TARTU ÜLIKOOL

Arvutiteaduse instituut

Informaatika õppekava

Sander Kangur

Mobiilirakendus allergeenide tuvastamiseks toidupakendilt tekstituvastuse abil

Bakalaureusetöö (9 EAP)

Juhendaja: Jakob Mass, MSc

Tartu 2021

Mobiilirakendus allergeenide tuvastamiseks toidupakendilt tekstituvastuse abil

Lühikokkuvõte:

Toiduallergia esineb üha rohkematel inimestel. Seega on rohkem allergikuid, kes peavad valima, mida nad söövad. Toidupakendite etiketid ei vasta alati normidele, mistõttu võib nendest vajaliku info filtreerimine olla aeganõudev protsess. Probleemi lahendamiseks arendati eesti keeles kasutatav mobiilirakendus Scannit, mis kontrollib tekstituvastuse abil allergeenide sisaldust toidus. Kasutaja skaneerib pakendi etiketti ja saab personaalset tagasisidet oma allergeenide kohta. Rakenduse kvaliteedi hindamiseks viidi läbi kasutajatega testimine. Üldiselt jäid testijad rakendusega rahule. Katsed näitasid, et koostisosade skaneerimine toimis paremini kui nende käsitsi lugemine. Peale testimist likvideeriti katsealuste väljatoodud puudused.

Võtmesõnad:

Mobiilirakendus, multiplatvormarendus, allergeenid, tekstituvastus, Flutter, ML Kit

CERCS: P175 Informaatika, süsteemiteooria

Mobile application for recognizing allergens on food packages with text recognition

Abstract:

The prevalence of food allergy keeps on rising. Therefore, more allergic people are trying to figure out, what they are allowed to eat. Food packaging labels do not always meet the standards, which makes finding necessary information a tedious task. To alleviate the problem, the mobile application Scannit, which works in Estonian, was developed. The app uses text recognition to check whether the chosen food contains a certain allergen or not. The user scans the label of a food package and receives personal feedback about their allergens. User testing was performed to assess the quality of the mobile application. Overall the users were satisfied with the application. The experiments concluded that scanning the label performed better than manually reading it. After testing, the main deficiencies pointed out by the subjects were removed.

Keywords:

Mobile application, cross-platform development, allergens, text recognition, Flutter, ML Kit

CERCS: P175 Informatics, systems theory

Sisukord

Sissejuhatus

Innovatsioon ja nutikad lahendused on lahutamatu osa inimkonna igapäevaelust. Tehnika kiire areng soodustab erinevate valdkondade integratsiooni - mis varasemalt võis tunduda võimatuna, on nüüdseks mõne nupuvajutuse kaugusel. R. Peters jt [1] on kirjutanud, et linnastumine ja inimkonna areng on toonud endaga kaasa toiduallergikute kiire kasvu. Inimestel, eriti heaoluühiskonnas, on hakanud rohkem esinema toiduallergiaid. Autorid kirjutavad, et toiduallergia on seisund, kus immuunsüsteem reageerib mõnele toidule ebatavaliselt. S.H. Sicherer leidis erinevate uuringute põhjal 2010. aastal [2], et Ameerika Ühendriikides on umbes kolmel protsendil elanikkonnast toiduallergia. A. Muraro jt [3] on toonud välja, et ligikaudu kahel protsendil täiskasvanutest esineb toiduallergia Euroopas 2012. aasta seisuga. Toiduallergikute arvu kiire kasvu tõttu on üha olulisem toidupakendite etikettide täpsus. Allergeenide vältimiseks tuleb kontrollida pakendil olevaid koostisosasid.

Etiketi märgistamisega seotud probleeme on uurinud mitmed autorid. Cornelisse-Vermaat jt. [4] leiavad oma uurimistöös, et toidupakendi üheks tavaliseks probleemiks on halvasti vormindatud ja liigne informatsioon. Selle filtreerimine ja enda jaoks vajaliku teabe leidmine võtab liiga kaua aega. Nad kirjutavad, et väikese kirja suuruse, ebaselge kirjatüübi või vale värvi tõttu võib mõni ohtlik või ebatervislik koostisosa märkamata jääda. Preeti jt [5] lisavad, et etiketid on tihti ebamäärased (nt „võib sisaldada maapähklit“) või nõuavad tugevaid teadmisi allergeenide valdkonnast (nt kasutatakse sõna „piim“ asemel „kaseiin“, mis on piimas leiduv valk). Mõlemas uuringus leiti, et toiduallergiatega tarbijatele on tähtis, et selliste disainivigade tõttu ei satuks ostukorvi allergiat tekitav toiduaine.

Kirjeldatud probleemide leevendamiseks ja vajaliku info kiiremaks filtreerimiseks võib rakendada optilist märgituvastust (ingl *Optical Character Recognition,* lüh *OCR*) ehk automaatset tekstituvastust. OCR töötab masinõppe (ingl *Machine Learning*) mudelist saadud andmete ja reeglite põhjal [6]. Turul on mitmeid lahendusi [7, 8, 9, 10, 11], mis kasutavad tekstituvastust toidu koostisosade välja selgitamiseks. Mainitud lahendused ei tööta eestikeelsete etikettidega. Peale tekstituvastuse kasutavad osad rakendused [7, 9, 12] veel triipkoodi skaneerimist. Rakenduste kasutamiseks peab inimesel olema kaameraga nutitelefon, mis tal üldjuhul on poes kaasas.

Käesoleva töö eesmärk on luua mobiilirakendus (nimega Scannit), millega saab toidupakendilt allergeenide sisaldust kontrollida märgituvastuse abil. Rakendus on arendatud multiplatvorm

arenduskeskkonnas Flutter, kus ühe koodibaasiga saab arendada korraga nii Android kui ka iOS äppe. Lõputöö raames keskenduti Android versiooni arendamisele ja testimisele. Lahendus kasutab tekstituvastuseks Firebase ML Kit masinõppe tööriistu. On tähtis, et mobiilirakendus oleks võimalikult lihtsasti kasutatav ja kiire, et minimeerida ajakulu poes. Töö käigus valminud rakendus on võimeline eristama iseseisvalt sõnu ning teavitab inimest, kas skaneeritud toit on allergikule ohutu. Kasutaja saab ise lisada soovimatu toiduaine või valida enimlevinud allergeenide seast. Scannit pakub isiklikku tagasisidet vastavalt kasutaja tehtud valikutele. Äpp kontrollib skaneeritud teksti nende ainete vastu eesti keeles. Mobiilirakendus töötab võrguühenduseta. Lahenduse kvaliteedi hindamiseks viidi läbi katsed, kus katsealused said testida rakendust erinevate pakendite peal.

Bakalaureusetöö on jaotatud kolmeks suuremaks peatükiks. Esimeses peatükis antakse ülevaade toiduallergiast, tekstituvastusest ja multiplatvormarendusest. Samuti on välja toodud arenduses kasutatud tehnoloogiad ning esitatakse sarnaste olemasolevate rakenduste ülevaade ja võrdlus. Teises peatükis kirjeldatakse arendusprotsessi ning rakenduse arhitektuuri. Kolmandas peatükis antakse ülevaade rakenduse testimisest ja järeldustest.

1. Taust

Järgnevalt antakse ülevaade uurimistööga seotud teemadest. Seletatakse lahti toiduallergia, tekstituvastus ja multiplatvormarendus. Samuti tuuakse välja kasutatud tehnoloogiad, sh Flutter, Dart ja ML Kit. Peatüki lõpus tutvustatakse olemasolevaid mobiilirakendusi allergeenide kontrollimiseks toidupakendilt.

1.1 Toiduallergia

Proviisor A. Teder on kirjutanud [13], et toiduallergiaks peetakse organismi ebaharilikku reaktsiooni mingile toiduainele. Enamasti on allergia vallandajaks allergeenis leiduv valk. Autori sõnul ei saa toiduallergiat välja ravida, aga on võimalik sümptomeid leevendada.

Alljärgnev materjal on refereeritud S. Rameshi ülevaate-artiklist [14]. Autori sõnul on terminit „toiduallergia“ raske piiritleda. Enamik reaktsioone toidule on tingitud toidutalumatusest mitte toiduallergiast. Ta on jaganud allergilised reaktsioonid vastavalt põhjustajatele kaheks. Immunoglobiin E (lüh *IgE*) antikehade põhjustatud reaktsioonid avalduvad organismis koheselt anafülaktilise šoki, oksendamise või nahaärrituse (nt kublad, lööve, punetus) kujul. Mitte immunoglobiin E (lüh *non-IgE*) antikehade põhjustatud reaktsioonid ei avaldu kohe ning on seotud peamiselt seedetrakti häiretega (nt kõhulahtisus, puhitus, oksendamine). Ramesh leiab oma artiklis, et paljud inimesed kasvavad lapsena põetud allergiatest välja.

R. Peters jt [1] on välja toonud, et toiduallergia tekke põhjus organismis on endiselt spekulatiivne, aga levinud teooriaks peetakse ekseemi põdemist noores eas. Sellisel juhul pole toidu suhtes tolerantsi tekkinud oraalsel teel ning esmane kokkupuude toidus oleva valguga toimub naha kaudu, mis omakorda vallandab immuunsüsteemi kaitserefleksi ja paneb aluse allergiale. Autorite sõnul võib alternatiivideks pidada keskkonnategureid ja geneetilist soodumust.

Põllumajandus- ja Toiduamet on välja toonud põhilised toidus sisalduvad allergeenid [15]: gluteen, piim, muna, kala, koorikloomad, molluskid, maapähklid, sojaoad, pähklid, seller, sinep, seesamiseemned, lupiin, vääveldioksiid ja sulfitid (lisaained E220 – E228). Mainitud ained tuleb selgelt eristada ülejäänud koostisosadest toidupakendil, mida alati ei tehta.

Internetis on olemas andmebaas, Open Food Facts (lüh *OFF*), erinevate toitude koostisosade jaoks. Järgnev materjal on refereeritud OFF andmebaasi kodulehelt [16]. Andmebaasi saavad

kõik tasuta kasutada ja oma toodetega uuendada. Hetkeseisuga on andmebaasis üle 1.7 miljoni toote, millest ligikaudu 800 000 on salvestatud Prantsusmaal, 350 000 USA-s ja 200 000 Hispaanias. Eestis on ainult 74 toodet salvestatud. Samas leidub hulganisti tooteid, mida müüakse erinevates riikides sama triipkoodiga (nt Nutella, Coca-Cola). Sellest võib järeldada, et Eestis pole rakendus veel iga päeva kasutajale mõistlik, sest peab tõenäoliselt ise oma tooted lisama, aga Prantsusmaal on mingil määral kasutatav juba. Andmebaasis on kirjas toidule vastav NOVA grupp ja Nutri-Skoor (ingl *Nutri-Score*).

C. Monteiro jt [17] on kirjutanud, et NOVA klassifikatsiooni järgi jaotatakse toit nelja gruppi selle põhjal, kuidas ja kui palju seda töödeldud on:

1. Esimene grupp on mittetöödeldud toit, mis kujutab endast naturaalset toitu, sh munad, seemned, piim, puuviljad jne.
2. Teise grupi moodustavad kulinaarsed koostisosad, mille alla kuulub või, õli, suhkur, sool jne.
3. Kolmas grupp on töödeldud toit, sh konservid, juust, leib jne.
4. Neljanda grupi moodustab kõrgrafineeritud (ingl *ultra-processed*) toit, mis hõlmab karastusjooke, sügavkülmutatud valmistoite, kummikomme jne.

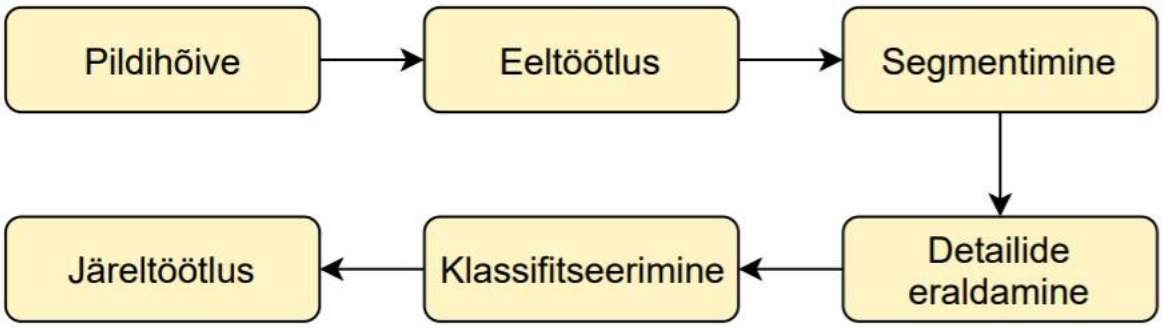
C. Julia ja S. Hercberg on kirjutanud [18], et Nutri-Skoor on hindamissüsteem toiteväärtuse jaoks. Süsteemis on 5 värvi ja tähed A kuni E. Süsteemis hinnatakse eraldi toiteaineid (suhkrud, rasvad, lipiidid jne) ning energiasisaldust. Seejärel arvutatakse punktide põhjal välja, millisesse kategooriasse toit kuulub. Aastal 2017 hakkas Prantsusmaa seda pakenditel kasutama ning nüüdseks on see kogunud populaarsust ka mujal maailmas.

