TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Aleksei Šohh IAIB213098

# Adaptiivse teadmiste hindamissüsteemi väljatöötamine IT-positsioonide kandidaatide esmaseks skaneerimiseks

Bakalaureusetöö

Juhendaja: Siim Rebane

BSc

Tallinn 2025

# Autorideklaratsioon

Kinnitan, et olen 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.

Autor: Aleksei Šohh

18.01.2025

# Annotatsioon

Käesoleva bakalaureusetöö eesmärk on välja töötada ja valideerida adaptiivse testimissüsteemi prototüüp. See rakendus on mõeldud kandidaatide sorteerimiseks nende teoreetiliste teadmiste taseme alusel, et valida välja kõige sobivamad. Rakendus on eriti kasulik suure konkurentsiga ametikohtade puhul, kus füüsilised või veebipõhised intervjuud võtavad ettevõtte töötajatelt palju aega.

Rakenduse arendamise peamine põhjus seisneb selles, et kuigi turul on saadaval mõningaid sarnaseid testimistooted, puudub sobiv lahendus, mida IT-ettevõtted saaksid reaalelus tõhusalt kasutada.

Prototüüp peab pakkuma funktsionaalsust testide koostamiseks, läbiviimiseks ning tulemuste statistika kuvamiseks.

Prototüüp koosneb kahest osast: tagarakendusest ja esirakendusest. Tagarakendus osa on realiseeritud *Spring Boot* raamistikus ja *Postgres* andmebaasis. See valik tuleneb nende stabiilsusest, populaarsusest ja laialdasest tööriistakomplektist, mis sobib kaasaegsete veebirakenduste loomiseks. Esirakendus on arendatud kasutades *Vue 3* ja *TypeScript*-i, mis tagab kaasaegse lähenemise liideste arendamisele, lihtsustades süsteemi hooldust ja edasiarendust. Rakenduse juurutamine toimub testserveris, kasutades *CI/CD* protsesse.

Prototüübi valideerimine toimub beetatestimise kaudu, kaasates potentsiaalseid kasutajaid. See protsess aitab tuvastada süsteemi peamised puudused ja parandada selle funktsionaalsust.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 33 leheküljel, 9 peatükki, 6 joonist, 2 tabelit.

# Abstract

## Development of an Adaptive Knowledge Assessment System for Initial Screening of Candidates for IT Positions

The goal of this bachelor thesis is to develop and validate a prototype of an adaptive testing system. This application is intended to sort candidates based on their level of theoretical knowledge to select the most suitable ones. The application is particularly useful for positions with high competition, where physical or online interviews require significant time investments from company employees.

The main reason for developing this application lies in the fact that, although there are some similar testing products available on the market, there is no suitable solution that can be effectively applied in real-life IT companies.

The prototype should provide functionality for creating tests, conducting tests, and displaying test result statistics.

The prototype consists of two parts: the backend and the frontend. The server-side part of the application is implemented using the Spring Boot framework and the Postgres database. This choice is based on their stability, popularity, and a wide range of tools suitable for modern web application development. The frontend is developed using Vue 3 and TypeScript, which ensures a modern approach to interface development, simplifying system maintenance and further development. The application is deployed on a test server using CI/CD processes.

The validation of the prototype will be carried out through beta testing involving potential users. This process will help identify the main shortcomings of the system and improve its functionality.

The thesis is in Estonian and contains 33 pages of text, 9 chapters, 6 figures, 2 tables.

# Lühendite ja mõistete sõnastik

| | |
|---|---|
| API | Rakendusliides (*Application Programming Interface*) |
| CI/CD | Jätkuintegreerimine ja juurutamine (*Continuous Integration and Continuous Deployment*) |
| CSS | Stiililehtede keel (*Cascading Style Sheets*) |
| Docker | Konteinerite virtualiseerimise platvorm |
| GitLab | Versioonihalduse platvorm ja CI/CD tööriist |
| Java | Programmeerimiskeel |
| JSON | Andmestruktuuri formaat (*JavaScript Object Notation*) |
| JWT | Veebipõhiste rakenduste turvatoken (*JSON Web Token*) |
| PostgreSQL | Relatsiooniline andmebaas |
| RESTful API | Veebipõhine rakendusliides (*Representational State Transfer API*) |
| Spring Boot | Java veebiraamistik rakenduste arendamiseks |
| Swagger | REST API dokumenteerimise tööriist |
| Tailwind CSS | CSS-põhine kasutajaliideste raamistik |
| TypeScript | Staatilise tüüpimisega programmeerimiskeel |
| Vue.js | JavaScripti raamistik kasutajaliideste loomiseks |
| XML | Andmete tähistamise keel (*eXtensible Markup Language*) |
| YAML | Andmete serialiseerimise formaat |

# Sisukord

# Jooniste loetelu

# Tabelite loetelu

# 1 Sissejuhatus

Selles peatükis antakse ülevaade töö eesmärgist, probleemi taustast ja vajadusest luua uus lahendus, mis automatiseerib kandidaatide teadmiste ja oskuste hindamise protsessi. Tutvustatakse peamisi väljakutseid ja motivatsiooni projekti elluviimiseks.

