabletu). Neovisno koji način pisanja odabran, nakon odrade i predaje vježbe, dolazi se na ekran s prikazom usporedbe napisanog i diktiranog teksta te prikazom pogreški. Klikom na gumb „Početna“ ponovno se otvara početni ekran, a klikom na gumb „Ponovna predaja“, ovisno o odabranom načinu pisanja, otvara se jedan od prethodnih ekrana za predaju teksta.



Slika 3.1 Slijedni dijagram podjele aplikacije

3.1. Pisanje na papiru

Odabirom pisanja na papiru, nakon skeniranje QR koda, dolazi se na ekran za reprodukciju audiozapisa. Ovisno o postavkama, korisniku je omogućeno usporavanje/ubrzavanje audiozapisa, zaustavljanje i ponovna reprodukcija audiozapisa. Odabirom strelice za povratak korisnik se vraća na ekran za skeniranje QR koda, a klikom na gumb „Nastavi“ otvara se ekran na kojem je omogućeno pohranjivanje i predaja teksta. Klikom na gumb „Pohrani“ otvara se kamera mobilnog uređaja koja nam omogućava slikanje napisanog teksta. Nakon uspješnog slikanja, slika se pohranjuje u internu memoriju aplikacije. Odabirom gumba „Predaj“, otvara se ekran s prikazom rješenja i pogreški vježbe.

Slika 3.1.1 Slijedni dijagram podjele aplikacije odabirom načina za pisanje „Papir“

3.2. Pisanje na mobilnom uređaju

Odabirom pisanja na mobilnom uređaju (tabletu), nakon skeniranje QR koda, dolazi se na ekran na kojem je omogućena reprodukcija audiozapisa te digitalno pisanje na ekranu pomoću olovke. Ovisno o postavkama, kao i kod pisanja na papiru, korisniku je omogućeno

usporavanje/ubrzavanje audiozapisa, zaustavljanje i ponovna reprodukcija audiozapisa. Odabirom strelice za povratak korisnik se vraća na ekran za skeniranje QR koda, a klikom na gumb „Predaj“ otvara se ekran s prikazom rješenja i pogreški vježbe.

Slika 3.2.1 Slijedni dijagram podjele aplikacije odabirom načina za pisanje „Uređaj“

4. Arhitektura rješenja

U sljedećim potpoglavljima opisana je baza podataka aplikacije, motivacija i razlog upotrebe korištenih tehnologija te je prikazana organizacija i arhitektura same aplikacije.

4.1. Baza podataka

4.1.1. Firebase

Firebase je platforma koja sadrži skup alata za „izgradnju, poboljšanje i razvoj aplikacija“ te oni pokrivaju veliki dio usluga koji bi programeri inače morali sami izgraditi. To uključuje stvari kao što su analitika, autentifikacija, baze podataka, konfiguracija, pohrana datoteka, push poruke, itd.

Sve usluge su smještene u oblaku što znači da nema potrebe za pisanjem middleware servisa između aplikacija i backend servisa, već se upiti na bazu mogu direktno pisati u klijentskoj aplikaciji. [5]

4.1.2. Firebase Firestore i Cloud Storage

Upravo zbog mogućnosti direktnih upita na bazu kroz aplikaciju, za pohranu podataka je korišten Firebase Firestore.

Firebase Firestore služi za pohranu podataka korištenih za rad i razvoj mobilnih uređaja koristeći NoSQL bazu podataka te nudi integraciju i s ostalim Firebase uslugama. Omogućuje realtime sinkronizaciju podataka te nudi rješenje za mogućnost rada i bez prisustva Internet veze. [6]

Struktura Firestore baze nije fiksna te ona podatke sprema u dokumente koji se sastoje od parova ključ-vrijednost. Dokumenti se mogu grupirati u kolekcije, a svaki dokument može sadržavati i sam svoju kolekciju.

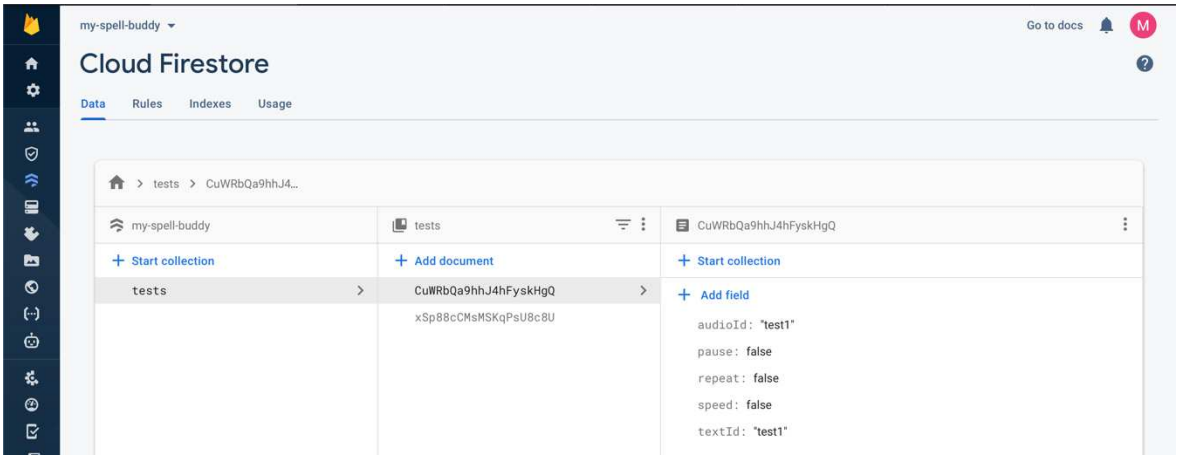
Za pohranu audiozapisa i tekstualnih datoteka koji se koriste za uvježbavanje diktata, korišten je Firebase Cloud Storage.