Kui Prantsusmaal ja Hispaanias on Nutri-Skoor juba riiklikult kasutusel, siis Eestis pole veel sellist hindamissüsteemi rakendatud poodides. Tarbija olukorra parandamiseks on võimalik kasutada lõputöö raames valminud lahendust.

1.2 Tekstituvastus

Optiline märgituvastus on masinõppe üks suuremaid harusid. Alljärgnev materjal tugineb Sharma jt uuringule [19]. OCR mõte seisneb teksti tuvastamisel pildilt. Seega võib selle sisendiks pidada mingit pilti ja väljundiks pildilt tuvastatud teksti. Nad jagavad tekstituvastuse tsükli kuueks etapiks (joonis 1a): pildihõive, eeltöötlus, segmentimine, detailide eraldamine, klassifitseerimine ja järeltöötlus. Pildihõive etapis saadakse digitaliseeritud versioon tekstist.

On tähtis, et digitaalne versioon oleks võimalikult kvaliteetne, sest moonutatud või liiga mürarohke pildi puhul ei suuda eeltöötlemine kõike korda teha. Eeltöötlemine korrastab digitaliseeritud teksti, et tekstituvastus protsess oleks võimalikult efektiivne. Antud etapis toimub binariseerimine (0-255 spektrist üleminek 0-1 spektrisse), kalde korrastamine, üleliigse tausta kärpimine, taustamüra eemaldamine jne. Järgmisena toimub segmentimine (joonis 1b), kus eeltöödeldud tekst jagatakse väiksemateks osadeks. Tekst jagatakse lõikudeks, lauseteks, sõnadeks, märkideks ja alammärkideks. Detailide eraldamise etapis jagatakse andmed detailide hulkadesse, et leida olulisi omadusi ja mustreid. Selle etapi eesmärk on vähendada pinda, mida peab tuvastama. Autorid on toonud välja kolm kategooriat mustrite leidmiseks: globaalsed muutused, struktuursed omadused, statistilised omadused. Klassifitseerimise etapis jagatakse andmed vastavalt eelnevalt saadud mustritele ja omadustele sõnadeks. Järeltöötlus kontrollib väljastatud teksti ja proovib leida vigu (nt „m“ ja „rn“) [20].

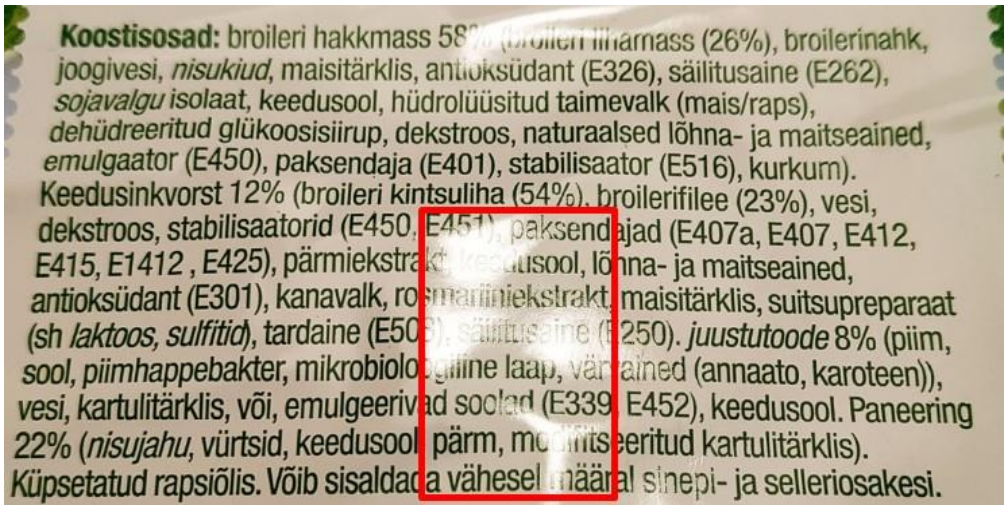


(a)

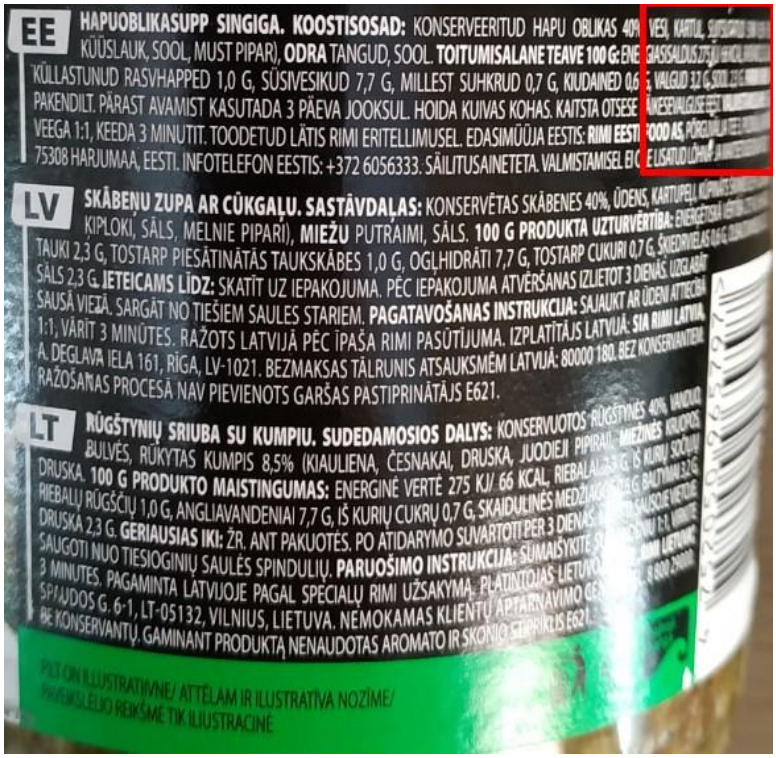
(b)

Joonis 1. Tekstituvastus. (a) Tekstituvastuse põhietapid. (b) Originaalpildist saadakse must-valge pilt, millel toimub segmentimine [21].

Optiline märgituvastus hõlmab endas palju probleeme, mida Ye jt [6] on liigitanud keskkonna, pildihõive ja teksti sisuga seotud probleemideks. Välja toodud teemad on jagatud alamkategooriatesse. Keskkonnaga seotud probleemideks on tegevuskoha keerukus ja ebaühtlane valgustus. Tegevuskoha keerukus tegeleb tausta ja teksti eristamisega - kui dokumentide peal on lihtne eristada musta kirja valgest paberist, siis looduses sulandub tekst tihti taustaga ühte. Ebaühtlane valgustus on väga levinud looduses, mistõttu on värvid tihti moonutatud ja pildi detailid väärad (joonis 2). Pildihõivega seotud probleemid on pildi kvaliteedi langemine pilditihenduse meetodi ja halva kaamera tõttu ning moonutustase, mis oleneb sellest, kui terava nurga alt pilte teha (joonis 3). Teksti sisuga seotud probleemideks on formaadisuhe, šrift ehk kirjastiil ja mitmekeelsed keskkonnad. Formaadisuhe on otseselt seotud teksti füüsiliste mõõtmetega. Kirjastiil kirjeldab probleeme, kus näiteks erinevates šriftides on samad tähed väga erinevad. Ye jt [6] sõnul on mitmekeelse keskkonna probleemiks näiteks see, et osades keeltes muutub märgi tähendus vastavalt kontekstile.



Joonis 2. Valgusvihkude tõttu on osa teksti loetamatu (piiritletud punase kastiga).



Joonis 3. Purgi kumeruse tõttu on osa teksti loetamatu (piiritletud punase kastiga).

Seoses mainitud probleemidega võib tekkida olukord, kus ühe fotoga polegi võimalik saada ülevaadet tervest etiketist (joonis 2 ja 3). Sellisel juhul võib lahenduseks olla dünaamiline pildivoog, millest räägitakse hilisemas peatükis.

1.3 Multiplatvormarendus

Multiplatvormarendus (lüh *MPA*) võimaldab ühe koodi baasiga luua rakenduse erinevatele süsteemidele (nt Flutteri puhul Android ja iOS). Heitkötter jt [22] sõnul on MPA vastandiks omaarendus (ingl *native development*). Sellisel juhul kasutab arendaja OS-spetsiifilisi teeke, tööriistu ja programmeerimiskeeli (nt iOS puhul Swift ja Objective C). Nad eristavad MPA juures kahte lähenemist - operatsioonisüsteemispetsiifilise koodi genereerimine programmi kompileerimisel (ingl *ahead-of-time*, lüh *AOT*) ning käitusaegse keskkonna kasutamine (ingl *just-in-time*, lüh *JIT*). Antud uurimistöös kasutatakse AOT kompilatsiooni väljalaske ehitamiseks ning JIT kompilatsiooni arendamise jaoks. Autorid leiavad, et mobiiliarenduse puhul on MPA tihti optimaalsem valik kui omaarendus, sest MPA tööriistad on piisavalt võimsad ja kasutajasõbralikud, et arendada rakendusi erineva süsteemi peal. Üheks

võimalikuks variandiks, mille puhul võib omaarendus parem olla, tõid nad välja videomängude kasutajaliidesed.

Käesolevas lõputöös keskendutakse mobiilsele arendusele, seega käsitleb autor operatsioonisüsteeme iOS ja Android. Rakenduse testimiseks iOS peal on vaja nutitelefoni iOS süsteemiga ning arendusmasinat (macOS), millel on Xcode peal. Mobiilse MPA jaoks on turul mitmeid tarkvararaamistikke (ingl *framework*). Laialtlevinud raamistikud on veel näiteks Xamarin ja React Native.

Xamarin on Microsofti poolt loodud tarkvara, mis kasutab keelt C# ja töötab .NET raamistikul [23]. Xamarin kompileerib Android arenduse puhul C# koodi vahekeelde (ingl *intermediate language*, lüh *IL*) ja sealt edasi JIT meetodi abil assemblerkeelde. Platvormi iOS puhul kompileeritakse C# kood täielikult AOT meetodi abil täiustatud RISC-masina (ingl *advanced RISC machine*, lüh *ARM*) assemblerkoodi.

React Native on Facebooki poolt loodud raamistik, mis kasutab arendamiseks JavaScripti ja Reacti komponente [24]. React kasutab mõlema platvormi puhul JIT kompilaatorit.

1.4 Tehnoloogiate valik

Järgnevalt antakse ülevaade kasutatud tarkvararaamistikest ja keelest. Samuti kirjeldatakse töös käsitletud masinõppe tööriistu.

1.4.1 Flutter

Flutter on 2017. aastal Google poolt loodud avaliku lähtekoodiga (ingl *open source*) multiplatvormi tarkvararaamistik, mis kasutab programmeerimiskeelt Dart [25]. Arendamisel kompileeritakse kood virtuaalmasinas JIT, mis võimaldab kuuma taaslaadimist (ingl *hot reload*) ehk kogu programmi pole vaja uuesti kompileerida. Uue versiooni väljastamisel kasutatakse AOT kompileerimist, et saavutada parem jõudlus.

Antud rakenduse arendamiseks kasutati Flutterit. Xamarin ja React Native jäid valikust välja, sest autoril on liiga vähene kokkupuude keeltega C# ja JavaScript. Samuti on autoril juba varasem kogemus Flutteriga. Töö arengu käigus otsustati keskenduda ainult Android arendusele, sest mõlema süsteemi testimine on liiga ajakulukas.

1.4.2 Dart

Cleverismi artiklis [26] kirjutatakse, et Dart on 2011. aastal tutvustatud avaliku lähtekoodiga objekt-orienteeritud keel, mis sarnaneb süntaksilt C-keelega. Algselt oli keel mõeldud veebis kasutamiseks. Rakenduses implementeeritud Dart keele näide on joonisel 4.

```
_loadAllergens() async {
  //Creates the string version of json file, so it can be saved to local storage

  //reads json into json string
  String data = await DefaultAssetBundle.of(context).loadString("assets/allergens.json");

  //converts json string into list of allergens
  List<Allergen> allergens;
  allergens = await (json.decode(data) as List).map((i) =>
      Allergen.fromJson(i)).toList();

  //converts list of allergens into list of strings
  List<String> jsonList = allergens.map((allergen) => jsonEncode(allergen.toJson())).toList();

  //update global variable _allergens
  _allergens = jsonList;
}
```

Joonis 4. Koodinäide Dart süntaksist.

