Izrada mobilne aplikacije s TTS i STT značajkama za efikasniju komunikaciju s ChatGPT-om

Vidović, Ivan

Sveučilište u Zadru

Stručni prijediplomski studij Informacijske tehnologije

Ivan Vidović

Izrada mobilne aplikacije s TTS i STT značajkama za efikasniju komunikaciju s ChatGPT-om

Završni rad

Zadar, 2023.

Sveučilište u Zadru

Stručni prijediplomski studij Informacijske tehnologije

IZRADA MOBILNE APLIKACIJE S TTS I STT ZNAČAJKAMA ZA EFIKASNIJU KOMUNIKACIJU S CHATGPT-OM

Završni rad

Student:

Ivan Vidović

Mentor: prof. dr. sc. Dino Županović

Zadar, 2023.

Izjava o akademskoj čestitosti

Ja, **Ivan Vidović,** ovime izjavljujem da je moj **završni** rad pod naslovom **Izrada mobilne aplikacije s TTS i STT značajkama za efikasniju komunikaciju s ChatGPT-om**
rezultat mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na izvore i radove navedene u bilješkama i popisu literature. Ni jedan dio mojega rada nije napisan na nedopušten način, odnosno nije prepisan iz necitiranih radova i ne krši bilo čija autorska prava.

Izjavljujem da ni jedan dio ovoga rada nije iskorišten u kojem drugom radu pri bilo kojoj drugoj visokoškolskoj, znanstvenoj, obrazovnoj ili inoj ustanovi.

Sadržaj mojega rada u potpunosti odgovara sadržaju obranjenoga i nakon obrane uređenoga rada.

Zadar, 10. listopada 2023.

SADRŽAJ:

SAŽETAK

Ovaj rad prikazuje izradu Android mobilne aplikacije „PromptCommunicator“ u Android Studio integriranom razvojnom okruženju koristeći Kotlin programski jezik. Predstavljeno rješenje pruža korisniku interakciju s modelom umjetne inteligencije koja je olakšana koristeći alate za pretvorbu govora u tekst i alate za pretvorbu teksta u govor što je ujedno i prednost ove aplikacije u odnosu na druge slične aplikacije. Također, opisana su moguća unaprjeđenja aplikacije dobivena usporedbom iste s ostalim aplikacijama, a koja su: povezivanje s bazom podataka, kreiranje pregleda povijesti razgovora, dodavanje navigacijske trake, implementacija naprednih TTS i STT modela te implementacija kompleksnijih modela umjetne inteligencije koja podržavaju ulaz više tipova podataka.

Ključne riječi: Android Studio, Kotlin, umjetna inteligencija, mobilna aplikacija.

DEVELOPING A MOBILE APPLICATION WITH TTS AND STT FEATURES FOR MORE EFFICIENT COMMUNICATION WITH CHATGPT

SUMMARY

This paper presents the development of the Android mobile application „PromptCommunicator“ in the Android studio integrated development environment using the Kotlin programming language. The solution introduced provides the user with interaction with an artificial intelligence model facilitated by speech-to-text and text-to-speech conversion tools, which is an advantage of this application compared to the other similar applications. Furthermore, potential improvements to the application are described through a comparison with other similar applications. These improvements include database integration, creating a conversation history view, adding navigation bar, implementing advanced TTS and STT models, and integrating more complex artificial intelligence models that support multiple types of input.

Keywords: Android Studio, Kotlin, artificial intelligence, mobile application.

POPIS KORIŠTENIH KRATICA

AI Artificial Inteligence

IDE Integrated Development Environment

TTS Text-To-Speech

STT Speech-To-Text

GPT Generative Pre-Trained Transformer

XML Extensible Markup Language

OOP Object-oriented programming

RLHF Reinforcement Learning from Human Feedback

API Application Programming Interface

SDK Software Development Kit

KB KiloByte

MB MegaByte

HTTP Hypertext Transfer Protocol

1. Uvod

U novije vrijeme AI tehnologije postaju sve više javno dostupne, one se uvode u aplikativna rješenja kako bi se korisniku olakšala interakcija s određenom aplikacijom ili kao chat aplikacije gdje je omogućena izravna komunikacija s AI modelima. Vrijednost AI tržišta u 2023. godini iznosi približno 136 milijardi dolara dok je rast projektiran na trinaest puta veću vrijednost u sljedećih 7 godina.[23]. Također, procjenjuje se kako će približno 100 miliona ljudi do godine 2025. raditi u AI okruženju.[23].

Cilj ovog rada je izraditi potpuno funkcionalnu, javno dostupnu Android mobilnu aplikaciju pod nazivom „PromptCommunicator“ baziranu na najnovijim i najrelevantnijim tehnologijama u svijetu IT-a koristeći dostupne AI alate koji pružaju mogućnost interakcije s modelom umjetne inteligencije koja je ostvarena putem pretvorbe govora u tekst te pretvorbe teksta u govor u svrhu olakšavanja komunikacije i interakcije krajnjeg korisnika s AI modelom. U izradi navedene aplikacije koristi se Android Studio integrirano razvojno okruženje primjenom Kotlin programskog jezika. AI tehnologija s kojom sama aplikacija komunicira dio je ChatGPT alata, odnosno GPT-3 modela kreiranog od strane tvrtke OpenAI. Vezano za interakciju u samoj aplikaciji koriste se TTS i STT moduli koji su besplatni te su dio Android platforme: TTS komponenta dio je Android Studio razvojnog okruženja, dok je STT komponenta dio Android operacijskog sustava. Svrha „PromptCommunicator“ Android mobilne aplikacije je pružanje stabilne usluge povezivanja korisnika s AI modelom uz olakšanu komunikaciju i interakciju.

U prvom poglavlju iznesen je kratak osvrt i uvod u sve tehnologije koje su korištene u izradi aplikacije uz objašnjenje načina na koji rade. Iduće poglavlje odnosi se na problem za koji se postavlja aplikativno rješenje te je iznesen razlog korištenja tehnologija koje su dio istog. Nakon toga, u radu se opisuje način na koji je izrađena aplikacija, problemi do kojih je dolazilo prilikom izrade i način na koji je cjelokupan projekt strukturiran te se daje pregled dizajna aplikacije i uvid u funkcionalni dio aplikacije. U zadnjem dijelu rada nalazi se dio koji se tiče nadogradnje rješenja te se vrši usporedba s postojećim rješenjem. Rad završava zaključkom.

2. Pregled tehnologija

U ovom poglavlju navedeni su osnovni podatci o tehnologijama koje su se koristile za izradu aplikacije koja je dio ovog rada. Završni rad realiziran je kao android mobilna aplikacija izgrađena u Android studio integriranom razvojnom okruženju u Kotlin programskom jeziku. Ona povezuje uslugu ChatGPT tvrtke OpenAI uz pomoć Text-To-Speech i Speech-To-Text tehnologija te različitih programskih biblioteka s korisnikom u svrhu poboljšanja komunikacije ljudi s umjetnom inteligencijom.

2.1 Android Studio IDE

Android Studio je integrirano razvojno okruženje razvijeno na IntelliJ IDEA platformi, a namijenjeno je izradi android mobilnih aplikacija.[1]. Android Studio predstavljen je 16. svibnja 2013. godine na Google I/O konferenciji kao verzija 1.0. Godine 2020. predstavljena je verzija 4.0 koja je ujedno i posljednja s navedenim formatom naziva verzije. Godine 2021. verzije dobivaju ime po životinjama. Tako je prva verzija bila Arctic Fox, zadnja dostupna verzija je Giraffe u kojoj je aplikacija koja je dio završnog rada dovršena.

2.1.1 Emulator

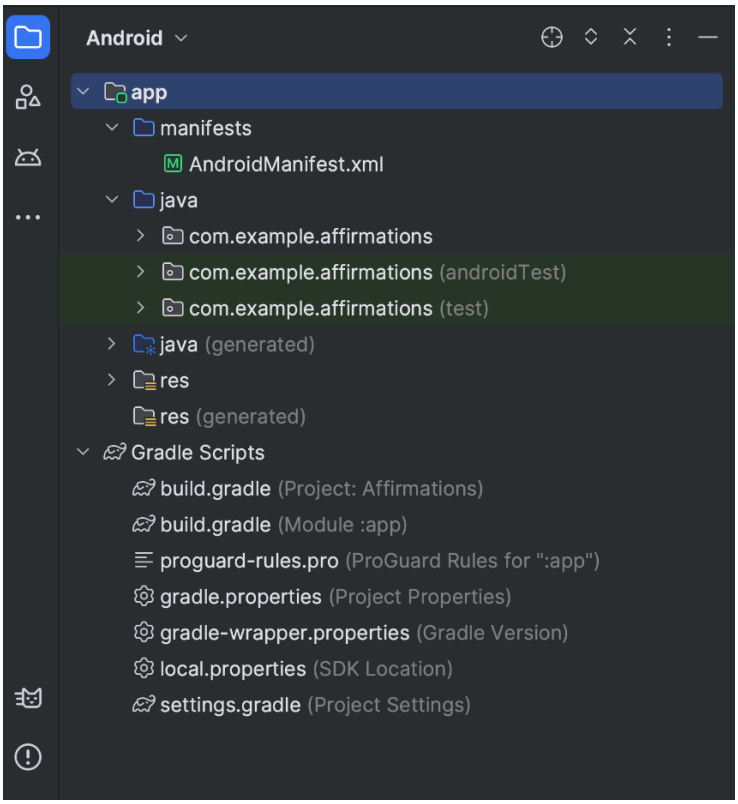
Android studio sadrži integrirani android emulator u kojemu je moguće kreirati više virtualnih uređaja s različitim verzijama android operativnog sistema što uvelike olakšava proces testiranja pogotovo na različitim veličinama zaslona. Kod samog kreiranja izgleda aplikacije omogućeno je dijeljenje frontend dijela aplikacije na više različitih prikaza za različite zaslone, od malih, srednjih do velikih zaslona.

2.1.2 Struktura projekta

Struktura projeta u Android Studio-u podijeljena je u više mapa:

- manifests – sadrži AndroidManifests.xml datoteku
- java – sadrži datoteke u kojima se nalazi izvorni Java ili Kotlin kod te JUnit test kod
- res – sadrži sve datoteke koje nisu programski kod, odnosno bitmap slike i UI string-ove
- Gradle Scripts – sadrži module projekta prema kojima je pregled organiziran zbog lakšeg pristupa ključnim datotekama:
    - Android App moduli
    - Moduli biblioteka
    - Google App Engine moduli

Prikaz i interakcija s navedenom stukturom omogućena je u lijevom dijelu grafičkog korisničkog sučelja aplikacije kao Project menu:



Slika 1 Projektne datoteke u prikazu projekta [1]

Navedeni prikaz strukturiran je na način prilagođen korisniku, a kako bi prikazali stvaran prikaz mape projekta potrebno je odabrati Projekt umjesto Android u meniju projekta.

2.2 Kotlin programski jezik

Kotlin je višeplatformski, objektno orijentirani programski jezik koji koristi statičke tipove podataka. Prvi put je predstavljen 2011. godine kao potencijalno bolje rješenje od Java programskog jezika, a 7. svibnja 2019. godine Google je Kotlin proglasio preferiranim jezikom za izradu Android aplikacija, a već 2020. godine Google iznosi procjenu kako je više od 70% od 1000 najboljih aplikacija koje se nalaze na Google Play Store-u izrađeno upravu u Kotlinu dok je trenutan postotak dosegao visokih 95%.[2]. Najveći razlog tomu je što je kod pisan u Kotlin-u čitljiviji te manjeg opsega. Međutim, Kotlin je interoperabilan s Java programskim jezikom te je u istom projektu moguće koristiti kodove pisane u oba jezika. Uz navedeno, ostale beneficije kreiranja aplikativnih rješenja u Kotlin-u su:

- Manje grešaka – 20% manje mogućnosti rušenja aplikacije [2]
- Podrška za Jetpack biblioteke
- Manje zahtjevan jezik za učenje (pogotovo za Java developere)
- Velika baza korisnika

Kotlin također nudi i mogućnost izrade aplikacija za više platformi kao što su web i iOS development.[2].