Cloud Storage je Firebaseova usluga skladištenja objekata koja na jednostavan i isplativ način omogućuje pohranu i skidanje korisnički generiranog sadržaja, kao što su fotografije, video i audiozapisi, itd. [7]

4.1.3. Fizički model podataka

Baza podataka aplikacije sadrži samo jedan entitet koji je u Firestore bazi pohranjen pod kolekciju *tests*. Kolekcija *tests* sadrži dokumente koji sadrže informacije koje će biti pohranjene u QR kodu koji će se isčitavati prilikom pokretanja vježbanja diktata u aplikaciji:

- *audioId*, predstavlja jedinstveni identifikator audiozapisa diktata pohranjenog u Cloud Storageu (tipa *string*)
- *textId*, predstavlja jedinstveni identifikator tekstualne datoteke diktata pohranjenog u Cloud Storageu (tipa *string*)
- *pause*, varijabla koja nam govori je li korisniku omogućeno pauziranje audiozapisa tijekom vježbe diktata (tipa *boolean*)
- *repeat*, varijabla koja nam govori je li korisniku omogućeno ponovno reproduciranje audiozapisa tijekom vježbe diktata (tipa *boolean*)
- *speed*, varijabla koja nam govori je li korisniku omogućeno mijenjanje brzine reprodukcije audiozapisa tijekom vježbe diktata (tipa *boolean*)



Slika 4.1.3.1 Prikaz kolekcije *tests* u bazi podataka

4.2. Korištene tehnologije

Mobilna aplikacija My Spelling Buddy napisana je za operacijski sustav Android te je korišteno razvojno okruženje Android Studio. Za programski jezik odabran je Kotlin, dok

su za skeniranje koda, prepoznavanje teksta iz slika te prepoznavanje digitalno pisanog teksta korišteni alati iz Google razvojnog kompleta ML Kit.

4.2.1. Android Studio

Android Studio je službeno razvojno okruženje (IDE) za Android operativni sustav. Najbolji je odabir za razvijanje nativnih Android aplikacija radi inteligentnog uređivača koda, brzog emulatora, mogućnosti korištenja aplikacijskih predložaka, itd. [8]

Za razvoj ove aplikacije, odabran je upravo radi jednostavnosti i brzine korištenja te postojeće integracije s Firebase klijentom.

4.2.2. Kotlin

Kotlin je višeplatformski, statički tipiziran, programski jezik opće namjene koji se koristi u JVM-u (Java Virtual Machine) te se temelji na Java platformi i okruženju. Danas je on prvi izbor za razvoj aplikacija za operacijski sustav Android te je preporučen za korištenje i od strane Google-a. [9]

Zbog sažetosti koda, lake čitljivosti, pružanja nulte sigurnosti te odlične integracije s Android Studijom odabran je kao programski jezik za izgradnju ove aplikacije.

4.2.3. Google ML Kit

Google ML Kit je mobilni SDK koji donosi rješenja za razne izazove kojim se susrećemo pri izgradnji aplikacija korištenjem strojnog i dubokog učenja kroz jednostavne API-je za vid i prirodni jezik. [10]

Za potrebe skeniranje koda, prepoznavanje teksta iz slika te prepoznavanje digitalno pisanog teksta iskorišten je i implementiran Vision API od ML Kit-a. U dodatku rada je detaljno objašnjena implementacija i funkcionalnost svakog algoritma prepoznavanja koji je korišten pri izradi aplikacije.

4.3. Organizacija i arhitektura aplikacije

4.3.1. Arhitektura aplikacije

Za arhitekturu aplikacije odabran je arhitekturni obrazac MVVM (Model-View-View Model) radi olakšavanje implementacije i čišćeg koda. Osnovne karakteristike obrasca su da:

- Pogled je pretplaćen na podatke modela te osluškuje model ako dođe do promjene u podacima. Prilikom promjena, ažurira se korisničko sučelje (pogled).
- Model pogleda osluškuje vijesti o promjenama modela te emitira nove informacije na koje se pogled pretplaćuje te nema direktnu referencu na pogled.
- Model je odgovoran apstrakciju izvora podataka. [11]

4.3.2. Organizacija aplikacije

Projekt android aplikacije se sastoji od različitih vrsta modula aplikacije, datoteka izvornog koda i datoteka resursa. U rootu projekta nalazimo `root` i `Gradle scripts` folder u kojem se nalaze datoteke za build konfiguraciju aplikacije.

`Root` folder sadrži foldere:

- `Manifests`: U njemu nalazimo datoteku AndroidManifest.xml koja služi za kreiranje android aplikacije te djeluje kao posrednik između android OS-a i aplikacije. U njoj nalazimo informacije kao što su korištena verzija androida, metapodaci, itd. Tu su dodane i potrebne dozvole za korištenje kamere uređaja, korištenje vanjskog spremišta i spajanje na Internet.
- `Java`*:* Sadrži sve datoteke Kotlin izvornog koda koje smo podijelili na više direktorija i poddirektorija ovisno o namjeni:
    1. *anaylzer:* Sadrži datoteke s implementiranim algoritmima za skeniranje QR koda (`QRCodeAnalyzer.kt`), prepoznavanje teksta iz slika (`TextAnalyzer.kt`) te prepoznavanje digitalno pisanog teksta (`DigitalInkAnalyzer.kt`). Svaki od algoritama će biti objašnjen detaljnije u *Dodatku* rada.

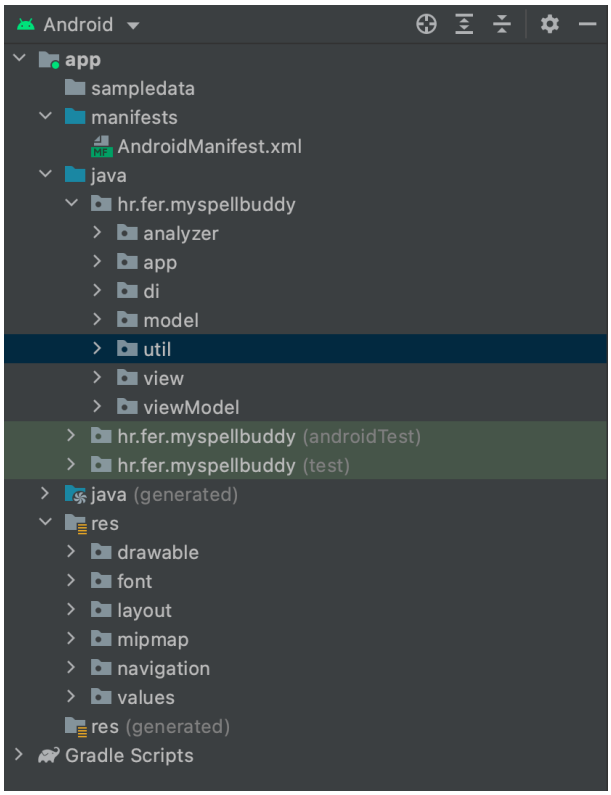
2. `app`: Sadrži datoteku *MySpellingBuddyApp.kt* koja predstavlja ulaznu točku u aplikaciju.
3. `di`*:* Sadrži datoteku u kojoj smo za instanciranje klasa za parsiranja JSON-a koristili *princip inverzije kontrole*. Za injekciju ovisnosti korištena je biblioteka *Hilt*.
4. `util`*:* Sadrži pomoćne datoteke kao što su pomoćne ekstenzije, adapteri, audio player, itd. Jedna od datoteka je `SessionManager.kt` koja služi kao lokalno spremište podataka. Podaci ostaju spremljeni i ponovnim pokretanjem aplikacije te se gube tek deinstalacijom same aplikacije. Korištena je za spremanje odabranog jezika aplikacije i načina pisanja diktata.

```
class SessionManager @Inject constructor(@ApplicationContext private val appContext: Context) {

    companion object {
        private const val PREFERENCES_APP = "prefs_app"
        private const val PLAYBACK = "playback"
        private const val WRITING_METHOD = "writing_method"
        private const val LANGUAGE = "language"
        private const val PAUSE_AUDIO = "pause_audio"
        private const val BARCODE_VALUE = "barcode_value"
    }

    private val appPreferences: SharedPreferences by lazy {
        appContext.getSharedPreferences(PREFERENCES_APP, Context.MODE_PRIVATE)
    }

    var language: String?
        get() = appPreferences.getString(LANGUAGE, "hr")
        set(value) {
            appPreferences.edit().putString(LANGUAGE, value).apply()
        }
```

