TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Fiona Eliis Protas 222478IBB
Ksenija Jarmuhhamedova 22742IABB
Elena Bogdanova 222881IABB

Tallinna Kiirabi mobiilirakenduse arendamine kriitiliste juhendite ja protokollide kättesaadavuse parandamiseks

Bakalaureusetöö

Juhendajad: Viljam Puusep
MSc
Karl-Erik Karu
MSc

Tallinn 2025

Autorideklaratsioon

Kinnitame, et oleme koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autorid: Fiona Eliis Protas, Ksenija Jarmuhhamedova, Elena Bogdanova

20.05.2025

Annotatsioon

Lõputöö eesmärk oli arendada Tallinna Kiirabi jaoks mobiilirakendus, mis koondab kiirabitöötajatele vajalikud tegevusjuhendid, ravimite info ja annustamiskalkulaatori ühte võrguühenduseta toimivasse mobiilirakendusse, et kiirendada info kättesaadavust ja toetada ajakriitilisi otsuseid.

Rakendus loodi .NET MAUI platvormil, järgides puhta arhitektuuri põhimõtteid ja MVVM-mustrit. Arendusprotsessis rakendati agiilseid meetodeid ning tehti tihedat koostöö Tallinna Kiirabiga, et tagada funktsionaalsuste vastavus reaalsele töökeskkonnale.

Rakendus võimaldab kuvada ja otsida tegevusjuhendeid PDF-vormingus, kasutades pdf.js teeki, arvutada ravimite annuseid vastavalt patsiendi andmetele ning pääseda ligi RHK-10 veebilehele. Tulemuseks on mitmeplatvormiline funktsionaalne mobiilirakendus, mis parandab kiirabitöötajate tööprotsesse ja on valmis avalikustamiseks Google Play’s ja App Store’i.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 48 leheküljel, 6 peatükki, 16 joonist, 5 tabelit.

Abstract

Development of Tallinn Ambulance mobile application to improve the availability of critical guidelines and protocols

This thesis aims to develop a mobile application for Tallinn Ambulance to consolidate essential work-related information, including guidelines, medication data, and a dosage calculator, into a single offline-capable environment, thereby enhancing information accessibility and supporting time-critical decision-making.

The application was built using the .NET MAUI framework, adhering to clean architecture principles and the MVVM pattern. The development process employed agile methodologies, combining Scrum and Extreme Programming elements, and involved close collaboration with Tallinn Ambulance to ensure alignment with real-world operational needs.

The application enables the display and search of guidelines in PDF format using the pdf.js library, calculates medication dosages based on patient data, and provides access to the ICD-10 website. Testing included manual and automated unit tests to verify system reliability. Key challenges included integrating a language model for offline use and automatically detecting PDF table of contents, both of which proved time intensive. The result is a fully functional mobile application that streamlines ambulance workers' processes and is prepared for official deployment on Google Play and App Store platforms.

The thesis is in Estonian and contains 48 pages of text, 6 chapters, 16 figures, 5 tables.

Lühendite ja mõistete sõnastik

Backlog	Tööde nimekiri, mis vajavad teostamist
CRUD	*Create-Read-Update-Delete*
GoF	*Gang of Four*, neli autorit (Gamma, Helm, Johnson, Vlissides), kes defineerisid klassikalised objektorienteeritud kujundusmustrid
LLM	*Large Language Model*, suur keelemudel
LM	*Language Model*, väiksem keelemudel
MVVM	*Model-View-ViewModel*, arhitektuurimuster
O/RM	*Object-Relational Mapping*
PDF	*Portable Document Format*
POCO	*Plain Old CLR Object*, lihtne C# klass, mis esindab andmestruktuuri ega sisalda eripärast loogikat ega sõltuvusi .NET-i raamistike suhtes.
RHK-10	Rahvusvaheline haiguste klassifikatsioon, 10. versioon
SPC	*Summary of Product Characteristic*, ravimi omaduste kokkuvõte
SQLite	Kergekaaluline relatsiooniline andmebaasimootor, mis talletab kogu andmebaasi ühe failina seadme lokaalsesse mällu
UI	*User Interface*, kasutajaliides
URL	*Uniform Resource Locator*, ühtne ressursilokaator
XP	*Extreme Programming*

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Tervishoiusektori digitaliseerumine on viimastel aastatel olnud olulise tähelepanu all, kuna digilahendused aitavad parandada meditsiiniteenuste kvaliteeti, suurendada efektiivsust ja tagada paremat teabe kättesaadavust ajakriitilistes olukordades. Kiirabivaldkond on üks neist, kus töö iseloomu tõttu on infole kiire ligipääs eriti oluline ning paberpõhised lahendused ei vasta enam töö praktilistele vajadustele [1].

Mobiilirakenduste kasutamine meditsiini- ja päästevaldkonnas on tõestanud oma tõhusust, aidates parandada situatsiooniteadlikkust, kiirendada info leidmist ja toetada otsuste tegemise kvaliteeti [2] [3]. Tallinna Kiirabi on samuti väljendanud vajadust digilahenduse järele, mis koondaks kogu vajaliku info ühte kasutajasõbralikku ja turvalisse keskkonda.

1.1 Probleem

Kiirabitöö on ajakriitiline ja otsustusprotsessid peavad toimuma võimalikult kiiresti ja täpselt, sest iga viivitus võib mõjutada patsiendi ellujäämist ja ravi edukust. Samas on kiirabitöötajatel sageli vaja ligipääsu erinevatele ravijuhistele, protseduuridele ning ravimite annustamisreeglitele, mis on hetkel saadaval kas paberkandjal või PDF-dokumentidena. See tähendab, et olulist informatsiooni tuleb otsida mitmest eraldiseisvast allikast, mis aeglustab otsustamist ja suurendab vigade riski.

Digitaalsete lahenduste kasutuselevõtt meditsiinivaldkonnas on viimastel aastatel oluliselt kasvanud. Uuringud on näidanud, et elektrooniliste juhendite ja otsustustoe süsteemide kasutamine võib vähendada ravivigu ja parandada meditsiinitöötajate efektiivsust. Samuti on tõestatud, et mobiilirakenduste kasutamine meditsiini- ja päästevaldkonnas aitab parandada situatsiooniteadlikkust, kiirendada info leidmist ja parandada otsuste tegemise kvaliteeti [4].

Tallinna Kiirabi töötajad on ise märkinud, et neil puudub efektiivne lahendus, mis võimaldaks kiiret juurdepääsu olulistele juhistele, kuna olemasolevad materjalid on kas paberkandjal või PDF-kujul, kus mobiiltelefonis puudub kohene otsingufunktsioon.

Probleemi lahendamisele suunatud digitaalsed lahendused on muutumas hädavajalikuks tervishoiusüsteemis laiemalt. Näiteks on e-tervise süsteemide kasutuselevõtt vähendanud dokumenteerimisvigade arvu ja parandanud patsiendiandmete [5]. Samuti on näidatud, et mobiilsed lahendused võivad parandada koostööd ja informatsiooni jagamist kiirabibrigaadide ja haiglate vahel [6].

1.2 Eesmärk

Käesoleva lõputöö eesmärk on arendada Tallinna Kiirabi jaoks mobiilirakendus, mis koondab kiirabitöötajatele vajaliku tööalase informatsiooni – tegevusjuhendid, ravijuhised ja annustamisreeglid – ühte keskkonda. Rakendus võimaldab kiiret ja intuitiivset ligipääsu materjalidele nii võrgus kui ka võrguühenduseta, toetades seeläbi ajakriitiliste otsuste tegemist välitööde käigus.

Rakendus arendatakse varasema meeskonnaprojekti tulemusel loodud ärianalüüsi ja prototüüpide põhjal, kasutades .NET MAUI platvormi. Töö käigus keskendutakse süsteemi põhifunktsioonide, nagu juhendite kuvamine, otsing ja annustamiskalkulaator, arendamisele ning testimisele. Lisaks pööratakse tähelepanu kasutusmugavusele, andmete turvalisusele ja info ajakohasuse tagamisele.

Töö eesmärk on luua funktsionaalne mobiilirakendus, mida saab testida reaalses töökeskkonnas ning mis loob aluse süsteemi laiemaks kasutuselevõtuks Tallinna Kiirabis.

1.3 Töö struktuur

Käesolev bakalaureusetöö koosneb kuuest peamisest peatükist. Sissejuhatus tutvustab kiirabivaldkonna probleeme, töö eesmärki ja annab ülevaate lõputöö ülesehitusest. Metoodika peatükk kirjeldab arendatavat objekti, kasutatud tööriistu, arendusmetoodikat ning meeskonna rollijaotust. Nõuded peatükk käsitleb rakenduse ärilisi, funktsionaalseid ja mittefunktsionaalseid nõudeid, kasutusjuhte, kasutajalugusid, protsessimudelit ja esialgset prototüüpi. Töö tulemused peatükk esitab rakenduse arhitektuuri, andmemudeli, valminud funktsionaalsused, testimistulemused ja keelemudelite analüüsi. Analüüs ja järeldused hindab arendusprotsessi, funktsionaalsuste vastavust eesmärkidele, kasutajate tagasisidet, pakub edasiarendusvõimalusi ning toob esile meeskonna hinnangu ja iga liikme isikliku panuse. Kokkuvõte esitab töö peamised tulemused ja järeldused.

2 Metoodika

Käesolevas peatükis antakse ülevaade lõputöö käigus kasutatud arendusmetoodikast, rakendatud tööriistadest ja tehnoloogiatest. Kirjeldatakse arendusprotsessi üldjoontes ning tuuakse välja projekti raames tehtud valikud, mis toetasid eesmärgiks seatud kiirabi mobiilirakenduse loomist.

2.1 Objekti kirjeldus

Käesolev lõputöö on edasiarendus meeskonnaprojekti aine „ITB1706 Infosüsteemide arendamise meeskonnaprojekt: tellimus“ raames Tallinna Kiirabile loodud projektile. Meeskonnaprojekti raames töötati välja prototüüp mobiilirakendusele (vt peatükk 3.7), mille eesmärk oli pakkuda kiirabitöötajatele lihtsat ligipääsu tööks vajalikele juhendmaterjalidele.

Prototüüp võimaldas kasutajatel sirvida ja otsida teavet nelja peamise juhendmaterjali seast: „Kiirabitöötaja taskuraamat“, „Kiirabi tegevusjuhised“, „Kiirabi tegevusjuhendi skeemid“ ja „TraPS kaart“.

Lisaks juhendmaterjalide haldamisele (sh lisamine ja eemaldamine) sisaldas prototüüp ka annustamiskalkulaatori funktsionaalsust, mis võimaldas sisestada patsiendi vanuse, kehakaalu, ravimi ning manustamisviisi ja arvutada selle põhjal vastava ravimikoguse.

Meeskonnaprojekt keskendus esmajoones kasutajaliidese ja põhifunktsionaalsuse prototüüpimisele, kaasates ka esialgse visiooni andmevoogudest ja arhitektuurist.

Ettevõttega saavutatud kokkuleppe tulemusel otsustasid meeskonnaprojekti liikmed jätkata alustatud arendustööd bakalaureusetöö raames. Tööd jätkati samas kolmeliikmelises meeskonnas, kellega viidi läbi ka varasem meeskonnaprojekt.

Lõputöö käigus alustati rakenduse terviklikku arendamist, mis hõlmas töökeskkondade seadistamist, tarkvaraarhitektuuri kujundamist ning praktiliselt rakendatava lahenduse realiseerimist. Valminud rakendus sai nimeks Kiirabi App.

Üheks keskseks nõudeks oli see, et rakendus peab olema kasutatav olukorras, kus puudub internetiühendus, kuna kiirabitöötajad viibivad sageli piirkondades, kus andmeside pole tagatud. Sellest tulenevalt kujundati rakenduse arhitektuur nii, et kõik juhendid ja vajaminevad andmed oleksid lokaalselt seadmes kättesaadavad.

Projekti käigus toimus tihe koostöö tellijaga – Tallinna Kiirabi esindajatega korraldati regulaarseid kohtumisi, mille käigus esitleti arenduse vahetulemusi ning koguti tagasisidet nii funktsionaalsuse kui ka kasutajamugavuse osas. Kõik rakendusse lisatud funktsionaalsused kohandati vastavalt kiirabitöötajate reaalsetele töövajadustele, arvestades nende igapäevaseid tööprotsesse ja tehnilisi piiranguid välitöös.

2.2 Kasutatud tööriistad

Käesoleva lõputöö raames arendatud Tallinna Kiirabi mobiilirakenduse loomisel rakendati mitmesuguseid tööriistu, raamistikke ja tehnoloogiaid, mis toetasid arendusprotsessi kiirust, töökindlust ning lõplikult seatud nõuete täitmist. Arendustööriistade valik lähtus ühelt poolt projekti spetsiifilistest vajadustest – vajadusest töötada võrguühenduseta režiimis, võimaldada kiiret ja intuitiivset ligipääsu meditsiinilistele juhenditele ning integreerida tehisintellektil põhinevat otsingufunktsionaalsust –, teisalt aga ka autorite varasemast kogemusest ja tuttavlikkusest valitud tehnoloogiatega, mis võimaldas tõsta arendustöö efektiivsust ja kvaliteeti.

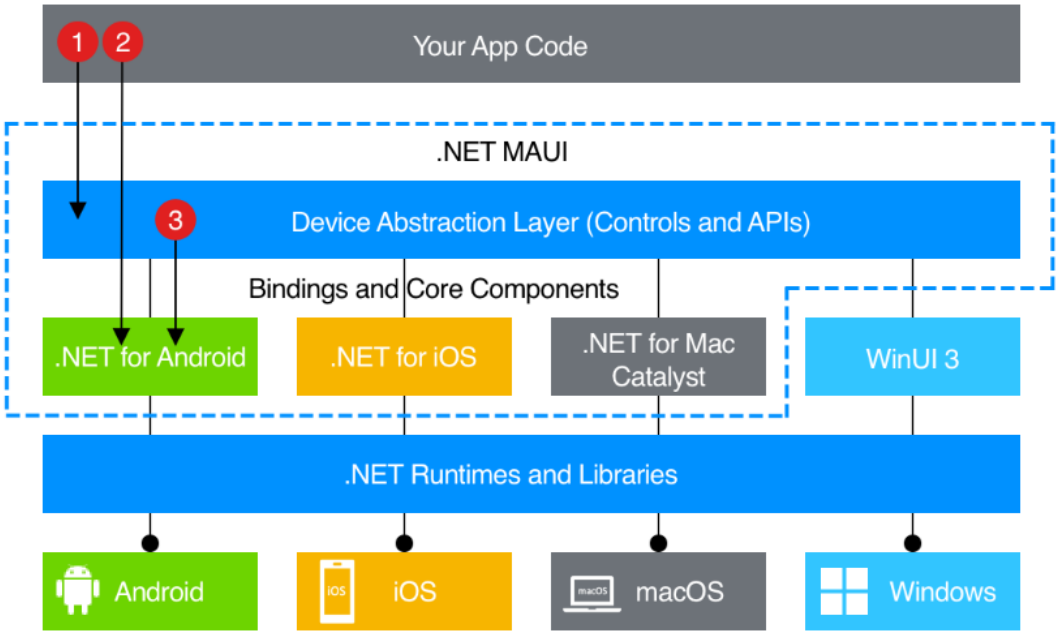
Tööriistade ja platvormi valikul oli oluline kriteerium ka see, et loodav rakendus peab olema kasutatav nii Androidi, iOS-i kui ka Windowsi seadmetes, tagades ühtlase kasutuskogemuse kõigil platvormidel.

Käesolevas alapeatükis antakse ülevaade kasutatud raamistikest ja programmeerimiskeeltest, tehnoloogilistest lahendustest, versioonihaldusest ning testimise ja valideerimise põhimõtetest.

2.2.1 Raamistikud ja programmeerimiskeeled

Projekti arendamiseks kasutati .NET MAUI (Multi-platform App UI) raamistikku, mis võimaldab luua ristplatvormilisi mobiilirakendusi ühise koodibaasi alusel [7]. MAUI sobib hästi olukordadesse, kus soovitakse arendada nii Androidi, Windowsi, MacOS-i kui ka iOS-i rakendust samaaegselt (vt Joonis 1).

Rakenduse loogika kirjutati C# programmeerimiskeeles, kasutades .NET-i ökosüsteemi. Kasutajaliidese loomiseks kasutati XAML-i, mis on deklaratiivne keel UI elementide struktureerimiseks .NET MAUI kontekstis [8].



Joonis 1. .NET MAUI rakenduse arhitektuuri kõrgetasemeline ülevaade [9].

Arendustöö toimus Visual Studio keskkonnas, mis pakkus tuge .NET MAUI projektide koostamiseks, testimiseks ja silumiseks erinevates seadmekeskkondades [10].

2.2.2 Tehnoloogiad

Rakenduse andmete talletamiseks ja päringute teostamiseks kasutati SQLite-i, mis võimaldab töödelda andmeid lokaalselt seadmes ilma internetiühenduseta. See valik oli tingitud kiirabitöötajate tööspetsiifikast, kus ühendusvõimalused võivad olla piiratud.

Rakenduses kasutati ka WebView komponenti, mille abil kuvati PDF-juhendeid kasutajale arusaadaval ja navigeeritaval kujul. WebView integreeriti avatud lähtekoodiga pdf.js teegiga, et võimaldada juhendite sisukorra kuvamist ja märksõnapõhist otsingut [11].

PDF-failidest teksti ekstraheerimiseks kasutati .NET-põhist teeki PdfPig, mis võimaldas lugeda juhendmaterjalide sisu programmiliselt ning töödelda neid otsinguks ja hilisemaks keelemudeli põhjal vastamiseks. PdfPig oli sobiv valik tänu oma heale .NET-

integreeritavusele ja võimalusele töödelda PDF-faile lokaalselt ilma väliste teenusteta [12].

Rakenduse prototüüp loodi esialgu Figma disainitööriistas, kus kujundati kasutajaliidese struktuur, visuaalsed komponendid ning kasutusvood [13]. Figma prototüüp võimaldas enne arendust tellijale ja meeskonnale visualiseerida, kuidas lõplik rakendus võiks toimida.

2.2.3 Versioonihaldus

Versioonihaldus toimus Git platvormil, kasutades GitLabi. Git võimaldas arendajatel jälgida muudatusi, taastada varasemaid versioone ja säilitada arendusajaloo läbipaistvust [14].

GitLabi kasutati ka tööde planeerimiseks ja kasutajalugude kirjeldamiseks. Tööde nimekirja haldamine ja arendustsüklite planeerimine toimus GitLabi ülesannete kaudu, mis võimaldas meeskonnal hoida ühist ülevaadet projekti edenemisest.

2.2.4 Testimine

Rakenduse arenduse käigus viidi läbi mitmetasandiline testimine, et tagada funktsionaalsuse korrektsus, töökindlus ja vastavus kasutajate ootustele. Kasutati nii ühikteste, manuaalset testimist kui ka lõppkasutajatepõhist valideerimist. PDF-juhendite korrektsele kuvamisele.