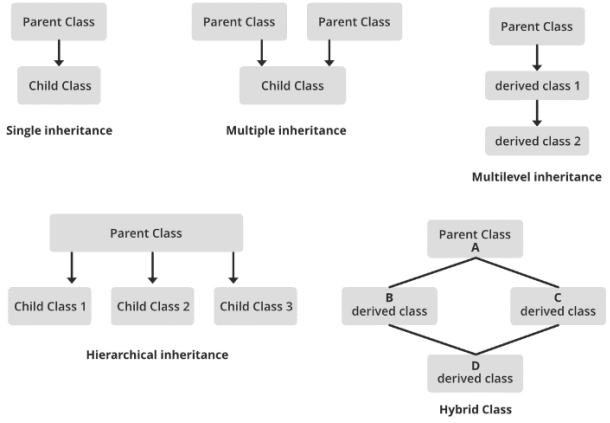
2.2.1 Objektno orijentirano programiranje

Objektno orijentirano programiranje je naziv za pristup razvoju aplikativnih rješenja na način da se aplikacija strukturira kao skup objekata (objekti iz stvarnog života) koji su realizirani u klasama koji međusobno komuniciraju na način da svaki objekt ima atribute i metode koje se koriste u daljnjem razvoju što je i glavni cilj OOP. [3]. Ovaj princip razvoja, za razliku od proceduralnog programiranja, daje mogućnost rješenja problemima iz stvarnog svijeta.

- Objekt – predstavlja objekt iz stvarnosti
- Atributi – opisuju objekt
- Metode – funkcije specifične za objekt koje mogu mijenjati njegove atribute

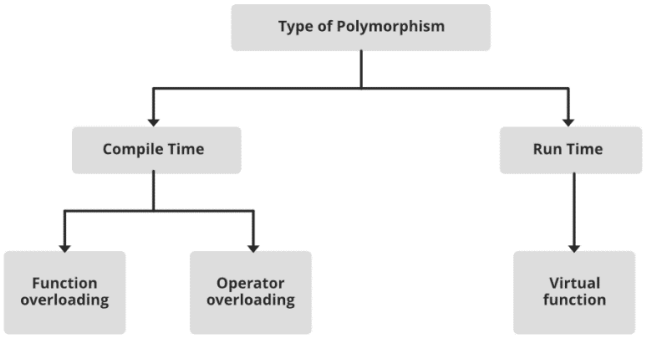
Glavni „stupovi“ OOP-a su:

- Učahurivanje (enkapsulacija) – nije moguće direktno pristupiti privatnim varijablama objekta iz drugih dijelova koda već se pristupanje vrši pomoću ugrađenih metoda za pisanje i čitanje te se na ovaj način osigurava da objekt ne dođe u nepredviđeno stanje.
- Apstrakcija – kreiranje početnog objekta čije „osobine“ (funkcije i atribute) nasljeđuju objekti sličnih vrijednosti u svrhu smanjenja redudancije i količine programskog koda. Na primjer, klasa Životinja je apstraktna klasa čiji će osobine naslijediti klase Pas ili Mačka.
- Nasljeđivanje – opisano u prijašnjoj točki, objekt koji je podskup nekog definiranog objekta može ga naslijediti u svrhu uštede vremena te također smanjenja redudancije koda



Slika 2 Nasljeđivanje u OOP [3]

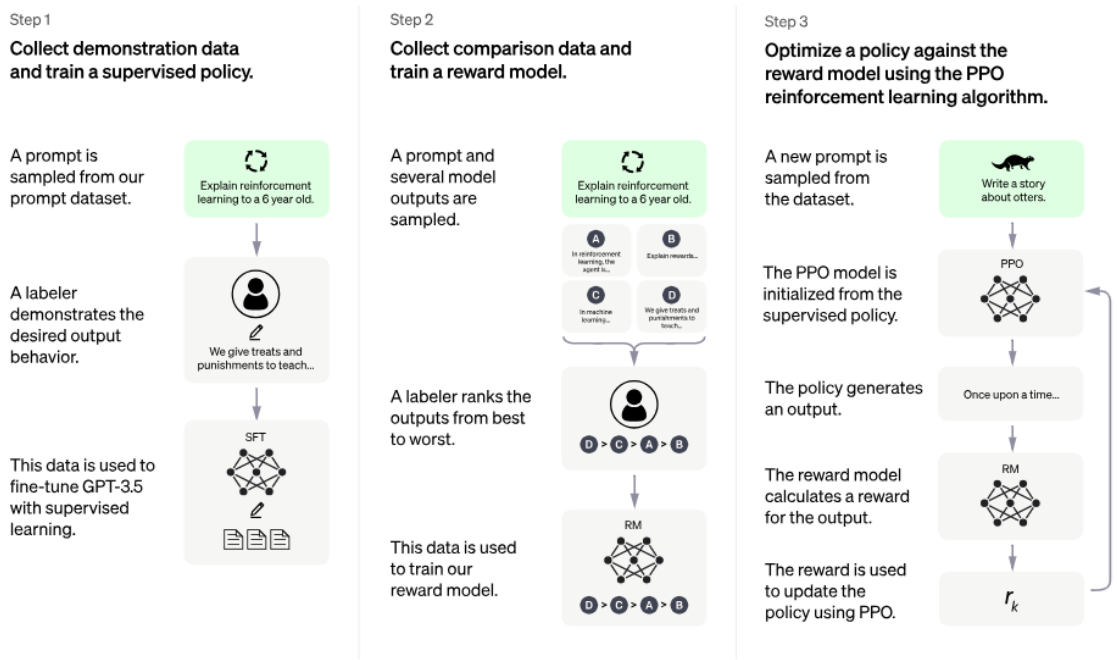
- Višeobličje (polimorfizam) – preopterećivanje metoda ili method overloading, pomoću ovog principa moguće je definirati više metoda istog imena, a proslijediti im različite paramtere, to jest objekte različitog tipa.



Slika 3 Polimorfizam u OOP [3]

2.3 OpenAI: ChatGPT

ChatGPT (Generative Pre-Trained Transformer) je oblik umjetne inteligencije, odnosno umjetna inteligencija uparena s obradom prirodnog jezika te strojnog učenja čime se dobiva tekstualna interakcija koja nalikuje međuljudskom dijalogu. Tvorac ChatGPT-a je tvrtka OpenAI koja radi na razvoju i istraživanju umjetne inteligencije, čija je misija omogućiti korist umjetne inteligencije čovječanstvu.[4]. Tvrtku OpenAI osnovalo je više znanstvenika 2015. godine, a neki od njih su Elon Musk i Sam Altman. ChatGPT dostupan je od 2022. godine, a radi tako da pomoću algoritama koji obrađuju velike količine teksta generira konkretan odgovor. ChatGPT treniran je koristeći Reinforcement Learning from Human Feedback (RLHF).[5]. Procesi RLHF treniranja prikazani u sljedećoj slici:



Slika 4 Proces RLFH treniranja [5]

Alat se razlikuje po sljedećim modelima:

- GPT-3 – prvi modeli specijalizirani za generiranje čistog i jasnog teksta/instrukcija.
- GPT-3.5 – temeljen na GPT-3 modelima uz veću usmjerenost na konverzacije.
- GPT-4 – najnoviji model i najnapredniji u smislu rješavanja problema koji omogućuje dvije vrste input-a, uz tekst ovom modelu moguće je proslijediti i slike.

Također, razlikuju se i po cijeni (1000 token ≈ 750 riječi):

- GPT-3 – $0.0004-$0.0020 / 1000 token
- GPT-3.5 – $0.0015-$0.003 / 1000 token
- GPT-4 - $0.03-$0.06 / 1000 token

2.4 Text-To-Speech i Speech-To-Text

Text-To-Speech (TTS) i Speech-To-Text (STT) su dvije tehnologije koje omogućuju računalima interakciju s ljudskim govorom. TTS pretvara tekst u zvuk ili ljudski govor, dok STT pretvara ljudski govor u tekst. Iako su navedene tehnologije razvijene prije više od 70 godina, uz pomoć razvoja umjetne inteligencije i strojnog učenja očekuje se eksponencijalan rast razvijenosti i primjene što je i trenutno vidljivo u obliku imitiranja govora bilo koje osobe, kreiranja glazbe kao i razinom stvarnosti u govoru dobivenom iz teksta te razinom preciznosti i prepoznavanja jezika kod pretvorbe govora u tekst.

2.4.1 TTS

TTS tehnologija radi na način da tekst razdvaja u foneme koji su najmanja jedinica zvuka u govoru.[6] Nadalje, fonemi se pretvaraju u zvučne valove koji tvore zvuk. Procesi pretvaranja teksta u zvuk dijele se na dva sistema:

- Statistički sistem – prema statističkom modelu vrši se pretpostavka mogućnosti za svaki fonem u danom kontekstu.[6].

- Sistem baziran na pravilima – predstavlja unaprijed zadana pravila prema kojima se zvuk pretvara u tekst.[6].

Neki od primjera korištenja TTS tehnologija su: virtualna i proširena stvarnost, audio knjige i E-čitači, edukativne aplikacije, govorni asistenti i slično.

2.4.2 STT

STT tehnologije rade na principu razdvajanja govornog signala u foneme koji se nadalje pretvaraju u tekst koristeći rječnike i skupove pravila. Ovi sistemi dijele se prema ovisnosti o govorniku, tako postoje sistemi koji komuniciraju samo s jednim govornikom, to jest s jednim glasom i oni koji komuniciraju s više govornika, odnosno više glasova.[6].

2.5 Biblioteke

Prilikom izrade aplikacije koja je dio ovog završnog rada bilo je potrebno uključiti više vanjskih biblioteka bez kojih bi se opseg koda aplikacije drastično povećao, a samim time i vrijeme potrebno za izradu iste. Programska biblioteka je kolekcija već napisanog programskog koda koja razvojnom programeru omogućuje brži i lakši razvoj, a obično sadrži više različitih komponenti te je specifična za određeni učestali problem.[7].

2.5.1 Korištenje biblioteka

Programeri koriste biblioteke za efikasniju izradu aplikacija. Svaka biblioteka je rješenje određenog problema. Specifični problemi koje biblioteke rješavaju su autentikacija i autorizacija, serverske konekcije, algoritmi, animacije, komunikacija s bazom podataka i slično. One se ne izvršavaju samostalno, već se implementiraju po potrebi, a njihov kod strukturiran tako da se može koristit u različitim aplikacijama. Programer koristi njihove funkcije i podatke na način koji je potreban u određenom kontekstu aplikacije te mu nije potreban uvid u cijeli kod biblioteke već je bitno znati samo na koji način ona funkcionira.

3. Opis problema

Svaka aplikacija osvrt je na određeni problem u stvarnom svijetu, produkt aplikacije je efikasnije rješenje koje zahtijeva manje resursa, bilo to vrijeme, novac ili nešto treće. Aplikacija kreirana kao dio ovog rada iznosi rješenje na problem koji je bio aktualan tokom početka izrade iste. Prilikom razvoja aplikacije, razvijena su razna rješenja fokusirana na isti problem što je dobar primjer brzine razvoja tehnologije.