Slika 4.3.2.1 Isječak koda lokalnog spremišta SessionManager

5. `model`
6. `view`
7. `viewModel`

- `res`*:* Sadrži sve resurse aplikacije kao što su slike, ikone, XML datoteke za layout pogleda, UI stringove, itd.



Slika 4.3.2.2 Organizacija aplikacije po direktorijima

4.3.3. Model

Direktorij *model* sadrži dva razreda: `ExerciseEntry` i `ResultPair`.

`ExcersiceEntry` predstavlja model podataka koji je sadržan unutar QR koda koji se skenira prije početka svake vježbe. Varijable `soundId` i `textId` čine jedinstvene identifikatore preko kojih dohvaćamo audio i tekstualne zapise iz baze podataka (Firebase Storage-a). Ostale varijable su tipa `Boolean` te predstavljaju dodatne postavke koje će biti konfigurirane korisniku tijekom izvođenja vježbe:

- `pause` (`true` ako je omogućeno pauziranje audiozapisa tijekom vježbe, inače `false`)
- `repeat` (`true` ako je omogućeno ponavljanje reproduciranja audiozapisa tijekom vježbe, inače `false`)
- `speed` (`true` ako je omogućeno mijenjanje brzine reprodukcije audiozapisa tijekom vježbe, inače `false`)

Slika 4.3.3.1 Dohvaćenje URL-a audiozapisa iz Cloud Storage-a preko varijable soundId

`ResultPair` sadrži par vrijednosti `userInput` i `solution` te služi za pohranjivanje napravljenih pogrešaka korisnika u diktatu. Varijabla `userInput` predstavlja korisnikov zapis riječi, dok `solution` predstavlja gramatički i pravopisni točno napisanu riječ iz diktata.

4.3.4. View

Direktorij `View` se dijeli dodatno na nove poddirektorije te on sadrži implementacije svih ekrana aplikacije i dodatne pomoćne klase za njihovo stvaranje:

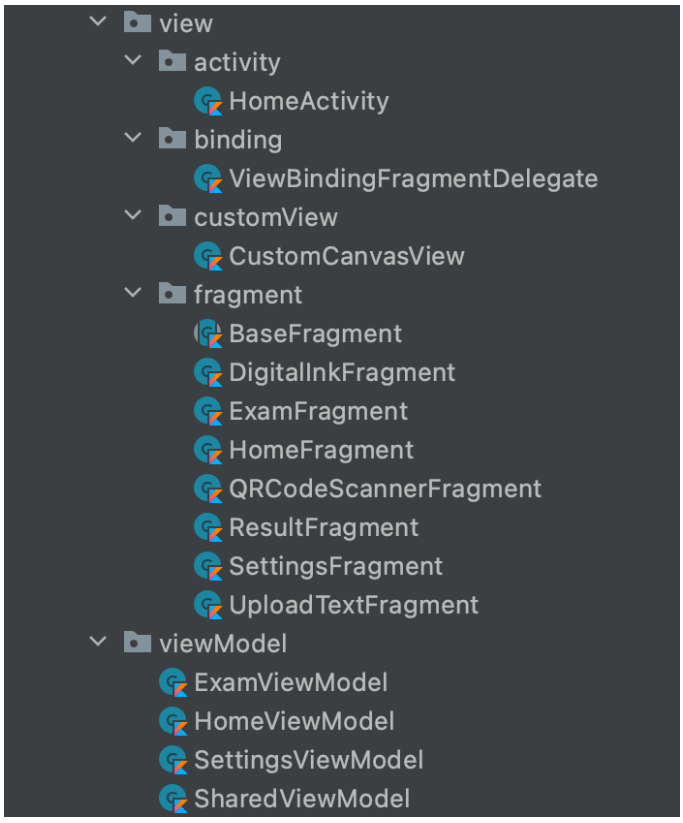
- `activity`
- `binding`
- `customView`
- `fragment`

`HomeFragment` predstavlja početni ekran i sadrži izbornike za mijenjanje jezika aplikacije i biranja načina pisanja diktata te gumb `Skeniraj` s kojim se prelazi na ekran `QRCodeScannerFragment` na kojem se otvara kamera za skeniranje i detektiranje

QR koda. Nakon uspješnog skeniranje QR koda, ovisno o odabranom načinu pisanja diktata, otvara nam se:

- ako je odabran način pisanja *Papir*: `ExamFragment` koji sadrži upute za pisanje vježbe, gumbove za reprodukciju audiozapisa te gumb `Predaj`. Odabirom gumba `Nastavi` otvara se ekran `UploadTextFragment` koji sadrži gumb `Pohrani` koji otvara kameru za slikavanje rješenja korisnika te gumb `Predaj` koji nas vodi na ekran s prikazom rješenja i pogreški vježbe.
- ako je odabran način pisanja *Uređaj*: `DigitalInkFragment` koji sadrži upute za pisanje vježbe, gumbove za reprodukciju audiozapisa, prostor namijenjen za digitalno pisnanje vježbe olovkom te gumb `Predaj`. Odabirom gumba `Predaj` otvara ekran s prikazom rješenja i pogreški vježbe.

`ResultFragment` je ekran koji nudi prikaz usporedbe napisanog i diktiranog teksta te prikaz pogreški. Sadrži gumb `Ponovna predaja` koji omogućava ponovnu predaju teksta nakon prikaza rješenja. Ovisno o tome koji je način pisanja odabran, korisnik će biti vraćen na `DigitalInkFragment` (uređaj) ili `UploadTextFragment` (papir).