Ühiktestide loomiseks loodi eraldi testiprojekt .NET 9 platvormil. Testiraamistikuna kasutati xUnit, mis on sobilik kaasaegsete .NET projektide testimiseks. Sõltuvuste testimiseks kasutati Moq teeki. .NET MAUI UI-komponentide testimiseks kasutati Microsoft.Maui.Mocks teeki.

Lisaks automatiseeritud testidele viidi läbi manuaalne testimine, mille käigus kontrolliti visuaalset käitumist, otsingu- ja sirvimisfunktsioone ning kasutajaliidese loogikat. Testimise käigus kasutati Excelit testjuhtumite jälgimiseks ning tulemuste dokumenteerimiseks. Ka annustamiskalkulaatori väljundeid valideeriti Exceli abil, kus testiti erinevaid sisendite kombinatsioone ja võrreldi neid kontrollandmetega.

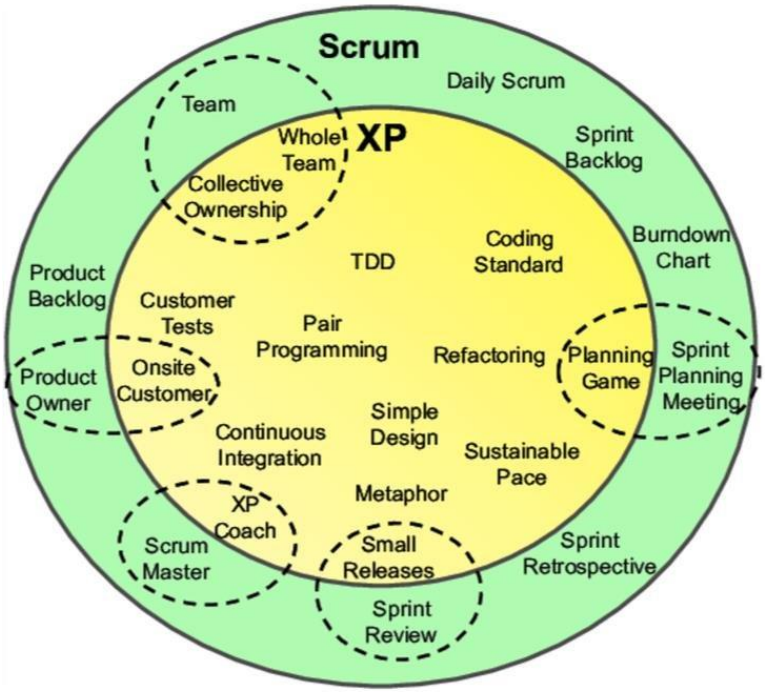
Keelemudeli täpsuse ja rakendatavuse hindamisel kasutati toetava tööriistana ka ChatGPT-d, mille abil võrreldi erinevate mudelite vastuseid ja hinnati semantilise sobivuse taset.

2.2.5 Dokumenteerimine ja info otsimine

Lõputöö raames kasutati mitmete ülesannete jaoks keelemudeleid, nagu ChatGPT [15], DeepSeek [16], Grok [17], Gemini [18] ja Claude [19]. Keelemudeleid rakendati eeskätt ideede struktureerimiseks ja akadeemilises stiilis sõnastamise, näiteks peatükkide esialgsete versioonide loomisel kasutatud sõnastuse lihvimisel. Samuti kasutati neid ka sihipäraseks infootsinguks, et leida asjakohaseid tehnilisi allikaid ja selgitusi, mis toetasid rakenduse arhitektuuri kujundamist ning andmehalduse lahenduste valikut. Kõiki vastuseid käsitleti abimaterjalina ning neid analüüsiti ja kohandati iseseisvalt, et tagada töö korrektsus ja vastavus teadustöö nõuetele.

2.3 Arendusmetoodika

Arendustöö käigus rakendati agiilse-tarkvaraarenduse põhimõtteid, kombineerides Scrumi ja Extreme Programming'u (XP) elemente [20]. Agiilne lähenemine võimaldas kiiresti kohaneda muutuva kasutajasisendiga ning säilitada pideva väärtuse loomise fookuse. Scrum pakkus raamistiku iteratiivseks tööks, kus fookus oli regulaarsetel arendussprintidel, tööde prioriseerimisel ning tiimikoosolekutel, samas kui XP tõi juurde tehnilisi praktikaid nagu pidev integreerimine, paarisprogrammeerimine ning testipõhine arendus [21]. XP ja Scrum'i elementide kombineerimine võimaldas leida tasakaalu struktuuri ja paindlikkuse vahel (vt Joonis 2). Just XP praktikad sobitusid hästi väikese arendustiimi tööviisiga, kus rollid olid paindlikumalt jaotatud ja arendus kulges intensiivses koostöös juhendajatega.



Joonis 2. Scrum/XP hübriid [22].

Metoodika valikul lähtuti vajadusest tagada nii paindlikkus kui ka arendusprotsessi tehniline kvaliteet, arvestades samal ajal väikeprojekti piiratud ressursse ja tiimikoosseisu. Kolmeliikmeline arendusmeeskond kohandas metoodikaid vastavalt enda suurusele ja tööspetsiifikale, säilitades paindlikkuse, iteratiivse lähenemise ning pideva koostöö kiirabipoolsete juhendajatega.

2.3.1 Töökorraldus ja planeerimine

Arendustöö alguses lepiti ülikoolipoolsete juhendajatega kokku konkreetne tööplaan koos tähtajaliselt fikseeritud vahe-eesmärkidega. See aitas tagada, et arendus edeneks järjepidevalt ning et kõik olulised funktsionaalsused oleksid valmimise hetkeks realiseeritud ja testitud.

Arendusprotsess jaotati viieks järjestikuseks etapiks, millel olid selged eesmärgid ja tähtajad:

1. **Keskkondade seadistamine ja koodi repositooriumi loomine:** eesmärgiks oli luua projektistruktuur, seadistada versioonihaldus Git platvormil ning arenduskeskkonnas Visual Studio. Samuti tuli ette valmistada kõik vajalikud tehnilised tingimused .NET MAUI platvormi kasutamiseks.

2. **Taaskasutatavate komponentide ja mallide loomine:** arenduse algfaasis seati eesmärgiks luua üldised kasutajaliidese mallid ja korduvkasutatavad UI-komponendid (sh nupud, otsinguribad, kaardid), et tagada hilisema arenduse ühtne disain ja kiirem töövoog.
3. **Juhendite kuvamise funktsionaalsuse kavandamine ja realiseerimine:** selle etapi sihiks oli võimaldada kasutajatel rakenduses tegevusjuhendeid sirvida ja lugeda kasutajasõbralikul kujul, kasutades sobivaid visuaalseid lahendusi (nt WebView-komponenti või PDFiumSharp).
4. **Otsingufunktsionaalsuse arendamine juhendmaterjalides:** eesmärgiks oli võimaldada märksõnapõhine otsing juhendites ning katsetada loomuliku keele põhist otsingut, kasutades keelemudeleid (LM ja LLM), et toetada kiiret ja sisupõhist infokättesaadavust.
5. **Annustamiskalkulaatori kavandamine ja arendus:** selle etapi fookuses oli arendada tööriist, mis võimaldaks kasutajal arvutada sobiv ravimidoos lähtudes patsiendi andmetest ja manustamisviisist, muutes selle praktiliseks töövahendiks välitingimustes.

Kõik funktsionaalsused planeeriti arendusiteratsioonidena kestusega keskmiselt 2–3 nädalat. Iga iteratsiooni lõpuks seati eesmärgiks saavutada kooskõlastatud ja testitud vahetulemus, et toetada järgneva arendusetapi sujuvat jätkumist. Iteratsioonide lõppu planeeriti regulaarseid hindamisi, mille kaudu analüüsiti tehtud tööd, hinnati meeskonnatööd ning tehti ettevalmistused järgmisteks tegevusteks.

2.3.2 Keelemudelite testimise protsess ja valikukriteeriumid

Ühe arendusprotsessi osana käsitleti semantilise otsingu võimalust juhendmaterjalides, kasutades keelemudeli-põhist lähenemist. Enne lahenduse lõplikku teostamist oli oluline hinnata, kas taoline süsteem on tehniliselt üldse rakendatav mobiilse rakenduse kontekstis – arvestades nii seadmete võimekust, töökiiruse nõudeid kui ka võrguühenduseta töötamise piiranguid.

Selleks viidi läbi eelnev analüüs ja katsetused, mille eesmärk oli vastata järgmistele uurimisküsimustele:

- Kas keelemudeli-põhine semantiline otsing on tehniliselt realiseeritav lokaalselt (ilma serveripõhise töötluseta)?

- Milline mudeli suurus ja tüüp sobib nutiseadmetel kasutamiseks, ilma et see mõjutaks jõudlust negatiivselt?
- Milline lahendus pakub sobiva tasakaalu täpsuse, kiiruse ja ressursikasutuse vahel?
- Kas sellise lähenemise rakendamine toob kasutajale praktilist väärtust võrreldes traditsioonilise stringiotsinguga?

Testimisse kaasati kolm erinevat lähenemist, mille võimekust hinnati mitmest aspektist:

- REST + lokaalne kasutajaliides – rakendus kasutab lokaalselt seadmes töötavat kasutajaliidest, mis suhtleb serveriga REST API kaudu. See võimaldab vajadusel serveripoolset töötlust, kuid kasutajaliidese loogika ja põhifunktsionaalsus toimuvad seadmes kohapeal.
- LLM + lokaalne töötlemine – rakendusse on integreeritud võimalus jooksutada suuri keelemudeleid lokaalselt seadmes.
- LM – väiksem ja optimeeritud mudel, mille saab integreerida otse rakendusse.

Analüüsi käigus selgitati välja, millised mudelid on üldse tehniliselt sobilikud .NET MAUI kontekstis, milline on nende maht, tööaeg, mälukasutus ja täpsus, ning kas neid on võimalik jooksutada lokaalselt erinevates operatsioonisüsteemides (Android, iOS, Windows).

Valikuprotsessi tulemus, võrdlusmudelite omadused ja tehniline analüüs on esitatud peatükis 4.5.

2.3.3 Kommunikatsioon ja tagasiside

Töötamine toimus igapäevase suhtluse kaudu Microsoft Teamsi keskkonnas ja Facebook Messengeri grupis, kus jagati vahetult infot probleemide, lahenduste ja progressi kohta. Lisaks toimusid iga iteratsiooni lõpus koosolekud Tallinna Kiirabi esindajatega, mille käigus tutvustati vahepealseid arendustulemusi ning koguti sisulist tagasisidet. Tagasiside põhjal korrigeeriti funktsionaalsusi, kohandati otsingulahendust ning arendati kasutusloogikat vastavalt kiirabitöötajate tegelikele töötingimustele ja vajadustele.

2.4 Rollid ja meeskonnadünaamika

Erinevalt klassikalisest Scrumist ei jaotatud meeskonnas rangeid rolle (nt Scrum-meister, tooteomanik ja arendustiim), vaid kõik liikmed panustasid vastavalt oma tugevustele ja töövoo vajadustele [23]. Igaüks tegeles nii arenduse, testimise, dokumenteerimise kui ka disainiga, mis suurendas tiimisisest paindlikkust ja teadmiste jagamist.

Töö alguses koostati juhendajate ja meeskonnaga koosolekul eesmärkide ja tähtaegade jaotus, mille alusel planeeriti tööetapid. Iga tähtaja järel toimusid vahekokkuvõtete koosolekud nii juhendajatega kui ka tellija (Tallinna Kiirabi) tiimiga, et esitleda tehtud tööd ja saada tagasisidet. Lõputöö lõppfaasis tihenes juhendajatega suhtlus, et tagada töö vastavus akadeemilistele nõuetele ja sisulistele ootustele.

Lisaks toimusid kogu arendustöö vältel meeskonna sisemised koosolekud, kus vahetati infot juba tehtud ülesannete kohta, seati uued prioriteedid ning jaotati järgmised tegevused. Selline regulaarne suhtlus aitas hoida tempot, vähendada arusaamatusi ning soodustas tiimitööd.

3 Nõuded

Selles peatükis antakse ülevaade rakenduse nõuetest, kasutusjuhtudest, protsessimudelitest ja prototüübist. Rakenduse esialgsed ärilised, funktsionaalsed ja mittefunktsionaalsed nõuded töötati välja Tallinna Kiirabi meeskonnaprojekti raames. Nõuete määratlemisel koguti sisendit Tallinna Kiirabi projektimeeskonnalt, kellel on igapäevane kokkupuude kiirabitöö korralduse ja tööprotsessidega. Lisaks analüüsiti turul olemasolevaid rakendusi ja tööriistu, et kaardistada funktsionaalsusi, mida praegused lahendused ei kata, kuid mis on kiirabitöötajate jaoks olulised. Selline kombinatsioon võimaldas määratleda esialgse funktsionaalsusnõuete kogumi, millele tugines ka varajane prototüüp.

Lõputöö käigus on neid nõudeid kohandatud ja täiendatud vastavalt arenduse jooksul ilmnenud vajadustele ning kasutajate tagasisidele. Eelkõige lähtuti Tallinna Kiirabi töötajate (rakenduse lõppkasutajate) valideerimisest ja kasutajatestidest. Muudatused hõlmasid uute funktsionaalsuste lisamist, mittevajalikuks osutunud lahenduste eemaldamist ning olemasolevate töövoogude optimeerimist.

Rakenduse nõuete eesmärgiks on kirjeldada süsteemi toimimist ja funktsionaalsust. Üldjuhul jaotatakse nõuded kolmeks – ärilised, funktsionaalsed ja mittefunktsionaalsed. Äriliste nõuete eesmärk on kirjeldada süsteemi loomise põhjuseid, ärilisi eesmärke ja vajadusi, mida arendatav rakendus peab täitma. Käesoleval juhul oli eesmärk parandada kiirabitöötajate ligipääsu tööalastele juhenditele, infole ja töövahenditele ning vähendada vajadust käsitsi PDF-ide haldamiseks. Funktsionaalsed nõuded kirjeldavad süsteemi käitumist ja funktsioone kasutaja seisukohalt [24]. Mittefunktsionaalsete nõuete eesmärk on täpsustada tarkvara toimimist üldisemalt, nagu jõudlus, kasutatavus ja ligipääsetavus [25].

Kasutusjuhtude koostamisel modelleeriti tüüpilised stsenaariumid, kuidas kiirabitöötajad rakendust reaalses töös kasutavad – näiteks konkreetse juhendi kiire leidmine tööväljakutsel. Kasutajalood täiendavad kasutusjuhte, tuues esile kasutajate vajadusi lihtsas ja narratiivses vormis. Protsessimudelid kirjeldavad millised on töövood

konkreetsete funktsioonide kasutamisel. Prototüüp loodi Figma keskkonnas ning see aitas testida varajasi kujundusi ja saada tagasisidet kasutajaliidese loogika ning funktsionaalsuse kohta enne lõpliku süsteemi arendamist.

3.1 Ärilised nõuded

Alljärgnevas tabelis (vt Tabel 1) on esitatud rakenduse ärilised nõuded, mis kirjeldavad süsteemi poolt täidetavaid ärilisi eesmärke ja ootusi.

Tabel 1. Rakenduse ärilised nõuded.

Kood	**Äriline nõue**
Ä1	Optimeerida tööprotsesse, vähendades aega ja ressursse, mis kuluvad manuaalsele infootsingule ja paberdokumentide haldamisele ja ravijuhiste leidmisele.
Ä2	Tagada, et kogu asutusesisene info oleks hõlpsasti ja kiiresti kättesaadav ühest tsentraliseeritud mobiilirakendusest.

3.2 Funktsionaalsed nõuded

Alljärgnevas tabelis (vt Tabel 2) on esitatud rakenduse funktsionaalsed nõuded, mis kirjeldavad süsteemi käitumist ja selle poolt pakutavaid funktsioone vastavalt kasutaja ootustele.

Tabel 2. Rakenduse funktsionaalsed nõuded.

Kood	**Funktsionaalne nõue**
Tegevusjuhenditega seotud funktsionaalsed nõuded	
F1	Rakendus peab sisaldama järgnevate tegevusjuhendite põhjalikke digitaalseid versioone: „Kiirabi taskuraamat“, „Kiirabi tegevusjuhendi skeemid“, „Kiirabi tegevusjuhend“ ning „TRAPS-kaart“. Need juhendid peavad olema üles laaditud ja kasutajale kergesti kättesaadavad.
F2	Juhendite kuvamine rakenduse avalehel peab toimuma kindlas järjestuses: 1) „Kiirabi tegevusjuhendi skeemid“, 2) „Kiirabi tegevusjuhend“, 3) „Kiirabi taskuraamat“, 4) „TRAPS-kaart“. Järjestus peab kajastama kiirabitöötajate töövoogu ja praktilist eelistust enimkasutatud materjalide põhjal.

Kood	Funktsionaalne nõue
Tegevusjuhenditega seotud funktsionaalsed nõuded	
F3	Kogu rakenduses sisalduv informatsioon peab tuginema Tervisekassa poolt kinnitatud tegevusjuhenditele ning ravimi omaduste kokkuvõttele ehk SPC-le.
F4	Kuvatav teave ravimi näidustuste, manustamisviiside ja annustamissoovituste kohta peab vastama üks ühele üleslaetud juhendites sisalduvale sisule. Rakendus ei tohi seda teavet iseseisvalt tõlgendada ega täiendada.
F5	Kui tegevusjuhendis on sisukord (PDF-failis), peab rakendus võimaldama kasutajal sirvida juhendit peatükkide kaupa, kasutades sisukorra struktuuri. Sisukorra alusel peab kasutaja saama liikuda otse soovitud peatükini.
Otsingu funktsiooni funktsionaalsed nõuded	
F6	Kasutajad saavad otsida haiguste diagnoose ja ravimite infot rahvusvaheliste diagnoosikoodide (nt ICD-10) ja märksõnade abil. Avalehe otsing suunab kasutaja otse RHK-10 (rahvusvahelise haiguste klassifikatsiooni) veebiotsingule.
F7	Kasutaja saab valitud tegevusjuhendi piires otsida märksõnu, mille tulemused kuvatakse rippmenüüs koos vastava tekstikatkendi ja leheküljenumbriga. Valides tulemuse, suunatakse kasutaja otse juhendis vastavasse kohta.
F8	Juhendi sees teostatud otsing võimaldab kasutajal liikuda vastete vahel „eelmine“ ja „järgmine“ nuppude abil, mis on rakendudes kuvatud vastavate ikoonidega.
Annustamiskalkulaatori funktsionaalsed nõuded	
F9	Kasutaja saab valida või otsida ravimit nimekirjast. Pärast ravimi valimist kuvatakse kasutajale vastava ravimi üldinfo, manustamisinfo, kõrvaltoimed ning vastunäidustused. Ravimite nimekiri ja nendega seotud andmed on pärit failist Kiirabi_taskuraamat_2024_v4_1.
F10	Kasutaja saab sisestada patsiendi kehakaalu (kg) ning annustamise aluseks oleva doseeringu (nt mg/kg, µg/kg), mille ta võtab valitud ravimi juures kuvatavast infost. Selle põhjal arvutab rakendus automaatselt ravimi annuse (nt mg, µg, g) järgmise valemiga: Annus = Kehakaal × Doseering.
F11	Lahuse korral saab kasutaja sisestada ka ravimi kontsentratsiooni (nt mg/mL, µg/mL). Sellisel juhul arvutab rakendus ka vajaliku vedeliku hulga milliliitrites valemiga: Vedeliku doos = (Kehakaal × Doseering) / Kontsentratsioon.