Darti dokumentatsioonis on kirjutatud [27], et masinkoodi kompileerimiseks on kasutusel Dart Native, mille tüüpideks on JIT ja AOT kompilaatorid. Veebi kompileerimiseks on Dart Web, mis kompileerib koodi JavaScript koodiks.

1.4.3 ML Kit

Järgnev materjal on refereeritud ML Kiti dokumentatsioonist [28] [29]. ML Kit on nii seadmel (ingl *on-device*) kui ka pilvel (ingl *cloud*) põhinev Google'i arendatud masinõppe tööriist. Rakenduses implementeeriti ainult seadmel põhinevat masinõppe tarkvaraarenduspaketti (ingl *software development kit*, lüh *SDK*), sest seda on võimalik kasutada võrguühenduseta ja on tasuta. Tabelis 1 on näha seadmel ja pilvel põhineva paketi erinevusi. SDK sisaldab eelnevalt empiiriliste andmetega treenitud mudeleid, mida kasutatakse järgnevates rakendusliidetes (ingl *Application Programming Interface*, lüh *API*): tekstituvastus, pildi sildistamine, triipkoodi skaneerimine, näotuvastus, objekti tuvastamine ja jälgimine, keele tuvastamine, tõlkimine, nutikas vastamine sõnumitele ja vaatamisväärsuste tuvastamine. Pilvel põhinevaid rakendusliideseid on võimalik kasutada Firebase platvormi kaudu. Kuna ML Kit pole avatud

lähtekoodiga ja tegemist on valdkonnaga, kus on tihe konkurents, pole võimalik kindlalt teada, millist algoritmi uurimistöös implementeeritud mudel kasutab.

Tabel 1. ML Kit seadmel ja pilves võrdlus [30].

Firebase ML Kit variant **Omadused**	**Seadmel**	**Pilves**
Kasutamise hind	Tasuta	Esimesed 1000 päringut kuus tasuta, edasi on tasuline
Peamine otstarve	Parim hõreda teksti tuvastamiseks piltidelt	Hea hõreda teksti tuvastamiseks piltidelt ja tiheda teksti tuvastamiseks dokumentidelt
Tuvastamise võimekus	Tuvastab ladina tähti	Tuvastab paljusid keeli ja erinevaid sümboleid
Tuvastamise kiirus	Madal latentsus - reaalajas kaadrite töötlemine	Kõrge latentsus - parem täpsus
Võrguühendus	Töötab võrguühenduseta - andmed jäävad seadmele	Töötab pilves

Firebase teenuste [31] hulgas on veel palju muud peale ML Kiti. Tagakomponendis (ingl *back end*) kasutatavad tooted on näiteks Cloud Firestore, mis on NoSQL andmebaas, ja Authentication, mis pakub turvalist autentimist. Väljalaske ja jälgimise jaoks on näiteks App Distribution, mis võimaldab rakendust testijatele jagada, ning Crashlytics, mida kasutatakse vigade haldamiseks. Kasutajakogemuse parendamiseks on kasutusel näiteks Predictions, mis rakendab masinõpet, et kasutajate käitumist ennustada, ja A/B Testing, mis võimaldab uuendustega eksperimenteerida. Antud töö jaoks piisas ainult ML Kit kasutusest.

1.5 Sarnased lahendused

Turul on palju erinevaid lahendusi, mis pakuvad kasutajale võimalust toidus sisalduvaid aineid kontrollida. Rakendusi valiti nii Google Playst kui ka Apple App Store'ist. Valikust jäeti välja äpid, mis ei läinud tööle või olid liiga madala hinnanguga. Rakendusi võrreldi kindlate omaduste põhjal. Võrreldavad omadused käsitlevad küsimusi nagu milline skaneerimismeetod on kasutusel, kas rakendus pakub lisainfot toiduainete kohta, kas kasutaja saab enda allergeene kontrollida, kas kasutaja tehtud valikud salvestatakse, millisel platvormil rakendus töötab, kas võrguühendus on vajalik rakenduse kasutamiseks.

Edaspidi on kasutusel järgmised lühendid: Infood (I) [7], OpenFood (OF) [12], Food Scanner (FS) [8], Foodi (F) [9], Food Ingredient Scanner (FIS) [10], Allergy & Vegan Scanner (AVS) [11] ja Scannit (S). OF jagab oma nime OFF andmebaasiga, mida tutvustati peatükis 1.1. Scannit on lõputöö raames kirjutatud lahendus.

- **Skaneerimine:** Skaneerimismeetodite all eristatakse triipkoodi ja koostisosade skaneerimist. OF kasutab triipkoodi skaneerimist, FS, FIS, AVS ja S kasutavad koostisosade skaneerimist. I ja F lubavad mõlemat. Triipkoodi skaneerimise suutlikkus on suuresti mõjutatud olemasolevast andmebaasist. Testimise käigus avastati, et I ja OF kasutavad OFF andmebaasi.
- **Lisainfo toiduainete kohta:** I, OF, F, FIS ja AVS pakuvad kasutajale lisainfot toiduainete kohta. FS ja S sellist funktsionaalsust ei oma. F puhul sooritatakse iga koostisosa kohta päring Vikipeediasse ning rakendus tagastab pildi, kui see eksisteerib, ja artikli esimese lõigu. FIS ja AVS puhul navigeeritakse kasutaja lihtsalt koostisosa Vikipeedia leheküljele. Kuna I ja OF kasutavad mõlemad OFF andmebaasi, esitatakse toidu NOVA grupp ja Nutri-Skoor.
- **Kasutaja allergeenide lisamine ja kontrollimine:** Kõik peale F võimaldavad kasutajal allergeene valida. Valitud nimekirja kontrollitakse skaneeritud koostisosade vastu. FS puhul saab valida allergeene viie kategooria (maapähklid, gluteen, kala, muna, piim) hulgast. Peale allergeenide saab rakenduses valida ka 4 dieedi (veganlus, taimetoitlus, loomne, jainism) vahel. FIS puhul pidi allergeenide lisamiseks maksma. Rakenduses AVS saab valida kolm suuremat toidugruppi tasuta, aga piiramatu variant oli tasuline. AVS ja S puhul on kasutajal võimalik lisada enda allergeene rakendusse.

- **Valikute salvestamine:** Valikute salvestamise all on peetud silmas, kas rakenduse sulgemisel ja uuesti käivitamisel on tehtud valikud alles. FS ei salvesta kasutaja valikuid. Kuna F ei lase kasutajal allergeene valida, siis ei ole antud äpil vaja midagi salvestada. Teised rakendused salvestavad kasutaja valikud.
- **Toetatud platvormid:** I, OF, FS ja F on toetatud ainult Androidi peal ning AVS iOS peal. FIS puhul leidub mitu Androidi versiooni erineva nimega ning ka üks iOS versioon. S on arendatud mõlemale platvormile, aga lõplik versioon on testitud ainult Androidi peal.
- **Võrguühendus:** FS ja S töötavad võrguühenduseta. I ja OF vajavad internetti, et luua ühendus andmebaasiga. FIS ja AVS vajavad internetti, et sooritada päringuid Vikipeediasse, aga rakenduse põhifunktsionaalsus säilib internetita. F ei esita ühtegi tuvastatud allergeeni võrguühenduseta.

Tabelis 2 on välja toodud sarnaste lahenduste omadused. Horisontaalselt on kirjutatud omadused ja vertikaalselt rakenduste nimed. Märge „+“ tähendab, et funktsionaalsus on olemas ning „-“, et antud funktsionaalsus puudub.

Tabel 2. Sarnaste lahenduste võrdlus. (antud rakendus sinises kirjas)

Omadus **Nimi**	**Skaneerimine**	**Lisainfo toiduainete kohta**	**Nimekirja kontrolli-mine**	**Valikute salvesta-mine**	**Toetatud platvorm**	**Kasutatav võrgu-ühenduseta**
Infood [7]	Triip Koostisosad	+	+	+	Android	-
OpenFood [12]	Triip	+	+	+	Android	-
Food Scanner [8]	Koostisosad	-	+	-	Android	+

Foodi [9]	Triip Koostisosad	+	-	-	Android	-
Food Ingredient Scanner [10]	Koostisosad	+	+	+	Android/ iOS	+
Allergy & Vegan Scan [11]	Koostisosad	+	+	+	iOS	+
Scannit	Koostisosad	-	+	+	Android/ iOS[1]	+

Käesoleva töö raames valminud lahendus aitab kasutajal etiketi manuaalset lugemist automatiseerida. Rakendust saab kasutada internetita ja tasuta. Lisaks ülaltoodule prooviti ka üldist skaneerimiskogemust võrrelda. Erinevalt teistest lahendustest võimaldab Scannit eestikeelset etiketti kontrollida. Samuti märkab rakendus allergeene, mis on liitsõnana kirjas (nt „nisu“ ja „nisujahu“) või on tuvastatud tekstilt vigaselt sisse loetud (nt pakendil on „lupiin“, aga loeti sisse „lupim“). Teised rakendused jäid hätta, kui tuvastatav sõna oli algoritmi jaoks n-ö valesti kirjas, ehk võib eeldada, et äpid rakendavad üks-ühele kontrolli.

[1] iOS versioon pole veel lõplik

2. Valminud lahenduse kirjeldus

Järgnevalt kirjeldatakse arendusega seotud protsesse ning antakse ülevaade arhitektuurist. Samuti näidatakse, kuidas rakendus töötab.

2.1 Rakenduse arendusprotsess ja nõuded

Programmeerimiseks kasutati arenduskeskkonda *Android Studio*. Samuti puututi kokku *Xcode* keskkonnaga, et jooksutada äppi iOS seadme peal. Jälgiti häid tarkvaraarenduse tavasid nagu versioonihaldussüsteemi *Git* kasutamine ja probleemide haldamine[2] (ingl *issue tracking*).

Töö algne idee ning esialgsed nõuded tekkisid seoses Estonian Business School konkursiga, mis toimus 2020. aasta mais. Nüüdseks on paljud nõuded muutunud ning rakendus ei ole enam kolmanda osapoolega seotud.

2.1.1 Funktsionaalsed nõuded

Järgnevalt on toodud nimekiri funktsionaalsetest nõuetest, mida rakendusega kindlasti teha peab saama.

1. Kasutaja saab allergeene nimekirjast valida
2. Kasutaja saab ise allergeene hallata
   2.1 Kasutaja saab allergeene ise lisada, kui neid nimekirjas pole
   2.2 Kasutaja saab enda lisatud allergeene kustutada
3. Kasutaja saab koostisosade etiketti skaneerida
4. Kasutaja näeb tuvastatud allergeene peale skaneerimist
5. Kasutajat teavitatakse kui ühtegi sõna ei tuvastatud pildilt
6. Kasutajat teavitatakse kui pildilt tuvastati vähemalt üks sõna
   6.1 Kasutajat teavitatakse kui ühtegi allergeeni ei tuvastatud tekstist
   6.2 Kasutajat teavitatakse kui tuvastati vähemalt üks allergeen tekstist
7. Tuvastatud allergeenid on pildil eraldi märgitud
8. Kasutaja valitud allergeenid salvestatakse
9. Kasutaja saab valida levinumaid allergeene

[2] https://github.com/SanderKangur/scannit/issues

10. Rakendus töötab eesti keeles

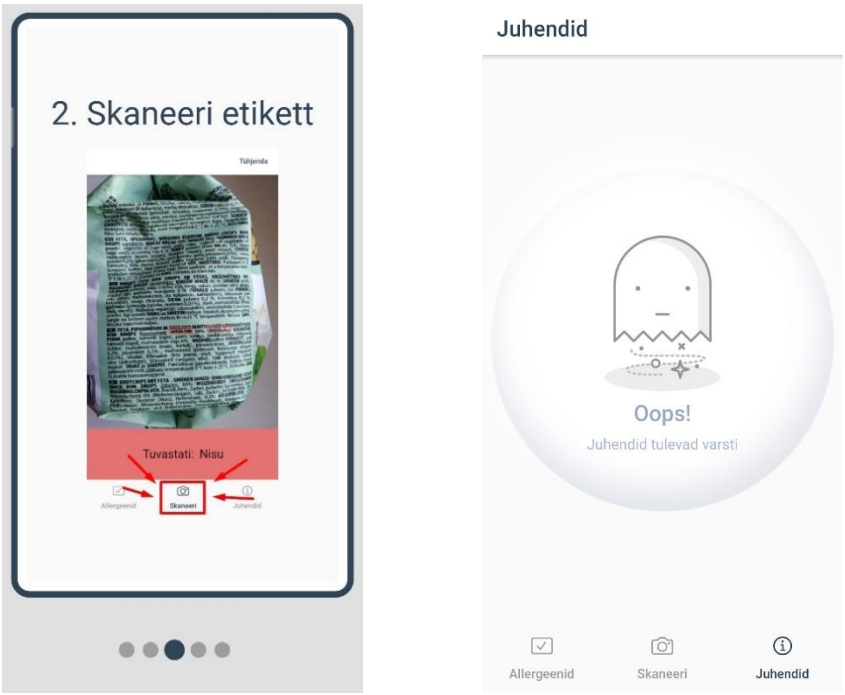
2.1.2 Mittefunktsionaalsed nõuded

Järgnevalt on loetletud rakenduse kohta kehtivad mittefunktsionaalsed nõuded:

1. Skaneerimine võtab alla 5 sekundi
2. Rakendus hoolitseb elutsüklite (ingl *lifecycle*) [32] eest
3. Andmeid hoitakse lokaalselt telefonis
4. Rakendust saab kasutada internetiühenduseta
5. Rakendus töötab vähemalt Android versioonil 7.0

2.2 Rakenduse kasutamine

Näitestsenaarium: kasutaja on poes ning leiab toote, mille koostisosasid ta tahab kontrollida. Kasutaja käivitab rakenduse, kuhu ta on varasemalt enda jaoks ohtlikud allergeenid lisanud. Kasutaja suunab kaamera etiketi poole. Kui rakendus leidis pakendilt kasutaja valitud allergeene, loetletakse tuvastatud allergeenid üles ja märgitakse punase kastiga. Kui rakendus ei leidnud ühtegi ohtlikku allergeeni, antakse kasutajale teada, et antud toidupakend on ohutu.