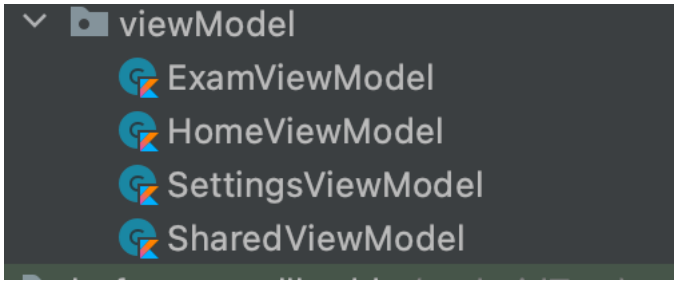


Slika 4.3.4.1 Sadržaj i organizacija direktorija View

4.3.5. ViewModel

Direktorij `ViewModel` sadrži klase s poslovnom logikom koji osluškuju promjene sadržaja modela te o tome obavještavaju ekran koji je pretplaćen na te promjene. Sadrži 4 klase:

- `ExamViewModel.kt` (delegira informacije o mogućim funkcionalnostima audio playera koje smo dobili očitavajući QR kod)
- `HomeViewModel.kt` (dohvaća trenutno odabrani jezik aplikacije)
- `SettingsViewModel.kt` (dohvaća iz `SessionManager`-a trenutno aktivne postavke aplikacije kao što su način pisanja diktata te omogućene funkcionalnosti audio playera)
- `SharedViewModel.kt` (služi za delegiranje informacija o navigaciji među ekranima)



Slika 4.3.5.1 Sadržaj i organizacija direktorija ViewModel

5. Korisničke upute

Mobilna aplikacija My Spelling Buddy omogućava samostalno vježbanje diktata u različitim jezicima koristeći automatsko prepoznavanje teksta. Kroz aplikaciju korisnik može raditi na svojim vještinama pisanja kao i na poboljšanju svog pravopisa i gramatike.

Kroz sljedeća potpoglavlja dane su jednostavne upute za korištenje.

5.1. Početni ekran i izbor načina pisanja

Nakon pokretanja aplikacije, otvara se početni ekran (Slika 5.1.1) koji sadrži dva izbornika: izbornik za biranje jezika aplikacije (Slika 5.1.2) i izbornik za biranje načina pisanja diktata (Slika 5.1.3).

Izbornik za jezik nam nudi odabir između hrvatskog i engleskog jezika, dok u izborniku za biranje načina pisanja biramo između pisanja na papiru i na mobilnom uređaju (tabletu). Jezik i način pisanja se mogu jedino promijeniti na početnom ekranu te se ne mogu promijeniti tijekom trajanja vježbe diktata. Klikom na gumb „Skeniraj“ korisniku se otvara ekran za skeniranje QR koda.

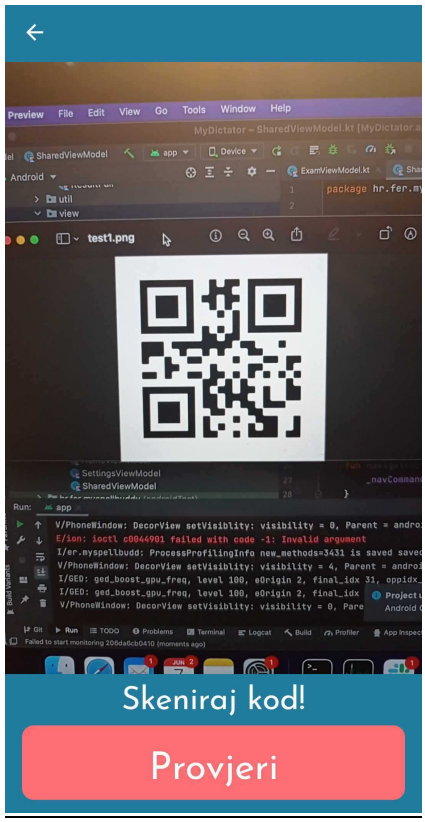
Slika 5.1.1 Početni ekran - izbornik za odabir načina pisanja

Slika 5.1.2 Početni ekran - izbornik za odabir jezika

5.2. Skeniranje QR koda

Kako bi se pristupilo vježbi diktata, korisnik mora skenirati validni QR kod izgeneriran u svrhu korištenja aplikacije (Slika 5.2.1). Ako korisnik prvi put pokreće aplikaciju, na ekranu će se pojaviti dijalog za odobrenje korištenja kamere uređaja (Slika 5.2.2).

Korisnik mora odobriti pristup kameri kako bi nastavio s korištenjem aplikacije. Nakon skeniranja QR koda, korisniku se otvara ekran ovisno koji je način pisanja diktata odabrao na početnom ekranu.



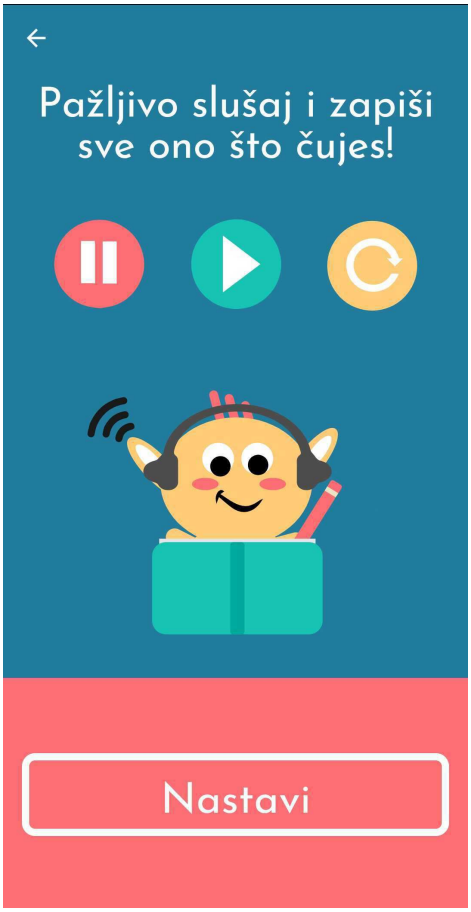
Slika 5.2.1 Ekran za skeniranje QR koda – pokušaj skeniranja