Kood	Funktsionaalne nõue
Annustamiskalkulaatori funktsionaalsed nõuded	
F12	Kui kasutajal on juba teada ravimi annus, saab ta selle sisestada käsitsi. Sellisel juhul arvutab rakendus vedeliku doosi ainult annuse ja kontsentratsiooni alusel: Vedeliku doos = Doos / Kontsentratsioon.
F13	Kasutaja saab muuta ühikuid ning vastavad väärtused värskendatakse automaatselt.
F14	Kasutaja saab klõpsata nuppu „Lähtesta andmed“, mis puhastab kõik sisestusväljad ja taastab vaikeseaded.

3.3 Mittefunktsionaalsed nõuded

Alljärgnevas tabelis (vt Tabel 3) on esitatud rakenduse mittefunktsionaalsed nõuded, mis määratlevad süsteemi kvaliteediomadused.

Tabel 3. Rakenduse mittefunktsionaalsed nõuded.

Kood	Mittefunktsionaalsed nõuded
MF1	Süsteem peab olema kasutajasõbralik ja intuitiivne.
MF2	Süsteemi värskendamine ja vigade parandamine peab olema lihtne.
MF3	Süsteem peab olema juurdepääsetav kõigi levinud operatsioonisüsteemide kasutajate jaoks (Android, iOS, Windows)
MF4	Rakendus peab olema kasutajatele ligipääsetav ka ilma internetiühenduseta, võimaldades kasutada lokaalselt salvestatud juhendeid ja kalkulaatorit.
MF5	Rakendus peab tagama, et tundlikku teavet ei saa volitamata isikud muuta.

3.4 Kasutusjuhud

Alljärgnevas tabelis (vt Tabel 4) on kirjeldatud rakenduse peamised kasutusjuhud, mis toovad välja tüüpilised interaktsioonid kasutaja ja süsteemi vahel konkreetsete eesmärkide täitmiseks.

Tabel 4. Rakenduse kasutusjuhud.

Kasutusjuhud
Tegevusjuhendite vaatamine Tegutseja: Kiirabitöötaja Eeldused: Rakendus on avatud, kasutaja on avalehel. Kirjeldus: 1. Kasutaja liigub tegevusjuhendite menüüsse. 2. Kasutaja sirvib tegevusjuhendite ja materjalide loetelu. 3. Kasutaja valib vajamineva tegevusjuhendi. 4. Süsteem kuvab valitud tegevusjuhendi täissisu.
Ühest tegevusjuhendist otsimine märksõnade alusel Tegutseja: Kiirabitöötaja Eeldused: Rakendus on avatud, kasutaja on avalehel tegevusjuhendite menüüs. Kirjeldus: 1. Kasutaja valib vajaliku tegevusjuhendi. 2. Süsteem kuvab valitud tegevusjuhendi täissisu. 3. Kasutaja sisestab otsinguväljale märksõna (nt sümptom, ravim). 4. Süsteem kuvab otsingutulemused koos tekstilõigu ja leheküljenumbriga. 5. Kasutaja valib tulemuse ja rakendus navigeerib vastavale leheküljele. 6. Kasutaja saab liikuda tulemuste vahel „Eelmine“ ja „Järgmine“ nuppude abil. 7. Kui juhendil on sisukord, saab kasutaja ka sisukorra kaudu liikuda soovitud peatükki.

Kasutusjuhud
Annustamiskalkulaatori kasutamine Tegutseja: Kiirabitöötaja Eeldused: Rakendus on avatud, kasutaja on avalehel. Kirjeldus: 1. Kasutaja liigub annustamiskalkulaatori vaatesse. 2. Kasutaja valib või otsib ravimi nimekirjast vajaliku ravimi. 3. Süsteem kuvab valitud ravimi üldinfo, manustamisinfo, kõrvaltoimed ja vastunäidustused. 4. Kasutaja sisestab patsiendi kehakaalu (kg). 5. Kasutaja sisestab doseeringu (nt mg/kg, µg/kg). 6. Süsteem arvutab automaatselt ravimi annuse (nt mg, µg, g). 7. Kui tegemist on lahusega, saab kasutaja sisestada ka lahuse kontsentratsiooni (nt mg/mL). 8. Süsteem arvutab automaatselt vajaliku lahuse koguse. 9. Kui kasutajal on teada annus, saab ta selle sisestada käsitsi ja süsteem arvutab lahuse koguse. 10. Kasutaja saab klõpsata nuppu „Lähtesta andmed“, mis puhastab kõik sisestusväljad ja taastab kalkulaatori vaikeseaded.
RHK-10 lingi kasutamine Tegutseja: Kiirabitöötaja Eeldused: Rakendus on avatud, kasutaja on avalehel. Kirjeldus: 1. Kasutaja klõpsab avalehel „Ava RHK-10 veebilehe otsing“. 2. Süsteem kontrollib, kas kasutajal on internetiühendus. 3. Kui internetiühendus on olemas, kasutajat suunatakse RHK-10 ametliku veebilehele. 4. Kui internetiühendus puudub, süsteem kuvab veateate

3.5 Kasutajalood

Alljärgnevas tabelis (vt Tabel 5) on esitatud rakenduse kasutajalood, mis kirjeldavad süsteemi funktsionaalsust lõppkasutaja vaatenurgast, tuues esile nende vajadused ja ootused.

Tabel 5. Rakenduse kasutajalood.

Kasutajalood
Tegevusjuhendite vaatamine Kiirabitöötajana, tahan pärast rakenduse avamist näha loetelu tegevusjuhenditest ja materjalidest, et saaksin kiiresti leida ja avada vajaliku juhendi ning tutvuda selle sisuga.
Ühest tegevusjuhendist otsimine märksõnade alusel Kiirabitöötajana, soovin otsida märksõna konkreetse juhendi sisust, et leida kiiresti vajalik info ja säästa aega kriitilises olukorras
Annustamiskalkulaatori kasutamine Kiirabitöötajana, soovin arvutada ravimi annuse patsiendi kehakaalu põhjal, et manustada õige kogus ravimit ja vältida vigu.
RHK-10 lingi kasutamine Kiirabitöötajana, ma tahan rakenduse avalehel klõpsata RHK-10 veebilehe lingile, et saaksin kiiresti ja lihtsalt avada RHK-10 veebilehe ning otsida haiguste diagnoose ja klassifikatsioone.

3.6 Protsessimudel

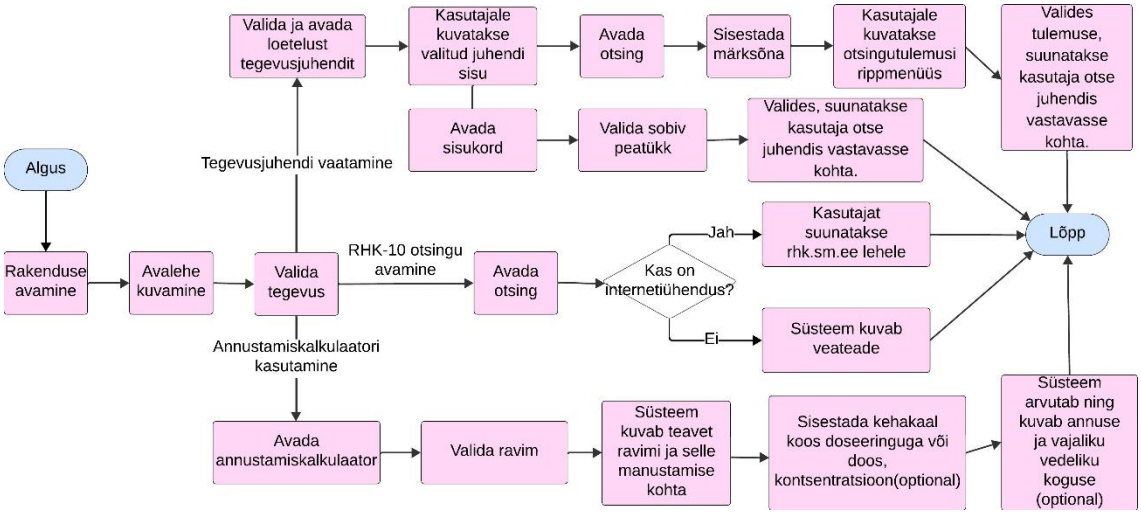
Kiirabitöötaja haldusprotsessi mudel kirjeldab töövoogu, mille abil kiirabitöötajad saavad kasutada annustamiskalkulaatori, vaadata tegevusjuhendeid ja otsida tegevusjuhenditest märksõnade alusel. Protsess algab rakenduse avamisega, mille järel kuvatakse avaleht. Avalehel on teenuste kataloog, mis sisaldab tegevusjuhendite loetelu, annustamiskalkulaatorit ja RHK-10 linki. Kiirabitöötaja saab valida ühe järgmistest tegevustest:

1. Tegevusjuhendi vaatamine**:** Kiirabitöötaja valib loetelust sobiva tegevusjuhendi ja avab selle sisu. Lisaks on võimalik teha märksõnaotsing konkreetse tegevusjuhendi sees, mille puhul kuvatakse vasted ning nende vahel saab valida ja liikuda. Võimalusel saab sisukorra kaudu navigeerida.
2. Annustamiskalkulaatori kasutamine**:** Kiirabitöötaja avab kalkulaatori, valib sobiva ravimi ning sisestab kas patsiendi kehakaalu koos ravimi doseeringuga või

otse ravimi annuse. Vajadusel saab sisestada ka ravimi kontsentratsiooni. Süsteem kuvab täpset teavet ravimi ja selle manustamise kohta ning arvutab automaatselt ja kuvab täpse annuse ning vajaliku vedeliku koguse, kui tegemist on lahusega. Saadud tulemust saab kasutada otse patsiendi ravimisel.

3. RHK-10 otsingu avamine**:** Kiirabitöötaja vajutab nupule avalehel, mille pärast suunatakse ta veebilehele rhk.sm.ee [26]. Ilma internetiühenduseta jääb see toiming teostamata.

Allpool on visualiseeritud vooskeem (vt Joonis 3), mis näitab tegevuste järjestust ja olulisi otsustuskohti, et hõlbustada rakenduse tööprotsesside mõistmist. Protsessimudel on vastavuses funktsionaalsete nõuetega.



Joonis 3. Kiirabitöötaja haldusprotsessi protsessimudel.

3.7 Prototüüp

Tulevikulahenduse väljatöötamise üheks olulisemaks komponendiks ning arenduse eeltööks oli prototüübi loomine, mis vastab Tallinna Kiirabi töötajate vajadustele ning toetab nende tööprotsesse. Prototüübi disainis ja funktsionaalsustes lähtuti kiirabitöötajate kui lahenduse peamiste kasutajate soovidest ja vajadustest, mis selgitati välja analüüsi ja intervjuude käigus kiirabimeeskonna töötajatega.

Prototüübi loomine toimus mitmes iteratsioonis, millest igaühele eelnes põhjalik vajaduste analüüs Tallinna Kiirabi esindajatega.

Prototüübi väljatöötamisel lähtuti järgmistest põhimõtetest:

1. **Kasutajaliides:** prototüübi kasutajaliides järgib kaasaegseid disainipõhimõtteid, tagades kasutajasõbralikkuse ja funktsionaalsuse.
2. **Mobiilne kohaldatavus:** kasutajaliides on optimeeritud mobiilseadmetele, et tagada sujuv kasutuskogemus ka välitingimustes.
3. **Ligipääsetavus:** disain vastab ligipääsetavuse standarditele, võimaldades rakenduse kasutamist erinevate vajadustega töötajatel.
4. **Kasutajate soovid:** kasutajaliidese loomisel lähtuti kiirabitöötajate peamistest soovidest:
    a. Lihtne ja intuitiivne navigeerimine kiireloomulistes olukordades;
    b. Selge ja ülevaatlik disain olulise teabe esiletoomiseks;
    c. Kõik vajalik ühes kohas, vähendades vajadust manuaalse infootsingu järele.

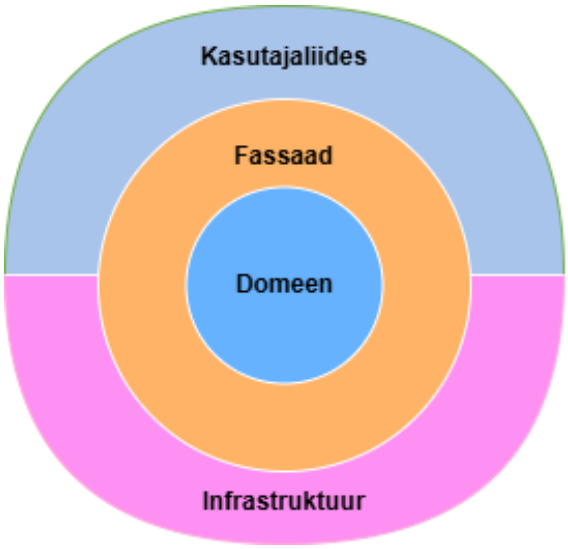
Prototüüp kujutab endast rakenduse varajast versiooni, mis loodi meeskonnaprojekti aine raames ning eelnevalt arendusprotsessile, et visualiseerida põhifunktsionaalsusi ja kasutajaliidest. Arenduse käigus tehti olulisi muudatusi, mistõttu valminud rakenduse välimus ja funktsionaalsus erinevad prototüübist märkimisväärselt. Näiteks eemaldati avalehelt üldine otsingu-funktsionaalsus ja lisati juhendi-sisene otsing, mis vastab kiirabi meeskonna tagasisidele. Protüübi kujundus on näidatud Lisas 2, mis illustreerib lahenduse esialgset visiooni.

4 Töö tulemused

Selles peatükis esitatakse Tallinna Kiirabi mobiilirakenduse arendustöö tulemused. Kirjeldatakse rakenduse arhitektuuri, andmemudelit, valminud funktsionaalsusi, väliste teekide kasutust ning testimise tulemusi. Eraldi tuuakse välja ka LLM- ja LM-analüüsi tulemused, mille eesmärk oli hinnata keelemudelite kasutusvõimalusi PDF-otsingu täpsuse parandamisel. Peatükk annab tervikliku ülevaate rakenduse arendusprotsessist ja saavutatud tulemustest.

4.1 Rakenduse arhitektuur

Rakenduse arhitektuur järgib puhta arhitektuuri põhimõtteid (vt Joonis 4), mis jagavad süsteemi kihtideks, et eraldada ärireeglid, andmehaldus ja kasutajaliides. See lähenemine vähendab sõltuvusi kihtide vahel, suurendab testitavust ja hõlbustab süsteemi laiendamist. Rakendus kasutab ka MVVM-arhitektuurimustrit, mis aitab hoida kasutajaliidese loogika ja andmetöötluse eraldatuna.

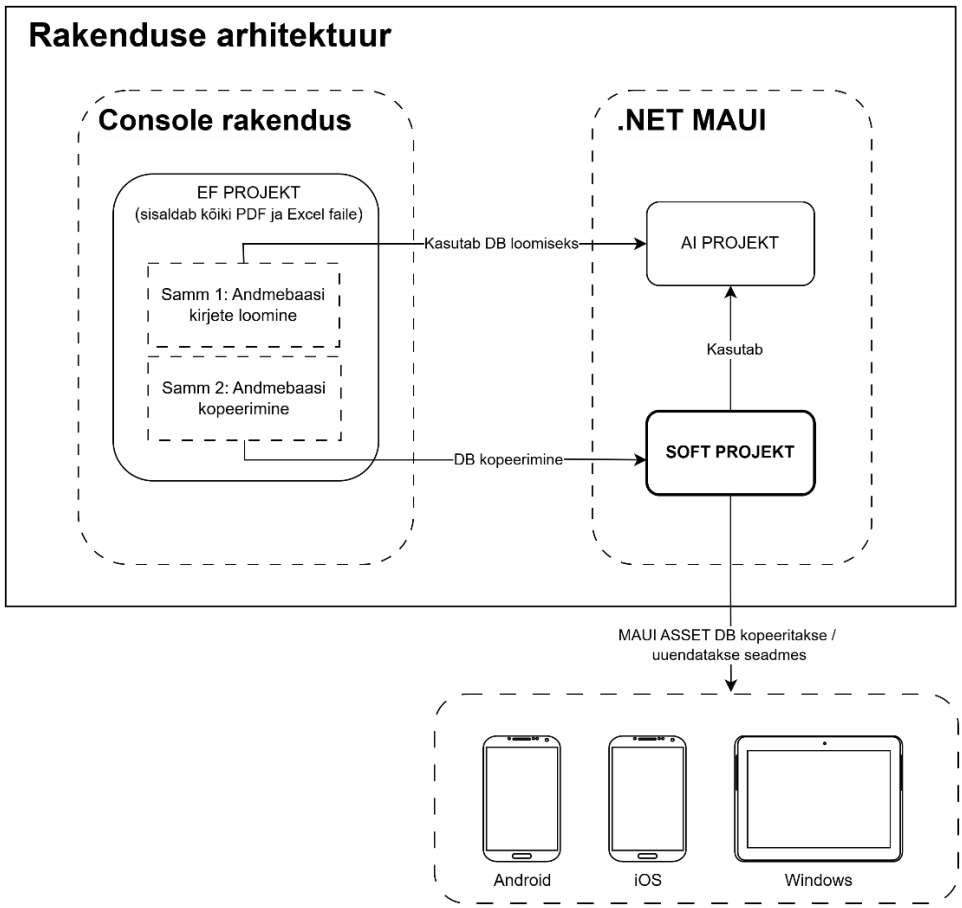


Joonis 4. Puhas arhitektuur.

Rakenduse arhitektuur põhineb mitmetasandilisel ülesehitusel, mis jaguneb esmalt kolmeks iseseisvaks alamprojektiks, millest igaühel on selgelt määratletud eesmärk ja roll:

- **EF projekt (Entity Framework konsoolirakendus):** Vastutab andmebaasi loomiseks vajalike lähtefailide (PDF, Excel) töötlemise eest. Projekt loeb andmed, teisendab need domeeni- ja andmestruktuurideks ning salvestab need SQLite andmebaasi, kasutades EF Core O/RM-raamistikku. Andmebaas luuakse lokaalselt ja kopeeritakse hiljem mobiilirakenduse varade hulka (asset DB).
- **SOFT projekt (MAUI mobiilirakendus):** Mobiilne lõppkasutaja rakendus, mis töötab Android, iOS ja Windows seadmetel. SOFT projekt kasutab EF projektis loodud ja varadesse kopeeritud SQLite andmebaasi. Andmebaasi kopeerimine või uuendamine seadmesse toimub rakenduse käivitamisel, vajadusel varasemat faili üle kirjutades. Rakendus sisaldab funktsionaalsusi nagu tegevusjuhendite kuvamine, otsing, RHK-10 link ja annustamiskalkulaator.
- **AI projekt (äriloogika ja domeeni arhitektuur):** Selle projekti sees rakendatakse kogu rakenduse loogiline arhitektuur. AI projekt sisaldab süsteemi põhiloogikat ja on struktureeritud puhta arhitektuuri põhimõtete järgi. Siin paiknevad kõik ärireeglid, andmetöötlus, domeeniobjektid ning kasutajaliidese mudelid vastavalt mustrile. AI-projekt on ühendatud nii EF- kui ka SOFT-projektiga: esimeselt saab see andmed ja teisele annab funktsionaalsuse.