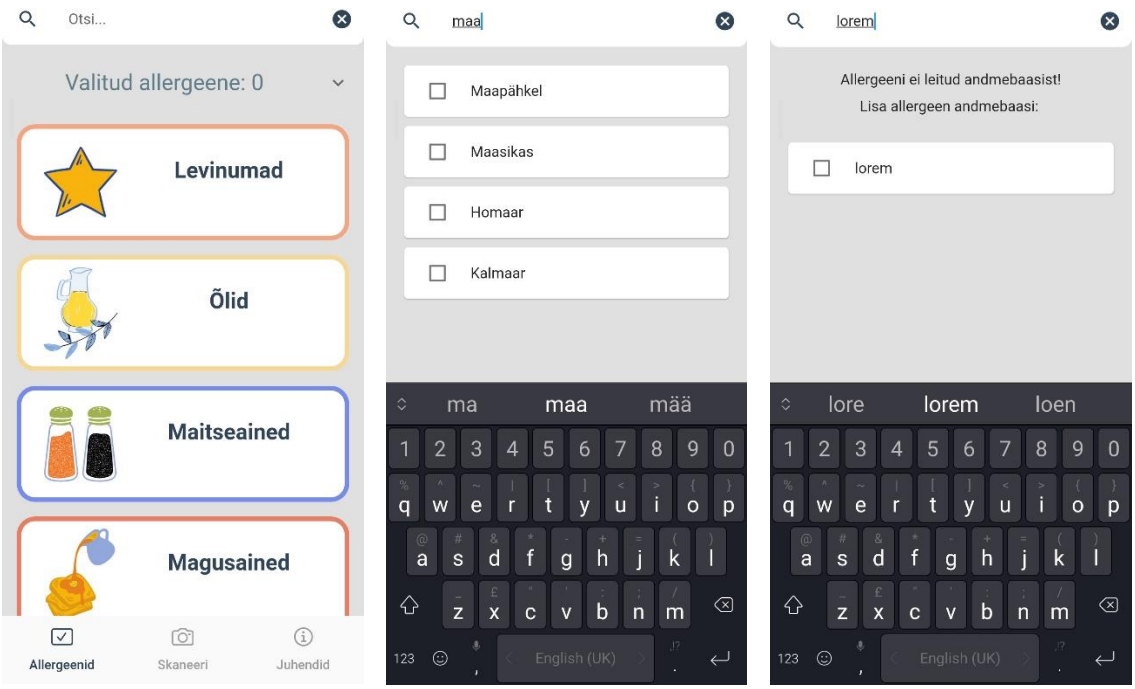


(a) (b)

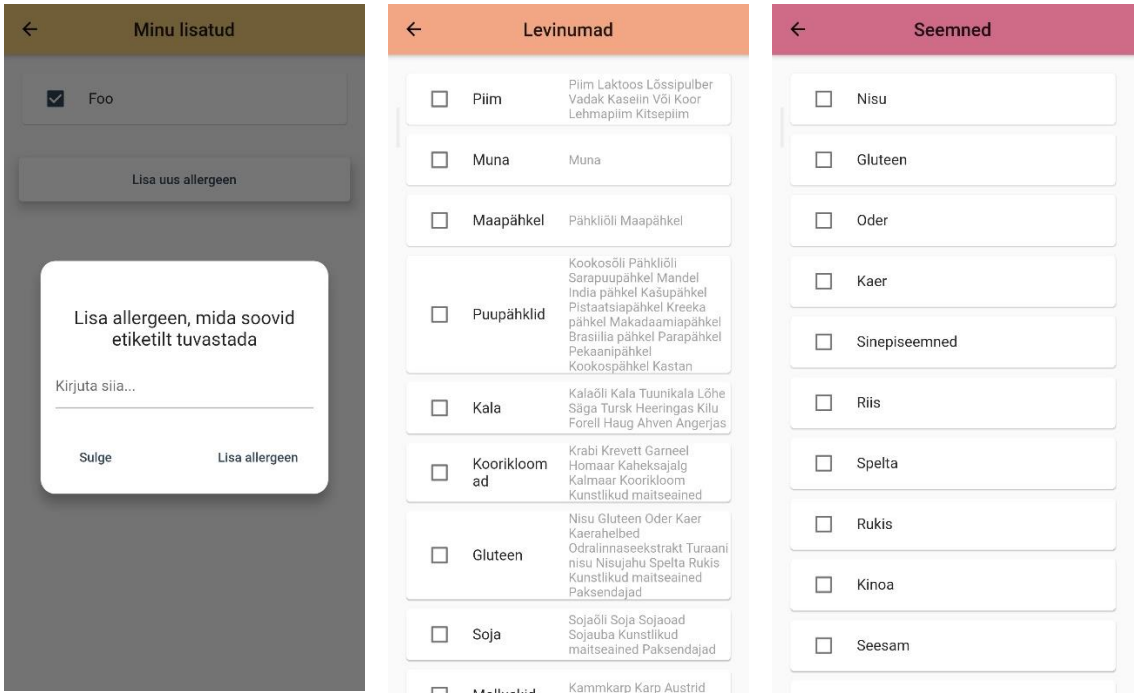
Pilt 1. Vahendid kasutaja aitamiseks. (a) Rakenduse tutvustus. (b) Spetsiifilised juhendid.

Kui kasutaja tõmbab rakenduse alla ning esimest korda selle tööle paneb, kuvatakse talle lühike rakenduse tutvustus (pilt 1a). Vahelehe “Juhendid” alt peaks tulevikus olema võimalik üles leida juhendid rakenduse kasutamiseks, praeguses versioonis ei jõutud neid implementeerida.

Järgmisena tuleks kasutajal lisada allergeenid, mida rakendus kontrollima hakkab. Valiku tegemiseks tuleb kasutajal minna vahelehele „Allergeenid“ (pilt 2a). Vahelehel on kasutajal võimalik allergeene otsida otsinguribalt (pilt 2b) või kategooriate kaupa. Kui otsitavat sõna andmebaasis ei leidu, saab kasutaja ise selle lisada (pilt 2c). Lisatud allergeen salvestatakse kategooria „Minu lisatud“ alla (pilt 2d). Siin lehel saab kasutaja hallata enda lisatud allergeene. Kategooriad saab mitut moodi rakendada. Üks võimalus on leida allergeen üles kategooriast „Levinumad“ (pilt 2e). Kasutajal on võimalik leida spetsiifilisemaid allergeene ülejäänud kategooriate hulgast (nt „Seemned“, pilt 2f).



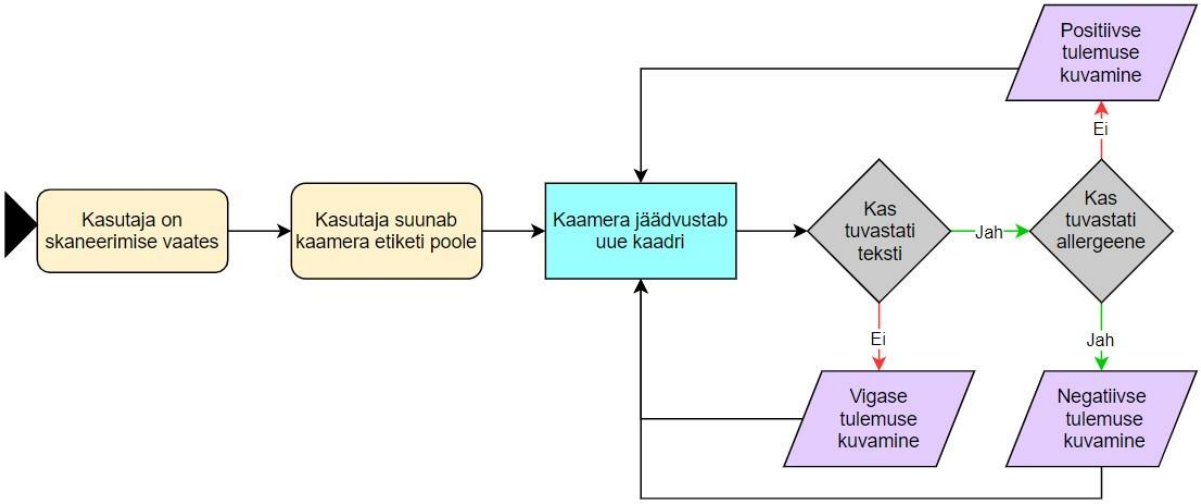
(a) (b) (c)



(d) (e) (f)

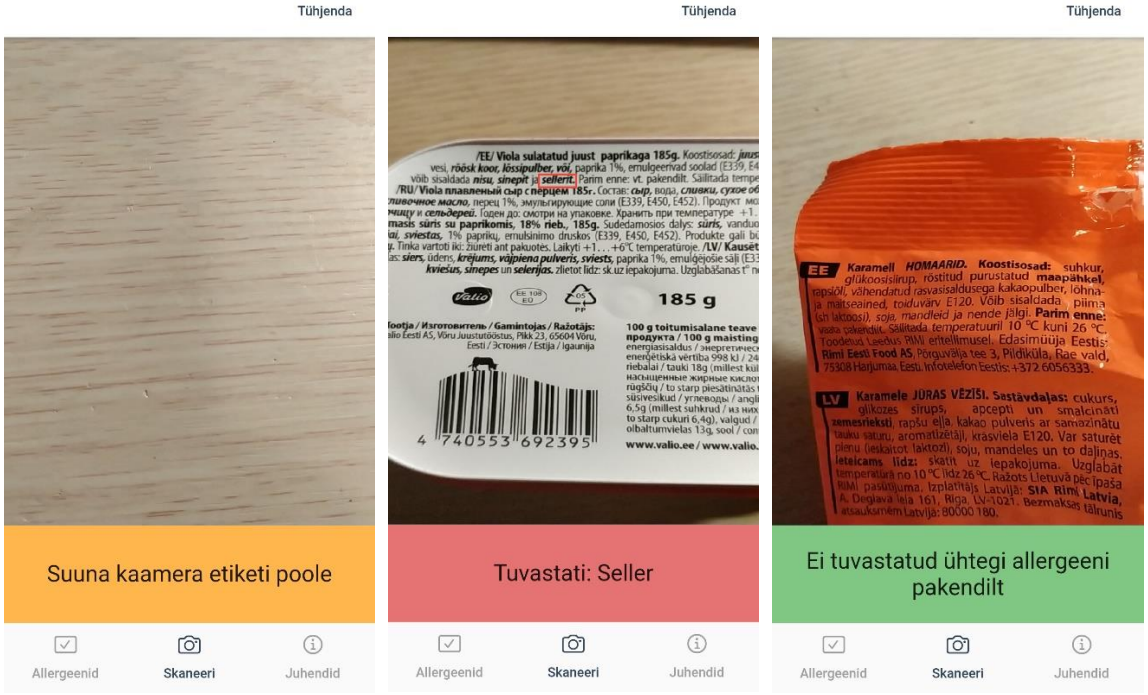
Pilt 2. Allergeenide haldamine. (a) Üldine vaade. (b) Andmebaasist otsimine. (c) Lisamine otsingust. (d) Lisamine käsitsi. (e) Levinumad allergeenid. (f) Seemnetüüpi allergeenid.

Rakenduse põhifunktsioon, etiketi skaneerimine, on välja toodud vooskeemina joonisel 5. Kasutaja alustab skaneerimise vaatest (pilt 3), kus tekstituvastuse algoritm proovib juba teksti leida. Kui kaamera pole etiketi poole suunatud või rakendus ei suuda lihtsalt teksti tuvastada, kuvatakse kasutajale vigase tulemuse vaade (pilt 3a). Kui pildilt leitakse tekst, hakkab tööle allergeenide sõnetöötlus algoritm, mis seletatakse lahti järgmises alapeatükis. Kui tuvastati allergeene, kuvatakse negatiivne tulemus (pilt 3b) ehk kasutaja on mingi koostisosa vastu allergiline. Kui allergeene ei tuvastatud, kuvatakse positiivne tulemus (pilt 3c) ehk kasutaja võib antud toodet tarbida. Järgmise sammuna saab kasutaja ette võtta uue toidupakendi.