Slika 5.2.2 Ekran za skeniranje QR koda – dozvola za korištenje kamere

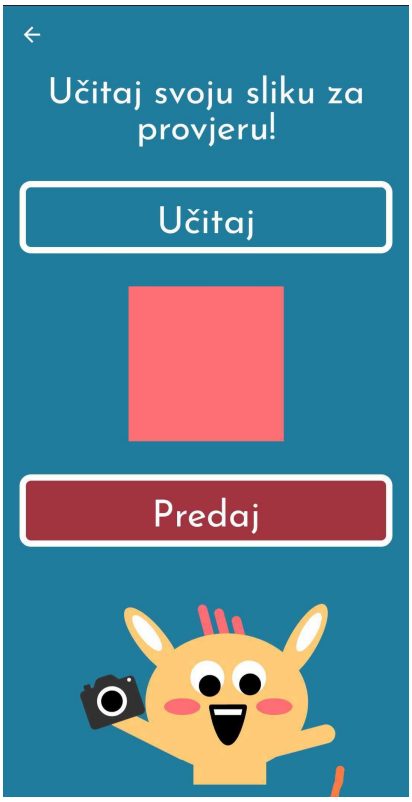
5.3. Pisanje teksta na papiru

Nakon skeniranja QR koda, ako je odabran način pisanja teksta na papiru, otvara se ekran za reprodukcije audiozapisa kojeg korisnik treba zapisivati na papir kao vježbu (Slika 5.3.1). Tijekom reprodukcije audiozapisa, ovisno o postavkama skeniranih s QR koda, korisnik može pauzirati, ponovno reproducirati i mijenjati brzinu reprodukcije audiozapisa.

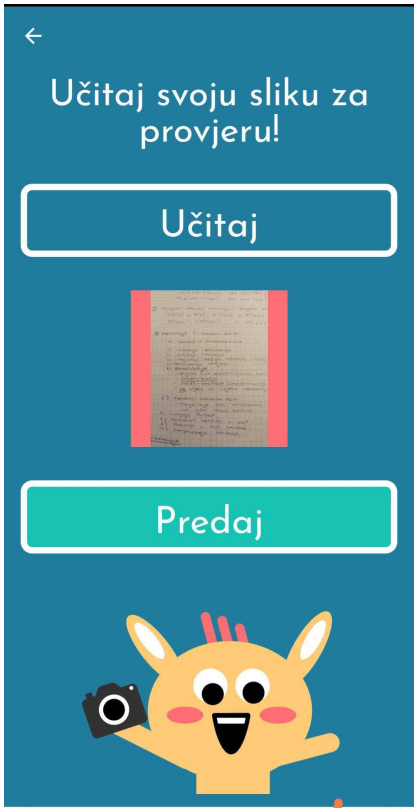
Klikom na gumb „Nastavi“, otvara se ekran za pohranjivanje rješenja vježbe. (Slika 5.3.2). Korisnik ne može ići dalje na sljedeći ekran prije pohrane rješenja (gumb Predaj je onemogućen i naznačen crvenoj bojom). Da bi pohranio rješenje svoje vježbe, korisnik mora kamerom uslikati tekst i to čini odabirom gumba „Učitaj“ s čime se otvara ekran s prikazom rješenja i greški. Nakon uspješnog učitavanja, gumb „Predaj“ postaje zelene boje i korisnik može predati vježbu na evaluaciju (Slika 5.3.3).



Slika 5.3.1 Ekran za pisanje teksta na papiru



Slika 5.3.2 Ekran za pohranjivanje rezultata – Onemogućena predaja



Slika 5.3.3 Ekran za pohranjivanje rezultata – Omogućena predaja

5.4. Pisanje teksta na uređaju (tabletu)

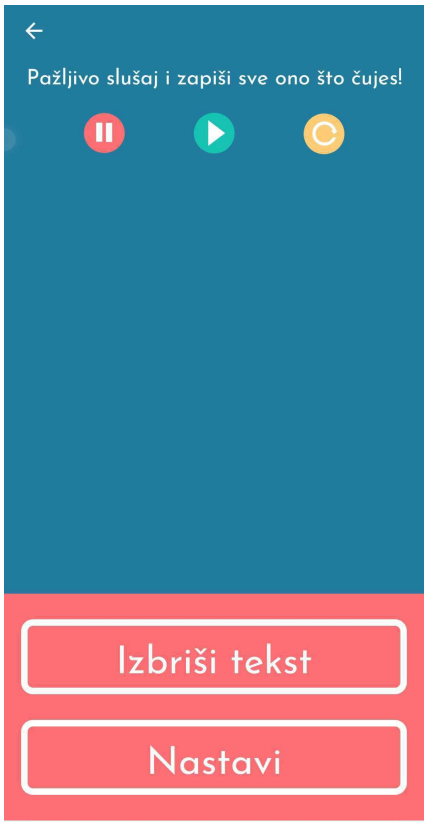
Nakon skeniranja QR koda, ako je odabran način pisanja teksta na mobilnom uređaju (tabletu), otvara se ekran koji na vrhu sadrži audio player za reprodukciju audiozapisa za vježbu diktata, dok se ispod njega nalazi prostor namijenjen za digitalno pisanje olovkom (Slika 5.4.1).

Tijekom reprodukcije audiozapisa, ovisno o postavkama skeniranih s QR koda, korisnik može pauzirati, ponovno reproducirati i mijenjati brzinu reprodukcije audiozapisa isto kao i kod pisanja na papiru.

Klikom na gumb „Izbriši“ tekst, prostor za pisanje će se očistiti te korisnik može ponoviti pisanja teksta. (Slika 5.4.2)

Klikom na gumb „Nastavi“, dolazi do automatskog očitavanja teksta s ekrana te se on učitava kao rješenje vježbe te ostvara ekran s prikazom rješenja i greški.

Slika 5.4.1 Ekran za pisanje teksta na uređaju – Primjer izgleda napisanog teksta



Slika 5.4.2 Ekran za pisanje teksta na uređaju – Primjer praznog ekrana za pisanje

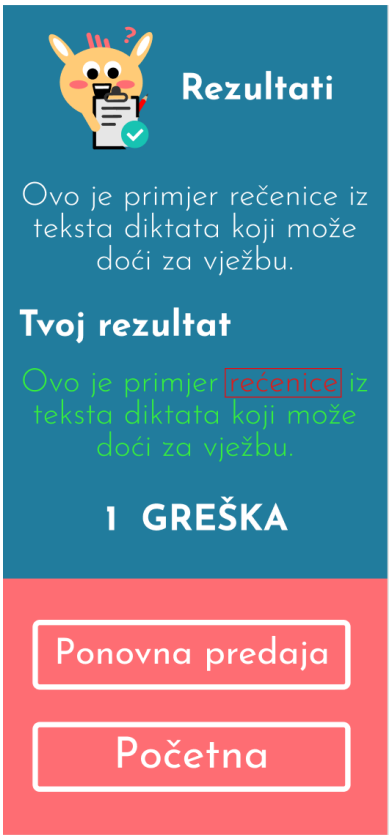
5.5. Prikaz rješenja vježbe i greški

Odabirom gumba predaj s prethodnih ekrana, otvara se ekran s prikazom usporedbe napisanog i diktiranog teksta te prikaz pogreški. (Slika 5.5.1).