Allpool skeemil (vt Joonis 5) on kujutatud nende projektide koostöö ja andmevoog.



Joonis 5. Projekti arhitektuur.

Just AI projekti sees rakendub kihiline arhitektuur, mille alusel on projekt jagatud viieks loogiliseks kihiks: kasutajaliidese kiht, fassaadi kiht, domeeni kiht, andmekiht ja infrastruktuuri kiht. Alljärgnevates alapeatükkides käsitletakse iga kihi rolli ja tehnilist ülesehitust eraldi.

4.1.1 Kasutajaliidese kiht

Kasutajaliides on realiseeritud .NET MAUI platvormil, mis toetab platvormiülest arendust Androidi, iOS-i ja Windowsi platvormidele. Kasutajaliidese kiht järgib lisaks MVVM arhitektuurimustrit (vt Joonis 6), mis võimaldab vaadete ja nende loogika eraldamist. MVVM eraldab andmemudeli (*Model*), kasutajaliidese (*View*) ning andmeside ja loogikakihi (*ViewModel*), võimaldades paremat koodi korduskasutatavust, testitavust ja hooldatavust. MVVM-mustri rakendamisel kasutati CommunityToolkit.Mvvm teeki, mis lihtsustab ja lühendab tüüpilist ViewModeli koodi [27].

Joonis 6. MVVM arhitektuurimuster.

Visuaalsed vaated on defineeritud XAML-failides, mis määratlevad iga lehe struktuuri ja kujunduse. Iga XAML-failil on vastav C#-põhine tagaklass, mis sisaldab vajadusel sündmuste käsitlemist või sidumist.

Peamised XAML-lehed rakenduses:

- MainPage.xaml – rakenduse avaleht, kus kuvatakse link RHK veebilehele, nupp navigeerimiseks annustamiskalkulaatori vaatesse ja juhendite nimekiri.
- ManualPage.xaml – juhendi vaade PDF-kujuliseks kuvamiseks, koos otsingu ja sisukorraga.
- DosageCalculatorPage.xaml – ravimite info ning annustamiskalkulaatori vaade.
- LoadingPage.xaml – vaade, mis kuvatakse andmete laadimise ajal.
- AppShell.xaml ja App.xaml – rakenduse navigeerimise ja üldise visuaalse stiili konfiguratsiooniks.

Kõiki vaateid toetavad XAML-i tagakoodifailid (*.xaml.cs), mis vajadusel lisavad käitumist või ühendavad vaate vastava ViewModeliga:

- MainPageViewModel.cs – haldab juhendite nimekirja kuvamist ja navigeerimislinke.
- ManualPageViewModel.cs – vastutab PDF-faili kuvamise, otsingu ja sisukorra loogika eest.
- DosageCalculatorViewModel.cs – sisaldab annustamiskalkulaatori arvutusloogikat.

ViewModelid suhtlevad fassaadikihiga ning kasutavad andmeid, mis on vormindatud kasutajaliidesele sobivaks.

Kogu rakenduse navigeerimine on struktureeritud Shell-navigeerimismustri alusel, mida konfigureeritakse failis AppShell.xaml. See võimaldab määrata rakenduse peavaated ja

nende vahelised navigeerimisreeglid. Rakenduse alglaadimine toimub failis App.xaml.cs, kus seotakse sõltuvused, initsialiseeritakse andmebaas ja määratakse lähteloogika. Rakenduse käivitamisel kontrollitakse meetodi Config.SetDatabase() abil, kas vajalik andmebaasifail on seadme lokaalses kaustas olemas. Vajadusel kopeeritakse see rakenduse sisemisest paketist. Lisaks kontrollitakse, kas seadmes asuv andmebaasifail vastab rakenduse paketis olevale versioonile; kui erinevused tuvastatakse, uuendatakse lokaalne andmebaas automaatselt.

4.1.2 Fassaadi kiht

Fassaadi kiht teisendab domeeni andmed kasutajaliidesele sobivaks, vähendades otseseid sõltuvusi domeeni kihiga. See sisaldab vaateklasse FileView.cs, ManualView.cs, ManualWithThumbnailView.cs, MedicationRecordView.cs, TagView.cs. Vaateklassidel on vastav *factory*-klass, mis loob domeeniobjekti põhjal kasutajaliidesele sobiva andmevaate. ViewModel Factory (GoF disainimuster) võimaldab vaateobjekti loomisel kasutada vajadusel keerukamat loogikat või arvutusi, lisasõltuvusi (*dependency injection*) komponentide eraldatuse ja testitavuse ning vaateobjektide loomise loogika on koondatud kesksetesse *factory*-klassidesse, vähendades duplikaatkoodi ja võimaldades hõlpsamat hooldust.

4.1.3 Domeeni kiht

Domeenikihi ülesanne on kirjeldada rakenduse olemid, mis esindavad süsteemi põhikontseptsioone. Olem (ingl. *entity*) on objekt, millele on unikaalne identifikaator ja mis inkapsuleerib äriloogikaga seotud andmed ja käitumise. Rakenduses on defineeritud järgmised põhilised domeeniklassid:

- File.cs – esindab rakenduses üldist andmefaili, millel puudub äriline kontekst. Tegemist on domeeniobjektiga, mis kapseldab FileData andmestruktuuri ning võimaldab ligipääsu faili metaandmetele (sh nimi, tüüp, sisu baitidena, räsi). Seda klassi kasutatakse erinevates kontekstides failide hoidmiseks ja haldamiseks, sõltumata nende otstarbest rakenduses.
- Manual.cs – esindab domeenis konkreetset meditsiinilist juhendmaterjali, mis põhineb FileData andmetel, kuid lisab sellele täiendava ärilise tähenduse ja loogika. Klass sisaldab meetodeid ja omadusi, mis võimaldavad juhendi lehekülgede arvu määramist, kuvamisjärjekorra haldamist ning seotud siltide

(Tag) *lazy* laadimist andmebaasist. Olemit kasutatakse otseselt kasutajaliidese juhendivaadetes ja märksõnaotsingus.

- MedicationRecord.cs – klass esindab ühte ravimiinfo kirjet domeenis. See pärineb baasklassist Entity<MedicationRecordData>, mille kaudu kapseldab ravimiga seotud andmed, nagu toime, manustamisviis, kõrvaltoimed ja vastunäidustused.
- Tag.cs – klass esindab märksõna või kategooriat, mis on seotud kindla failiga. Tegemist on domeeniobjektiga, mis pärineb üldisest Entity<TagData> baasklassist ning kapseldab TagData andmestruktuuri andmed. Klass sisaldab kahte atribuuti: Name – sildi nimi (nt „meditsiinilised skeemid", „erakorraline meditsiin" jt) ja FileId – viide failile, millega silt on seotud.

Kõik eeltoodud olemid pärinevad kas otse või kaudselt abstraktsest Entity klassist, mis määratleb ühtse identiteedi käsitluse (Id) ning võimaldab vajadusel realiseerida seotud andmete asünkroonse laadimise mehhanismi (LoadLazy()). Domeenikiht ei sõltu teistest rakenduse kihtidest, mistõttu saab selles defineeritud äriloogikat testida ja arendada sõltumatult andmekihist või kasutajaliidesest.

4.1.4 Andmekiht

Andmekiht vastutab rakenduse püsivate andmete haldamise eest ning realiseerib ühenduse andmebaasiga, kasutades Entity Framework Core O/RM-raamistikku. Kõik rakenduse andmed salvestatakse SQLite andmebaasi, mis sobib hästi mobiilirakendustele tänu oma kergusele ja lokaalsusele.

Andmekiht sisaldab kolme peamist andmemudelit, mis on realiseeritud POCO põhimõtete järgi. Andmemudelid pärinevad abstraktsest EntityData<T> baasklassist:

- FileData.cs – esindab faili metaandmeid, nagu nimi, tüüp, sisu baitidena, räsi ja kuvamisjärjekorda.
- MedicationRecordData.cs – salvestab ravimiinfo: toime, manustamisviis, kõrvaltoimed ja vastunäidustused.
- TagData.cs – esindab märksõna, mis seotakse failiga selle FileId kaudu.

Kõik andmeklassid pärinevad üldisest EntityData<T> baasstruktuurist, mis tagab ühtse identifitseerimisskeemi (Id) ja võimaluse andmeobjekti kloonimiseks. Need

andmemudelid on seotud EF Core kaudu SQLite andmebaasitabelitega. Nende põhjal luuakse domeeni olemid, mis kapsuleerivad andmeid ning lisavad äriloogika ja käitumise.

4.1.5 Infrastruktuuri kiht

Infrastruktuurikiht sisaldab rakenduse tehnilise toimimise jaoks vajalikke komponente, mis toetavad domeeni- ja andmekihi loogikat, kuid ei kuulu otseselt äriloogikasse. Siia kuuluvad andmebaasi kontekstid, hoidlad (*repos*), andmebaasi initsialiseerimine ja ressursihaldus. Infrastruktuurikiht on tihedalt seotud rakenduse käivitamise, andmete CRUD operatsioonidega ning välistest allikatest andmete laadimisega.

Hoidlad – infrastruktuurikihis asub üldine hoidlaabstraktsioon Repo<TObject, TData>, mis implementeerib liidest IRepo<TObject> ning realiseerib tüüpilised CRUD-operatsioonid, nagu lisamine, muutmine, kustutamine ja andmebaasipäringud. See klass kasutab EF Core DbContext-i kaudu andmete lugemiseks ja salvestamiseks DbSet<TData> kogumit ning toetab ka dünaamilist otsingut omaduse nime alusel. Hoidlad rakendavad geneerilist loogikat, mis on korduv erinevate andmeklasside puhul. Konkreetsetele andmetüüpidele luuakse eraldi hoidlad, mis laiendavad üldist Repo<TObject, TData> klassi.

Andmebaasikontekst ja EF Core – rakenduse andmebaasihaldus toimub klasside AppDbContext.cs ja BaseDbContext.cs kaudu, kus AppDbContext on tuletatud BaseDbContext-ist ning omakorda BaseDbContext on tuletatud DbContext-ist. AppDbContext määratleb tabelite kogumid ning on EF Core poolt kasutatav andmebaasi tööks. BaseDbContext võimaldab konteksti laiendamist ja seadistamist. Lisaks sisaldab kiht DesignTimeDbContext.cs klassi, mida EF Core kasutab andmebaasi migratsioonide genereerimiseks.

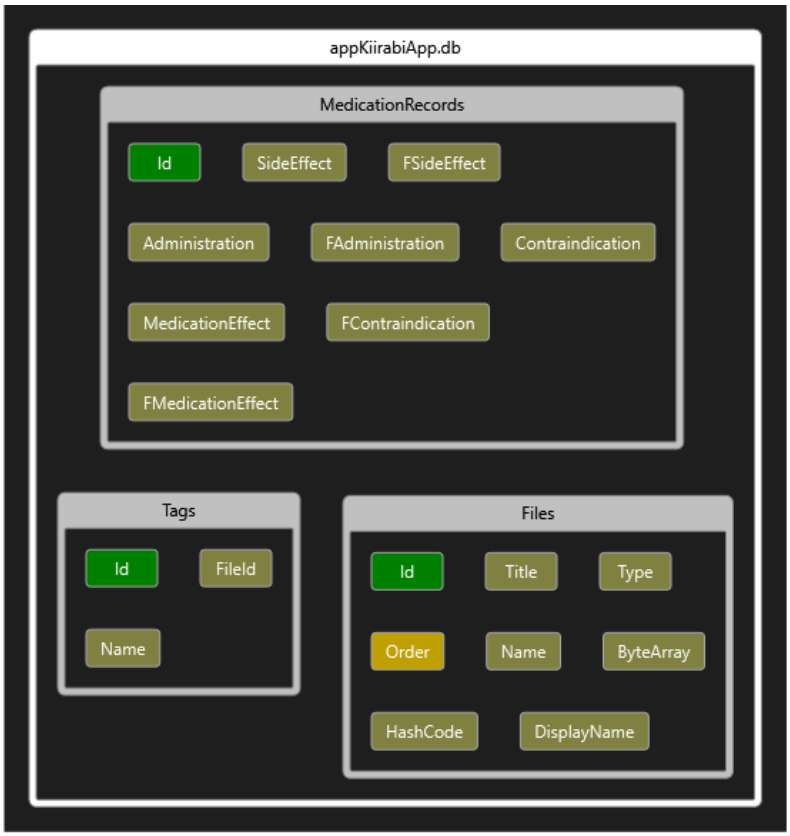
Andmebaasi initsialiseerimine – DbInitializer.cs vastutab andmebaasi loomise, migreerimise ning algandmetega täitmise eest. Initsialiseerimine toimub rakenduse käivitamisel ning sisaldab kolme põhifunktsiooni:

- PDF-failide import rakenduse sisemistest ressurssidest
- Siltide (TagData) automaatne genereerimine
- Ravimikirjete (MedicationRecordData) importimine Excel-failist

Ravimite import toimub kasutades ClosedXML teeki, mille abil loetakse failist annustamisKalkulaator_Ravimid.xlsx tabeliandmed ja teisendatakse need MedicationRecordData objektideks. Lisaks töödeldakse Exceli tekstivormingut JSON-tekstiks, et säilitada andmete struktuur ja kujundus.

4.2 Andmemudel

Rakenduse andmebaasiks on SQLite, mille struktuur modelleeriti kolme põhitabeli abil: Files, Tags ja MedicationRecords. Diagrammil (vt Joonis 7) on kujutatud iga tabeli väljad ning nende tüüp.



Joonis 7. appKiirabiApp.db andmemudel.

Rakenduse andmebaasi andmemudel järgib relatsioonilist struktuuri, kus tabelite vahelised seosed on defineeritud välisvõtmete abil. Näiteks seob Tags tabeli väli FileId end Files tabeli primaarvõtmega Id, moodustades ühe-mitmele (1:N) seose. Selline määratlus tagab referentsiaalse terviklikkuse.

4.3 Valminud funktsionaalsused ja vaated

Käesolevas alampeatükis antakse ülevaade Tallinna Kiirabi mobiilirakenduse valminud funktsionaalsustest, mis realiseeriti vastavalt määratletud nõuetele (vt peatükk 3.2). Kirjeldatakse rakenduse põhivaateid, nende kasutajaliidese elemente ja toiminguid, keskendudes tegevusjuhendite kuvamisele, otsingufunktsioonile, annustamiskalkulaatorile ning RHK-10 lingi integreerimisele. Iga funktsionaalsus on illustreeritud ekraanipiltidega, mis näitavad rakenduse visuaalset esitust ja kasutajakogemust. Funktsionaalsused on optimeeritud kiirabitöötajate vajadustele, tagades kiire ja intuitiivse juurdepääsu infole võrguühenduseta keskkonnas.

4.3.1 Tegevusjuhendi kuvamine

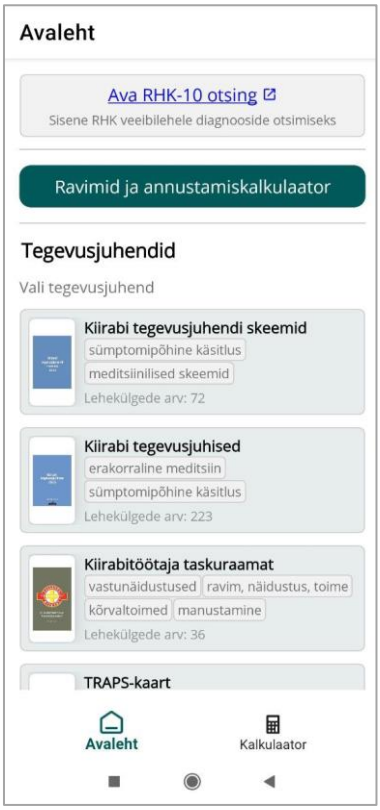
Tegevusjuhendi avamisel kuvatakse valitud dokument PDF-vormingus (vt Joonis 8), kasutades WebView komponenti ja pdf.js teeki, vastates nõuetele F1, F3, F4 ja F5. Vaade võimaldab juhendi sisu sujuvat kerimist ja suurendamist, tagades loetavuse mobiilseadmetel. Juhendi sisu põhineb Tervisekassa kinnitatud materjalidel, tagades vastavuse nõuetele F3 ja F4.

Joonis 8. Tegevusjuhendi sisu kuvamine.

Arenduse käigus katsetati ka PdfiumSharp teeki, mille abil prooviti PDF-failide kuvamist rakenduses. Kuigi algselt tundus see sobiva lahendusena, ilmnesid praktikas mitmed ühilduvusprobleemid, mistõttu see asendati pdf.js-l põhineva WebView-lahendusega. Sellest hoolimata aitas PdfiumSharpi testimine mõista PDF-failide renderdamise kitsaskohti ja suunata lõplikku tehnoloogilist valikut.

4.3.2 Avaleht ja navigeerimine

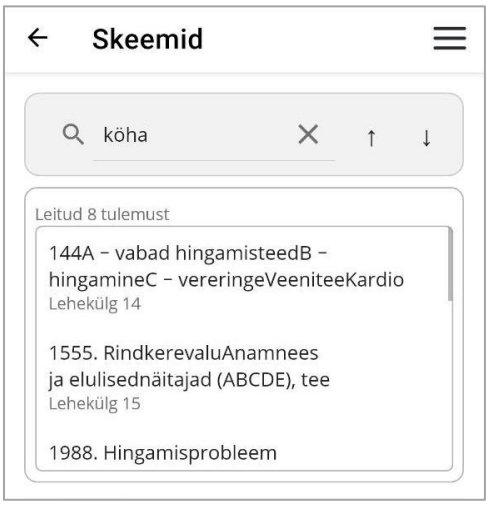
Rakenduse avaleht (vt Joonis 9) on kasutajaliidese keskne osa, mis koondab kiirabitöötajatele vajalikud funktsioonid ühtsesse vaatesse, vastates nõuetele F1, F2 ja F6. Avalehel kuvatakse tegevusjuhendite loetelu kindlas järjestuses: „Kiirabi tegevusjuhendi skeemid“, „Kiirabi tegevusjuhend“, „Kiirabi taskuraamat“ ja „TRAPS-kaart“, mis peegeldab kiirabitöötajate praktilisi eelistusi. Lisaks sisaldab avaleht nuppu „Ava RHK-10 otsing“, mis suunab kasutaja rahvusvahelise haiguste klassifikatsiooni veebilehele, ning nuppu navigeerimiseks annustamiskalkulaatori vaatesse. Lehe alaosas oleval navigeerimisribal vahelehed, mis võimaldavad kiiret liikumist avalehe ja kalkulaatori vaadete vahel.