Joonis 5. Skaneerimise vooskeem.

Kuna kaamera on pidevalt liikumises ja n-ö staatilist pilti ei tehta etiketist, siis jätab rakendus juba tuvastatud allergeenid meelde kontrollitava pakendi puhul. Vastasel juhul kaoksid mõned allergeenid ära, kui nad ei mahu kõik korraga kaamerasse. Kui eelmisest pakendist on leitud allergeene, siis tuleb väljund puhastada nupuga „Tühjenda“, mis asub üleval paremal (pilt 3).



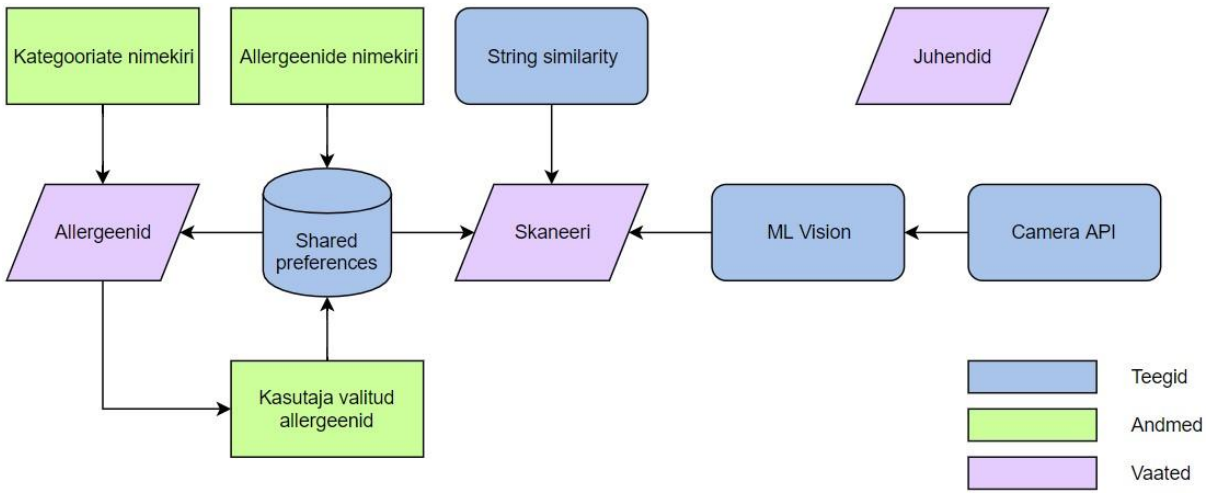
(a) (b) (c)

Pilt 3. Skaneerimise võimalikud tulemused. (a) Vigane tulemus. (b) Negatiivne tulemus. (c) Positiivne tulemus.

Pildil 3b on võimalik näha punast kasti ümber tuvastatud allergeeni. Rakendus joonistab iga kaader uue kasti. Kui rakendus leiab allergeeni mitmest kohast etiketilt, siis tehakse ka kast mitmesse kohta. Selle abil on kasutajal võimalik ise üle kontrollida, kas tuvastatud sõna on päriselt allergeen.

2.3 Arhitektuur

Joonisel 6 on välja toodud rakenduse arhitektuur. Käesolev lahendus koosneb kolmest peavaatest – „Allergeenid“, „Skaneeri“ ja „Juhendid“. Vaade „Juhendid“ on teistest eraldiseisev ja sisaldab ainult püsiprogrammeeritud (ingl *hardcoded*) informatsiooni. Vaated „Allergeenid“ ja „Skaneeri“ kasutavad teeki (ingl *library*) *Shared preferences* [33] lokaalse andmebaasina. Mobiilirakendus kasutab tekstituvastuseks paketti (ingl *package*) *Firebase ML Vision* (edaspidi *ML*) [34], mis saab töötlemiseks vajaliku pildi teegi *Camera* [35] abil. Vaade „Skaneeri“ võrdleb teegi *String similarity* [36] abil tuvastatud teksti ja kasutaja valitud allergeene.



Joonis 6. Rakenduse arhitektuur.

2.3.1 Tekstituvastuse algoritm

Eelnevalt käsitletud tekstituvastuse peatükis kirjeldati probleeme seoses valguse (joonis 2) ja pildistamise nurgaga (joonis 3). Probleemide lahendamiseks kasutatakse dünaamilist piltide voogu. Selle rakendamiseks on kasutusel teek *Camera*.

ML kasutab teegi *Camera* edastatud kaadreid, et reaalajas teksti tuvastada. Esiteks luuakse ML isend (ingl *instance*), mis rakendab seadmel põhinevat tekstituvastuse API-d. Antud kontekstis on kaader *CameraImage* isend, mille andmetest luuakse klassi *FirebaseVisionImage* isend. Antud isendi loomiseks on vaja baite ja metaandmeid. Baitide leidmiseks kasutatakse konkatenatsiooni (ingl *concatenation*) ehk kaadri tasandite (ingl *plane*) baidid loetakse ühte järjendisse. Metaandmeteks on pildi suurus, formaat ja pööre (ingl *rotation*) ning tasandi kõrgus, laius ja baitide arv ühes reas. Saadud *FirebaseVisionImage* isendil rakendatakse tekstituvastuse algoritmi. Kuna antud algoritmi tööprotsessi pole võimalik detailsemalt uurida, võib ainult oletada, et tööriist kasutab tavalisi tekstituvastuse meetodeid (nt eeltöötlus, segmentimine jne). Algoritmi tulemusena tagastatakse klassi *VisionText* isend, mis sisaldab pildilt tuvastatud teksti. Leitud tekst jaguneb blokkideks, mis jaguneb ridadeks, mis omakorda jaguneb sõnadeks. Iga tuvastatud sõna puhul rakendatakse sõnetöötluse algoritmi.

2.3.2 Sõnetöötluse algoritm

Sõnetöötluse algoritmi abil otsustatakse, kas koostisosa on allergeen või mitte. Kuna tekstituvastus ei ole alati perfektne, siis ei pruugi tuvastatud sõna olla täpselt sama, mis etiketil kirjas oli. Seega ei tohiks rakenduses võrrelda allergeene ainult üks ühele. Kahe sõna sarnasuse võrdlemiseks kasutatakse rakenduses teeki *String similarity*, mis rakendab Sørensen–Dice koefitsienti (edaspidi *DK*). Koefitsient on defineeritud järgmiselt [37]:

$$DK := \frac{2|A \cap B|}{|A| + |B|} \tag{1}$$

Valemis 1 tähistab $|A \cap B|$ tähtede paaride hulka, mis leiduvad mõlemas sõnas. $|A|$ tähistab võrdluse aluseks võetud sõna tähtede paare ning $|B|$ võrreldava sõna tähtede paare.

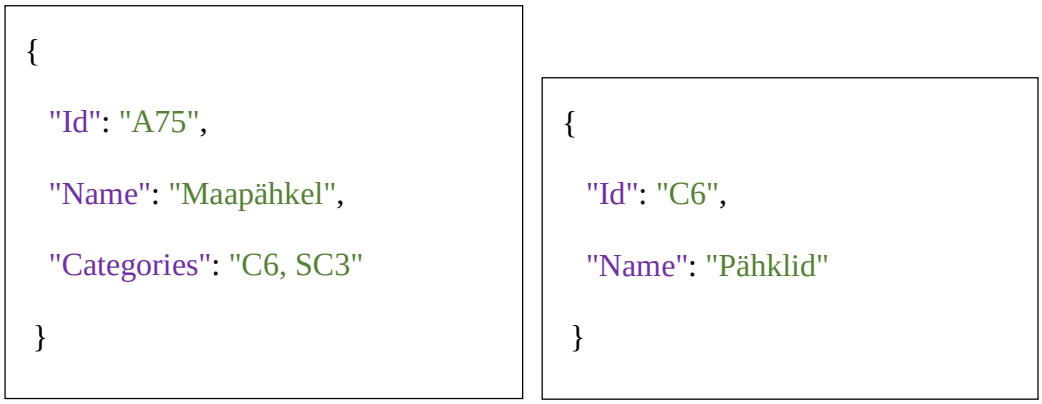
Järgnevalt toob autor näite DK paremini lahti mõtestamiseks. Olgu sõnad „seesam“ ja „seller“. Antud juhul oleks tähtede paarid vastavalt {se, ee, es, sa, am} ja {se, el, ll, le, er}. Kuna mõlemas sõnas on 5 tähtede paari, siis kehtib $|A| = |B| = 5$. On ainult üks paar, {se}, mis leidub mõlemas sõnas. Seega kehtib $|A \cap B| = 1$. Edasi võib asendada sümbolid numbritega $DK := \frac{2 \times 1}{5 + 5} = \frac{2}{10} = 0.2$. Antud näites on sõnade sarnasusfaktor 0.2.

Kuna *string_similarity* eristab suurt tähte, muudetakse kõik tähed nii skaneeritud tekstis kui kasutaja lisatud allergeenides väikesteks. Samuti muutub sarnasuslävend (arv, millest alates

loetakse sõnad sarnaseks) vastavalt sõna pikkusele. Kui skaneeritav sõna on lühem kui 6 tähte, seatakse lävend 0.8 peale. Kui sõna on vähemalt 6 tähte pikk, jääb piiriks 0.6. Mida lühem on sõna, seda tõenäolisem on, et sarnane tähtede kombinatsioon on pakendil olemas. Seega lühema sõna puhul peab sarnasuslävend kõrgem olema. Selline lisatingimus hoiab ära kaks probleemi. Esiteks, kui kaamera on suunatud nii, et ainult osa sõnast on kaadris, mis on väga tõenäoline, proovib rakendus ainult seda osa tuvastada. Teiseks, tihti on kaadris ka muude keelte etiketid, kust algoritm võib valepositiivseid leida. Täpsuse lisamiseks kontrollitakse, kas skaneeritav sõna sisaldab mõnda kasutaja valitud allergeeni. Sellist kontrolli ei saa vastupidi teostada, sest skaneeritav tekst sisaldab sagedasti üksikuid tähti. Sellisel juhul loeks rakendus üksiku tähe mingiks allergeeniks.

2.3.3 Andmemudel

Allergeene hoitakse lokaalselt telefonis teegi *Shared preferences* abil. Järgnev materjal tugineb mainitud teegi dokumentatsioonil [38]. *Shared preferences* võimaldab andmeid püsimälusse kirjutada ja neid sealt ka lugeda võti-väärtus paaridena (ingl *key-value pairs*). Võti on alati sõne ning väärtus võib olla sõne, sõnede järjend, täisarv, kümnendarv või tõeväärtus. Teeki on mõistlik kasutada, kui salvestatav andmete hulk on väike ning andmemudel pigem lihtne.



(a) (b)

Joonis 7. Lahenduses kasutatud JSON formaadid. (a) Allergeeni JSON formaat. (b) Kategooria JSON formaat.

Arendatud lahenduses kasutatakse väärtusena sõne ja sõnede järjendit. Rakenduse tööprotsessi vältel kasutatakse JSON formaati (ingl *JavaScript Object Notation*) andmete formaadiks.

Andmete salvestamiseks konverteeritakse JSON sõneks ja seatakse kindlale võtmele. Rakenduses implementeeritud JSON formaat on välja toodud joonisel 7. Lisaks kasutatakse rakenduses järjendit, kus hoitakse kasutaja valitud allergeene. Pistikmooduli abil salvestatakse valitud allergeenid sõnede järjendina ning üldine allergeenide nimekiri sõnena. Üldine nimekiri tuleb salvestada, kui kasutaja otsustab lisada allergeene, mida rakenduses veel olemas pole. Kuna kasutaja kategooriaid muuta ei saa, pole vaja neid lokaalselt meelde jätta.

Järgmises peatükis käsitletakse käesoleva töö raames korraldatud katseid.

3. Rakenduse testimine

Rakenduse kvaliteedi hindamiseks otsustati viia läbi katsealustega testimine. Katsealune asus tavakasutaja ehk allergiku rolli ning sooritas erinevaid ülesandeid. Selles peatükis antakse ülevaade testimise protsessist ja tulemustest. Samuti tuuakse välja tehtud järeldused ning kirjeldatakse plaane tuleviku jaoks.

3.1 Testimise läbiviimine

Testimise eesmärk oli hinnata kasutajasõbralikkust ja rakenduse täpsust. Testimisel võrreldi etiketi rakenduseta kontrollimise ehk manuaalselt lugemise ja rakendusega skaneerimise kiirust ning erinevusi. Samuti uuriti, kuidas lahendus toimib erinevate toidupakenditega pidades silmas etiketi disaini ja pakendi materjali.