Pod „Tvoj rezultat“, nalazi se ispravljeno korisnikovo rješenje. Zelenom bojom su naznačene točno napisane riječi, dok su crvenom istaknute gramatičke i pravopisne pogreške. Ako korisnik nije napravio ni jednu pogrešku, ispisat će se na ekranu odgovarajuća poruka. (Slika 5.5.2).

Korisnik može ponoviti predaju rješenja odabirom gumba „Ponovna predaja“. Ovisno o tome koji je način pisanja odabran, korisnik će biti vraćen ili na ekran Slika 5.3.2 (pisanje na papiru) ili na ekran Slika 5.4.2 (pisanje na uređaju).

Odabirom gumba „Početna“, korisnik se vraća na početni ekran (Slika 5.1.1 i 5.1.2).



Slika 5.5.1 Ekran za prikaz rješenja i grešaka– Primjer rezultata vježbe s greškom

Slika 5.5.2 Ekran za prikaz rješenja i grešaka– Primjer rezultata vježbe bez greške

6. Upute za instalaciju aplikacije

Izvorni kod aplikacije može se pronaći na sljedećoj poveznici: https://github.com/mdulibic/MySpellBuddy. Kako bi se nakon skidanja, kod mogao koristiti potrebno je također instalirati razvojno okruženje Android Studio te konfigurirati i napuniti podatcima Firebasu bazu podataka.

Za konfiguriranje baze, prvo moramo kreirati Firebase projekt klikom na „Add project“ kroz Firebase konzolu na sljedećoj poveznici: https://console.firebase.google.com/. Nakon izrade projekta, potrebno je registrirati aplikaciju s kreiranim Firebase projektom. Za to ponovno otvaramo Firebase konzolu te odabiremo gumb sa Andoroid ikonom nakon čega ćemo morati upisati ime android paketa aplikacije koji možete pronaći unutar datoteke *AndroidManifest.xml* te postaviti neki proizvoljni nadimak za aplikaciju.

Sljedeći korak je skidanje konfiguracijske datoteke *google-service.json* koji trebamo dodati u *root* direktorija *app* unutar našeg projekta. Za omogućavanje rada Firebase usluga, potrebno je u *build.gradle* fileove projekta dodati određene dependencies:

- *Pod build.gradle* (*project*-level):

```
buildscript {

  repositories {
    // Check that you have the following line (if not, add
it):
    google()  // Google's Maven repository
  }

  dependencies {
    // ...

    // Add the following line:
    classpath 'com.google.gms:google-services:4.3.10'  //
Google Services plugin
  }
}

allprojects {
  // ...
```

```
  repositories {
    // Check that you have the following line (if not, add
it):
    google()  // Google's Maven repository
    // ...
  }
}
```

- *Pod build.gradle* (*module*-level):

```
apply plugin: 'com.android.application'
// Add the following line:
apply plugin: 'com.google.gms.google-services'  // Google
Services plugin

android {
  // ...
}
```

Detaljnije upute za dodavanje Firebase projekta aplikaciji, mogu se pronaći na sljedećoj poveznici: https://firebase.google.com/docs/android/setup.

Kako bi isprobali funkcionalnosti aplikacije, potrebno je i u Cloud Firestore bazu dodati kolekcije navedene u poglavlju 4.1.3 te ispuniti ih s testnim podacima. Na temelju tih podataka. potrebno je pomoću dodatnog alata (npr. preko web stranice https://goqr.me), izgenerirati QR kod koji se zatim skenira pri korištenju aplikacije.

7. Zaključak

U sklopa rada izrađena je mobilna aplikacija My Spelling Buddy koja omogućava samostalno vježbanje diktata u različitim jezicima koristeći automatsko prepoznavanje teksta. Kroz aplikaciju korisnik može raditi na svojim vještinama pisanja kao i na poboljšanju svog pravopisa i gramatike.

Aplikacija nam nudi dva načina pisanja diktata: na papiru ili mobilnom uređaju (tabletu) te je lokalizirana za hrvatski i engleski jezik. Kako bi se pristupilo vježbi, korisnik skenira prethodno izgenerirani QR kod koji sadrži jedinstvenu referencu na tekstualni i audio zapis diktata te također neke druge specifikacije kao što su mogućnost pauziranja i ponovne reprodukcije audiozapisa kao i mijenjanje brzine reprodukcije. Nakon odrađene vježbe, korisnik predaje tekst vježbe ili slikavanjem teksta ili automatskim očitavanjem teksta s ekrana ovisno koji je način pisanja odabrao. Nakon predaje teksta, korisniku se otvara pregled usporedbe napisanog i diktiranog teksta te prikaz napravljenih pogreški.

Mogući nedostatak ovog programskog rješenja je nemogućnost dodavanja vlastitih zapisa za vježbanje diktata. U budućim iteracijama aplikacije, mogla bi se implementirati funkcionalnost dodavanja vlastitog teksta i audiozapisa za vježbanje diktata bez potrebe za skeniranjem vanjskog QR koda. Također, mogla bi se dodati mogućnost izvoza rezultata vježbe u .PDF dokument koji bi se zatim mogao skinuti te spremiti na uređaj ili podijeliti s ostalim korisnicima.

Iako postoje slična programska rješenja koja pomažu pri pisanju tekstova i poboljšavanju pravopisa i gramatike korisnika, prednost aplikacije My Spelling Buddy je da donosi novitet mogućnosti pisanja unaprijed određenog teksta koji se zatim automatski obrađuje te ispisuje korisniku napravljene pogreške prilikom pisanja u usporedbi s originalnim tekstom. Takav način obrade teksta je najsličniji pravom iskustvu pisanja diktata u školama te nije implementiran u postojećim programskim rješenjima na tržištu.

Literatura

[1] Pernar, M. *Svrha diktata i učestalost primjene u nastavi Hrvatskog jezika u primarnom obrazovanju*. Diplomski rad. Sveučilište u Zagrebu Učiteljski fakultet, 2016.

[2] Miletić, J. *Slobodni diktat u nastavu hrvatskoga jezika u srednjim strukovnim školama*. Diplomski rad. Sveučilište u Zadru Odjel za kroatistiku, 2016.