Joonis 9. Rakenduse avaleht.

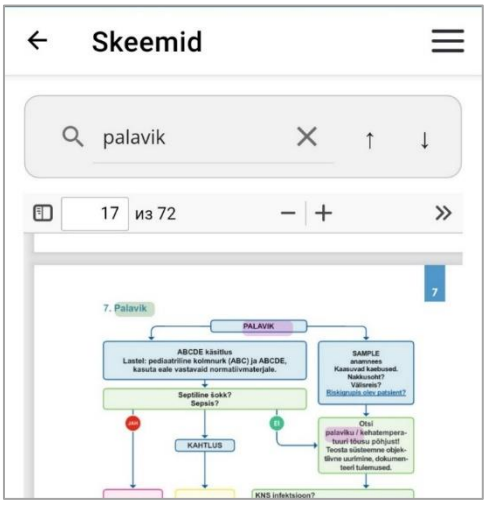
4.3.3 Otsingu võimalused tegevusjuhendi sees

Otsingufunktsioon juhendi sees vastab nõuetele F7 ja F8, võimaldades kiirabitöötajatel leida vajalikku teavet märksõnade alusel. Kasutaja saab sisestada otsinguväljale märksõna või sõnaosa, mille järel süsteem kuvab rippmenüüs tulemused koos tekstikatkendi ja leheküljenumbriga (vt Joonis 10). Kasutaja saab valida tulemuse, mille järel rakendus navigeerib automaatselt vastavale leheküljele.



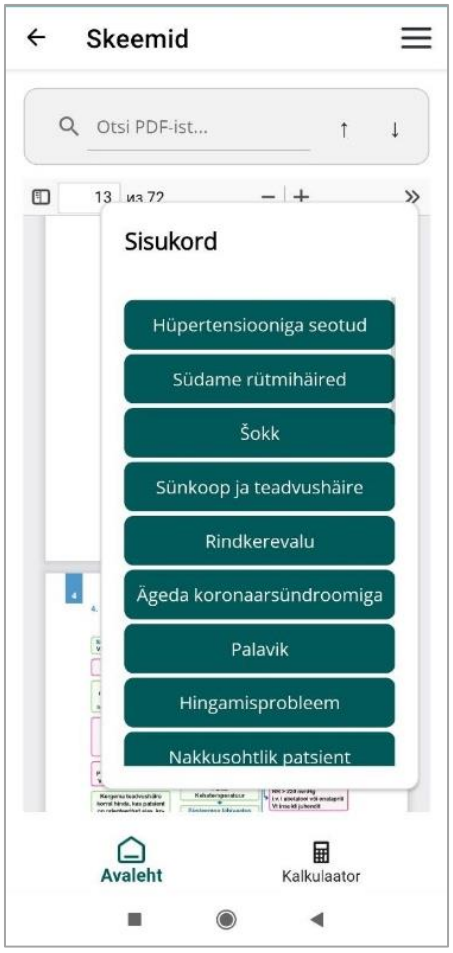
Joonis 10. Otsing tegevusjuhendi sees.

Tulemused märgistatakse dokumendis värvilise esiletõstuga (vt Joonis 11), et hõlbustada nende leidmist. Lisaks võimaldavad nupud „ ↑ “ ja „ ↓ “ liikuda tulemuste vahel, tagades kiire ja paindliku navigeerimise.



Joonis 11. Otsingu tulemused tekstis.

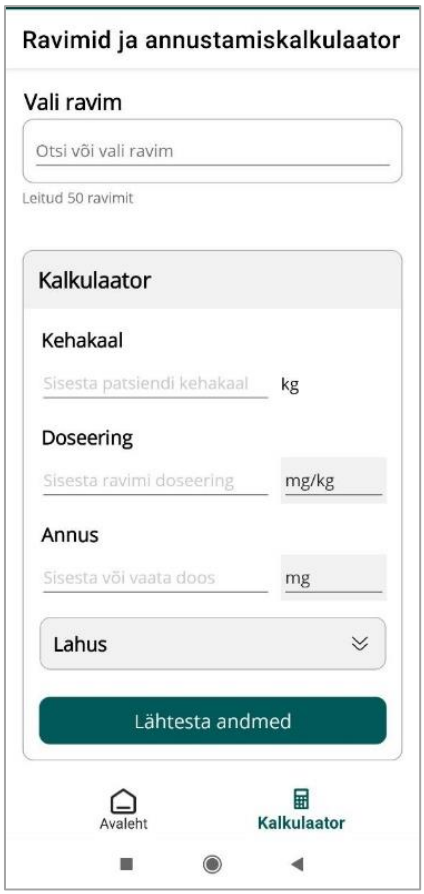
Kui tegevusjuhend sisaldab sisukorda, kuvatakse see eraldi navigeerimispaneelil, mis avaneb nupuvajutusega (vt Joonis 12). Kasutaja saab valida peatüki, mille järel rakendus suunab otse vastavale leheküljele. Paneel sulgub automaatselt pärast valiku tegemist. Sisukorra struktuur kuvatakse loendina, tagades selguse ja kasutusmugavuse.



Joonis 12. Sisukorra paneel.

4.3.4 Ravimid ja annustamiskalkulaator

Annustamiskalkulaatori vaade (vt Joonis 13) on loodud toetama kiirabitöötajate otsuste tegemist ravimite manustamisel, vastates nõuetele F9–F14. Vaatesse pääseb avalehe nupu kaudu või navigeerimisriba vahelehelt. Vaade koosneb kahest põhiosast: ravimite otsingu- ja infopaneelist ning kalkulaatorist.



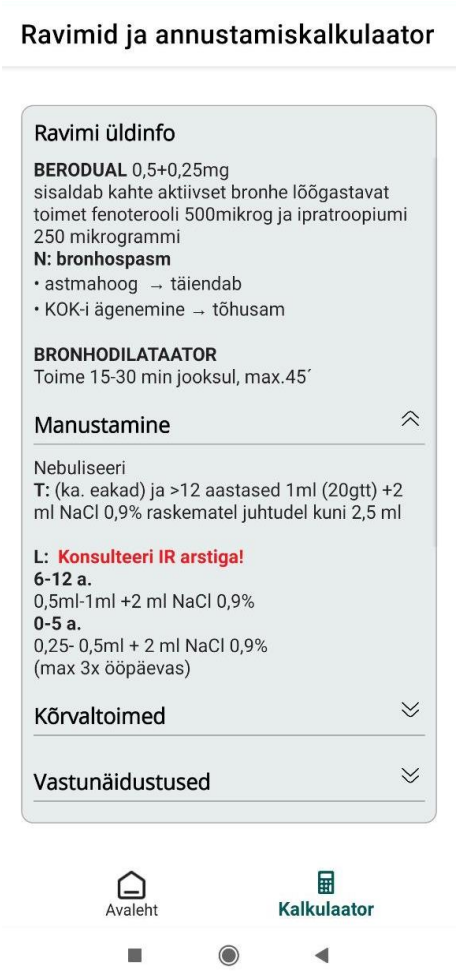
Joonis 13. Ravimite ja annustamiskalkulaatori vaade.

Ravimi valimiseks kuvatakse otsinguväli, mis avab rippmenüüna ravimite nimekirja (vt Joonis 14). Kasutaja saab sirvida nimekirja, valida ravimi rippmenüüst või sisestada ravimi nime, sõnaosa või märksõna, mille järel süsteem filtreerib vasted automaatselt.

Joonis 14. Ravimi valik.

Pärast ravimi valimist kuvatakse selle üldinfo, manustamisjuhised, kõrvaltoimed ja vastunäidustused, mis põhinevad dokumendil „Kiirabi taskuraamat 2024“. Kõrvaltoimed

ja vastunäidustused on vaikimisi kokku volditud, et vähendada ekraanikoormust, kuid neid saab vajadusel avada (vt Joonis 15).



Joonis 15. Ravimi info kuvamine.

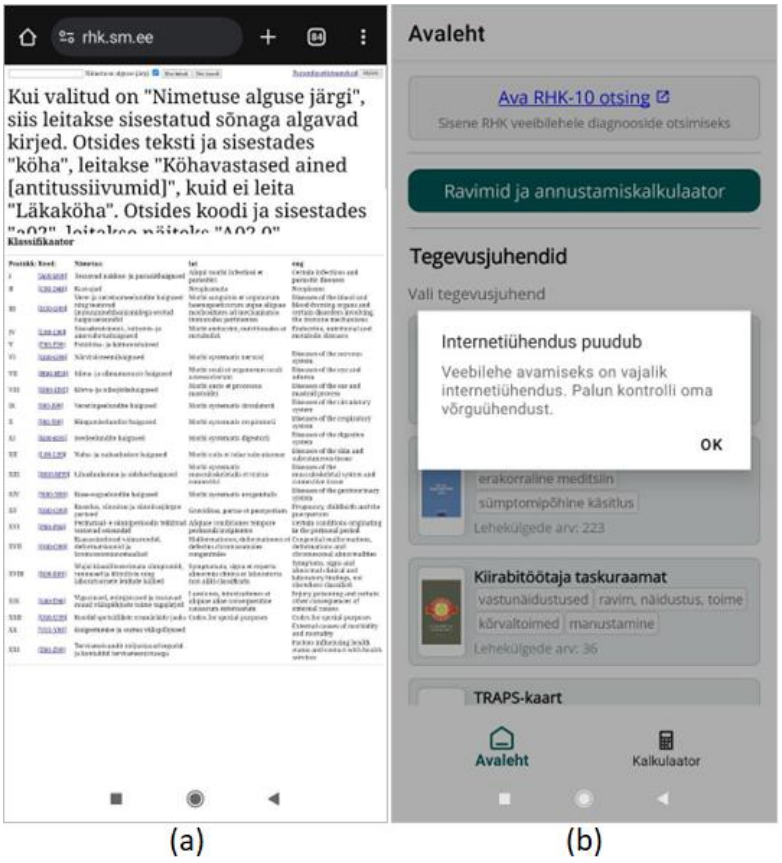
Kalkulaator ise sisaldab sisestusvälju patsiendi kehakaalu (kg), doseeringu (nt mg/kg, µg/kg) ja vajadusel lahuse kontsentratsiooni (nt mg/mL) jaoks. Arvutused toimuvad automaatselt vastavalt valemitele:

- Annus = Kehakaal × Doseering
- Vedeliku doos = (Kehakaal × Doseering) / Kontsentratsioon (lahuste korral)
- Vedeliku doos = Doos / Kontsentratsioon (käsitsi sisestatud annuse korral)

Kasutaja saab muuta ühikuid, mille järel väärtused värskendatakse reaalajas. Nupp „Lähtesta andmed“ tühjendab kõik sisestusväljad ja taastab vaikeseaded, tagades kiire taaskasutuse.

4.3.5 RHK-10 lingi kasutamine

RHK-10 otsingu integreerimine võimaldab kiirabitöötajatel kiiresti pääseda rahvusvahelise haiguste klassifikatsiooni veebilehele, vastates nõudele F6. Avalehel asuv nupp „Ava RHK-10 otsing“ kontrollib internetiühendust. Kui ühendus on olemas, avaneb seadme vaikimisi brauseris veebileht rhk.sm.ee (vt Joonis 16, (a)). Internetiühenduse puudumisel kuvatakse veateade (vt Joonis 16, (b)), mis teavitab kasutajat võrgu puudumisest.



Joonis 16. RHK-10 lingi avamise tulemused: (a) RHK-10 veebileht, (b) Internetiühenduse puudumise veateade.

Funktsionaalsus on lihtne, kuid oluline diagnooside kiireks kinnitamiseks ja klassifitseerimiseks.

4.4 Testimine

Rakenduse testimine viidi läbi mitmel tasandil: manuaalselt arenduse käigus, automatiseeritud üksustestide kaudu ning lisaks on planeeritud lõppkasutajate testimine ettevõtte siseselt pärast lõputöö valmimist.

4.4.1 Manuaalne testimine

Manuaalset testimist viidi läbi kogu arenduse vältel, et kontrollida funktsionaalsuste korrektsust ja kasutajaliidese loogilist toimimist erinevates seadmetes. Testimisel keskenduti eelkõige kasutusmugavusele, andmesisestuse voogudele ja visuaalsete komponentide toimimisele.

Lisaks funktsionaalsele kontrollile kasutati manuaalset testimist ka selleks, et hinnata rakenduse jõudlust erinevates seadmetes – sealhulgas mõõta, kui kiiresti rakendus avaneb ning kui kaua kestab erinevate funktsionaalsuste täitmine.

Manuaalse testimise käigus kasutati järgmisi tüüpi stsenaariume:

- **Viide RHK lehele:** Kasutaja vajutab RHK lingile ning teda suunatakse vastavale veebilehele. Kui seadmel puudub võrguühendus, peab süsteem kuvama vastavasisulise teavituse. Kontrolliti, et teavitus kuvatakse õigesti võrguühenduse puudumisel.
- **Tegevusjuhendi avamine ja sirvimine:** Kasutaja valib loendist ühe tegevusjuhendi ning kontrollib, kas WebView komponent kuvab vastava PDF-faili korrektselt, sh sisukorra olemasolu ja toimivus.
- **Otsing PDF-ide sisus:** Kasutaja sisestab märksõna otsingusse. Süsteem otsib vastava koha juhendi sisus ja kuvab selle kasutajale PDF-faili sees korrektselt esile tõstetuna.
- **Ravimi valimine loendist:** Kasutaja valib ravimi loendist, mille järel süsteem kuvab ravimi kohta vajaliku teabe (üldinfo, manustamisviis, kõrvaltoimed ja vastunäidustused). Kontrolliti, et kuvatav teave vastab allikmaterjalile ning säilib algne tekstiformaat.
- **Annustamiskalkulaatori testimine:** Kasutaja sisestab patsiendi vanuse, kehakaalu ning ravimi soovitusliku doseeringu, mis põhineb ravimi infol. Süsteem arvutab ja kuvab vajaliku annuse. Kui ravim nõuab kontsentratsiooni

arvestamist, sisestab kasutaja ka kontsentratsiooni ning süsteem arvutab vedeliku koguse, mida tuleb manustada.

- **Võrguühenduseta töövõime testimine:** Rakendus suletakse ja avatakse lennurežiimis. Kontrollitakse, kas tegevusjuhendid ja annustamiskalkulaator töötavad korrektselt ka ilma võrguühenduseta.

Testjuhtumite tulemusi ei dokumenteeritud eraldi testitabelis, vaid avastatud probleemide korral loodi GitLabis vastavad ülesanded. Need probleemid fikseeriti ja lahendati arenduse käigus.

4.4.2 Ühiktestid

Automatiseeritud üksustestid loodi .NET 9 platvormil kasutades xUnit raamistikku. Sõltuvuste simuleerimiseks kasutati Moq teeki ning .NET MAUI spetsiifiliste komponentide testimiseks kasutati Microsoft.Maui.Mocks teeki.

Testprojektis loodi üksustestid kolmele ViewModelile:

- **DosageCalculatorPageViewModel:** Testiti annustamiskalkulaatori loogikat erinevate sisenditega. Samuti kontrolliti, et infot ravimite kohta ja vedeliku annuse arvutamise aken saab kinni panna.
- **MainPageViewModel:** Kontrolliti, et tegevusjuhendid laaditakse õigesti erinevate sisendandmete korral.
- **ManualPageViewModel:** Testiti omaduste muutmise teavituste korrektset toimimist, konstruktori poolt käskude ja kollektsioonide initsialiseerimist ning algväärtuste (nt otsinguteksti ja otsingu rippmenüü nähtavuse) õigsust.

Testide eesmärk oli tagada äriloogika korrektsus ja süsteemi stabiilsus sõltumata kasutajaliidesest. Testid kirjeldasid selgelt süsteemi käitumist erinevates sisendolukordades ning võimaldasid varakult tuvastada loogika- või andmetöötlusvigu. Tegemist on esmaste ja kriitilisemate üksustestidega, mis loodi prioriteetsete funktsionaalsuste katmiseks – vajadusel saab testide hulka täiendada rakenduse edasise arenduse käigus.

4.4.3 Lõppkasutajate testimine

Lõppkasutajate testimine toimub 21.–30. mai vahemikus Tallinna Kiirabi siseselt, kasutades spetsiaalselt koostatud .apk faili, mis võimaldab rakendust installeerida Android-seadmetesse ilma rakenduste poest alla laadimiseta. Testimises osalevad valitud kiirabitöötajad, kes annavad tagasisidet kasutajakogemuse, kasutusloogika ja vajalike täienduste kohta. Kuigi rakendus oli algselt kavandatud Tallinna Kiirabi vajadustele, kinnitasid kiirabitöötajad selle sobivust kogu Eesti kiirabiteenistustele, kuna see katab üldised tegevusjuhendid ja ravimiannustamise vajadused.

Ent lõppkasutajate kaasatus ei piirdu vaid selle etapiga. Kogu arendusprotsessi vältel toimusid regulaarsed kohtumised Tallinna Kiirabi esindajatega, kus esitleti vahetulemusi ja koguti tagasisidet. Näiteks mõjutas kiirabitöötajate tagasiside annustamiskalkulaatori lisamist, mis algselt polnud plaanis, kuid osutus vajalikuks nende praktiliste vajaduste põhjal. See jooksev koostöö tagas, et rakendus vastab reaalses töökeskkonnas esinevatele vajadustele.

Pärast mai testimist kogutud tagasiside põhjal planeeritakse teha parandusi või täiustusi, mis ei kuulu enam lõputöö mahu hulka, kuid võivad olla aluseks rakenduse tulevasele versioonile.

4.5 Keelemudelite analüüsi tulemused

Keelemudeli-põhise semantilise otsingu rakendamise otstarbekuse hindamiseks viidi läbi eraldi loodud testkeskkonnas võrdlev analüüs, mille käigus testiti erinevaid mudeleid nii nende töökindluse, täpsuse kui ka jõudlusnõuete täitmise osas. Analüüsi eesmärk oli välja selgitada, milline mudeli tüüp sobib kõige paremini .NET MAUI rakendusega integreerimiseks olukorras, kus töötlemine peab toimuma lokaalselt, ilma serveripoolse tugisüsteemita.

Testimine viidi läbi rakenduses kasutatavate PDF-juhendite alusel. Eesmärk oli hinnata, kas erinevad mudelid suudavad kasutaja esitatud päringule leida asjakohase vastuse reaalsetest dokumentidest, arvestades nii sisulist konteksti kui ka sõnastuslikke variatsioone.