Testimiseks küsitleti ja vaadeldi 10 katsealust. Kõik testijad lahendasid samu probleeme ja kasutasid sama seadet testimiseks. Testiti kuut pakendit ja iga pakendi puhul sooritati kolm ülesannet (Lisa I).

Käimasoleva SARS-COVID-19 pandeemia tõttu otsustati koostada valim noortest inimestest. Katsealused valiti pigem ühest seltskonnast, et vähendada viiruse riski. Testijad on siiski erinevate taustadega.

Testimiseks vajalik varustus:

- 6 pakendit
- Kontroll-allergeenide (lüh *K-A* edaspidi) tabel
- Stopper
- Nutitelefon (Xiaomi Redmi Note 6 Pro)

Testimisel kasutati kuut autori poolt valitud pakendit. Toidupakendid valiti nende etiketi (nt kirjasuurus, reavahe, värvide kontrast, teksti paigutus) ja pakendi (nt materjal, kuju, värv) erinevuste järgi, et saaks testida nii selgema kujundusega kui ka mürarohkemaid pakendeid. Kõikidel pakenditel oli olemas eestikeelne etikett. Kõik pakendid olid avamata ja poeriiulilt kättesaadaval kujul. Iga pakendi kohta on loodud n-ö allergik, kelle jaoks on autori poolt valitud K-A. Selline valik tagab selle, et kõik testid toimuvad kontrollitud keskkonnas. Valitud allergeenid antakse katsealusele kontrollimiseks. Iga pakendi puhul sooritati kolm ülesannet.

Enne testimist esitati katsealusele mõned üldised küsimused (Lisa II A), et saada parem ülevaade katsealuse taustast. Esimese ülesandena mõõdeti, kui palju võtab aega manuaalselt etiketi lugemine ja K-A sisalduse kontrollimine. Manuaalselt allergeenide tuvastamine pakendilt:

1. Kasutaja ees on pakend ja K-A tabel
2. Korraldaja käivitab stopperi
3. Kasutaja valib pakendi
4. Kasutaja peab iga K-A puhul otsustama, kas toidupakendi etikett sisaldab seda või mitte
5. Korraldaja peatab stopperi

Seejärel mõõdeti, kui palju aega läheb katsealusel K-A lisamiseks rakendusse. K-A lisamist rakendusse mõõdeti sellepärast, et see kujutab üht põhifunktsionaalsust ning on tähtis osa, kui mõõta rakenduse kiirust ja lihtsust. Allergeenide lisamine rakendusse:

1. Kasutaja ees on K-A tabel
2. Korraldaja käivitab stopperi
3. Kasutaja käivitab rakenduse
4. Kasutaja lisab K-A rakendusse
5. Korraldaja peatab stopperi

Viimasena mõõdeti, kui palju võtab aega rakendusega K-A tuvastamine etiketilt. Rakendusega allergeenide tuvastamine pakendilt:

1. Kasutaja ees on pakend
2. Korraldaja käivitab stopperi
3. Kasutaja valib pakendi
4. Kasutaja skaneerib pakendit kuni kõik allergeenid on käes või üks minut on täis
5. Korraldaja peatab stopperi

Pärast testimist esitati katsealusele mõned küsimused rakenduse kohta (Lisa II B), et selgitada välja lahenduse nõrkused ja tugevused ning hinnata rakendust.

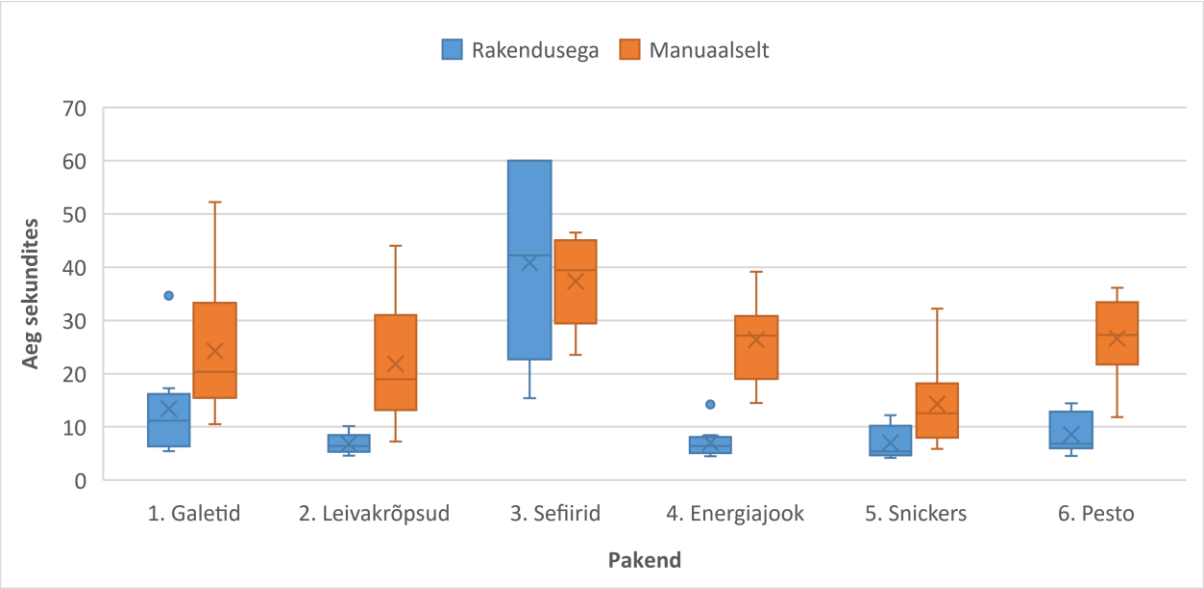
3.2 Testimise tulemused

Osalenud katsealused olid kõik 22-23 aastat vanad. Valimis oli 4 naissoost ja 6 meessoost isikut. Kõik peale ühe testija olid vähemalt ühte skaneerimisega seotud rakendust (nt QR koodi

skaneerija) varem kasutanud. Keskmine kogemus nutitelefoniga oli 8.6 aastat. Kuus testijat eelistasid platvormi iOS ja neli testijat Androidi. Kõigi katsete tulemused on esitatud Lisas III.

Testimisest tuli välja, et üldiselt läks kasutajatel palju vähem aega rakendusega skaneerides, kui ise K-A manuaalselt otsides. Joonisel 8 on toodud mõlema meetodi kiiruse võrdlus. Keskmiselt läks testijatel manuaalseks otsimiseks 25.1 sekundit ning rakendusega skaneerimiseks 13.9 sekundit. Seega oli skaneerimine 1.8 korda kiirem.

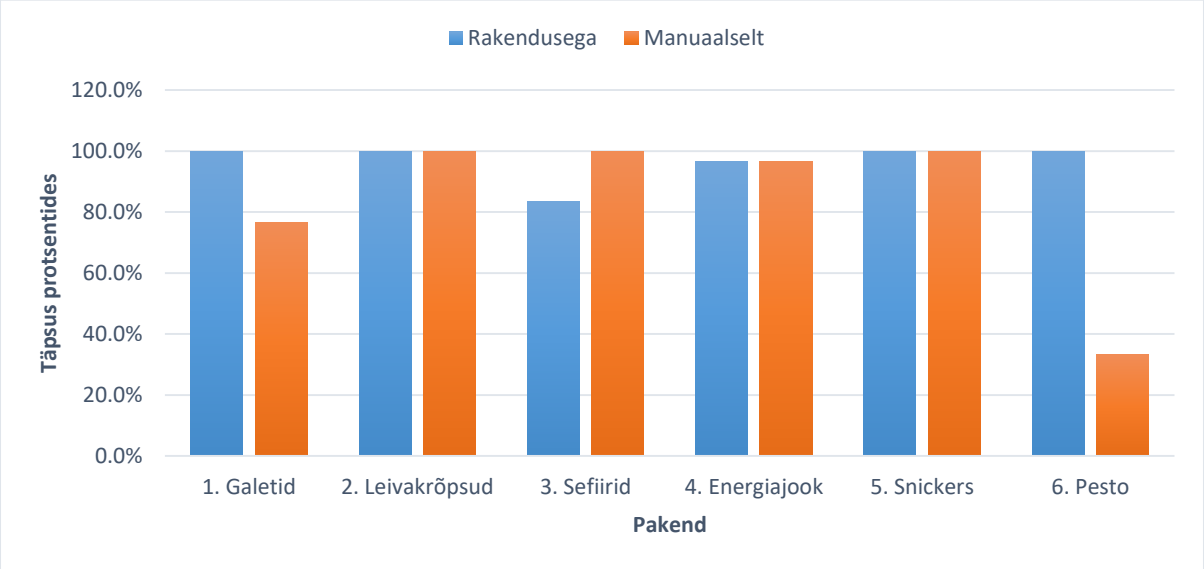
Sefiiripakk oli selgelt kõige raskem nii kasutaja kui ka rakenduse jaoks. Kasutajale pakkus raskust see, et etikett polnud keelte kaupa lõikudeks jaotatud, ning tihe kuldne tekst lillal taustal, mida oli küllaltki raske lugeda. Rakenduse jaoks oli otsustavaks peegelduv materjal ja väike hägune kiri. Neljal katsealusel ei õnnestunud alla minuti kõiki allergeene rakendusega üles leida. Sefiiripaki puhul mängis suurt rolli valgus ja pakendi kortsud. Kui K-A jäi täpselt varju, siis ei suutnud rakendus seda tuvastada. Veel on jooniselt näha, et käsitsi kontrollimise puhul varieerus soorituse aeg kordades rohkem kui rakendusega skaneerides.



Joonis 8. K-A kontrollimise kiirus pakendilt. Kasti sees tähistab joon mediaani ning *X* aritmeetilist keskmist. Ülemine kvantiil on 0.75 ja alumine 0.25. Vurrud tähistavad maksimumi ja miinimumi. Üksikud punktid on erindid, mis on keskmisest liiga kaugel.

Joonisel 9 on esitatud kontrollimise täpsus protsentides. Täpsuse all peetakse siin silmas korrektselt tuvastatud allergeenide arvu. Kuna katsealuseid oli 10 ja iga pakendi puhul 3 K-A, siis 100% võrdub 30-ga. Kui katsealune või rakendus jätab ühe allergeeni tuvastamata või leiab

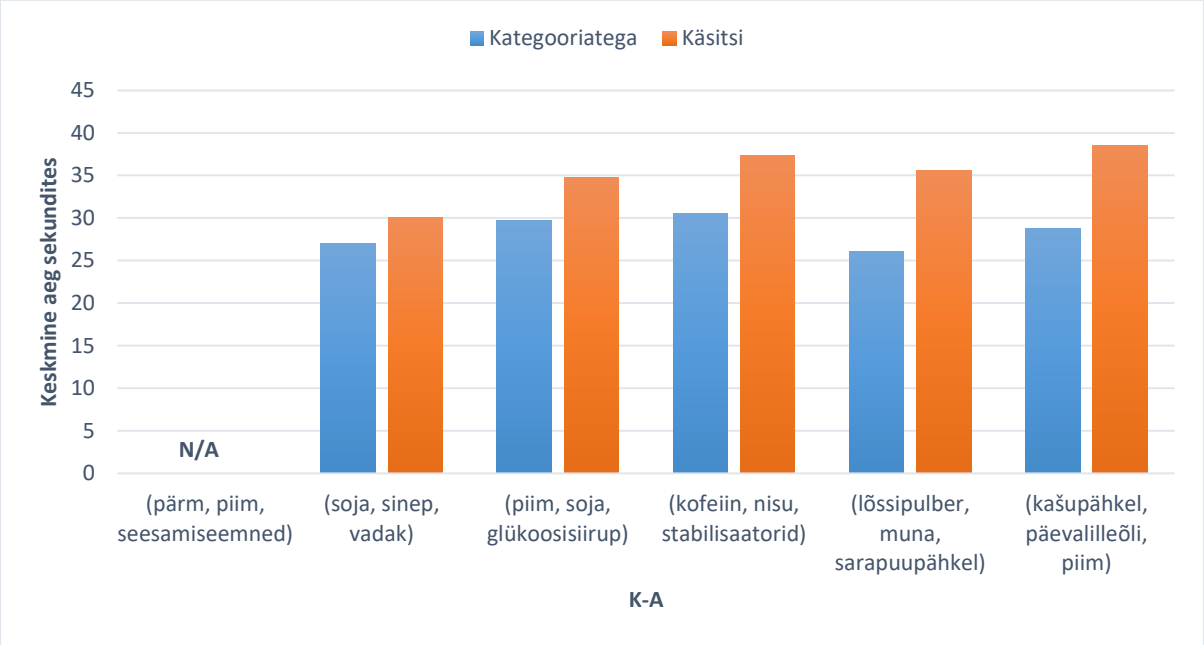
valepositiivse, siis on täpsus 29/30 ehk 96.7%. Jooniselt on näha, et rakendusega skaneerimine toimis täpsemini. Kuna galetipakk oli kõigi jaoks esimene testitav pakend, siis võib arvata, et inimesed ei olnud valmis veel etiketist infot filtreerima. Mitu testijat ei pannud alguses sõna „piim“ tähele, mis oli ainuke allergeen rasvases šriftis pakendil. Pakendite 2 ja 5 puhul olid K-A rasvases kirjas välja toodud. Sefiiripakk võttis manuaalselt kontrollides kõige rohkem aega, aga kõik K-A leiti lõpuks üles. Energiajoogi puhul ei leidnud üks katsealune sõna „stabilisaatorid“. Pesto purgilt pidi leidma sõna „kašupähkel“, mis leidus pakendil tema teise nime all – India pähkel. Mitte ükski katsealune ei olnud sellest faktist teadlik. Pakendiga 1, 2, 5 ja 6 sai rakendus väga hästi hakkama. Sefiiripaki puhul ei leidnud rakendus üles sõna „piim“ või „glükoosisiirup“. Energiajoogi puhul tuli välja katsete ainuke valepositiivne, kus rakendus leidis sõna „nisu“, mida tegelikult etiketil polnud. Sõna asukoht märgiti pildil ainult korraks ära ning rohkem ei suudetud viga taasluua, et kontrollida, mis sõna vastu eksiti.