3.1 Problem

Razvojem umjetne inteligencije, u ovom radu, konkretno, ChatGPT-a koji postaje javno dostupan AI alat dolazi do brzog rasta baze korisnika i ogromnog opterećenja na web aplikaciju koja omogućuje interakciju s navedenim modelom, a koju je kreirala tvrtka OpenAI koja je ujedno i tvorac ChatGPT-a. Što je u početku dovodilo do stalnog rušenja web servisa te njegove nedostupnosti pri čemu je dolazilo do otežane komunikacije između korisnika i modela. Nadalje, aplikacija nije pružala integriranu mogućnosti pretvorbe teksta koji je produkt interakcije u govor kao i unos glasovnog ulaza od strane korisnika što je također utjecalo na efikasnost korištenja aplikacije. Uzevši sve navedeno u obzir, dolazi do potrebe za razvojem rješenja koje će rasteretiti sistem i omogućiti bolju povezanost te veću korist za krajnjeg korisnika koji pokušava koristiti ovaj model umjetne inteligencije makar kao specijalizirano rješenje za manju ciljanu skupinu korisnika. Kako web rješenje već postoji dolazi do potrebe za rješenjem na različitim platformama, konkretno na mobilnim uređajima. Međutim, izrada ove vrste rješenja nije bila moguća u početku. Naime, ChatGPT postaje javno dostupan 2022. godine, dok ChatGPT API tada nije bio javno dostupan.

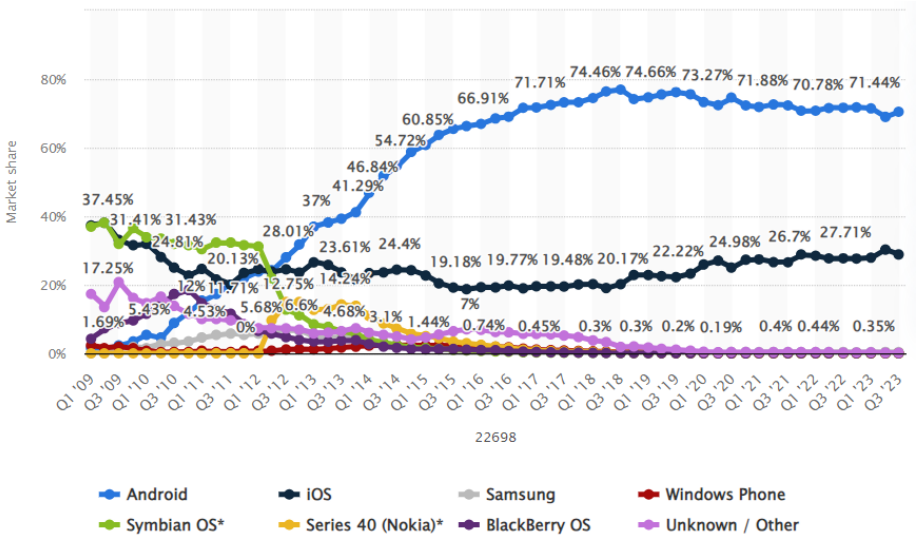
Godinu dana kasnije, 1. ožujka 2023. OpenAI postavlja ChatGPT API kao javno dostupan prilikom čega dolazi i do teme ovog završnog rada kako je sada bilo moguće izraditi aplikaciju koja će koristiti ovaj model na razini mobilne aplikacije uz implementiranje mogućnosti za pretvorbu teksta u govor i govora u tekst u svrhu povećanja efikasnosti.

4. Rješenje

Nakon detaljnije analize problema, dolazimo do teorijske razrade rješenja. Svako rješenje je unikatan proizvod u kojemu do izražaja dolaze snalažljivost, iskustvo i kreativnost svakog programera. Programer shvaća zahtjeve problema te prema njiva gradi tok rješavanja istog. Kao što građevinski objekt ne možemo graditi od krova prema temelju, tako je i za izradu aplikativnog rješenja potrebno postaviti dobar temelj koji se bazira na detaljnom proučavanju problema kako bi mogli vizualizirati tok i sve tehnologije prema kojima vodimo aplikaciju do implementacije, to jest, puštanja u rad.

4.1 Razrada rješenja i odabir platforme

Kada smo odredili problem ili dijelove problema za koje želimo postaviti aplikativno rješenje, potrebno je razmotriti tehnologije te njihovu dostupnost koje će nam biti potrebne kako bi došli do rješenja prije samo razvoja. U ovom primjeru znamo da je potrebno rasteretiti web aplikaciju putem koje se vrši interakcija sa ChatGPT modelom te olakšati istu. Prvo je potrebno odabrati ciljanu platformu, a kako već postoji web aplikacija kojoj je lako pristupiti putem osobnog računala u uži izbor dolaze mobilne platforme kao što su Android i iOS. U ovom dijelu potrebno je analizirati potencijalno tržište.



Slika 5 Prikaz postotka korisnika mobilnih operacijskih sustava [8]

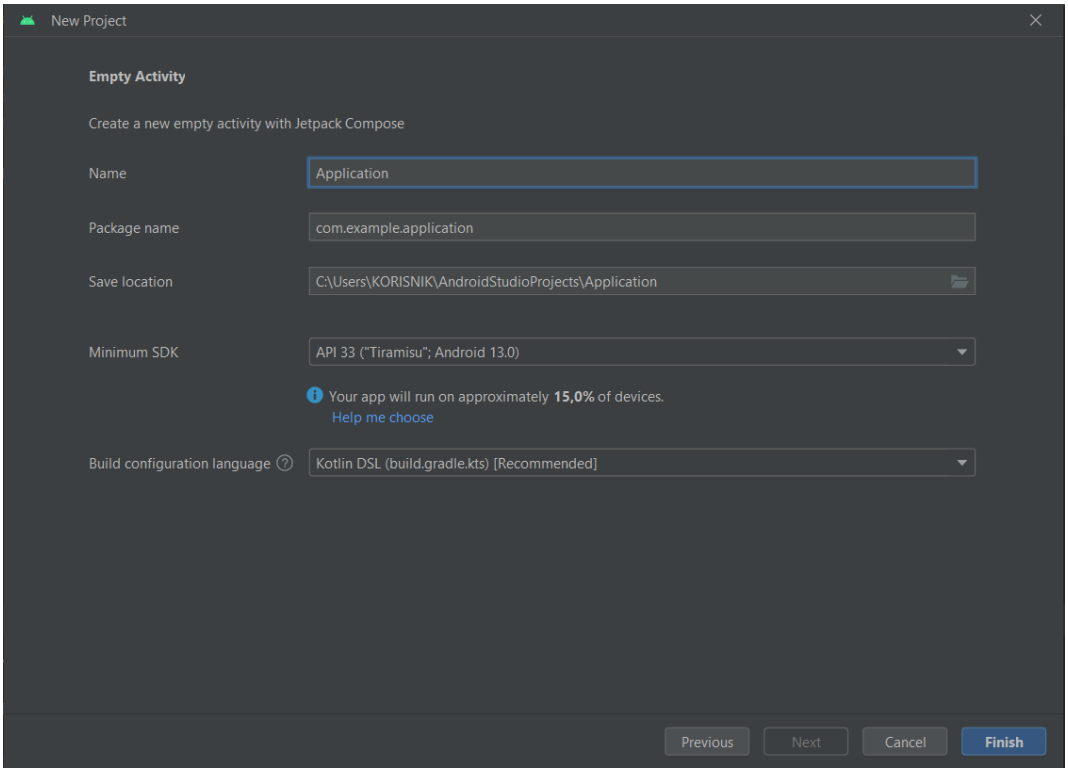
Na slici 5 vidljivo je kako preko 70% korisnika mobilnih uređaja koristi Android platformu, tek nešto više od 27% korisnika koristi iOS, dok su ostali operacijski sustavi zastupljeni tek kod manje od 0.5% slučajeva. Sukladno s navedenim, odabrana platforma za izradu aplikacije postaje Android. Nakon odabira ciljane platforme, potrebno je odabrati i integrirano razvojno okruženje ili IDE u kojemu kreiramo aplikaciju. Bitno je uzeti o obzir više stavki, kao što su: cijena, dostupnost, učestalost nadogradnje, mogućnost daljnjeg razvoja u istom razvojnom okruženju i slično. [9]. Uzevši nadvedeno u obzir, odabir se svodi na Android Studio kao trenutno najpopularnije, javno dostupno i besplatno integrirano razvojno okruženje koje uz integrirani emulator pruža daleko najbolje uvijete za izradu aplikacije. Što se tiče programskog jezike, na izboru imamo Java programski jezik i Kotlin programski jezik, kao što je navedeno u poglavlju 2.2, Kotlin je trenutno najpopularniji jezik za izradu Android aplikacija i smatra se preferiranim jezikom za izradu istih postaje logičan odabir iz izradu.

4.1.1 Dostupnost vanjskih biblioteka i API-a

Prvo pitanje ili problem s kojim se susrećemo je dostupnost ChatGPT API-a što je omogućeno nakon što je API postao javno dostupan u tekućoj godini. Nadalje, postavlja se pitanje dostupnosti TTS i STT modula i biblioteka. Pretvorba teksta u govor moguća je putem Android Studio okruženja još od 2009. godine.[10]. Međutim, razvojem umjetne inteligencije i strojnog učenja, AI alati za TTS i SST postaju sve više javno dostupni putem korištenja raznih API-a kao što je whisper.ai API tvrtke OpenAI. Korištenje navedenih API-a skupo je za krajnjeg korisnika te se za potrebe ovog projekta implementira integrirana mogućnost Android Studio-a koja je besplatna i daleko naprednija od besplatnih solucija baziranih na AI tehnologijama. Druga bitna stavka je samo povezivanje koje je omogućeno putem okhttp3 biblioteke.[11].

4.2 Razvoj aplikacije

Nakon uvida u dostupne alate, odabira platforme i programskog jezika, započeta je izrada same aplikacije koja je podijeljena na frontend (sam izgled aplikacije) i backend (procesi koji se odvijaju u pozadini aplikacije). Prije svega potrebno je kreirati novi projekt u Android Studio integriranom razvojnom okruženju:

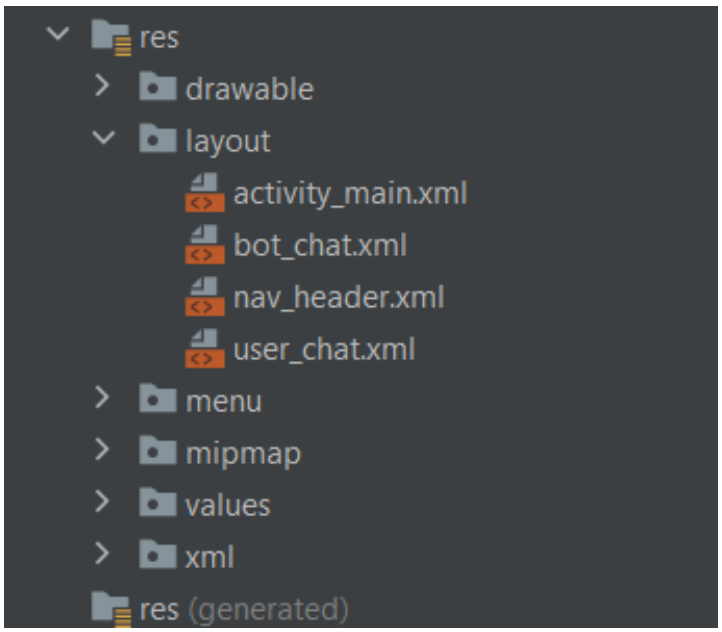


Slika 6 Kreiranje projekta Android Studio

Prilikom kreiranja projekta, potrebno je odabrati naziv projekta, naziv paketa, direktorij u kojemu će projekt biti spremljen te minimalan SDK, to jest, verziju Android sustava od koje je moguće pristupiti aplikaciji. Za potrebe ovog projekta, odabrana je najnovija dostupna verzija sustava Android: Android 13, „Tiramisu“. Iako je navedena verzija trenutno zastupljena u tek oko 15% uređaja, odabrana je zbog mogućnosti korištenja najnovijih značajki Android sustava te također iz razloga što je Android 13 relativno nova verzija kojoj korisnici tek počinju pristupati. Razvoj same aplikacije podijeljen je na frontend dio (izgled i dizajn sučelja) te backend dio (svi pozadinski procesi za koje korisnik ne treba znati) što su ujedno i najpopularniji termini u IT svijetu.