[3] Grammarly, Poveznica: https://www.grammarly.com; pristupljeno 4.lipnja 2022.

[4] InstaText, Poveznica: https://instatext.io, pristupljeno: 4.lipnja 2022.

[5] Stevenson D., What is Firebase? The complete story, abridged, Medium (2018, rujan). Poveznica: https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0, pristupljeno: 4.lipnja 2022.

[6] Firebase, Cloud Firestore, Poveznica: https://firebase.google.com/docs/firestore, pristupljeno: 4.lipnja 2022.

[7] Firebase, Cloud Storage , Poveznica: https://firebase.google.com/docs/storage, pristupljeno: 4.lipnja 2022.

[8] Agicent App Development Company., Why Android Studio is Awesome?, Medium (2019, kolovoz). Poveznica: https://medium.com/@agicent/why-android-studio-is-awesome-c606c94366e6, pristupljeno: 5.lipnja 2022.

[9] Kotlin, Kotlin Documentation, Poveznica: https://kotlinlang.org, pristupljeno: 5.lipnja 2022.

[10] Google ML Kit, ML kit Documentation, Poveznica: https://developers.google.com/ml-kit, pristupljeno: 5.lipnja 2022.

[11] Zagoršćak, M. *Usporedba MVVM i MVP arhitektura pri izradi Android aplikacija na primjeru vođenja skladišta*. Završni rad. Sveučilište Josipa Jurja Strossmayera u Osijeku FERIT, 2020.

Dodatak

QRCodeAnalyzer

Za detekciju i obradu QR koda, korišten je *Barcode Scanning* API podržan kao dio mobilnog SDK-a Google ML kit (službena dokumentacija: https://developers.google.com/ml-kit/vision/barcode-scanning).

Za potrebe procesiranja svi okvira dobivenih od kamere tijekom detektiranje QR koda, stvorena je klasa `QrCodeAnalyzer.kt` koja nasljeđuje klasu `ImageAnalysis.Analyzer` u kojoj smo nadjačali metodou `analyze`.

Metoda `analyze` će primati svaki snimljeni okvir kao `ImageProxy` objekt iz kojeg ćemo zatim kreirati `InputImage` objekt koji dalje koristimo za procesiranje:

```
val inputImage = InputImage.fromMediaImage(img,
image.imageInfo.rotationDegrees)
```

Zatim instanciramo objekt tipa `BarcodeScanner` koji će procesirati instancirani `inputImage` objekt:

```
// Process image searching for barcodes
        val options = BarcodeScannerOptions.Builder()
            .setBarcodeFormats(Barcode.FORMAT_QR_CODE)
            .build()

        val scanner = BarcodeScanning.getClient(options)
```

Nakon njegove inicijalizacije, koristimo ga za procesiranje dobivenih barkodova.
U slučaju uspješnog parsiranja, dobivamo listu objekata tipa Barcode iz kojih iščitivamo rawValue vrijednost koja predstavlja vrijednost pohranjenu u QR kodu:

```
scanner.process(inputImage)
                .addOnSuccessListener { barcodes ->
                    for (barcode in barcodes) {
                        rawValue = barcode.rawValue
                    }
                }
```

Tu vrijednost dalje koristimo u aplikaciji za dohvaćanje audio i tekstualnog zapisa iz baze podataka te postavljanje dodatnih mogućnosti aplikacije.

U slučaju neuspjeha ispisujemo dobivenu iznimku, a nakon završetka skeniranja pozivamo `image.close()` kako bi oslobodili red analiziranja i nastavili procesirati ostale okvire:

```
            .addOnFailureListener {
                                Toast.makeText(
                                    context,
                                    "An exception occurred: $it",
                                    Toast.LENGTH_SHORT
                                )
                                    .show()
                            }
                            .addOnCompleteListener {
                                image.close()
                            }
```

TextAnalyzer

Za prepoznavanje teksta iz slika, korišten je *Text Recognition* API podržan kao dio mobilnog SDK-a Google ML kit (službena dokumentacija: https://developers.google.com/ml-kit/vision/text-recognition/android).

Za detektiranje teksta, kreiran je razred `TextAnalyzer` koji zatim izlaže svoju metodu `processImage` kako bi bilo moguće slanje okvira slike (u našem slučaju objekata tipa `Bitmap`) uhvaćenih kamerom koji se zatim procesiraju unutar metode. Za procesiranje slika instancirali smo objekt tipa `FirebaseVisionTextDetector` koji sadrži metodu `detectInImage` koja prima objekt tipa `InputImage`:

```
private val recognizer =
TextRecognition.getClient(TextRecognizerOptions.DEFAULT_OPTIONS)
val result = recognizer.process(image)
```

Za spremanje detektiranje riječi, instancirali smo listu tipa `String` te nju poslije završetka detekcije šaljemo dalje na obradu u aplikaciji.
U slučaju uspješnog parsiranja, dobivamo objekt tipa `Text` iz kojeg prvo uzimamo vrijednost `text` za iščitavanje blokova teksta. Iz blokova teksta dobivamo pojedine linije iz kojih naposljetku uzimamo riječi i spremamo ih u svoju listu riječi:

```
            val wordsList = arrayListOf<String>()
```

```
.addOnSuccessListener { result ->
                val resultText = result.text
                for (block in result.textBlocks) {
                    for (line in block.lines) {
                        for (element in line.elements) {

wordsList.add(element.text.trim())
                        }
                    }
                }
                onSuccess(wordsList)
            }
```

`OnSuccess` je callback funkcija s kojom vraćamo listu riječi na daljnju obradu u aplikaciji. U slučaju neuspjeha ispisujemo dobivenu iznimku:

```
.addOnFailureListener { e ->
                Toast.makeText(
                    context,
                    "An exception occurred: $e!",
                    Toast.LENGTH_SHORT
                )
                    .show()
            }
```

DigitalInkAnalyzer

Za prepoznavanje digitalno pisanog teksta, korišten je *Digital Ink Recognition* API podržan kao dio mobilnog SDK-a Google ML kit (službena dokumentacija: https://developers.google.com/ml-kit/vision/digital-ink-recognition/android) .

Za potrebe prepoznavanja digitalno pisanog teksta, kreiran je razred `DigitalInkAnalyzer` koji sadrži četiri metode: `download`, `addNewTouchEvent`, `recognize` i `clear`.