Joonis 9. K-A kontrollimise täpsus pakendilt.

Joonis 10 annab ülevaate K-A lisamise kiirusest. Üldiselt oli lisamiseks kaks võimalust – kategooriatest valides või ise kirjutades. Antud juhul loetakse „ise kirjutamiseks“ ka stsenaariumi, kus kasutaja proovis ühe korra kategooriaid kasutada, aga leidis, et saab käsitsi ikkagi kiiremini. Tuleb silmas pidada, et testimise käigus polnud veel otsimisfunktsiooni implementeeritud ehk kasutaja pidi kõik sõnad ise lõpuni kirjutama. Osalejatele näidati kõiki lisamisvõimalusi pärast esimest pakendit. Galetipaki puhul pole mõtet tulemust hinnata, sest

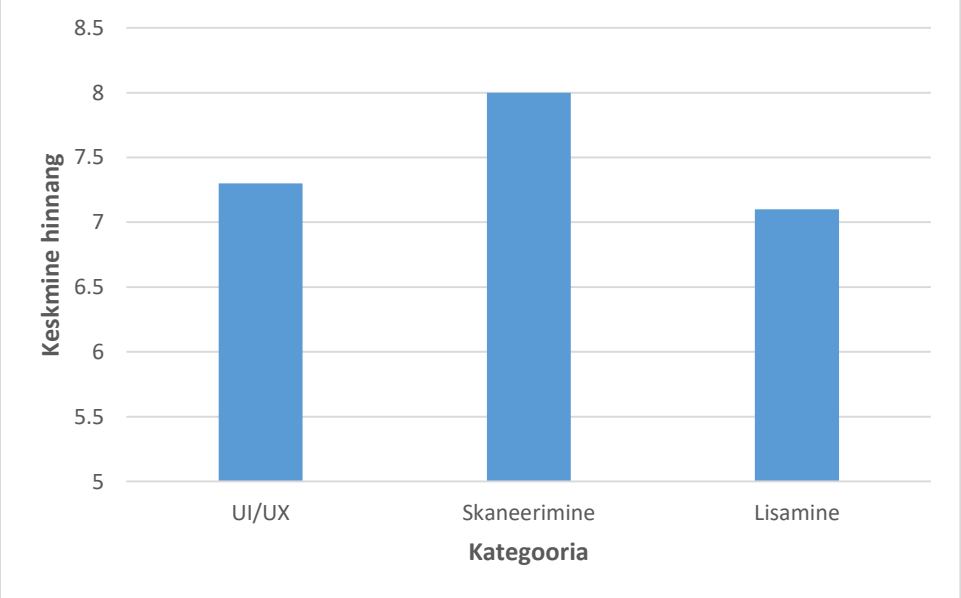
osad katsealused vajasid abi. Kategooriaid otsustas kasutada 5 inimest ja käsitsi lisas samuti 5 inimest. Üldiselt said kategooriaid kasutanud testijad paremini hakkama. Nad proovisid esimesena lisada allergeeni kategooria „Levinumad“ alt. Kui seda seal kirjas polnud, siis mõni otsis allergeeni teisest kategooriast ja mõni hakkas käsitsi lisama. Iga katsega läks meetodite ajaline vahe suuremaks. Sellest võib järeldada, et kategooriaid kasutanud katsealused said iga iteratsiooniga paremaks. Nad teadsid, kus osad allergeenid asuvad, ning said seetõttu kiiremini valmis. Üks asi, mida tuleks silmas pidada, on see, et lühikese sõna puhul (nt nisu) võib olla kiirem seda käsitsi lisada. See-eest pikem sõna (nt glükoosisiirup) pakkus paljudele kirjutajatele raskust.



Joonis 10. K-A lisamise kiirus rakendusse. Esimese pakendi puhul aeg ei loe, sest osad katsealused vajasid abi.

Joonisel 11 on näha testimise järel rakendusele antud hindeid. Testijatel paluti hinnata rakenduse erinevaid aspekte skaalal 1 kuni 10, kus 1 on väga halb ja 10 väga hea. Üldmuljele anti kümnepallisüsteemis hinne 7.3. Üldmulje kujutas endast kasutajakogemust (ingl *user experience*, lüh) ja kasutajaliidest (ingl *user interface*, lüh *UI*). Skaneerimisele anti 8.0 ning lisamisele 7.1. Kokku sai rakendus hindeks 7.5. Katsealused leidsid, et rakendus oli intuitiivne ja loogiliselt üles ehitatud. Üks testija väljendas rahulolematust liigse värvikasutuse üle, kuid teisele jällegi meeldis selline disain. Toodi välja, et skaneerimine on testijate arvates rakenduse

kõige tugevam külg. Mitu inimest kiitsid skaneerimise kiirust ning seda, et rakendus joonistab kastid ümber sõnadele, mida ta tuvastab. Lisamise puhul peeti positiivseks kategooriate üldist olemasolu. Üks katsealune leidis, et kategooria „Levinumad“ olemasolu on kasulik funktsionaalsus. Mõni katsealune ei märganud alguses, et rakenduse all paremas nurgas on nupp (ingl *floating action button*) allergeenide käsitsi lisamiseks. Kõige suuremaks miinuseks toodi välja rakenduse üldine kiirus. Äpp käivitus ligikaudu 7 sekundit, mis oli paljude kasutajate jaoks liiga pikk aeg. Samuti on viivitus vahelehtede vahel liikudes liiga suur. Aeglane sooritus võis olla tingitud nii 2.5 aastat vanast telefonist kui ka faktist, et rakendust testiti silumisrežiimis (ingl *debug mode*) mitte väljalaskerežiimis (ingl *release mode*). Vahe on selles, et väljalaskerežiimis optimeeritakse koodi, et tagada kiirem käivitus, parem jõudlus ja väiksem faili suurus [39].



Joonis 11. Rakendusele antud hinnangud.

3.3 Tulemuste analüüs

Testimisest saab järeldada, et allergeenide tuvastamine toidupakendilt mobiilirakendusega on kiirem ja täpsem kui manuaalne lugemine. Katsealustele meeldis skaneerimise kiirus ja äpi intuitiivsus. Katsealused oleks soovinud paremat lisamissüsteemi ning üldiselt kiiremat rakendust.

Kategooriaid kasutanud testijad suutsid kiiremini allergeene lisada ning see ajaline vahe aina suurenes. Seda saab põhjendada sellega, et testijad teadsid, kus mingi allergeen juba asub. See-eest käsitsi lisamise kiirus kellelgi ei paranenud, pigem tuli kiirustamise tõttu sisse kirjavigu.

Arendamise lõpufaasis oli üheks probleemiks valepositiivsete suur hulk. Selle parandamiseks prooviti optimeerida sõnetöötlust, mis allergeene omavahel võrdles. Selle tulemusena tuli katsete käigus ainult üks valepositiivne tulemus. Siiski on võimalik, et rakendus leiab allergeene, mida pakendil tegelikult pole. Kuna antud lahendus on seotud inimeste tervise ja heaoluga, siis on mõistlik lasta rakendusel tuvastada pigem rohkem allergeene kui vähem.

Tuleb silmas pidada, et ühelgi katsealusel polnud probleeme otsitavate allergeenidega ega üldse toiduallergiaga. Katsete käigus ei teadnud ükski testija, et kašupähkel ja India pähkel on samad pähklid. Rakendus oli siin olukorras kasulik, kuna sisaldas lisainfot allergeenide sünonüümide kohta, mida inimene ei pruugigi teada. See-eest võib eeldada, et pähkliallergiaga kasutaja sellist viga ei tee. Samuti ei pannud mitu testijat pakendi ainukest paksus kirjas olevat allergeeni tähele. Mainitud katsealused põhjendasid seda väitega, et nende silmad sõitsid lihtsalt üle sellest. Jällegi võib aimata, et etikette lugema harjunud inimene näeb paksus kirjas allergeene just esimesena.

Lisaks eelnevalt väljatoodule, tehti katsete käigus järgmised märkused:

- Kui kõik K-A leidusid pakendil, oli ajakulu väiksem sellest, kui mõni oli puudu.
- Mõni katsealune võttis testimist kui võistlust ning ei leidnud kiirustamise tõttu esimeselt pakendilt kõiki allergeene üles.
- Allergeenide lisamisel käsitsi polnud erilist vahet, kas testija oli harjunud iOS või Android klaviatuuriga.

3.3 Rakenduse puudujäägid ja edasiarendus

Katsealused andsid lõpus ka ettepanekuid, kuidas võiks rakendust parandada. Need on järgmisena välja toodud:

- Lisada mingi otsimisvõimalus või automaatjätkamine (ingl *autocompletion*) allergeenide lisamiseks. (4 katsealust)
- Grupeerida valitud allergeenid ühte kohta, et kasutajal oleks parem ülevaade.
- Allergeene käsitsi lisades võiks kohe kirjutusriba ja klaviatuur aktiivne olla.

- Teeks rakenduse kiiremaks. (2 katsealust)
- Muuta välimus tagasihoidlikumaks.

Peale testimist implementeeriti otsingufunktsioon allergeenide lisamiseks ning koondati kasutaja lisatud allergeenid rohkem nähtavasse kohta. Lisati lühike õpetus, mida näidatakse kasutajale rakenduse esmakordsel käivitamisel. Rakendusest ehitati ka väljalase, kus polnud enam probleeme rakenduse kiirusega. Ainuke koht, kus rakendus mõnikord viivitus sekundi, oli kaamera initsialiseerimine. Autori peamiseks plaaniks on lisada puuduolevad juhendid kasutaja aitamiseks, iOS tööle saada ning inglise keel lisada. Samuti vajab edasist arendamisst skaneerimine.

Skaneerimise täpsuse seisukohast tuleks tegeleda erinevate disaini ja grammatika probleemidega. Kui allergeeni ees on väljend „ei sisalda“, allergeeni taga on sõna „vaba“ või allergeen ise on ilmaütlevas käändes, peaks rakendus need allergeenid läbi laskma. Kui allergeen on rea lõpus ja sidekriipsuga poolitatud, kontrollib rakendus poolitatud osasid eraldi. Esiteks on oht, et mingi soovimatu allergeen jääb tuvastamata. See puudutab pigem lühemaid sõnu, sest pikema sõna puhul on sarnasuslävend madalam ning tõenäoliselt tunneb rakendus selle ikkagi ära. Teiseks tõstab poolitamine valepositiivse tulemuse tõenäosust, sest tuvastatavas tekstis on rohkem ebaharilikke tähtede kombinatsioone, mis ei moodusta sõna. Tuvastatud tekst koosneb blokkidest, ridadest ja sõnadest. Sõnad on üksteisest tühikuga eraldatud elemendid reas. Seega ei tuvastata kahest sõnast koosnevat tühikuga eraldatud allergeeni koos. Sellisel juhul peaks kasutaja lisatud allergeenide hulgas olema mõlemad sõnad. Teine võimalus on valida ainult silmapaistvam/vähem levinud sõna (nt väljendi „paakumisvastased ained“ puhul on esimene sõna eristatavam). Kirjeldatud probleemide lahendamiseks tuleb sõnetöötluse algoritmi edasi arendada.

Kokkuvõte

Bakalaureusetöö raames loodi mobiilirakendus, millega saab toidupakendilt allergeenide sisaldust kontrollida tekstituvastuse abil. Rakendus on arendatud multiplatvorm arenduskeskkonnas Flutter. Lõputöö raames keskenduti Android versiooni arendamisele ja testimisele. Lahendus kasutab tekstituvastuseks Firebase ML Kit teenust.