4.3 Frontend

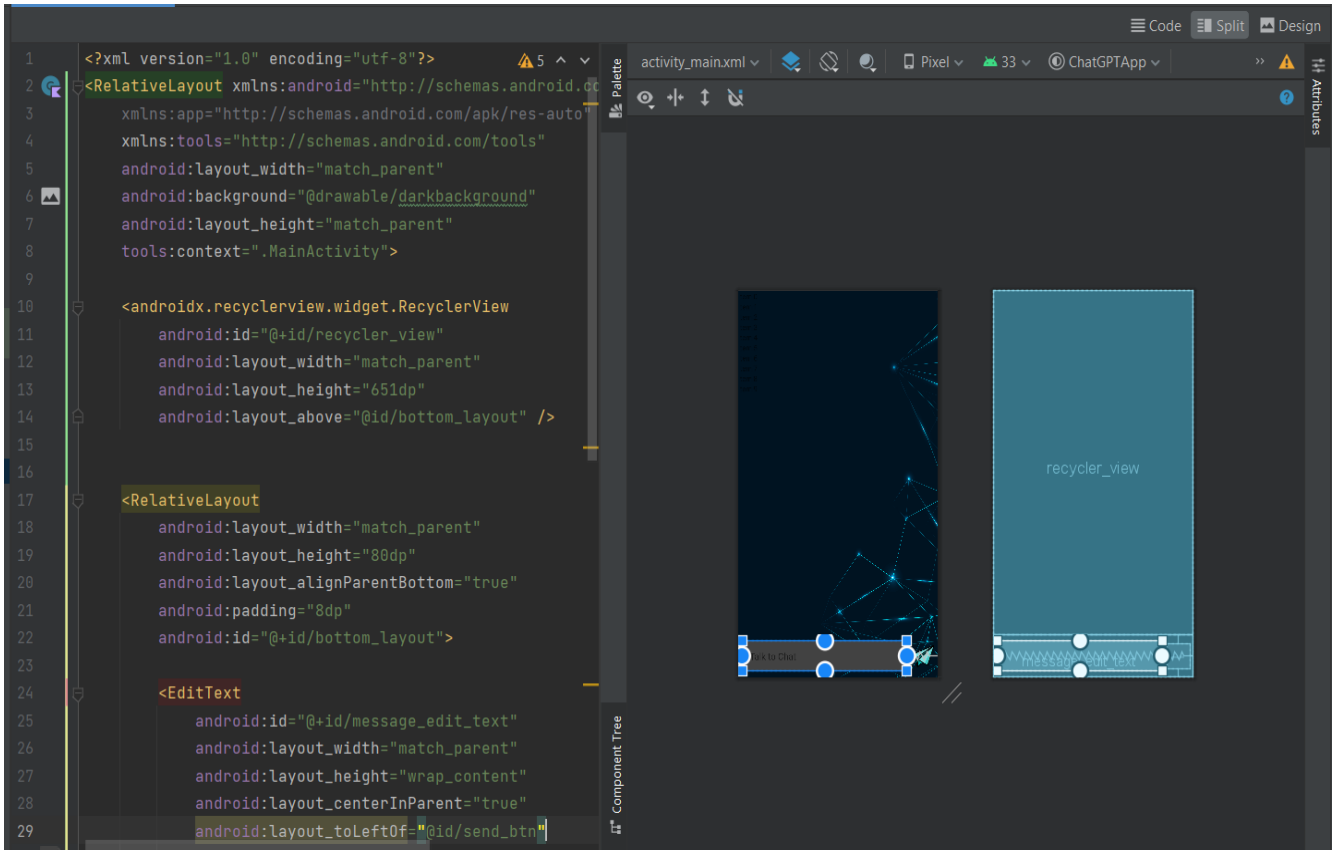
Frontend se odnosi na vizualni dio aplikacije, odnosno grafičko korisničko sučelje ili GUI. U frontend dijelu aplikacije radi se na onomu što korisnik vidi, dizajnira se i postavlja izgled ekrana, to jest, dodaju se komponente koje oni sadrže kao što su: gumbi, tekstni okviri, slike, pozadine i slično. Vezano za izgled aplikacije koja je dio ovog rada, pažnja se obraća na način korištenja aplikacije. Ona će se koristiti za čitanje i unos teksta što može biti naporno za oko. Iako je implementirano rješenje za pretvorbu govora u tekst i teksta u govor, i dalje je potrebno gledati ekran, uzevši navedeno u obzir postavljana je tamna pozadina, dok je tekst svijetle boje kako bi se postigao kontrast i lakša čitljivost.



Slika 7 Frontend mapa aplikacije

Kao što je navedeno u poglavlju 2.1.2, struktura projekta u Android Sudio-u podijeljena je u mape. Mapa res sadrži sve datoteke koje se tiču izgleda aplikacije. Tako u mapi drawable imamo ikone koje se koriste u aplikaciji. Menu mapa sadrži stavke navigacijske trake. Mipmap mapa sadrži ikone vezane za pokretanje i prikaz aplikacije. Values mapa sadrži informacije o tekstu koji se prikazuje u aplikaciji, bojama i temama aplikacije. Xml mapa sadrži skup pravila vezanih za lokalan i cloud backup informacija koje čine aplikaciju. Layout mapa sadrži .xml datoteke u kojima se kreira izgled zaslona. Pogledom na sliku 7 vidimo četiri različite datoteke formata .xml čiji će sadržaj biti prikazan u nastavku rada.

Izgled aplikacije kreiran je u četiri navedene .xml datoteke unutar layout mape gdje je sam izgled moguće uređivati vizualno ili putem koda. U slici ispod prikazan je dio sadržaja activity_main.xml datoteke te je opisan način na koji je moguće birati između preferiranog oblika uređenja dizajna:



Slika 8 Uređenje layout datoteka

U gornjem desnom kutu slike 8 vidimo tri gumba:

- Code gumb – prikazuje zaslon na kojemu je moguće unositi samo xml kod (lijeva polovica slike)
- Split gumb – prikazuje podijeljen zaslon kao na slici gdje je lijeva strana xml kod, a desna strana grafički prikaz koda
- Design gumb – prikazuje zaslon na kojemu je moguće vizualno uređenje sučelja (desna polovica slike)

Acitvity_main.xml datoteka sadrži kod za postavljanje pozadine aplikacije, okvira za unos teksta koji se prosljeđuje ChatGPT API-u i ikone za slanje istog:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:background="@drawable/darkbackground"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recycler_view"
        android:layout_width="match_parent"
        android:layout_height="651dp"
        android:layout_above="@id/bottom_layout" />


    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="80dp"
        android:layout_alignParentBottom="true"
        android:padding="8dp"
        android:id="@+id/bottom_layout">

        <EditText
            android:id="@+id/message_edit_text"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_centerInParent="true"
            android:layout_toLeftOf="@id/send_btn"
            android:background="@drawable/rounded_corner_right"
            android:hint="@string/talk_to_chat"
            android:padding="16dp" />

        <ImageButton
            android:id="@+id/send_btn"
            android:layout_width="48dp"
            android:layout_height="48dp"
            android:layout_alignParentEnd="true"
            android:layout_centerInParent="true"
            android:layout_marginStart="10dp"
            android:background="?attr/selectableItemBackgroundBorderless"
            android:padding="8dp"

            android:src="@drawable/send_svgrepo_com" />


    </RelativeLayout>


</RelativeLayout>
```

BotChat.xml datoteka sadrži izgled tekstnog okvira u komu se prikazuju tekstovi koji su produkt interakcije s API-em. Svaki povratni tekst kreira novi okvir:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:padding="8dp">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/rounded_corner_left"
        android:padding="8dp">

        <TextView
            android:id="@+id/left_chat_text_view"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="@color/white"
            android:textSize="18sp" />

    </LinearLayout>

    <TextView
        android:id="@+id/left_chat_timestamp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="2dp"
        android:textColor="#a9a9a9"
        android:textSize="12sp" />
</LinearLayout>
```

User_chat.xml datoteka realizirana je na isti način, a jedina razlika je položaj tekstnog okvira koji se sada nalazi na desnoj strani zaslona:

```
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@drawable/rounded_corner_right"
        android:padding="8dp">
        <TextView
            android:id="@+id/right_chat_text_view"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="@color/white"
            android:textSize="18sp" />
    </LinearLayout>
```

Nav_menu.xml datoteka sadrži kostur vezan za navigacijsku traku koja je predmet daljnje nadogradnje aplikacije s obzirom da trenutne cijene korištenja ChatGPT API-a to ne dopuštaju na ovoj razini aplikacije, odnosno za aplikaciju koja nije komercijalna:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

</androidx.constraintlayout.widget.ConstraintLayout>
```

Dok se elementi navigacijske trake nalaze u menu mapi kao zasebna .xml datoteka:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/fstItem"
        android:title="First item"/>
    <item
        android:id="@+id/sndItem"
        android:title="Second item"/>
    <item
        android:id="@+id/trdItem"
        android:title="Third item"/>
</menu>
```

Na ovaj način postavljen je uvid u daljnje mogućnosti razvoja i smjer u kojem bi razvoj potencijalno mogao krenuti. Vezano za values mapu u kojoj se nalaze boje, teme i string podatci projekta omogućeno je smanjenje redudancije i opsega količine proizvedenog koda na način da se svaki od navedenih elemenata postavljaju samo jednom te im je moguće pristupiti iz različitih dijelova aplikacije:

```
<color name="purple_200">#FFBB86FC</color>
```

```
<string name="app_name">ChatGPTApp</string>
```

Pomoću navedene strukture na projektnim mjestima gdje je potrebno pristupiti nekima od navedenih resursa potrebno je upisati samo njegov naziv kao što je „purple_200“ za boju čija heksadekadska vrijednost iznosi #FFBB86FC iz čega je odmah vidljiva korist ovog načina zapisa podataka i njihovih vrijednosti.

Posljednja mapa vezana za izgled korisničkog sučelja je mapa „themes“. U ovoj mapi nalaze se prije generirane teme aplikacije koje su dostupne programeru za uređenje. Teme su podijeljene u dvije datoteke:

- Themes.xml – svijetla tema
- Themes.xml (night) – tamna tema

Navedene datoteke postavljaju primarne i sekundarne boje u odnosu na temu koju korisnik postavlja na vlastiti uređaj za cjelokupan operacijski sustav. Specifičnosti svijetle teme su svijetla pozadina i taman tekst, dok je za tamnu temu specifična tamna pozadina i svijetli tekst. Izgled same datoteke u kojima su smješteni parametri teme izgleda ovako:

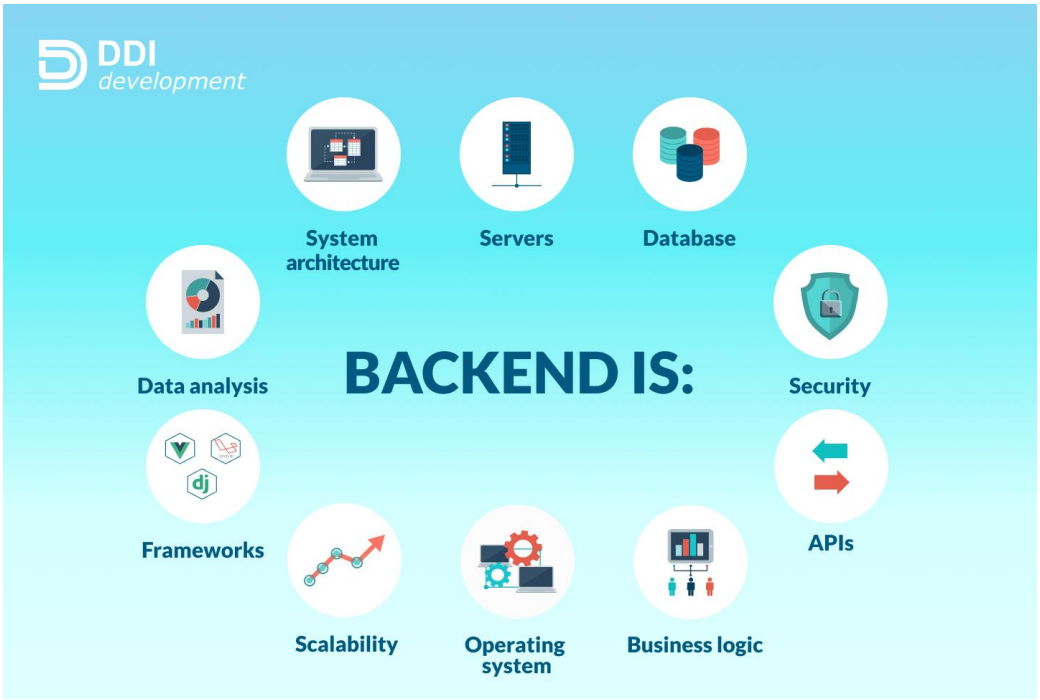
```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.ChatGPTApp"
parent="Theme.MaterialComponents.DayNight.NoActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/black</item>
        <item name="colorPrimaryVariant">@color/black</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>
```