Unutar metode `download`, preuzimamo model koji ćemo koristiti za prepoznavanje skica korisnika. U slučaju neuspješnog skidanja, ispusuje se odgovarajuća poruka s iznimkom. Nakon skidanja modela, može se koristiti i bez aktivne internetske veze:

```
fun download() {
        var modelIdentifier: DigitalInkRecognitionModelIdentifier? = null
        try {
            modelIdentifier =

DigitalInkRecognitionModelIdentifier.fromLanguageTag("HR")
        } catch (e: MlKitException) {

Timber.d("DigitalInkRecognitionModelIdentifier.fromLanguageTag error:
$e")
        }

        model =
            DigitalInkRecognitionModel.builder(modelIdentifier!!).build()

        val remoteModelManager = RemoteModelManager.getInstance()
        remoteModelManager.download(model,
DownloadConditions.Builder().build())
            .addOnSuccessListener {
                Timber.d("DigitalInkAnalyzer : Model downloaded")
            }
            .addOnFailureListener { e: Exception ->
                Timber.d("DigitalInkAnalyzer : Error while downloading a
model: $e")
            }
    }
```

Metoda `addNewTouchEvent` se poziva unutar metode klase `CustomCanvasView` na detekciju novog dodira po ekranu za pisanje te prima koordinate putanje poteza korisnikovog pisanja.

Klasa `CustomCanvasView` predstavlja pogled unutar ekrana koji služi kao platno za digitalno pisanje.

Dobivene koordinate koristimo za stvaranje objekta tipa `Ink` koji je vektorska reprezentacija nacrtanog poteza te sadrži koordinate i točnu vremensku oznaku nastajanja poteza.

Raspoznajemo tri različite vrste akcije detekcija pisanja ovisno o tome je li pisanje u tijeku ili je završeno. Akcija `MotionEvent.ACTION_UP` se poziva podizanjem prsta sa ekrana čime se poziva metoda `addPoint` objekta tipa `Ink.Stroke.builder()` koji služi za

pamćenje trenutno napravljene pozicije i njihovih točnih vremena nastajanja te metoda `addStroke` objekta tipa `Ink.builder()` koji služi za spremanje svih dosad napravljenih poteza.

```
fun addNewTouchEvent(event: MotionEvent) {
        val action = event.actionMasked
        val x = event.x
        val y = event.y
        val t = System.currentTimeMillis()

        when (action) {
            MotionEvent.ACTION_DOWN,   MotionEvent.ACTION_MOVE   ->
strokeBuilder.addPoint(
                Ink.Point.create(
                    x,
                    y,
                    t
                )
            )
            MotionEvent.ACTION_UP -> {
                strokeBuilder.addPoint(Ink.Point.create(x, y, t))
                inkBuilder.addStroke(strokeBuilder.build())
                strokeBuilder = Ink.Stroke.builder()
            }
            else -> {
                // Action not relevant for ink construction
            }
        }
    }
```

Metoda `recognize` služi za automatsku detekciju teksta iz skice nastale digitalnim pisanjem. Instanciramo objekt tipa `DigitalInkRecognizer` koji sadrži metodu `recognize` koja prima objekt tipa `Ink.Builder()` u kojem je sadržana skica sa ekrana.

U slučaju uspješnog parsiranja, dobivamo objekt tipa `RecognitionResult` koji sadrži listu potencijalnih kandidata (`candidates`) za detektiranu riječ. Iz te liste uzimamo prvog kandidata (jer je to kandidat sa najvećom vjerojatnosti točnosti) te iz vrijednosti `text` dobivamo digitalno prepoznati teksta koji šaljemo dalje u aplikaciju na obradu.

U slučaju neuspjeha, ispisujemo dobivenu iznimku.

```
fun recognize(context: Context, onSuccess: (List<String>) -> Unit) {
        val recognizer: DigitalInkRecognizer =
            DigitalInkRecognition.getClient(
                DigitalInkRecognizerOptions.builder(model).build()
            )
        val ink = inkBuilder.build()
        recognizer.recognize(ink)
            .addOnSuccessListener { result: RecognitionResult ->
                val list = result.candidates[0].text.replace("\n",
"").replace("\r", "").split(" ")
                list.forEach { it.trim() }
                onSuccess(list)
                Timber.d("DigitalInkAnalyzer recognized text:
${result.candidates[0].text}")
                clear()
            }
            .addOnFailureListener { e: Exception ->
                Timber.d("DigitalInkAnalyzer : Error during recognition:
$e")
            }
    }
```

Pozivom metoda `clear` čisti se platno za crtanje te reinicijaliziramo objekt za spremanje napravljenih poteza.

```
                fun clear() {
                        inkBuilder = Ink.builder()
                    }
```

Sažetak

U ovom je radu opisana izrada i organizacija mobilne aplikacije My Spelling Buddy. Svrha aplikacije je da služi kao pomoćni alat pri vježbanju pisanja diktata te tako pomaže pri poboljšavanju pravopisnih i gramatičkih znanja korisnika. Sučelje aplikacije je jednostavno i intuitivno te je stoga pogodna za korištenje za sve dobne skupine.

Glavna funkcionalnost aplikacije je automatsko ispravljanje teksta diktata te prikaz napravljenih pogreški. Za ostvarivanje automatskog ispravljanja diktata i skeniranje QR koda korišten je Google ML Kit, mobilni SDK koji implementacijom algoritama iz strojnog i dubokog učenja rješava probleme vezane za vid i prirodni jezik.

Aplikacija je napisana za operacijski sustav Android te je korišten programski jezik Kotlin radi sažetosti koda i lake čitljivosti. Kao baza podataka, korišten je Firebase radi jednostavnosti implementacije i održavanja bez potrebe za pisanjem vlastitog backend servisa.

Ključne riječi: mobilna aplikacija, diktat, Android, Kotlin, Firebase, Google ML Kit, pravopis, automatsko prepoznavanje, gramatika, QR kod

Summary

This paper describes the development and organization of the mobile application My Spelling Buddy. The purpose of the application is to serve as an assistant learning tool for practicing dictation writing and thus helps to improve the spelling and grammar knowledge of the users. The application interface is simple and intuitive and therefore convenient to use for all age groups.

The main functionality of the application is the automatic correction of dictation text and the display of made errors. The Google ML Kit, a mobile SDK that solves vision and natural language problems by implementing algorithms from machine and deep learning, was used to achieve automatic dictation correction and QR code scanning.

The application was built for the Android operating system and the Kotlin programming language was used because of its code conciseness and great readability. Firebase was used as the database because of its ease of implementation and maintenance without the need of writing your own back end service.

Keywords: mobile application, dictation, Android, Kotlin, Firebase, Google ML Kit, spelling, automatic recognition, grammar, QR code