Testides võrreldi kolme lähenemist:

1. Suur keelemudel (LLM): kasutati TinyLlama 1.1B mudelit formaadis tinyllama-1.1b-chat-v1.0.Q4_K_M.gguf.
2. ONNX-põhine tähendusvektorite mudel: kasutati mudelit model_qint8_avx512.onnx koos semantilise sarnasuse arvutamiseks vajalikku vektorruumi loogikaga .
3. Kombineeritud lahendus, kus ONNX-mudeli abil leiti tähendusvektorite põhjal päringule sobivaim tekstilõik ning sellele lõigule tuginedes koostas vastuse suurem keelemudel (LLM).

Andmete sobivust ehk relevantsust hinnati tekstilõikude tähenduste matemaatilise võrdluse abil.. Selleks teisendati kõik lõigud mitmemõõtmelisteks arvulisteks vektoreiks. Need moodustavad nn *embedding*-ruumi, kus sisuliselt sarnased tekstid paiknevad üksteisele lähedal. Ka kasutaja päring teisendati samasse ruumi ning selle põhjal leiti kõige lähedasem ehk tähenduslikult sobivaim lõik. Sarnasus arvutati koosinussarnasuse alusel, mis võimaldab hinnata kahe vektori vahelise nurga suurust ehk sisulist sarnasust. Kui sobivaim lõik oli leitud, genereeris keelemudel selle põhjal vastuse – sõltuvalt konfiguratsioonist kas väiksem või suurem mudel.

Analüüsi tulemused näitasid, et kuigi suure keelemudeli (TinyLlama 1.1B) kasutamine võimaldas semantiliselt korrektsemaid ja kontekstitundlikumaid vastuseid, ei sobinud see lähenemine lokaalse rakenduse konteksti. Mudeli laadimine oli ajamahukas, aeg vastuse genereerimiseks ulatus pea minutini isegi lauaarvutis ning mobiilses seadmes oleks see põhjustanud märkimisväärset jõudluse langust. Lisaks ilmnes, et suure keelemudeli kasutamisel oli lõpptulemuse kvaliteet tugevalt sõltuv täpsest sisendi päringust ning kasutajalt nõuti oskust esitada struktureeritud ja hästi suunatud küsimusi. Arvestades rakenduse sihtkasutajat – kiirabitöötajat, kes tegutseb ajasurves –, ei ole selline kasutusmudel realistlik ega sobitu süsteemi eesmärgiga pakkuda kiiret ja intuitiivset tuge.

Väiksemate mudelitega lahendus osutus samas piisavalt täpseks, kiireks ja ressursisäästlikuks. ONNX-mudeli abil genereeritud vektorid võimaldavad kiiret semantilist võrdlust, mille abil leitakse kasutaja päringule vastav sobiv tekstilõik. Kuna otsing ei sõltu täpsest märksõnast, vaid sisulisest sarnasusest, paraneb märgatavalt kasutusmugavus võrreldes traditsioonilise stringipõhise otsinguga.

Kuigi algselt oli plaanis semantilise otsingu funktsionaalsus rakendusse integreerida, lükkus selle realiseerimine edasi, kuna testimise käigus ilmnes mitmeid aspekte, mis vajaksid enne täiendavat viimistlemist. Lisaks tuli arendusperioodil keskenduda teistele kriitilisematele funktsionaalsustele ja töökindlusega seotud küsimustele, mille tõttu jäi keelemudeli põhine otsing antud etapis rakendusse lõplikult lisamata. Samas on kogu eeltöö tehtud ning süsteem arhitektuurselt valmis võimaldamaks selle funktsiooni lisamist Arvestades jõudluse ja kasutusmugavuse tasakaalu, oleks tulevikus esmavalikuks just väiksem keelemudel.

Kokkuvõttes järeldati, et kuigi LLM-põhine täielik vastuse genereerimine pakub tulevikuperspektiivis potentsiaali, ületab see antud rakenduse tööulatuse ning vajaks eraldi süsteemi- ja ressursiarhitektuuri. Antud töö kontekstis osutus kõige sobivamaks kombinatsioon väiksemast mudelist ja lokaalsest semantilisest otsingust, mis tagas kiire vastuse, madala ressursikasutuse ning rahuldava täpsuse reaalses kasutusolukorras.

5 Analüüs ja järeldused

Selles peatükis antakse ülevaade Tallinna Kiirabi mobiilirakenduse arendusprotsessist, saavutatud eesmärkidest, kasutajate tagasisidest ning tehtud valikutest. Analüüs põhineb arenduse käigus saadud kogemustel, valideerimistulemustel ja rakenduse testimisel.

5.1 Hinnang arendusprotsessile

Autorite hinnangul täidab loodud mobiilirakendus oma eesmärki – parandada kiirabitöötajate ligipääsu olulisele tööalasele infole, vähendada juhendmaterjalide käsitsi haldamise koormust ning toetada operatiivset otsustamist välitööde käigus. Lõpptulemuseni jõuti läbi mitmete tehnoloogiliste valikute ja arendusetappide, mis hõlmasid samuti suunamuutusi võrreldes algse kavandiga.

Arenduse alguses oli fookus pigem staatilise PDF-kuvamise ja lihtsa otsingu lahendamisel. Kasutajatega konsulteerimisel ning praktiliste vajaduste analüüsimisel selgus siiski, et pelgalt failide kättesaadavus ei ole piisav – kiirabitöötajad vajasid tööriista, mis toetab kiiret ja täpset ravimiannuste arvutamist. Selle tulemusel lisati projekti arenduse käigus annustamiskalkulaator, mida algses tööplaanis ei olnud, kuid mille kasulikkus tuli ilmsiks teiste sarnaste lahenduste analüüsimisel. Uurides alternatiivseid lahendusi, leiti inspiratsiooni veebipõhisest tööriistast OmniCalculator’i annustamiskalkulaator [28], mis pakkus intuitiivset lähenemist ravimiannuste arvutamisele ja aitas kujundada kalkulaatori kasutajaliidest ning funktsionaalsust.

Projektis järgiti agiilse arenduse põhimõtteid – kasutajate tagasiside põhjal toimusid mitmed iteratiivsed kohandused. Näiteks vähendati liidese keerukust, eemaldati üleliigseid vaateid ja täpsustati otsingutulemuste esitust. Kõik need muudatused tehti selleks, et tagada sujuv ja intuitiivne kasutuskogemus.

Olulise arendusotsusena jäeti lõplikus rakenduses kõrvale nii suure keelemudeli (LLM) kui ka kerge keelemudeli (LM) ONNX-vormingus integreerimine. Kuigi testimisel näitasid mõlemad lähenemised potentsiaali semantilise otsingu toetamisel, oleks nende rakendamine eeldanud lisaarendusi ja ressursse, mis ületasid lõputöö mahtu ning

eesmärke. Valminud lahendus keskendus funktsionaalsusele, mida oli võimalik realiseerida ilma mudelipõhise töötlemiseta, tagades seejuures süsteemi stabiilsuse ja kasutusmugavuse.

Tagantjärele hinnates oleksid autorid võinud mõnes kohas arenduse keerukusastmega paremini arvestada – näiteks PDF-failide automaatse sisukorra tuvastamine ja toetamine osutus märksa ajamahukamaks, kui esialgu eeldati. Samuti oleks olnud kasulik varem planeerida põhjalikum testimine ja valideerimine struktureeritud testjuhtude alusel.

Olulise ajakulu põhjustas ka PdfiumSharp teegi kasutuselevõtt projekti alguses. Kuigi see tundus esialgu sobiva ja lihtsasti rakendatava lahendusena, selgus töö käigus, et teegi piirangud ja ühilduvusprobleemid muutusid arenduses takistuseks, mis nõudis märkimisväärselt aega testimiseks, probleemide mõistmiseks ja lõpuks alternatiivse lahenduse (pdf.js + WebView) leidmiseks ja rakendamiseks. See peegeldab laiemat mustrit, kus mitmed tehnilised valikud, mis esmapilgul näisid lihtsatena, osutusid ajamahukaks just teadmatusest või ebapiisavast eeltööst tingituna.

Lisaks kujunes väljakutseks rakenduse ametlik avalikustamine, mille puhul ilmnes mitmeid tehnilisi ja juriidilisi bürokraatlikke nõudeid (nt D-U-N-S numbrid, arendajakontode registreerimine, esindusõigus). Kui selliste piirangutega oleks arvestatud juba projekti alguses, oleks olnud võimalik vastavad tegevused aegsasti ette planeerida ja vältida viivitusi lõppjärgus.

5.2 Eesmärkide ja funktsionaalsuste täitmine

Rakenduse põhieesmärk oli luua Tallinna Kiirabi töötajatele töövahend, mis koondab igapäevases töös vajaliku info ühte kergesti ligipääsetavasse mobiilirakendusse. Eesmärk oli parendada tööprotsesse eeskätt olukordades, kus kiire otsustamine ja info leidmine on oluline, näiteks tööväljakutsete käigus. Funktsionaalsuste arendamisel lähtuti algselt meeskonnaprojekti käigus koostatud nõuete kogumist, mida lõputöö käigus täiendati ja kohandati.

Lõppkasutajate ehk kiirabitöötajate tagasiside põhjal saab väita, et rakendus mitte ainult ei vastanud ootustele, vaid mõnes osas isegi ületas neid. Kasutajad tõid esile, et annustamiskalkulaatori funktsionaalsus oli kasulikum, kui esialgu osati oodata. Annustamiskalkulaator asendab varasemalt kasutusel olnud paberkandjal kiirabitöötaja

taskuraamatu, pakkudes selle asemel kiiret ja digitaalselt ligipääsetavat lahendust. Selle asemel, et manuaalselt juhendis sirvida ja sobivat manustamisskeemi otsida, saab kasutaja nüüd lihtsa otsingu abil valida ravimi nime ning rakendus kuvab automaatselt olulisema info – sealhulgas ravimi üldinfo, manustamise, toimeaine ja vastunäidustused. Kalkulaator aitab vältida vigu, säästab väärtuslikku aega ning toetab kiirete ja täpsete otsuste langetamist kriitilistes olukordades.

Lisaks realiseeriti mitmeid täiendavaid funktsionaalsusi, mida ei olnud meeskonnaprojekti esialgses mahus määratletud, kuid mille vajadus ilmnes kasutajatestide ja konsultatsioonide käigus. Näiteks RHK-10 otsingu lisamine ning PDF-ide otsitavaks töötlemine pdf.js abil.

Samas jäi realiseerimata administraatorivaade. Selle eesmärk olnuks võimaldada sisu haldamist (nt juhendite lisamist või muutmist) otse rakenduse kaudu. Turvalisuse ja ligipääsu piirangute tõttu ei olnud arendusmeeskonnal võimalik luua ühendust Tallinna Kiirabi sisemise serverisüsteemidega, mistõttu puudus võimalus ühendada arenduses kasutatav andmebaas nende süsteemiga. Administraatoriliidese puudumine ei takistanud rakenduse peamise eesmärgi täitmist, kuid selle lisamine võiks olla osa edasistest arendustest, kui koostöö organisatsiooniga võimaldab ohutut andmeside integreerimist.

5.3 Valideerimine ja tagasiside

Tänaseks päevaks on rakenduse esmane sisu ja tehniline funktsionaalsus valideeritud koostöös Tallinna Kiirabi meeskonnaga. Arenduse käigus loodi rakenduse testversioon (*.apk-fail), mida esialgu jagati piiratud mahus oma lähikondlaste seas (2–3 isikut), kellel oli Android-seade. Selline varajane jagamine toimus eesmärgiga läbi viia esmatasandi funktsionaalne testimine, et tuvastada suuremad tehnilised probleemid enne beeta-testimise etappi sihtkasutajatega ehk kiirabitöötajatega.

Esmastest testversioonidest saadud tagasisidele on Tallinna Kiirabi meeskond väljendanud rahulolu rakenduse funktsionaalsuse ja ülesehitusega. Arendustöö on toimunud koostöös kiirabi esindajatega ning selle käigus on arvesse võetud sisulisi ootusi ja kasutatavuse nõudeid. Tellija on kinnitanud, et rakendus vastab nende ootustele ning on valmis toetama rakenduse ametlikku avalikustamist, sealhulgas kontode ja vormistuste haldamist.

Lisaks sisulisele valideerimisele on alustatud ka rakenduse ametliku avalikustamise ettevalmistusprotsessiga. Kuna lõputöö keskne eesmärk oli funktsionaalse rakenduse arendus, mitte selle lõplik turuletoomine, sõltub rakenduse ametlik avaldamine Tallinna Kiirabi otsustest ja võimalustest. Mõlema platvormi (Google Play ja Apple App Store) puhul tuleb konto registreerida organisatsiooni nimel, mis eeldab eraldi tehnilisi ja juriidilisi protsesse. Eriti keerukas on Apple'i arendajakonto loomine, kus on nõutud kehtiv D-U-N-S number ning esindusõigusega isiku registreerimine D&B andmebaasis. Kuna tudengitel puudub volitus Tallinna Kiirabi nimel juriidilisi toiminguid teha, peab kõik toimingud vormistama organisatsiooni esindaja.

Praeguse seisuga on Tallinna Kiirabi alles hindamas, kas rakenduse avalikustamine nende nime alt on juriidiliselt ja organisatsiooni sisepoliitika järgi üldse võimalik. Sellest tulenevalt on võimalik, et rakendus tuleb avalikustada eraisiku kontolt käenduslepingu alusel.

Rakenduse esmane kasutuselevõtt on planeeritud Eesti Kiirabi Liidu korraldatavale Kiirabi Kevadkonverentsile 30. maiks 2025, kus osalevad kiirabitöötajad üle kogu Eesti. Konverentsil võimaldatakse osalejatel rakendust QR-koodi abil alla laadida, et tutvustada selle võimalusi laiemale sihtrühmale. Kui selleks ajaks ei ole avalikustamine tehnilistel või juriidilistel põhjustel võimalik, on Tallinna Kiirabi ja autorite vahel sõlmitud kokkulepe, et rakendust tutvustatakse organisatsiooni sisemises uudiskirjas juuni 2025 jooksul.

5.4 Edasiarendusvõimalused

Kuigi rakenduse põhieesmärk sai lõputöö raames saavutatud, ilmnes arenduse käigus mitmeid ideid, mis võiksid tulevikus parandada selle funktsionaalsust ja kasutusmugavust. Järgnevalt on välja toodud edasiarendusvõimalused, mis põhinevad nii arendusprotsessi kogemustel kui ka Tallinna Kiirabi ettepanekutel. Kuigi need ei kuulunud lõputöö algsesse töömahtu, võiksid need tulevikus oluliselt suurendada rakenduse väärtust kiirabitöötajate igapäevatöös.

5.4.1 Sisselogimisfunktsionaalsus

Hetkel ei ole rakenduses sisselogimise võimalust, kuna koostööpartner (Kiirabi) ei soovinud anda ligipääsu oma serverile. Selle põhjuseks on turvakaalutlused ning asjaolu, et puudub ametlik leping, mis reguleeriks serveri ligipääsuga seotud riskid.

Tulevikus võiks lisada sisselogimisfunktsiooni, mis ei oleks mõeldud kõigile kasutajatele, vaid ainult piiratud õigustega administraatorile. Selle konto kaudu saaks volitatud isik lisada uusi või kustutada olemasolevaid tegevusjuhendeid rakenduses. Hetkel toimub juhendite uuendamine otse lähtekoodi tasemel ja nõuab tehnilisi oskusi. See tähendab, et vaid IT-taustaga inimene saab vajadusel muudatusi teha.

Administraatori sisselogimine lihtsustaks oluliselt tegevusjuhendite haldamist. Selle abil saaks määratud isik teha muudatusi rakenduses iseseisvalt, ilma arendaja abita, mis suurendaks süsteemi kasutusmugavust ja paindlikkust.

Sisselogimise kasutajaliidese võimaliku teostuse visuaalse kavandi on välja toodud Lisas 2 (vt Joonis 19 ja Joonis 20).

5.4.2 Suure keelemudeli integreerimine otsingusse.

Lõputöö algfaasis oli plaanitud täiendada otsingufunktsionaalsust keelemudeli toel. Idee seisnes selles, et kasutaja saaks esitada avatud küsimusi loomulikus keeles ning keelemudel vastaks nendele olemasolevate tegevusjuhendite põhjal. Samuti oli plaan, et kogu otsing toimuks LM-i abil, pakkudes kontekstitundlikumaid ja täpsemaid vastuseid kui lihtne märksõnapõhine otsing.

Pärast põhjalikku katsetamist ja eelanalüüsi jõuti siiski järeldusele, et selle funktsionaalsuse arendamine ei ole lõputöö raames teostatav, kuna see on liiga mahukas ning ei jõutaks lõpuni. Järgnevalt on toodud peamised väljakutsed, mis takistasid keelemudeli integreerimist:

1. **Töövõimekus väljaspool võrguühendust**: Kiirabitöötajad tegutsevad tihti piirkondades, kus puudub internetiühendus, mistõttu peab keelemudel toimima lokaalselt, sõltumatult võrguühendusest. See piirab oluliselt mudelite valikut, kuna enamik keelemudeleid töötab ainult internetiühenduse olemasolul.

2. **Keelemudeli suurus ja jõudlus**: Mobiilirakenduses ei ole võimalik kasutada suuri keelemudeleid, kuna need ületavad seadmete arvutusvõimekuse ja salvestusmahu. Kuna mudel peab olema võimeline töötama ka võrguühenduseta, tuleb see alla laadida seadmele, mis tähendab, et mudeli suurus peab olema väike, kuid samas piisavalt võimekas.
3. **Vastuste usaldusväärsus**: Tervishoiuvaldkonnas on äärmiselt oluline, et mudeli poolt antud vastused oleksid täpsed, usaldusväärsed ja vastaksid meditsiinilistele standarditele. Eksimisruumi ei tohi olla, kuna valeinformatsioon võib olla eluohtlik.

Nende tegurite tõttu lükati keelemudeli integreerimine edasi. Antud lahendus osutus liiga mahukaks, et seda lõputöö raames lõpule viia, ning vääriks eraldi bakalaureuse- või magistritöö käsitlust. Siiski jääb see lahendus perspektiivikaks arendusvõimaluseks tulevikus, eeldusel et suudetakse tagada süsteemi jõudlus, töökindlus ja usaldusväärsus.

5.4.3 Testid kiirabitöötajatele juhendmaterjalide põhjal

Üheks tulevikus kaalutavaks arendusvõimaluseks on kiirabitöötajate teadmiste kontrollimise funktsioon, mille abil saaks olemasolevate juhendmaterjalide põhjal koostada teste. Idee sai alguse Tallinna Kiirabi meeskonna ettepanekust ning hõlmab keelemudeli kasutamist, et analüüsida kõiki tegevusjuhendeid ja genereerida nende alusel valikvastustega küsimusi.

Selline lahendus võimaldaks töötajatel kontrollida oma teadmisi rakenduse sees, ilma et nad peaksid eraldi õppematerjale või teste otsima. Testid võiksid olla nii üldised, hõlmates kõiki juhendeid korraga, kui ka keskenduda konkreetsele juhendile, sõltuvalt sellest, millist osa töötaja soovib üle korrata või harjutada.