Pod stavkom „parent“ postavlja se izgled takozvanog „Action Bar-a“ odnosno zaglavlja aplikacije, a u nastavku koda postavljaju se boje specifične za temu gdje je također vidljiva referenca na postavljene vrijednosti colors.xml datoteke koja se nalazi u values mapi

4.4 Backend

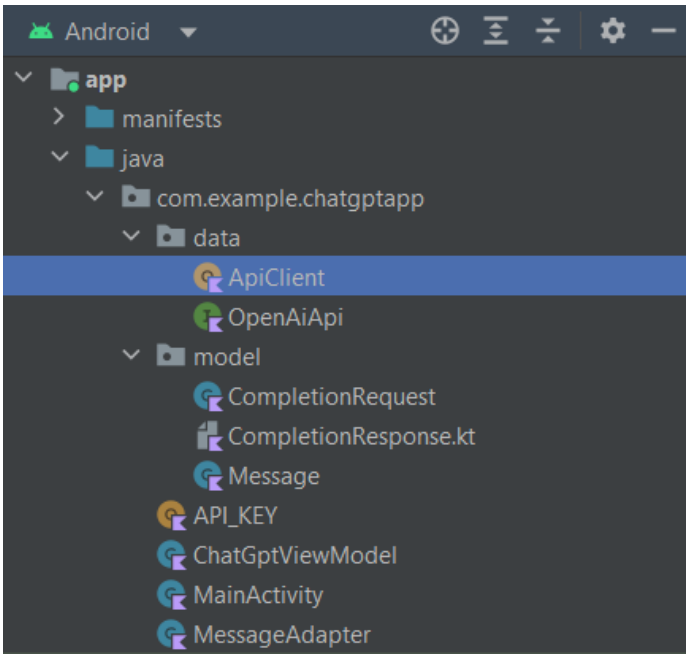
Backend strana razvoja aplikacije odnosi se na sve ono što se odvija u pozadini, odnosno procese koje korisnik ne vidi, a koji su krucijalni za rad same aplikacije.[13]. Na ovaj način razdavaju se dvije strane aplikacije (frontend i backend). U backend razvoj aplikacije ulaze svi elementi koji se tiču serverske strane aplikacije kao što su:

- Kreiranje i upravljanje serverskom stranom aplikacije
- Spajanje aplikacije s bazom podatka i komuniciranje s istom
- Kreiranje API-a
- Rukovanje autentikacijom i autorizacijom korisnika
- Procesiranje i odgovaranje na korisničke zahtjeve
- Slanje podataka na frontend dio aplikacije



Slika 9 What is backend development? [14]

Backend dio ove aplikacije zasniva se na pristupanju ChatGPT API-u putem okhttp2 biblioteke uz pomoć url-a samog API-a te API ključa koji se dobiva na zahtjev putem web stranice OpenAI [4] te korisničkog unosa dohvaćenog s frontend dijela aplikacije, parsiranju dohvaćenog koda pomoću retrofit2 biblioteke, prosljeđivanje dobivenog na dio aplikacije koja pretvara tekst u govor pomoću TextToSpeech biblioteke te slanje na frontend dio preko bot_chat.xml datoteke. Struktura backend dijela aplikacije prikazana je na slici ispod:



Slika 10 Struktura klasa

4.4.1 ChatGPT API

Kako bi bili u mogućnosti koristiti ChatGPT API, potrebno je zakupiti određenu količinu tokena (cijena navedena u poglavlju 2.3) te kreirati svoj API ključ za određeni ChatGPT model koji će se koristiti za generiranje odgovora. Nadalje, interakcija s API-em moguća je putem bilo kojeg programskog jezika koristeći http client. Način na koji je realizirana komunikacija sa API servisom prikazana je u sljedećim koracima:

- Nakon dohvaćanja API ključa isti je potrebno spremiti na sigurno mjesto unutar aplikacije pošto je isti vidljiv samo jednom nakon kreiranja iz sigurnosnih razloga (slika 15, klasa API_KEY)

- Nadalje, potrebno je generirati kod koji služi spajanju na API servis. Klasa ApiClient te sučelje OpenAiApi sadrže pristupni kod čija je dokumentacija dostupna na stranicama OpenAI-a [15].

```
import retrofit2

interface OpenAiApi {
    @Headers("Authorization: Bearer $MY_API_KEY")
    @POST("v1/completions")
    suspend fun getCompletions(@Body completionResponse:
CompletionRequest) : Response<CompletionResponse>
}
```

U sučelje OpenAiApi uvezena je biblioteka retrofit2, type-safe Http client biblioteka za Android i Javu. [16]. Kao header parametar proslijeđen je navedeni string tip podatka koji sadrži API ključ iz API_KEY objekta sa slike 10 dok post parametar prosljeđuje upit na određenu lokaciju. Kao posljednji parametar proslijeđen je CompletionRequest koji sadži naziv modela, upit i maksimalan broj tokena:

```
  val completionRequest = CompletionRequest(
          model = "text-davinci-003",
         prompt = question,
         max_tokens = 4000
)
```

Zatim je kreirana klasa ApiClient koja sadrži bazni url API servisa i OkHttpClient, ona koristi prethodno navedeno sučelje, pristupa servisu te dohvaća odgovor

```
import okhttp3.OkHttpClient
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

object ApiClient {
    private const val BASE_URL = "https://api.openai.com/"

    private val httpClient = OkHttpClient.Builder()
        .build()

    private val retrofit = Retrofit.Builder()
        .baseUrl(BASE_URL)
        .client(httpClient)
        .addConverterFactory(GsonConverterFactory.create())
        .build()

    val apiService : OpenAiApi = retrofit.create(OpenAiApi::class.java)
}
```

Pozivanje navedenog ostvareno je u klasi ChatGptViewModel vidljivoj na slici 10 na način da se pozivanje vrši putem metode callAPI():

```
try {
    val response =
ApiClient.apiService.getCompletions(completionRequest)
    handleApiResponse(response)
}catch (e: SocketTimeoutException){
    addResponse("Timeout :  $e")
}
```

Ukoliko aplikacija ne ostvari konekciju s API servisom funkcija javlja grešku, u drugom slučaju prosljeđuje dobiveni odgovor funkciji handleApiResponse() na daljnu obradu:

```
if (response.isSuccessful){
    response.body()?.let { completionResponse ->
        val result = completionResponse.choices.firstOrNull()?.text
        if (result != null){
            addResponse(result.trim())
        }else{
            addResponse("No choices found")
        }
    }
}else{
    addResponse("Failed to get response ${response.errorBody()}")
}
```

Kada funkcija uspješno dohvati odgovor, odabire se samo tijelo odgovora u kojemu se nalazi konkretna poruka dobivena od strane servisa. Poruka je „odgovor“ ChatGPT modela na prethodno poslan korisnički upit i onda je predmet daljnje obrade i pretvorbe teksta u govor. Ukoliko funkcija ne primi zadovoljavajući odgovor, javlja se greška.

- Nakon što uspješno dohvati odgovor, odnosno poruku odgovora funkcija addResponse() dodaje poruku u listu poruka dok funkcija addToChat() dodaje poruku u listu poruka koja također sadrži informacije o pošiljatelju te vrijeme zaprimanja poruke:

```
private fun addResponse(response : String){
   _messageList.value?.removeAt(_messageList.value?.size?.minus(1) ?: 0)
   addToChat(response,Message.SENT_BY_BOT,getCurrentTimestamp())
}
  fun addToChat(message : String , sentBy : String , timestamp:String){
    val currentList = _messageList.value ?: mutableListOf()
    currentList.add(Message(message,sentBy,timestamp))
    _messageList.postValue(currentList)
}
```

4.4.2 Postavljanje poruka na frontend

Nakon uspješnog slanja upita i zaprimanja poruke odgovora iste je potrebno proslijediti frontend dijelu aplikacije za koji dio je zadužena klasa MessageAdapter vidljiva na slici 10. Ova klasa ne sadrži biblioteke kao prethodno opisane klase. Ona služi kao adapter za obje vrste poruka za koje zatim provjerava vrstu „pošiljatelja“, ako je poruka zaprimljena od strane ChatGPT modela postavljaju ju kao takvu te prosljeđuje na tekstni okvir datoteke bot_chat.xml, odnosno ako je poruka zaprimljena od strane korisnika postavlja ju na tekstni okvir datoteke user_chat.xml. Navedene datoteke prikazane su u poglavlju 4.3. Prikaz koda pomoću kojega je navedeno realizirano prikazan je u nastavku:

```
inner class MessageViewHolder(itemView:View):RecyclerView.ViewHolder(itemView)
    {
    fun bind(message: Message) {
        when (message.sentBy) {
            Message.SENT_BY_ME -> {
                rightChatTextView?.text = message.message
                rightChatTimestamp?.text = message.timestamp
            }
            else -> {
                leftChatTextView?.text = message.message
                leftChatTimestamp?.text = message.timestamp
            }
        }
    }
```

Kako bi se omogućilo prosljeđivanje teksta u navedene okvire koristi se funkcija onCreateViewHolder() RecyclerView widgeta.[17]. RecyclerView odnosi se na jedan on mnogih view holder elemenata / biblioteka Android Studio razvojnog okruženja, koristi se u svrhu prikazivanja velikih količina podataka u listi, a kada lista elemenata postane duža od same visine zaslona RecyclerView ne uništava pogled već omogućuje korisniku scroll kroz listu elemenata.[18]. Svaki element liste definiran je putem view holder-a što u ovom slučaju znači da je svaka poruka koja se prikazuje zaseban view holder objekt. Dodavanje poruka u view holder objekte koji se dalje dodavaju na RecyclerView pomoću adaptera realiziran je prekoračenjem (override) RecyclerView.Adapter.onCreateViewHolder() funkcije na sljedeći način:

```
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):

MessageViewHolder {

   return when (viewType) {
        SENT_BY_ME -> {
            val view = LayoutInflater.from(parent.context)
                .inflate(R.layout.user_chat, parent, false)
            MessageViewHolder(view)

        }
        else -> {
            val view = LayoutInflater.from(parent.context)
                .inflate(R.layout.bot_chat, parent, false)

            MessageViewHolder(view)
        }
    }
}
```

U prikazanom kodu provjerava se izvor poruka (user ili bot) koje se dodavaju na view holder. Ažuriranje podataka, to jest, view holder objekata omogućeno je prekoračenjem (override) funkcije RecylcerView.Adapter.onBindViewHolder().[18]. Navedena funkcija kao parametre prima view holder objekte te indekse pozicije pomoću kojih se vrši ažuriranje same liste, to jest, RecyclerView widget-a:

```
override fun onBindViewHolder(holder: MessageViewHolder, position: Int) {

    val message = messageList[position]
    holder.bind(message)

}
```

Navedeni kod još uvijek ne šalje podatke na frontend dio aplikacije, ovaj proces vrši se u MainActivity klasi. Klasa MainActivity krucijalna je komponenta svake Android aplikacije, a način na koji se aktivnosti kreiraju i međusobno komuniciraju fundimentalan je dio svakog aplikacijskog modela. Tako MainActivity klasa kreira i pokreće aktivnost, odnosno pokreće aplikaciju pomoću metode koja se mora pozvati: onCreate(). Kada se pozove onCreate() funkcija, aktivnost se pokreće i ona je u stanju „Started“ te aplikacija postaje vidljiva krajnjem korisniku.[19].