Eesmärgi täitmiseks uuriti süvitsi tausta peatükis kirjeldatud teemasid ning võrreldi olemasolevaid rakendusi. Järgnevalt loodi rakenduse prototüüp ja viidi läbi rakenduse testimine erinevate kasutajatega. Lõpuks tehti rakenduses muudatusi vastavalt testijate tagasisidele.

Testimisest selgus, et rakenduse abil allergeenide otsimine osutus keskmiselt 1.8 korda kiiremaks kui käsitsi etikettide lugemine. Samuti saavutas lahendus parema täpsuse. Katsealused andsid mobiilirakendusele kümnepallisüsteemis hindeks 7.5 ning nõustusid rakendust allergikule soovitama. Kõige silmapaistvam oli kasutajate jaoks skaneerimise kiirus. Suurimaks probleemiks peeti otsingufunktsiooni puudust ja rakenduse kiirust. Mõlemad vead parandati peale testimist.

Tulevikus on plaanis täiustada sõnetöötluse algoritmi, et parendada skaneerimise täpsust. Veel peaks vaeva nägema juhendite kirjutamisega ja inglise keele lisamisega. Kindlasti areneb üldiselt tekstituvastus ja masinõpe, mis omakorda tõstab rakenduse jõudlust.

Viidatud kirjandus

[1] R. L. Peters, M. Krawiec, J. J. Koplin, A. F. Santos, Update on food allergy, *Pediatric Allergy and Immunology*, vol. 32, No 4, pp. 647-657, 2021.

[2] S. H. Sicherer, Epidemiology of food allergy, *Journal of Allergy and Clinical Immunology*, vol. 127, No 3, pp. 594-602, 2011.

[3] A. Muraro, T. Werfel, K. Hoffmann-Sommergruber, G. Roberts, K. Beyer, C. Bindslev-Jensen, V. Cardona, A. Dubois, G. duToit, P. Eigenmann, M. Fernandez Rivas, S. Halken, L. Hickstein, A. Høst, E. Knol, G. Lack, M. Marchisotto, B. Niggemann, B. Nwaru, N. Papadopoulos, L. Poulsen, A. Santos, I. Skypala, A. Schoepfer, R. Van Ree, C. Venter, M. Worm, B. Vlieg–Boerstra, S. Panesar, D. de Silva, K. Soares-Weiser, A. Sheikh, B. Ballmer-Weber, C. Nilsson, N. de Jong, C. Akdis, EAACI Food Allergy and Anaphylaxis Guidelines: diagnosis and management of food allergy, *Allergy: European Journal of Allergy & Clinical Immunology*, vol. 69, No 8, pp. 1008-1025, 2014.

[4] J. R. Cornelisse-Vermaat, J. Voordouw, V. Yiakoumaki, G. Theodoridis, L. J. Frewer, Food-allergic consumers' labelling preferences: a cross-cultural comparison, *European Journal of Public Health*, vol. 18, Oxford University Press, 2007, pp. 115-120.

[5] J. Preeti, M. Shideh, S. H. Sicherer, Interpretation of commercial food ingredient labels by parents of food-allergic children, *Journal of Allergy and Clinical Immunology*, vol. 109, New York, Mosby, 2002, pp. 1019-1021.

[6] Q. Ye, D. Doermann, Text Detection and Recognition in Imagery: A Survey, in *IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume: 37, Issue: 7, July 1 2015)*, S. Dickinson, eds. , Toronto, IEEE, 2014, pp. 1480-1500.

[7] Infood Team - Infood, 2021. https://play.google.com/store/apps/details?id=net.infood.app&hl=en&gl=US. (08.03.2021)

[8] A. Bhansali - Food Scanner, 2020. https://play.google.com/store/apps/details?id=com.getext.jain. (05.04.2021)

[9] Salty Nerd - Foodi, 2021. https://play.google.com/store/apps/details?id=com.SaltyNerd.Foodi. (05.04.2021)

[10] MaxSoft - Food Ingredients Scanner, 2019. https://apps.apple.com/us/app/food-ingredients-scanner/id1459475479. (16.04.2021)

[11] Good Snooze - Allergy & Vegan Scan - Soosee, 2021. https://apps.apple.com/us/app/allergy-vegan-scan-soosee/id1502026145. (16.04.2021)

[12] Open Food Facts - Open Food Facts, 2021. https://play.google.com/store/apps/details?id=org.openfoodfacts.scanner. (08.03.2021)

[13] A. Teder, Toiduallergia – kuidas ära tunda?, Südameapteek, 2019. https://www.sudameapteek.ee/tervisenouanded/toiduallergia-kuidas-ara-tunda/. (08.04.2021)

[14] S. Ramesh, Food allergy overview in children, *Clinical Reviews in Allergy and Immunology*, vol. 34, No 2, pp. 217-230, 2008.

[15] Põllumajandus- ja Toiduamet, Allergeenid, 2021. https://pta.agri.ee/ettevotjale-tootjale-ja-turustajale/toidu-tootmine/toidu-margistamine#allergeenid. (07.03.2021)

[16] Open Food Facts, Open Food Facts - World, 2021. https://world.openfoodfacts.org/. (16.04.2021)

[17] C. A. Monteiro, G. Cannon, J.-C. Moubarac, R. B. Levy, M. L. C. Louzada, P. C. Jaime, The UN Decade of Nutrition, the NOVA food classification and the trouble with ultra-processing, *Public Health Nutrition*, vol. 21, No 1, pp. 5-17, 2018.

[18] C. Julia, S. Hercberg, Development of a new front-of-pack nutrition label in France : the five-colour Nutri-Score, *Public Health Panorama*, vol. 3, No 4, pp. 712-725, 2017.

[19] R. Sharma, B. Kaushik, N. Gondhi, Character Recognition using Machine Learning and Deep Learning - A Survey, *2020 International Conference on Emerging Smart Computing and Informatics (ESCI)*, Pune, 2020.

[20] G. Khirbat, OCR Post-Processing Text Correction using Simulated Annealing (OPTeCA), *Proceedings of the Australasian Language Technology Association Workshop 2017*, Brisbane, 2017.

[21] A. Flôr, Text Segmentation, 2019. https://arthurflor23.medium.com/text-segmentation-b32503ef2613. (06.05.2021)

[22] H. Heitkötter, S. Hanschke ja T. A. Majchrzak, Evaluating Cross-Platform Development, *Web Information Systems and Technologies*, vol. 140, J. Cordeiro, K. H. Krempels, eds. , Porto, Springer, 2013, pp. 120-138.

[23] J. Johnson, D. Britch, C. Dunn, What is Xamarin?, 2020. https://docs.microsoft.com/et-ee/xamarin/get-started/what-is-xamarin. (15.04.2021)

[24] React Native, Core Components and Native Components, 2021. https://reactnative.dev/docs/intro-react-native-components. (15.04.2021)

[25] Flutter, Flutter architectural overview, 2020. https://flutter.dev/docs/resources/architectural-overview. (10.12.2020)

[26] Cleverism, Dart, https://www.cleverism.com/skills-and-tools/dart/. (10.12.2020)

[27] Dart, Platforms, 2020. https://dart.dev/platforms. (10.12.2020)

[28] Google, ML Kit, 2021. https://developers.google.com/ml-kit/guides. (15.04.2021)

[29] Google, Custom Models with ML Kit, 2020. https://developers.google.com/ml-kit/custom-models. (15.04.2021)

[30] Firebase, Text Recognition, 2021. https://firebase.google.com/docs/ml-kit/recognize-text. (17.04.2021)

[31] Firebase, Firebase, 2021. https://firebase.google.com/. (05.05.2021)

[32] Android Developers, Understand the Activity Lifecycle, 2020. https://developer.android.com/guide/components/activities/activity-lifecycle. (03.05.2021)

[33] Flutter, Shared preferences, 2021. https://pub.dev/packages/shared_preferences. (04.05.2021)

[34] Firebase, Firebase ML Vision, 2021. https://pub.dev/packages/firebase_ml_vision. (04.05.2021)

[35] Flutter, Camera, 2021. https://pub.dev/packages/camera. (04.05.2021)

[36] J. Landon, String similarity, 2021. https://pub.dev/packages/string_similarity. (29.04.2021)

[37] R. R. Shamir, Y. Duchin, J. Kim, G. Sapiro ja N. Harel, Continuous Dice Coefficient: a Method for Evaluating, Cornell University arXiv, New York, 2019.

[38] Android Developers, Save key-value data, 2020. https://developer.android.com/training/data-storage/shared-preferences. (27.04.2021)

[39] Flutter, Flutter's build modes, 2019. https://flutter.dev/docs/testing/build-modes. (05.05.2021)

Lisad

I Testimiseks kasutatud pakendid ja allergeenid

Pilt 4. Pakend 1 - Galetid (K-A: seesamiseemned, pärm, piim).

Pilt 5. Pakend 2 - Leivakrõpsud (K-A: soja, sinep, vadak).

Pilt 6. Pakend 3 - Sefiirid (K-A: piim, soja, glükoosisiirup).

Pilt 7. Pakend 4 - Energiajook (K-A: kofeiin, nisu, stabilisaatorid).

Pilt 8. Pakend 5 - Snickers (K-A: lõssipulber, muna, sarapuupähkel).

Pilt 9. Pakend 6 - Pesto (K-A: kašupähkel, päevalilleõli, piim).

II Testimise küsimustik

A. Küsimused enne testülesandeid:

1. Mis on sinu vanus?
2. Kui kaua oled nutitelefoni kasutanud?
3. Millisel ametil töötad/erialal õpid?
4. Kas praegu kasutad Android või iOS telefoni?
5. Mitut erinevat skaneerimisel põhinevat äppi oled varem kasutanud? (QR, Google Lens, Maxima äpp)

B. Küsimused pärast testülesandeid:

1. Mis hinnangu annad rakenduse üldmuljele kümnepallisüsteemis?
2. Mis hinnangu annad skaneerimisele kümnepallisüsteemis?
3. Mis hinnangu annad lisamisele kümnepallisüsteemis?
4. Kas kasutaksid rakendust tulevikus või soovitaksid allergikule?
5. Mis meeldis rakenduse juures?
6. Mis ei meeldinud rakenduse juures?
7. Milliseid ettepanekuid ja parendusi esitaksid rakendusele?

Tööga kaasa lisatud ZIP-failis on dokument nimega *vastused.xlsx*, milles on katsealuste vastused küsimustele.

III Testimise tulemused

Tabel 3. Testimise tulemused sekundites. Formaat on P*x*K*y*, kus *x* on pakendi number ja *y* katse number.

Testija	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
P0K1	52.18	19.95	17.76	10.5	32.18	20.71	15.43	36.72	21.61	15.48
P0K2	52.75	37.58	54.34	40.42	51.32	51.98	65	34	36.87	40.55
P0K3	17.27	15.21	5.44	10.63	34.66	10.83	6.07	15.79	6.43	11.48
P1K1	30.17	26.43	11.9	7.26	13.63	33.41	18.65	13.62	19.27	44.02
P1K2	21.28	27.45	53.39	23.83	24.6	27.22	23.62	33.95	21.22	28.62
P1K3	10.13	7.14	5.72	6.23	9.41	4.7	5.57	8.16	4.58	6.54
P2K1	44.54	40.94	38.03	28.91	23.51	46.51	40.86	46.47	34.08	29.58
P2K2	34.59	29.12	28.8	27.23	45.73	35.9	28.73	31.83	26.78	33.48
P2K3	60	60	21.11	23.23	60	58.17	23.82	15.42	26.27	60
P3K1	36.25	28.79	20.31	28.98	15.05	27.45	26.21	39.16	14.49	26.79
P3K2	31.04	38.33	28.77	31.03	58.86	43.05	25.22	31.13	25.48	26.89
P3K3	5.13	14.19	6.97	6.46	7.95	6.31	4.91	4.5	8.44	5.37
P4K1	32.2	19.17	8.25	7.11	11.25	5.89	16.76	10.9	13.85	17.79
P4K2	36.73	33.23	32.4	16.08	46.03	30.82	31.89	23.46	27.82	29.59
P4K3	4.93	8.13	5.71	12.2	5.07	4.42	4.18	4.78	10.38	10.15
P5K1	36.12	26.42	35.7	24.68	11.81	32.63	28.12	23.69	15.86	31.07
P5K2	43.81	39.13	33.2	26.97	47.91	32.28	35.59	20.34	24.34	33.09
P5K3	8.38	14.45	6.84	6.79	5.9	6.05	14.07	4.56	6.46	12.44
Lisamine	Man	Kat	Man	Kat	Man	Man	Man	Kat	Kat	Kat

IV Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Sander Kangur,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose „Mobiilirakendus allergeenide tuvastamiseks toidupakendilt“, mille juhendaja on Jakob Mass, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.

2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.

3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.

4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Sander Kangur

07.05.2021