Lisaks sellele võiks rakendus salvestada tehtud testide tulemused ja kuvada kasutajale isiklikku tagasisidet. See võimaldaks töötajatel jälgida oma arengut ja vajadusel testid uuesti läbi teha, et kinnistada teadmisi või parandada varasemaid vigu. Selline süsteem muudaks rakenduse lisaks infoallikale ka õppimise ja enesekontrolli vahendiks.

5.5 Logid ja meeskondlik hinnang

Üldjoontes tunnevad töö autorid, et kõik meeskonnaliikmed panustasid projekti vastavalt oma oskustele ja võimetele. Tööde jaotus toimis valdavalt hästi, kuigi mõnel juhul oleks rollid võinud olla selgemalt määratletud ja vastutusalad konkreetsemalt struktureeritud. Sellest hoolimata suudeti koostöö jooksul tagada ülesannete katvus ning lahendada kõik vajalikud arendustööd õigeaegselt.

Lõputööga hakati intensiivsemalt tegelema alates 2025. aasta veebruarist ning aktiivne arendusperiood kestis kuni mai lõpuni. Ka pärast lõputöö esitamise tähtaega jätkuvad arendustegevused, kuna see on osa arenduse loomulikust protsessist. Tegemist oli väga tiheda ja töömahuka ajaraamiga, mille jooksul toimus pidev suhtlus nii meeskonnaliikmete vahel kui ka juhendajate ja Tallinna Kiirabi esindajatega. Regulaarsete koosolekute ja vahetu suhtluse kaudu hoiti arendustöö järjepidevalt fookuses ning tagati, et lahendused vastaksid nii tehnilistele nõuetele kui ka tellija ootustele.

Töö kogumaht meeskonna peale kokku oli ligikaudu 490 tundi, mis hõlmas arendustööd, dokumentatsiooni koostamist, testimist, suhtlust ja projektijuhtimist. Iga meeskonnaliikme ajalogide kokkuvõtte ja eneseanalüüsiga saab tutvuda Lisas 3 kuni Lisas 5.

6 Kokkuvõte

Tervishoiusektori digitaliseerimine on toonud kaasa vajaduse kiirete ja tõhusate digilahenduste järele, eriti kiirabivaldkonnas, kus info kättesaadavus on ajakriitiline.

Tallinna Kiirabi töötajad seisavad silmitsi probleemiga, et olulised tegevusjuhendid ja ravimiandmed on kättesaadavad peamiselt paberkandjal või PDF-formaadis, mis aeglustab otsustusprotsesse ja suurendab vigade riski.

Lõputöö eesmärk oli arendada mobiilirakendus, mis koondab kiirabitöötajatele vajaliku tööalase info ühte kasutajasõbralikku ja võrguühenduseta toimivasse keskkonda.

Arendusprotsessis rakendati agiilseid meetodeid, sealhulgas Scrumi ja XP elemente, kasutades .NET MAUI platvormi koos SQLite andmebaasi ja semantilise otsingu integreerimisega. Projekti planeerimiseks ja versioonihalduseks kasutati GitLabi, mis toetas meeskonna koostööd ja iteratiivset arendust.

Tulemustena valmis täielikult töötav mobiilirakendus, mille põhifunktsionaalsused hõlmavad tegevusjuhendite kuvamist ja otsingut, ravimite annustamiskalkulaatorit ning RHK-10 lingi integreerimist. Rakendus töötab võrguühenduseta ning toetab sujuvat kasutuskogemust erinevates seadmetes. Tehti üksustestid põhikomponentide loogika kontrollimiseks ja viidi läbi manuaalne testimine kasutusvoogude hindamiseks.

Järeldusena võib öelda, et loodud rakendus lahendab reaalse tööalase probleemi ning loob Tallinna Kiirabi töötajatele olulist lisaväärtust. Arendustöö käigus õpiti tundma mobiilirakenduste arhitektuuri, MVVM-mudeli rakendamist, testimist ning koostööd tellijaga. Rakendus on laienemisvõimeline, näiteks sisselogimisvõimaluse ja keelemudeli integreerimise suunas, ning sobib hästi edasiseks arenduseks ja kasutuselevõtuks organisatsiooni sees.

Kasutatud kirjandus

[1] M. F. B. M. C. H. D. B. M. B. Buntin, „The benefits of health information technology: A review of the recent literature shows predominantly positive results,“ *Health Affairs,* kd. 30, nr 3, p. 464–471, 2011.

[2] C. S. S. M. e. a. F. Ehrler, „Impact of a shared decision-making mHealth tool on caregivers' team situational awareness, communication effectiveness, and performance during pediatric cardiopulmonary resuscitation: study protocol of a cluster randomized controlled trial,“ *Trials,* kd. 22, nr 277, 2021.

[3] D. V. J. M. C. H. V. N. Boltin, „Mobile Decision Support Tool for Emergency Departments and Mass Casualty Incidents (EDIT): Initial Study,“ *JMIR mHealth and uHealth,* kd. 6, nr 6, p. e10727, 2018.

[4] S. G. C. T. T. H. N. M. N. F. M. S. M. G. L. &. L. S. Agarwal, „Decision-support tools via mobile devices to improve quality of care in primary healthcare settings,“ *Cochrane Database of Systematic Reviews ,* 2021.

[5] Sotsiaalministeerium, „Eesti e-tervise strateegia 2025-2030,“ jaanuar 2025. [Võrgumaterjal]. Available: https://www.sm.ee/sites/default/files/documents/2025-02/E-tervise%20strateegia.pdf. [Kasutatud 04 mai 2025].

[6] „Tervise infosüsteemi ja regulaarse tervisestatistika andmete võrdlus. II analüüs,“ Tervise Arengu Instituut, 2012.

[7] Microsoft, „.NET MAUI,“ [Võrgumaterjal]. Available: https://dotnet.microsoft.com/en-us/apps/maui. [Kasutatud 29. Aprill 2025].

[8] „Learn Microsoft,“ Microsoft, 06 märts 2025. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/visualstudio/xaml-tools/xaml-overview?view=vs-2022. [Kasutatud 16 aprill 2025].

[9] Microsoft, „What is .NET MAUI?,“ [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui?view=net-maui-9.0. [Kasutatud 30. aprill 2025].

[10] „Visual Studio,“ Microsoft, [Võrgumaterjal]. Available: https://visualstudio.microsoft.com/.

[11] Mozilla, „pdf.js,“ GitHub, [Võrgumaterjal]. Available: https://github.com/mozilla/pdf.js/. [Kasutatud 27. aprill 2025].

[12] UglyToad, „PdfPig,“ GitHub, [Võrgumaterjal]. Available: https://github.com/UglyToad/PdfPig. [Kasutatud 27. aprill 2025].

[13] „Figma,“ Figma Inc., 2025. [Võrgumaterjal]. Available: https://www.figma.com/. [Kasutatud 30 märts 2025].

[14] GitLab, GitLab Inc., 2025. [Võrgumaterjal]. Available: https://about.gitlab.com/. [Kasutatud 30 märts 2025].

[15] „ChatGPT,“ OpenAI, [Võrgumaterjal]. Available: https://chatgpt.com/.

[16] „DeepSeek,“ [Võrgumaterjal]. Available: https://chat.deepseek.com/.

[17] „Grok,“ xAI, [Võrgumaterjal]. Available: https://x.ai/grok.

[18] „Google Gemini,“ Google Inc., [Võrgumaterjal]. Available: https://gemini.google.com/app.

[19] „Claude,“ Anthropic PBC, [Võrgumaterjal]. Available: https://claude.ai/.

[20] Tutorialspoint, „Extreme Programming,“ Tutorials Point, 2025. [Võrgumaterjal]. Available: https://www.tutorialspoint.com/extreme_programming/index.htm. [Kasutatud 16. märts 2025].

[21] C. A. Kent Beck, Extreme Programming Explained: Embrace Change (2nd Edition), Addison-Wesley Professional, 2004.

[22] F. Fuior, „Key elements for the success of the most popular Agile methods,“ *Revista Română de Informatică şi Automatică,* kd. 29, pp. 7-16, 2019.

[23] D. West, „Atlassian,“ [Võrgumaterjal]. Available: https://www.atlassian.com/agile/scrum/roles.

[24] J. B. Karl Wiegers, Software Requirements, Third Edition, Washington : Microsoft Press, 2013.

[25] B. A. N. E. Y. J. M. Lawrence Chung, on-Functional Requirements in Software Engineering, New York: Springer New York, NY, 2000.

[26] „RHK-10,“ Sotsiaalministeerium, [Võrgumaterjal]. Available: https://rhk.sm.ee/.

[27] „Learn Microsoft,“ Microsoft, [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm. [Kasutatud 12 mai 2025].

[28] Ł. Zaborowska, „Omni Calculator,“ [Võrgumaterjal]. Available: https://www.omnicalculator.com/health/dosage.

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

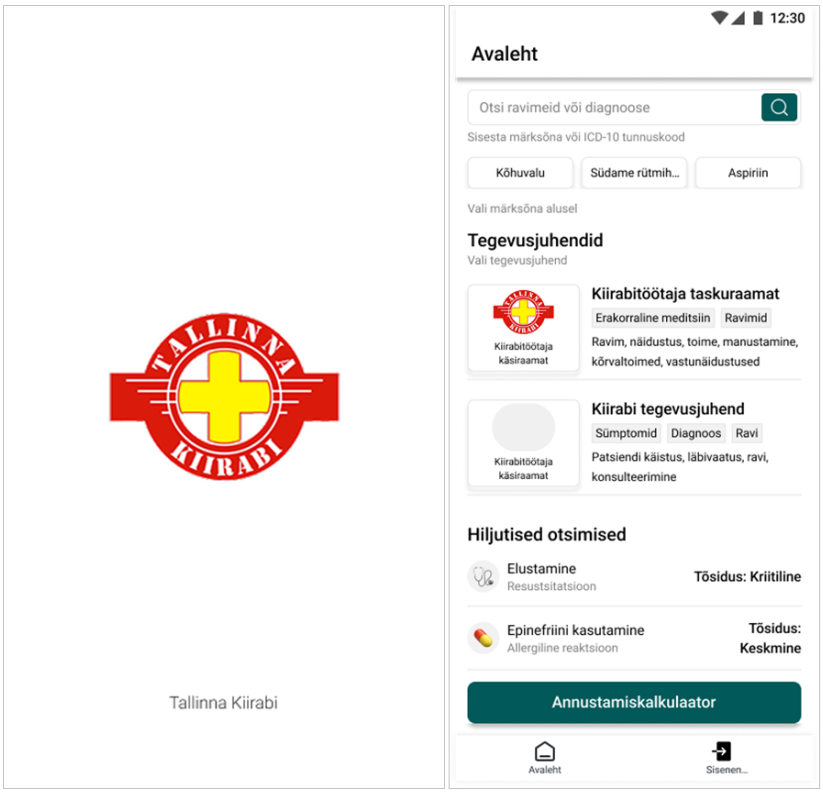
Meie, Fiona Eliis Protas, Elena Bogdanova ja Ksenija Jarmuhhamedova

1. Anname Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Tallinna kiirabi mobiilirakenduse arendamine kriitiliste juhendite ja protokollide kättesaadavuse parandamiseks“, mille juhendajad on Viljam Puusep ja Karl-Erik Karu
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

20.05.2025

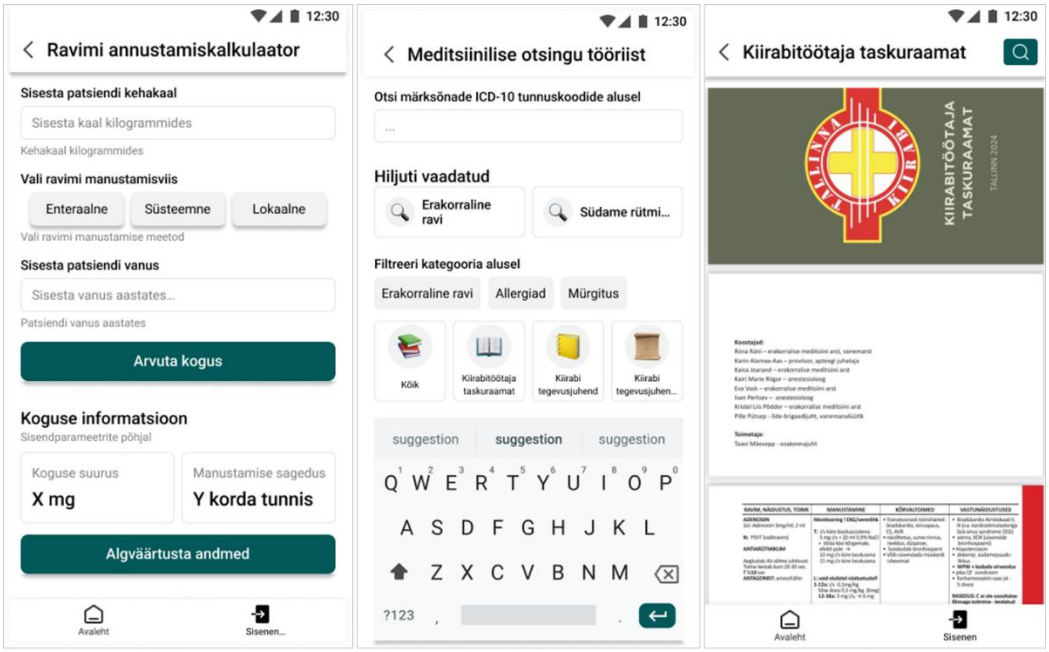
1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 – Mobiilirakenduse prototüüp



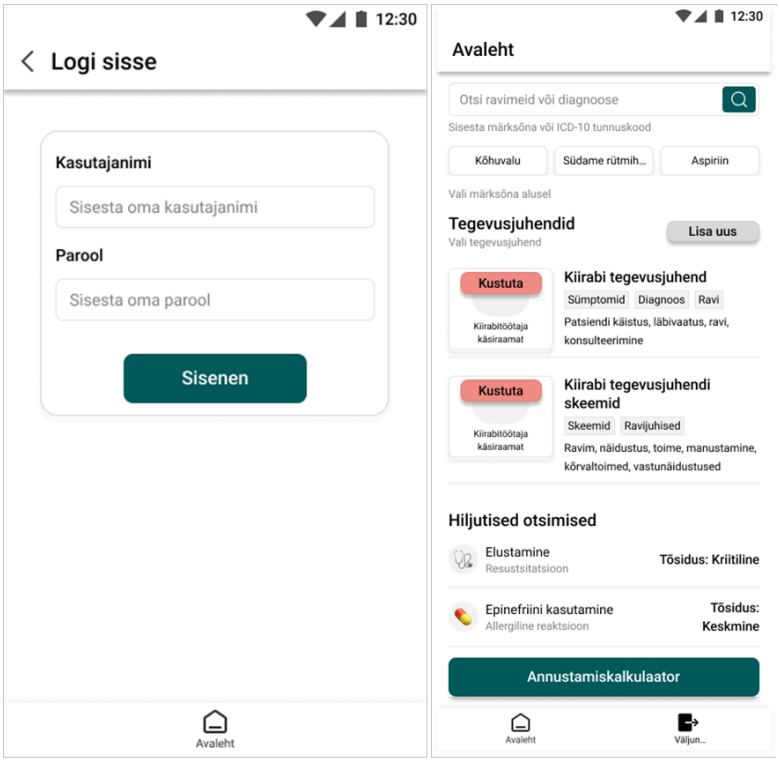
(a) (b)

Joonis 17. Prototüübi vaated: (a) „Laadimise leht", (b) „Avaleht".



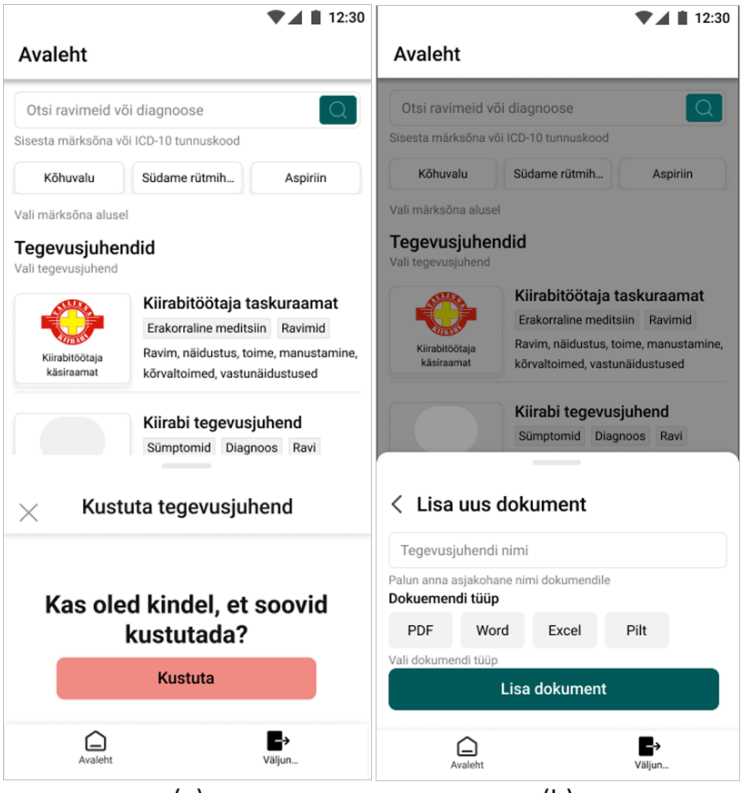
(a) (b) (c)

Joonis 18. Prototüübi vaated: (a) „Annustamiskalkulaator”, (b) „Otsimine”, (c) „Tegevusjuhendi vaade”.



(a) (b)

Joonis 19. Prototüübi vaated: (a) „Sisse logimise leht”, (b) „Avaleht (Admin vaade)”.



(a) (b)

Joonis 20. Prototüübi vaated: (a) „Tegevusjuhendi kustutamine”, (b) „Uue tegevusjuhendi lisamine”.

Lisa 3 – Fiona Eliis Protas ajalogide kokkuvõte ja eneseanalüüs

Projekti käigus täitsin mitmekesiseid rolle ning panustasin aktiivselt nii tehnilise arenduse kui ka meeskondliku suhtluse korraldamisse. Projekti alguses vastutasin arenduskeskkondade loomise eest – seadistasin GitLabi repositooriumi ning lõin projekti struktuuri Visual Studios. Sinna lisasin esmased kaustad ja failid, mis lõid aluse rakenduse arhitektuurile ja arenduse alustamisele.

Enne arendustöödega alustamist süvenesin .NET MAUI raamistikku ning läbisin Coursera kursuse „Mobile Development with .NET MAUI“, et mõista selle tehnilisi eripärasid ja piiranguid. Õppeprotsess võimaldas mul teadlikumalt rakenduse komponentide arendusega alustada. Olin vastutav mitmete kasutajaliidese komponentide loomise eest, sh taaskasutatavate nuppude, lehe päiste, otsingukasti ning lehevahelise navigeerimisloogika eest. Lõin rakendusele laadimisekraani ning tegevusjuhendite kuvamise komponendi.