Uz onCreate() funkciju ostale integrirane funkcije aktivnosti koje sistem poziva su:

- onStart() – izvodi se nakon onCreate(), sadrži finalnu pripremu nakon čega aplikacija postaje interaktivna
- onPause() – sistem poziva ovu funkciju kada aktivnost nije u fokusu te „pauzira“ aktivnost
- onResume() – sistem poziva ovu funkciju nakon što je pauzirana aktivnost ponovno pristupljena te ju ponovno postavlja u „Started“ stanje
- onStop() – sistem poziva ovu funkciju kada aktivnost više nije vidljiva korisniku, najčešće se poziva nakon rušenja aktivnosti
- onDestroy() – sistem poziva ovu funkciju kada se aktivnost uništava
- onRestart() - sistem poziva ovu funkciju kada se aktivnost treba vratiti u „Started“ stanje iz stanja „Stopped“ te vraća aktivnost na stanje prije stopiranja

MainActivity klasa sadrži onCreate() funkciju koja sadrži observer koji nadgleda listu poruka klase ChatGptViewModel, kada se dogodi promjena liste poziva se RecyclerView.Adapter klase MessageAdapter te na taj način poruke dohvaćene od strane API-a dolaze do korisnika aplikacije, odnosno postavljaju se na sučelje na sljedeći način:

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    _binding = ActivityMainBinding.inflate(layoutInflater)
    val binding = _binding.root
    setContentView(binding)

    chatGptViewModel = ViewModelProvider(this)[ChatGptViewModel::class.java]

    chatGptViewModel.messageList.observe(this){messages ->
        val adapter = MessageAdapter(messages, this, textToSpeech)
        _binding.recyclerView.adapter = adapter
    }
    _binding.sendBtn.setOnClickListener {
        val question = _binding.messageEditText.text.toString()
        chatGptViewModel.addToChat(question,
Message.SENT_BY_ME,chatGptViewModel.getCurrentTimestamp())
        _binding.messageEditText.setText("")
        chatGptViewModel.callApi(question)
    }
}
```

Kao što je prikazano u kodu, onCreate() sadrži i listener za gumb koji šalje upit API-u, ovdje je realizirano postavljanje korisničkih upita u korisničko sučelje.

4.4.3 Pretvorba teksta u govor

Nakon uspješnog dohvaćanja poruke i postavljanja iste na grafičko sučelje aplikacije, istu je potrebno pretvoriti u govor. Uvozom biblioteke android.speech.tts.TextToSpeech u MainActivity klasu potrebno je inicijalizirati TTS instancu.[10].

```
textToSpeech = TextToSpeech(this){ status->

    if(status == TextToSpeech.SUCCESS){
        val result = textToSpeech.setLanguage(Locale.getDefault())

        if(result == TextToSpeech.LANG_COUNTRY_AVAILABLE || result ==
TextToSpeech.LANG_NOT_SUPPORTED){

            Toast.makeText(this, "Language not supported",
Toast.LENGTH_SHORT).show()
        }
    }
}
```

U prikazanom dijelu koda inicijalizirana je TTS instanca te su postavljene obvezne provjere nakon kojih je moguće koristiti biblioteku, prva provjera odnosi se na provjeru statusa inicijalizacije, ukoliko je inicijalizacija uspješno provedena prelazi se na sljedeću provjeru. Iduća bitna provjera je dostupnost postavljenog jezika, za potrebe aplikacije se postavlja zadan jezik za podneblje u kojemu se nalazi uređaj putem kojega se pristupa aplikaciji. Ukoliko jezik nije dostupan, korisnik dobiva obavijest o grešci: „Language not supported“. Sama pretvorba teksta u govor realizira se putem speak() metode koja se poziva na instanciranim objektom TextToSpeech klase:

```
textToSpeech.speak(messageList[messageList.size-1].message,
TextToSpeech.QUEUE_FLUSH)
```

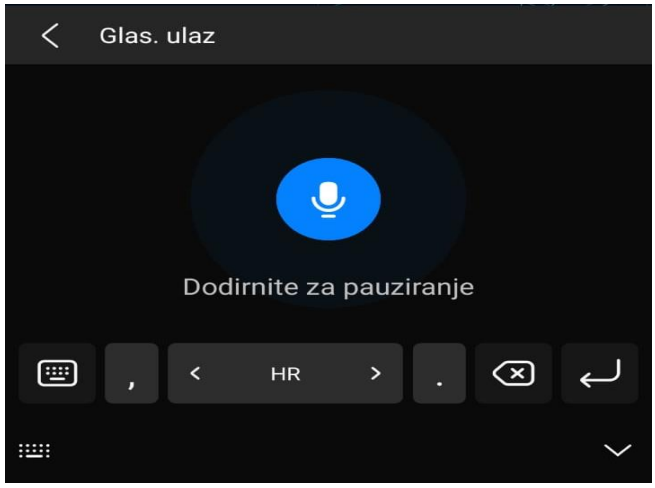
Kao ulazne parametre ova metoda prima:

- Tekst – tekst koji se pretvara u govor, u ovom slučaju to je poruka dobivena od strane ChatGPT modela
- Queue mode – način na koji će se popunjavati lista tekstova koji se pretvaraju u govor, za potrebe aplikacije odabran je QUEUE_FLUSH način tako se prethodno pretvoreni tekstovi neće ponavljati prilikom svake iduće pretvorbe

4.4.4 Pretvorba govora u tekst

Uzevši u obzir cijenu dostupnih alata koji omogućuju pretvorbu govora u tekst kao što je whisper.io alat tvrtke OpenAI te mogućnosti i performanse besplatnih STT servisa kao alat za pretvorbu prilikom izrade aplikacije koristila se integrirana funkcija pretvorbe govora u tekst Android operacijskog sustava koja je sastavni dio Android tipkovnice. Ponekad može biti teško koristiti tipkovnicu uređaja, na primjer, ako koristimo rukavice, ukoliko je tipkovnica mala ili ako unosimo velike količine teksta.[20]. Na ovaj način moguće je glasovno unositi upite u samu apliakciju. Prilikom unosa teksta upita u aplikaciji otvara se Android tipkovnica te je u gornjem meniju vidljiva ikona mikrofona, pritiskom na ovu ikonu uređaj prihvaća glasovni ulaz koji pretvara u tekst te ga postavlja u polje za slanje upita:

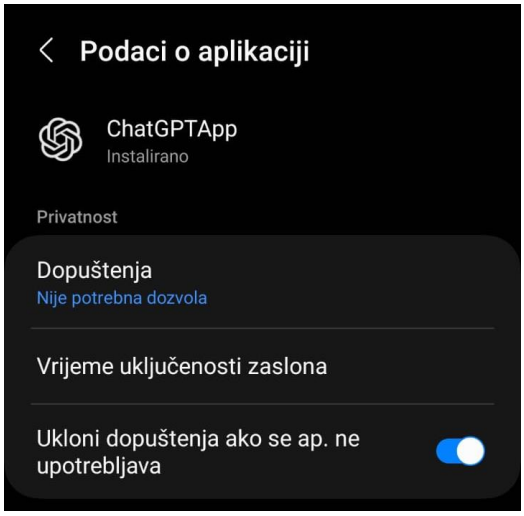
Slika 11 Prikaz Android tipkovnice



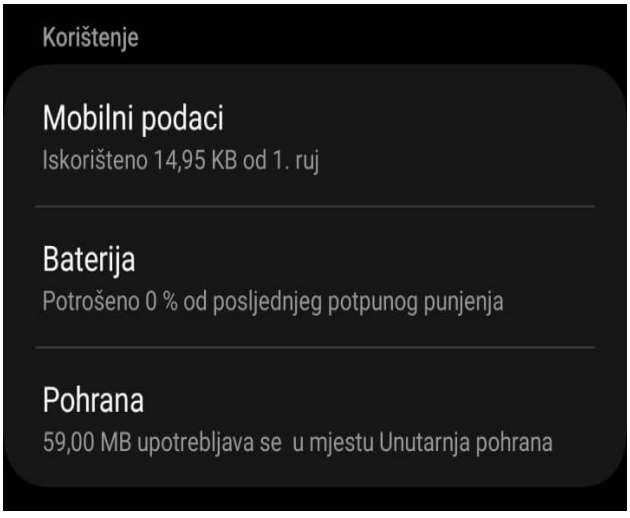
Slika 12 Prikaz Android tipkovnice nakon pritiska na ikonu mikrofona

5. Pregled mogućnosti aplikacije

Izrađena aplikacija koristi jednu aktivnost koja upravlja svim procesima iste. Ne zahtjeva dozvole od strane korisnika jer ne pristupa komponentama uređaja i operacijskog sustava za koje su dozvole potrebne.



Slika 13 Podaci o aplikaciji

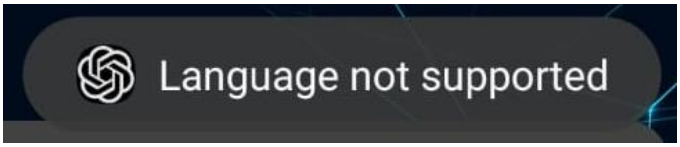


Slika 14 Prikaz potrošnje podataka, baterije te pohrane

Od datuma 1. rujna 2023. (najnovija verzija aplikacije) do sada ukupna količina iskorištenih mobilnih podataka iznosi 14.95 KB kako se radi o vrlo malim količinama podataka jer je u pitanju samo tekst kako se TTS i STT provodi direktno na uređaju te za te funkcije nije potrebna internetska veza. Aplikacija koristi minimalan broj pozadinskih resursa, s toga potrošnja baterije skoro nije mjerljiva, također, tamna tema doprinosi istom zbog smanjene svjetline zaslona. Veličina same aplikacije iznosi 59.00 MB dok je .apk datoteka aplikacije veličine 8.97 MB.

5.1 Ikona i naziv aplikacije

Naziv aplikacije je „PromptCommunicator“, a kao ikona aplikacije koristi se tamna verzija logotipa ChatGPT modela preuzeta s OpenAI web stranice.[5]. Iz preuzete ikone kreirane su .webp i .xml datoteke putem Android Studio razvojnog okruženja gdje su iste postavljene kao ic_launcher datoteke koje se dalje koriste za prikaz ikone aplikacije te kao ikona u Android porukama zvanim „Toast“ kojima aplikacija obavještava korisnika o trenutnom statusu neke operacije. Toast može biti kratkog trajanja (2 sekunde) i dugog trajanja (3.5 sekunde), prikazuje do dvije linije koda i zauzima onoliko mjesta koliko je potrebno za prikaz teksta koji sadrži.[21].Toast poruke u aplikaciji izgledaju ovako:



Slika 15 Toast poruka aplikacije

Ikona aplikacije u Android operacijskom sustavu prikazuje se na sljedeći način:

Slika 16 Izgled ikone aplikacije na Android uređaju

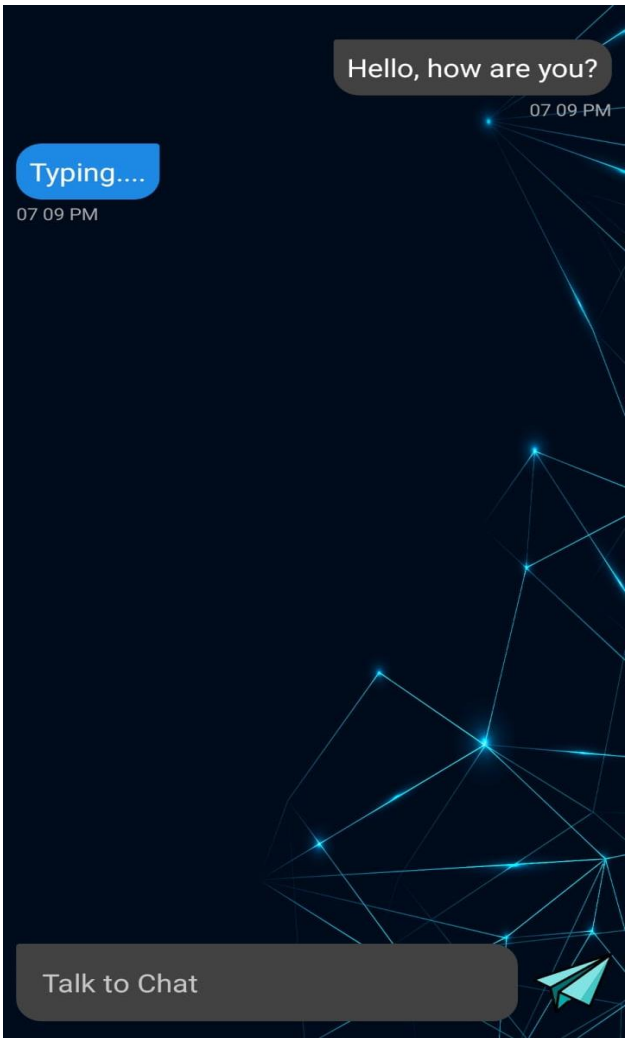
5.2 Izgled aplikacije

Slika 17 Izgled aplikacije

Aplikacija koristi tamnu pozadinsku temu i svijetlu boju fonta, u donjem dijelu aplikacije nalazi se tekstni okvir. Dodirom na navedeni okvir na uređaju se otvara tipkovnica gdje je korisniku omogućen unos teksta, bilo to ručno ili glasovno. Tekstualni ulaz prosljeđuje se na backend dio aplikacije pritiskom na ikonu papirnatog zrakoplova koja simbolizira slanje poruke. Pozadinski procesi obavljaju se bez da korisnik za njih zna. Jedina povratna informacija su „Toast“ poruke kojima se korisniku daje obavijest o stanju procesa. Obično je to greška ili uspjeh procesa.

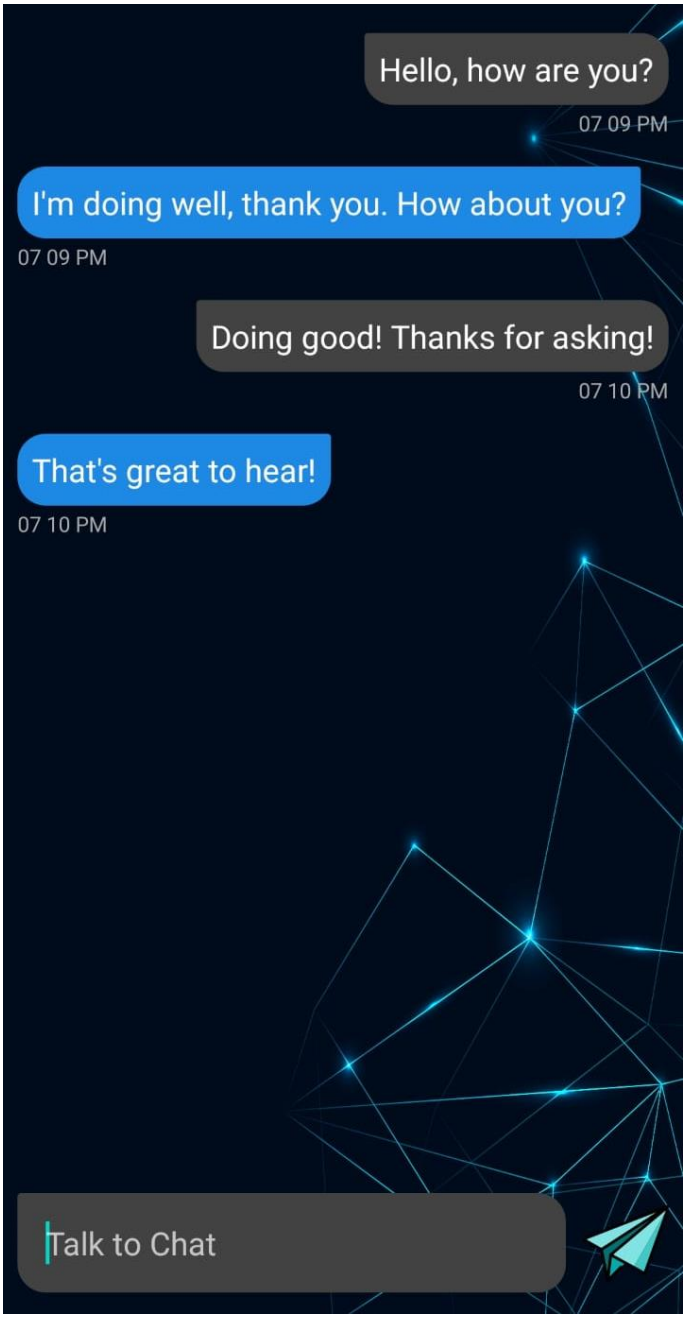
5.2.1 Slanje upita

Slanje upita vrši se na način opisan u prethodnom poglavlju, kada je upit poslan, a aplikacija uspješno uspostavi komunikaciju s API servisom, korisnički upit postavlja se u tekstni okvir na zaslonu nakon kojega se postavlja i dohvaćeni odgovor dobiven od strane servisa (za vrijeme dohvaćanja odgovora tekstni okvir prikazuje vrijednost „Typing….“ kao obavijest da se odgovor trenutno ispisuje:



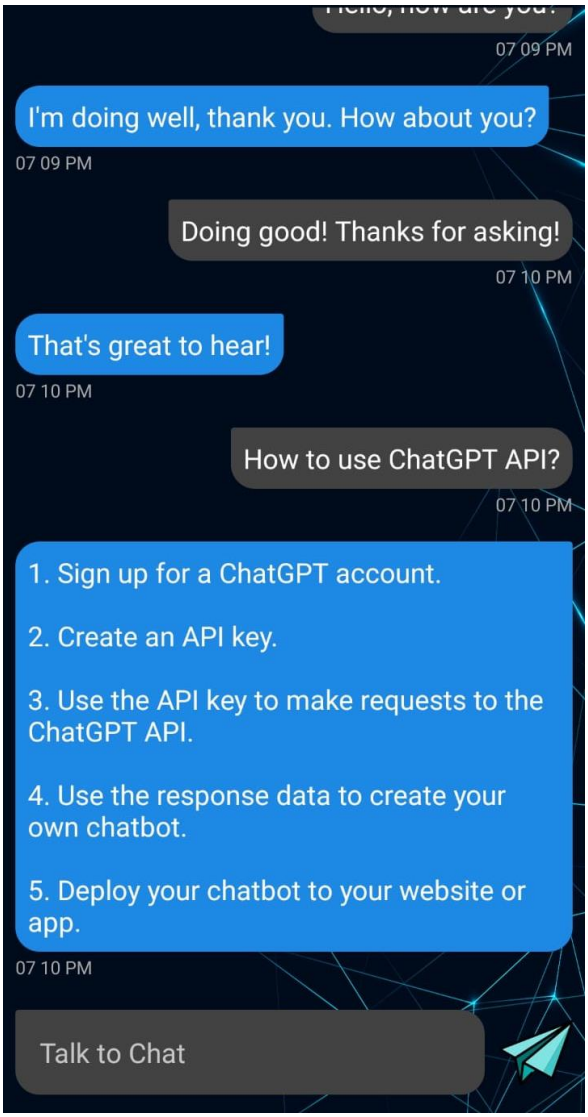
Slika 18 Slanje upita

Nakon uspješnog dohvaćanja i ispisa odgovora aplikacija on zamjenjuje inicijalnu poruku „Typing….“ kao što je vidljivo na niže navedenoj slici :



Slika 19Prikaz upita i odgovora u aplikaciji

Ukoliko količina prikazanih podataka (upita i odgovora) prelazi visinu zaslona, korisniku se omogućuje scroll na zaslonu (prelaskom prsta od dolje prema gore ili obrnuto) kako bi se prikazali podatci liste koji zbog veličine ne mogu stati na zaslon:



Slika 20 Prikaz veće količine podataka u aplikaciji

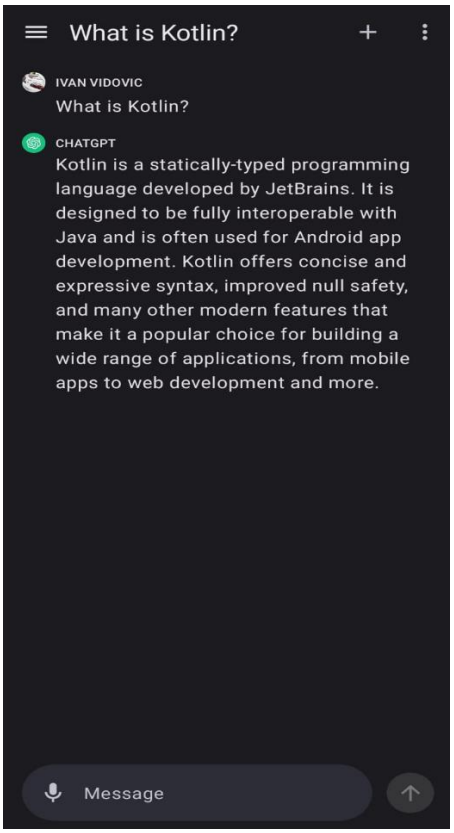
Prilikom interakcije s aplikacijom svaki odgovor generiran i dohvaćen od strane ChatGPT API servisa glasovno se reproducira korisniku putem TTS biblioteke gdje se u paru s STT servisom stiče dojam stvarnog ljudskog dijaloga.

5.3 Rasprava

U ovom poglavlju obavljena je usporedba izrađene aplikacije završnog projekta s mobilnom aplikacijom OpenAI tvrtke koja je objavljena u kolovozu 2023. godine kada je ova aplikacija bila izrađena. Također, izneseni su prijedlozi za daljnju nadogradnju i načini na koji se aplikacija može unaprijediti. Kao poboljšanje predloženo je mogućnosti povezivanja na bazu podatka gdje bi se zapisivala i čitala povijest razgovora, implementacija preciznijih TTS i STT modela većih mogućnosti te nadogradnja na noviji AI model GPT-4.

5.3.1 Usporedba aplikacija

Prilikom usporedbe aplikacije koja je dio ovog završnog rada i ChatGPT aplikacije tvrtke OpenAI uzimaju se u obzir fronted rješenje te funkcionalnost same aplikacije. Backend dio aplikacije nije predmet usporedbe kako kod ChatGPT aplikacije nije javno dostupan. Počevši od dizajna aplikacije, možemo zaključiti kako obje aplikacije koriste sličnu temu, koristi se tamna pozadina uz svijetlu boju fonta. U donjem dijelu zaslona, obje aplikacije sadrže polje za tekstualni unos upita te u donjem desnom kutu imaju gumb za slanje istog. Prva aplikacija koristi ikonu papirnatog zrakoplova dok ChatGPT aplikacija koristi ikonu strelice usmjerene prema vrhu ekrana. Prva bitna razlike je položaj poslanog upita i dobivenog odgovora na zaslonu. U prvoj aplikaciji upiti od strane korisnika poravnati u odnosu na desnu stranu zaslona, dok su odgovori API-a poravnati u odnosu na lijevu stranu zaslona, sadrže vremenske oznake te se razlikuju po boji okvira u kojemu se nalaze (korisnički upit – siva boja), (ChatGPT odgovor – plava boja). ChatGPT aplikacija sve poruke postavlja poravnate uz lijevu stranu zaslona, ne postoje vremenske oznake, a način na koji je moguće razlikovati korisničke upite od odgovora modela realizirano je dodavanjem korisničkog imena uz korisnikovu profilnu sliku iznad upita te naslova „CHATGPT“ i ikone koja sadrži logotip ChatGPT modela iznad odgovora GPT modela. Druga bitna razlika je postojanje navigacijske trake u aplikaciji OpenAI-a koja sadrži naslov razgovora baziran na prvom korisničkom upitu te hamburger meni u kojemu je dostupan pregled povijesti razgovora. Na slici 21 navedenoj ispod vidljiv je dizajn ChatGPT aplikacije:



Slika 21 Izgled ChatGPT aplikacije

Nakon što su navedene vizualne razlike, bitno je navesti funkcionalne razlike aplikacija. Obje aplikacije služe istoj svrsi, odnosno ostvaraju komunikaciju između krajnjeg korisnika i GPT modela slanjem upita i primanjem odgovora. Aplikacije se ne razlikuju u brzini slanja i dobivanja odgovora, a bitne funkcionalne razlike odnose se na TTS i STT komponente. ChatGPT aplikacija koristi vlastiti model STT komponente koja je preciznija i može prepoznati jezike uz manju brzinu pretvorbe, nasuprot tomu, aplikacija koja je predmet završnog rada koristi Android STT komponentu koja je manje precizna te nema mogućnost prepoznavanja jezika, ali je daleko brža u samoj pretvorbi govora u tekst. Najveća funkcionalna razlika ove dvije aplikacije jest u TTS komponentama. Naime, ChatGPT aplikacija ne pruža mogućnost pretvorbe odgovora dobivenih komunikacijom s modelom u govor, dok aplikacija završnog rada pruža navedenu mogućnost koristeći STT biblioteku integriranu u Android Studio razvojno okruženje.

5.3.2 Nadogradnja

Prilikom usporedbe funkcionalnog dijela aplikacija, vidljiva je mogućnost nadogradnje vezana za STT komponentu. Iako je trenutno rješenje brže od rješenja implementiranog u ChatGPT aplikaciji, ono je daleko manje funkcionalno te bi nadogradnja, odnosno korištenje whisper.io API koje se koristi u ChatGPT aplikaciji bilo bolja solucija. Bitno je napomenuti kako je trenutno rješenje besplatno, dok se korištenje whisper.io API-a naplaćuje, svaka minuta iznosi $0.006.[22]. Kod vizualnog dijela aplikacije, moguća je nadogradnja u obliku navigacijske trake koja postoji u ChatGPT aplikaciji, u samom kodu aplikacije postavljen je „kostur“ za istu koja bi sadržavala naslov razgovora, gumbe za brisanje i kreiranje novog razgovora te kada bi se aplikacija povezala s online bazom podatka ostvarila bi se mogućnost prikaza povijesti razgovora. Iduća bitna mogućnost daljnje nadogradnje je promjena GPT modela s kojim aplikacija trenutno komunicira, trenutno rješenje pruža brze i konkretne odgovore, ali noviji GPT-4 model koristi novije podatke, odgovori su kreativniji u smislu kreiranja realnijeg dijaloga nalik ljudskom. GPT-4 model također nudi mogućnost primanja slikovnih upita koje ovaj model može obrađivati. Nadogradnja na noviji model zahtjeva minimalnu izmjenu koda, dok bi u izgledu aplikacije jedina promjena bila gumb za dodavanje slike koji bi se nalazio uz tekstualni okvir za slanje upita. Međutim, kao i kod nadogradnje STT komponente i ova bi nadogradnja utjecala na troškove izrade aplikacije kako se trenutni model u odnos na GPT-4 model višestruko razlikuje u cijeni. Cijena trenutnog GPT modela iznosi približno $0.001 za 1000 tokena, dok GPT-4 cijena doseže iznos od približno $0.05 za 1000 tokena. Detaljan prikaz cijena nalazi se u poglavlju 2.3.

6. Zaključak

U ovom radu opisana je izrada cjelokupne Android mobilne aplikacije koristeći Android Studio IDE, Kotlin programski jezik, različite tipove biblioteka te TTS i STT komponente. Svaka od navedenih stavki detaljno je opisana te je naveden način na koji je svaka implementirana. Izrada rješenja crpila je vremensko opterećenje od približno 5 radnih dana u koje ulazi istraživanje dostupnih alata, kreiranje dizajna aplikacije te izrada backend dijela aplikacije uz stalno testiranje funkcionalnosti kao i savladavanje Kotlin programskog jezika koje je olakšano iz razloga što je isti sličan Java programskom jeziku. Prije početka izrade same aplikacije nije postojalo službeno rješenje mobilne aplikacije od strane tvrtke OpenAI čije se API usluge koriste u ovom radu. Službena aplikacija ChatGPT publicirana je nakon završetka izrade aplikacije koja je dio ovog završnog rada, a s kojom je na kraju izvršena usporedba. Uz službenu aplikaciju, u to vrijeme postaju dostupne razne aplikacije fokusirane na rješenje istog problema, međutim svaka od navedenih aplikacija sadrži neke bitne razlike. Navedeno ukazuje na rapidan razvoj umjetne inteligencije i raznih chat modela koji koriste istu kao i na razvoj aplikativnih rješenja od strane programera što treba uzeti u obzir u svakom aspektu razvoja aplikacija, ne samo kod AI chat modela. U poglavlju 5.3.1 izvršena je usporedba aplikacija pri čemu je zaključeno kako se vizualni dio ne razlikuje u bitnim aspektima dizajna, dok za funkcionalni dio postoje dvije bitne razlike. Naime, aplikacija završnog rada sadrži TTS komponentu koja nedostaje u aplikaciji tvrtke OpenAI, međutim ona sadrži STT komponentu putem whisper.io modela koji je daleko napredniji model od rješenje integriranog u Android tipkovnici koje je besplatno dok se korištenje whisper.io API-a naplaćuje. Nadalje, otvorene su mogućnosti daljnje nadogradnje aplikativnog rješenja u poglavlju 5.3.2 gdje je raspravljena mogućnost povezivanja s online bazom podataka kako bi se ostvarila mogućnost prikaza povijesti razgovora, otvorena je mogućnost implementacije whisper.io modela te mogućnost implementacije navigacijske trake. Sve navedene mogućnosti doprinose značajne promjene funkcionalnom dijelu aplikacije, kao također i vizualnom dijelu aplikacije te se iste trebaju implementirati ukoliko bi došlo do potencijalnog komercijaliziranja aplikacije. Nakon što je aplikacija izrađena, korisniku je omogućen stalan pristup API modelu dok god je isti dostupan od strane pružatelja usluga, bez obzira na dostupnost modela u web aplikaciji što također doprinosi poboljšanoj komunikaciji korisnika i modela.

Zaključno s navedenim, moje osobno mišljenje je da razvoj AI alata ne treba shvaćati kao potencijalnu prijetnju već moramo naučiti koristiti iste u svrhu olakšavanja rješenja različitih problema s kojima se susrećemo bilo to implementacijom alata u aplikacije ili samom interakcijom s AI modelima koji pružaju značajno unaprjeđenje prilikom obavljanja bilo koje vrste rada. Na ovaj način svatko od nas može smanjiti količinu vremenskog opterećenja prilikom ostvarivanja rješenja za određeni problem.

7. LITERATURA

[1] Google for Developers., „Meet Android Studio“, *developer.android.com* [Online]

Dostupno: https://developer.android.com/studio/intro [Pristupljeno: 9.10.2023.]

[2] Kotlin., „Kotlin docs“, *kotlinlang.org* [Online]

Dostupno: https://kotlinlang.org/docs/home.html [Pristupljeno: 9.10.2023.]

[3] @GeeksForGeeks., „Introduction to Object Oriented Programming“, *geeksforgeeks.org* [Online]

Dostupno: https://www.geeksforgeeks.org/introduction-of-object-oriented-programming/ [Pristupljeno: 9.10.2023]

[4] OpenAI., „About“, *openai.com* [Online]

Dostupno: https://openai.com/about [Pristupljeno: 9.10.2023.]

[5] OpenAI., „Introducing ChatGPT“, *openai.com* [Online]

Dostupno: https://openai.com/blog/chatgpt [Pristupljeno: 9.10.2023.]

[6] LinkedIn., „TTS and STT“, *linkedin.com* [Online]

Dostupno: https://www.linkedin.com/pulse/tts-stt-arshitha-suresh [Pristupljeno: 9.10.2023.]

[7] CFBlog., „What is a programming Library? A Beginner's Guide“, *careerfoundry.com* [Online]

Dostupno: https://careerfoundry.com/en/blog/web-development/programming-library-guide/ [Pristupljeno: 9.10.2023.]

[8] Statista., „Global market share held by mobile operating systems from 2009 to 2023, by quarter“, *statista.com* [Online]

Dostupno: https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/ [Pristupljeno: 9.10.2023.]

[9] TechnologyAdvice., „11 Best Android IDEs for Developers of 2022“, *developer.com* [Online]

Dostupno: https://www.developer.com/mobile/top-android-ides-for-developers/ [Pristupljeno: 9.10.2023.]

[10] Google for Developers., „Text to Speech“, *developer.android.com* [Online]

Dostupno: https://developer.android.com/reference/android/speech/tts/TextToSpeech?authuser=1 [Pristupljeno: 9.10.2023.]

[11] Block, Inc., „OkHttp“, *square.github.io* [Online]

Dostupno: https://square.github.io/okhttp/ [Pristupljeno: 9.10.2023.]

[12] @GeeksForGeeks., „Frontend vs Backend“, *geeksforgeeks.org* [Online]

Dostupno: https://www.geeksforgeeks.org/frontend-vs-backend/ [Pristupljeno: 9.10.2023]

[13] RoadMap.sh., „Backend Developer“, *roadmap.sh* [Online]

Dostupno: https://roadmap.sh/backend [Pristupljeno: 9.10.2023.]

[14] DDI Development., „Backend Development in 2020: key languages, technologies, features“, *ddi-dev.com* [Online]

Dostupno: https://ddi-dev.com/blog/programming/backend-development-key-languages-technologies-features-in-2020/ [Pristupljeno: 9.10.2023.]

[15] OpenAI., „Making requests“, *platform.openai.com* [Online]

Dostupno: https://platform.openai.com/docs/api-reference/making-requests [Pristupljeno: 9.10.2023.]

[16] Block, Inc., „Retrofit“, *square.github.io* [Online]

Dostupno: https://square.github.io/retrofit/ [Pristupljeno: 9.10.2023.]

[17] Google for Developers., „RecyclerView.Adapter“, *developer.android.com* [Online]

Dostupno: https://developer.android.com/reference/androidx/recyclerview/widget/RecyclerView.Adapter?authuser=1 [Pristupljeno: 10.10.2023]

[18] Google for Developers., „Create dynamic lists with RecyclerView“, *developer.android.com* [Online]

Dostupno: https://developer.android.com/develop/ui/views/layout/recyclerview [Pristupljeno: 10.10.2023.]

[19] Google for Developers., „Introduction to activites“, *developer.android.com* m [Online]

Dostupno:https://developer.android.com/guide/components/activities/intro-activities [Pristupljeno: 10.10.2023.]

[20] Authority Media., „How to use voice to text on Android“, *androidauthority.com* [Online]

Dostupno:https://www.androidauthority.com/how-to-use-voice-to-text-on-android-1195895/ [Pristupljeno: 10.10.2023.]

[21] Google for Developers., „Toast overview“, *developer.android.com* [Online]

Dostupno:https://developer.android.com/guide/topics/ui/notifiers/toasts [Pristupljeno: 10.10.2023.]

[22] OpenAI., „Pricing“, *openai.com* [Online]

Dostupno: https://openai.com/pricing [Pristupljeno: 10.10.2023.]

[23] Exploding Topics., „57+ Amazing AI Statistics (2023)“, *explodingtopics.com* [Online]

Dostupno: https://explodingtopics.com/blog/ai-statistics [Pristupljeno: 10.10.2023.]

8. POPIS SLIKA I TABLICA