Arendasin ka sisselogimislehe (kuigi see jäi lõplikust skoopist välja) ning picker-komponendi, mida kasutasime ravimite valikuks. Panin paika projektis kasutatavad *assembly*-d ja nimeruumid. Tegelesin esmase andmebaasiloogika loomisega, sh tegevusjuhendite kuvamiseks vajalike andmete haldusega. Katsetasin PdfiumSharp teeki, kuid selle mitteoptimaalse töö tõttu otsustasime üle minna WebView-põhisele lahendusele pdf.js-iga, mille integreerimise eest samuti vastutasin. Selle raames lisasin leheküljeloogika, siltide halduse juhenditele ja lõin ManualMap komponendi rakenduse avalehele ning PDF-ide eelvaatepildid ManualMap komponenti.

Lisaks lõin PDF-idesse otsingufunktsiooni, mis võimaldas klõpsates liikuda vastava tulemuse lehele ning tõstis otsisõna esile. Arendasin ka sisukorra kuvamise loogika PDF-ide jaoks. Ravimite annustamiskalkulaatori lehel lõin loogika Exceli-faili andmete lugemiseks ning nende salvestamiseks SQLite-andmebaasi. Samuti arendasin ravimiinfo kuvamist kalkulaatori ja ravimite lehtedel. Kuna arendusperioodil lisandus

juhendmaterjale ning muutus ka rakenduse sisu, tegelesin nende lisamise ja järjekorra korrastamisega.

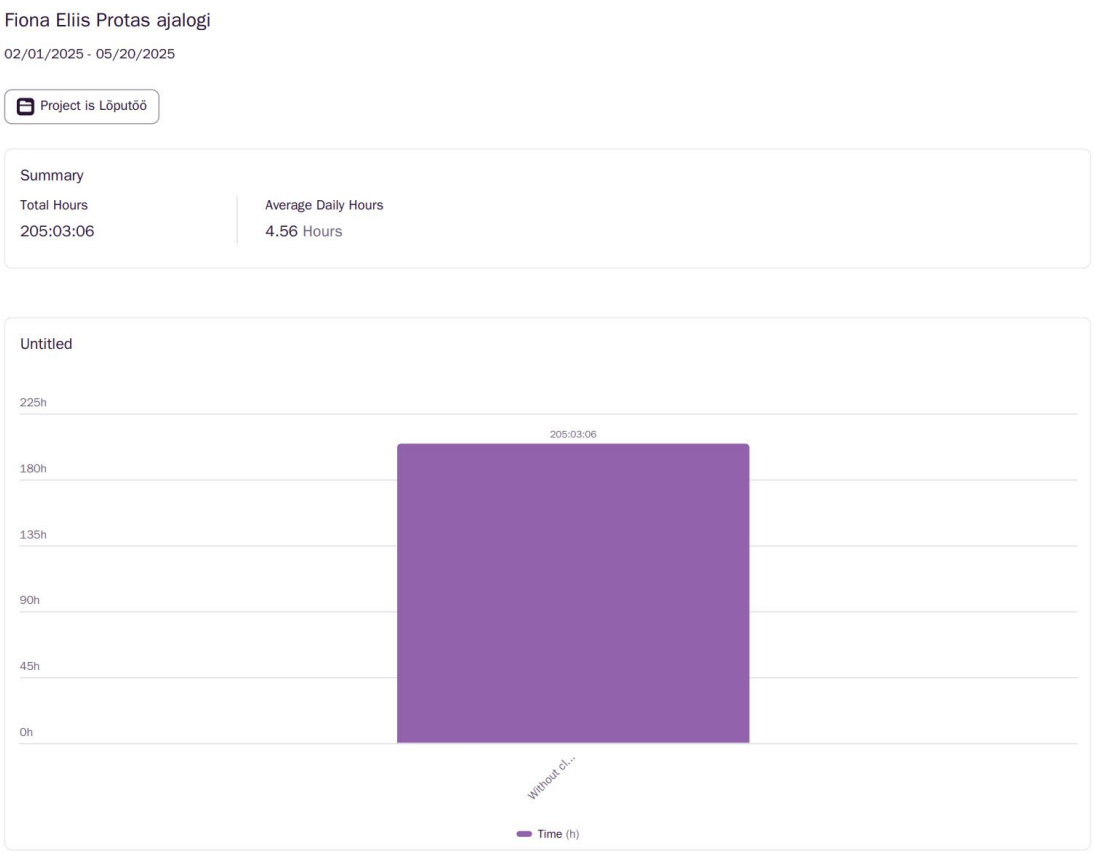
Projekti jooksul tegin jooksvaid kasutajaliidese täiustusi, andmete valideerimise arendusi kalkulaatoris, muutsin rakenduse logo ning puhastasin ja struktureerisin koodi vastavalt puhta koodi ja puhta arhitektuuri põhimõtetele. Lisaks täiendasin andmebaasi loomise loogikat ning dokumenteerisin arenduse käiku jooksvalt.

Lisaks kasutajaliidese arendusele ja andmebaasiloogikatele panustasin olulisel määral ka keelemudelite testimisse ja analüüsi. Testisin mitmeid keelemudelite lahendusi, keskendudes nende täpsusele, jõudlusele ja sobivusele võrguühenduseta kasutuseks.

Selle töö käigus lõin eraldi testkeskkonna, katsetasin päringute tulemusi erinevate juhendite põhjal ning hindasin saadud vastuste sisulist täpsust. Keelemudeli integreerimine rakendusse osutus oodatust raskemaks ning esmasest versioonist jäi funktsionaalsus välja. Sellegipoolest andis keelemudelite uurimine väärtusliku kogemuse edasisteks arendusteks ning on aluseks võimalikule tulevasele täiustusele.

Olin ka meeskonna peamine kontakt Tallinna Kiirabi esindajatega, tagades sujuva suhtluse ja vastastikuse mõistmise tehniliste ning sisuliste ootuste osas.

Olen tehtud tööga rahul – minu panus katab suure osa rakenduse võtmefunktsionaalsust ning kommunikatsiooniprotsessidest. Projekti intensiivne ajaraam (veebruarist maini) oli väljakutse, kuid sellest hoolimata suutsime meeskonnana edukalt täita töö eesmärgid. Tagantjärele vaadates oleks kasuks tulnud täpsem rollide jaotus projekti alguses, et tööprotsess oleks veelgi sujuvamalt kulgenud. Kokku investeeris projekti ligikaudu 205 tundi (vt Joonis 21).



Joonis 21. Fiona Eliis Protas ajalogi kokkuvõte.

Lisa 4 – Ksenija Jarmuhhamedova ajalogide kokkuvõte ja eneseanalüüs

Projekti käigus panustasin Tallinna Kiirabi mobiilirakenduse arendamisse nii kasutajaliidese disainis kui ka rakenduse loogika väljatöötamises. Täitsin mitmekesiseid rolle, sh funktsionaalsuste loogiline realiseerimine, otsingusüsteemi ja PDF-komponentide katsetamine, testimine ning meeskondlik kommunikatsioon. Lisaks osalesin aktiivselt juhendajate ja tellijaga peetud koosolekutel.

Projekti alguses süvenesin .NET MAUI raamistikku, kuna mul puudus varasem kogemus selle platvormiga. Tutvusin dokumentatsiooniga ja katsetasin väiksemaid komponente, et mõista raamistikku kui tervikut – selle tööpõhimõtteid, piiranguid ja tugevusi. See ettevalmistav töö aitas mul hiljem arendusprotsessis enesekindlamalt kaasa lüüa, eriti rakenduse loogika ja UI-komponentide loomisel.

Arenduse varases etapis lõin mitmeid taaskasutatavaid kasutajaliidese komponente, sealhulgas erinevat tüüpi label'eid ning töötasin selle nimel, et vähendada komponentide koodis esinevat dubleerimist. Lisaks koostasin rakenduse avalehe esmase vaate.

Projekti esimese funktsionaalsuse realiseerimise etapis katsetasin PdfiumSharp teeki eesmärgiga realiseerida PDF-failide kuvamine otse rakenduses. Analüüsi tulemusel – mida viisime läbi koos meeskonnaliikmete ja juhendajatega – jõudsime ühise otsuseni loobuda PdfiumSharp lahendusest ning kasutada WebView-põhist pdf.js integratsiooni, mille rakendamise võtsid üle teised meeskonnaliikmed.

Otsingu funktsionaalsuse arendamises osalesin koos teiste meeskonnaliikmetega nii loogika kui ka kasutajaliidese poolel. Töötasin koostöös teiste meeskonnaliikmetega otsingu visuaalse lahenduse kallal, sealhulgas otsingukasti paigutuse, tulemuste kuvamise loogika ja kasutajakogemuse sujuvuse arendamisel.

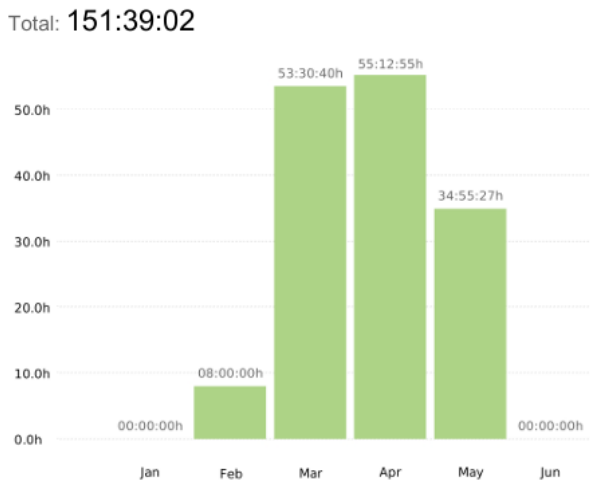
Annustamiskalkulaatori funktsionaalsuses töötasin välja algse loogika ning vastutasin selle kasutajaliidese kujunduse eest. Kalkulaatori esmane loogika hõlmas sisendite töötlemist, nagu vanus, kehakaal ja soovituslik annus, mille põhjal arvutati ravimikogus.

Kontsentratsiooni sisestamise ja arvestamise loogika lisas hiljem teine meeskonnaliige. Minu poolt lisati ka ravimiinfot täiendavad alajaotused nagu vastunäidustused ja kõrvaltoimed, et tagada kasutajale sisuliselt oluline ja praktiline info koondatult ühes vaates.

Osalesin ka keelemudelite potentsiaali analüüsimises seoses semantilise otsingu integreerimisega. Viisin läbi erinevate mudelite võrdlust, uurisin nende jõudlust ja lokaalse kasutamise võimalusi. Lõplik otsus keelemudeli funktsionaalsuse mitte rakendada sündis pärast ühiseid arutelusid kogu meeskonna ja juhendajatega, arvestades tehnilisi piiranguid ja töömahtu.

Projekti lõppfaasis tegin kasutajaliidese parandusi ning kirjutasin üksusteste rakenduse avalehe ViewModelile, tagamaks juhendite laadimisfunktsiooni (LoadManuals meetodi) töökindluse.

Olen peaaegu rahul oma panusega – mul oli võimalus kaasa lüüa nii kasutajaliidese, rakendusloogika kui ka testimise tasandil. Projekt andis väärtuslikku kogemust tehniliste lahenduste loomisel ja koostöös reaalse tellijaga. Tagantjärele hinnates saan aga aru, et oleksin saanud rohkem panustada ajaliselt ja võtta suuremat vastutust teatud töövaldkondades – näiteks tihedam suhtlus tellijaga. Nendest tähelepanekutest lähtudes tean, millele keskenduda rohkem tulevastes projektides. Kokku investeerisin projekti ligikaudu 151 tundi (vt Joonis 22).



Joonis 22. Ksenija Jarmuhamedova ajalogi kokkuvõte.

Lisa 5 – Elena Bogdanova ajalogide kokkuvõte ja eneseanalüüs

Käesoleva lõputöö raames panustasin Tallinna Kiirabi mobiilirakenduse arendusse mitmekülgselt, hõlmates nii kasutajaliidese disaini, funktsionaalsuste arendamist, andmehaldust kui ka testimist. Minu rollid hõlmasid tehniliste lahenduste uurimist, koodi kirjutamist, probleemide lahendamist ja iteratiivset täiustamist. Allpool on toodud ülevaade minu peamistest tegevustest ja panusest projekti.

Projekti algfaasis süvenesin .NET MAUI platvormi, kuna mul puudus varasem kokkupuude selle raamistikuga. Uurisin platvormi dokumentatsiooni, lugesin täiendavaid materjale ning katsetasin lihtsamaid tegevusi, et mõista selle toimimist ja võimalusi. See ettevalmistus aitas mul hiljem arendustöödes efektiivsemalt osaleda.

Arenduse varases etapis keskendusin taaskasutatavate kasutajaliidese komponentide loomisele, sealhulgas nuppude, märgendite (tagide) ja lehekülje mallide disainimisele. Lõin esmase annustamiskalkulaatori lehekülje koos sellele navigeerimise loogikaga, mis sai aluseks kalkulaatori edasisele arendamisele. Need komponendid ja mallid tagasid rakenduse visuaalse ühtsuse ja kiirendasid hilisemat arendusprotsessi.

Olin olulisel määral kaasatud tegevusjuhendite PDF-failide kuvamise funktsionaalsuse arendamisse. Algse lahendusena katsetasime PDFiumSharp teeki, kuid tuvastasin selle käivitamisel lokaalsetes seadmetes esinenud kuvamisprobleemid. Probleemide analüüsimise järel otsustasime meeskonnaga üle minna WebView-põhisele lahendusele, kasutades pdf.js teeki. Pärast pdf.js integreerimist seostasin WebView otsingufunktsiooni meie enda UI otsingukomponendiga, võimaldades märksõnapõhist otsingut PDF-faili sees. Lisasin ka navigatsiooniloogika, mis võimaldab kasutajal liikuda otsingutulemuste vahel. Ning projekti lõppfaasis täiustasin otsingut, lisades rippmenüü otsingutulemuste kuvamiseks, mis lihtsustab kasutajal leitud tulemuste sirvimist.

Annustamiskalkulaatori arendamisel oli mul keskne roll. Täiendasin kalkulaatori põhilised arvutusvalemid. Disainisin kalkulaatori kasutajaliidese, mis jäi rakenduse

lõppversiooni osaks, lisades täiendavaid funktsionaalsusi, nagu ühikute teisendamine, andmete lähtestamise nupp ja sektsioonide avamise-sulgemise võimalus ravimiinfo kuvamisel, et parandada kasutajakogemust. Ravimiinfo kuvamiseks lõin Excel-faili, mis põhines dokumendil „Kiirabi taskuraamat 2024“. Täiustasin faili käsitsi, lisades tühikuid, korrigeerides suur- ja väiketähti ning vormindades sisu, et tagada visuaalne korrektsus. Lisaks kohandasin andmebaasi loogikat, võimaldades uue Excel-faili lisamisel automaatset andmebaasi uuendamist, mis lihtsustas ravimiandmete haldust.

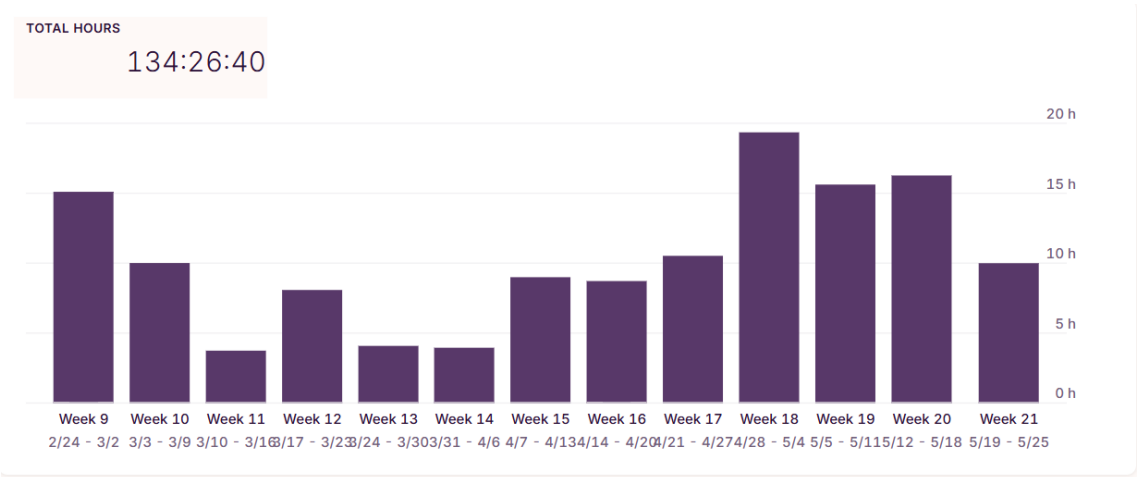
Arenduse lõppfaasis täiustasin ravimite valiku (picker) komponenti, lisades otsingufunktsionaalsuse, mis võimaldab kiirelt otsida ravimit nime või märksõna järgi. See oli vajalik, kuna ravimite nimekirjas on üle 50 kirje, mis ilma otsinguta raskendaks navigeerimist. Lisaks tegin jooksvaid kasutajaliidese täiustusi, näiteks visuaalse stiili ja paigutuse kohandusi.

Osalesin aktiivselt rakenduse testimises, keskendudes peamiselt manuaalsele testimisele. Kontrollisin uute funktsionaalsuste toimimist lokaalsetes seadmetes, kasutades erinevaid seadmemudeleid, et tagada platvormiülene ühilduvus. Näiteks testisin PDF-ide kuvamist, otsingufunktsiooni ja kalkulaatori arvutuste korrektsust. Lisaks kirjutasin üksusteste (xUnit) annustamiskalkulaatori lehekülje ja tegevusjuhendi lehekülje loogika kontrollimiseks, tagades äriloogika töökindluse.

Arenduse käigus kohtasin mitmeid väljakutseid, mis nõudsid süsteemset analüüsi ja lahenduste otsimist. Üheks näiteks oli PDF-failide vilkumine Android-seadmetes WebView komponendi kasutamisel. Teine mahukas ülesanne oli keelemudelite (LLM) uurimine semantilise otsingu võimaldamiseks. Lõin eraldi testkeskkonna, kus katsetasin erinevaid mudeleid, hindasin nende sobivust võrguühenduseta tööks ja uurisin nende jõudlust. Kuna teema oli mulle uus, nõudis see põhjalikku lisalugemist ja aega, kuid pakkus väärtuslikku kogemust, arvestades keelemudelite kasvavat tähtsust tänapäeva tarkvaraarenduses.

Olen rahul oma panusega projekti. Projekt võimaldas mul arendada tehnilisi oskusi, eriti .NET MAUI platvormil, ning õppida probleemide süsteemset lahendamist. Töö käigus sain selgema ettekujutuse oma tugevustest, ning valdkondadest, mida saan edasi arendada. See kogemus on motiveerinud mind süvendama teadmisi mobiiliarenduses ja

uurima tulevikus keerukamaid tehnoloogiaid, näiteks tehisintellekti integreerimist rakendustesse. Kokku investeerisin projekti ligikaudu 134 tundi (vt Joonis 23).



Joonis 23. Elena Bogdanova ajalogi kokkuvõte.