## 1.1 Probleem

Tänapäeval viiakse tehnilised tööintervjuud tavaliselt läbi kandidaatide ja ühe või kahe kogenud vanemarendaja vahelise isikliku vestluse vormis. Sellise suhtlemise käigus hinnatakse kandidaadi teoreetilisi teadmisi ning loogilist mõtlemisvõimet. Nagu iga protsess, on ka tehnilisel intervjuul nii positiivseid kui ka negatiivseid külgi.

**Positiivsed küljed**:

1. **Paindlik lähenemine igale kandidaadile**: Tavapärane praktika on esitada küsimusi kõigi tulevase ametikohaga seotud teemade kohta, et mõista, kui sügavalt kandidaat nendest tehnoloogiatest aru saab. Seejärel võib intervjueerija paluda kandidaadil kirjeldada viimast projekti, milles ta osales, ja süveneda selle teostuse detailidesse, et mõista tema teadmiste ja oskuste taset.

**Negatiivsed küljed**:

1. **Aja- ja rahakulud**: Intervjuu läbiviimiseks peab intervjueerija valmistuma, intervjueerima ning koostama kandidaadi kohta aruande. Need etapid nõuavad märkimisväärset ajakulu, mis tähendab ettevõtte jaoks ka rahalisi kulutusi. Probleem süveneb eriti olukordades, kus kandidaatide arv on suur, näiteks praktikantide värbamisel. Sellistel juhtudel võib ühele kohale kandideerida sadu inimesi, mis suurendab oluliselt intervjuude läbiviimise kulusid ja muudab need oluliseks kulureaks.
2. **Raskused sobiva aja leidmisel**: Arendajate põhiülesanded hõlmavad koodi kirjutamist ja silumist ning koosolekutel osalemist, mis teeb intervjuu läbiviimiseks sobiva

aja leidmise keeruliseks. Intervjuus osalejate ajakavade kokkusobitamine võib viia värbamisprotsessi venimiseni, mõnikord isegi mitme kuuni.

3. **Subjektiivne hinnang**: Isiklik suhtlus ei ole vaba subjektiivsusest. Intervjueerija otsus võib olla mitmeti mõjutatud — alates isiklikest eelistustest (näiteks parfüümi lõhn või kandidaadi riietumisstiil) kuni kandidaadi karismaatilisuseni, mis võib viia tema oskuste ülehindamiseni.
4. **Intervjueerija ebapädevus**: Harvadel juhtudel võib esineda olukordi, kus ettevõttes puuduvad piisava kvalifikatsiooniga spetsialistid, kes suudaksid kandidaadi professionaalseid oskusi objektiivselt hinnata, mis mõjutab negatiivselt ka intervjuu tulemusi.
5. **Piiratud võimalus kandidaatide objektiivseks võrdlemiseks**: Küsimused ja hindamiskriteeriumid võivad erineda olenevalt intervjueerijast, mis raskendab kandidaatide objektiivset võrdlemist ja parima kandidaadi määramist. See probleem ilmneb eriti suure arvu kandidaatide puhul, kus intervjueerijatel on raske säilitada ühtseid standardeid.
6. **Tehnilised probleemid**: Võimalikud seadmete või internetiühenduse rikked võivad samuti mõjutada intervjuu kvaliteeti negatiivselt, mis on eriti oluline kaugintervjuude puhul, kus ühenduse kvaliteet sõltub paljudest teguritest.

2023. aastal, töötades kesktaseme arendajana, viis autor läbi küsitluse, milles osales 12 vanemarendajat ja tehnoloogiajuhti, kes tegelevad tehniliste tööintervjuudega. Autor küsis neilt, kuidas nad hindavad kandidaatide teoreetilisi teadmisi. Kõik vastajad (100%) märkisid, et nad hindavad neid teadmisi isikliku intervjuu käigus. Küsimusele, miks seda osa protsessist ei võiks automatiseerida, vastas 17% vastanutest, et neil puudub selleks vajadus, 8% ütles, et nad olid otsinud vastavat lahendust, kuid ei leidnud sobivat, ja ülejäänud 75% märkisid, et see on nende ettevõttes juurdunud praktika, ja nad ei ole mõelnud teistele lähenemistele.

Autoril tekkis idee automatiseerida see osa protsessist. Järgneva olemasolevate toodete uuringu tulemusena, mis olid saadaval Eesti turul 2023. aastal, selgus, et ei olnud sobiva hinnaga ja/või funktsionaalsusega lahendusi, mis oleksid tööandjale vastanud. Enamik olemasolevatest toodetest pakkus testimise platvorme, kuid teste tuli koostada käsitsi, mis suurendas ajakulu. Mõned ettevõtted pakkusid tehnilise intervjuu teenust "teenusena", kuid

selline lähenemine ei vastanud samuti tööandja vajadustele.

Nende probleemide lahendamiseks jõuti autori tööandjaga kokkuleppele, mille kohaselt autor arendab süsteemi kandidaatide testimiseks, mis võimaldab objektiivselt ja kvaliteetselt hinnata nende teadmisi ja oskusi. Positiivsete tulemuste korral saaks toodet edasi arendada täieõiguslikuks kommertslahenduseks.

## 1.2 Taust

**Testimisvaldkonna keerukus**: Kandidaatide testimise protsess hõlmab vajadust arvestada mitmeid muutujaid, nagu testitavad teemad, küsimuste keerukuse tase, vastuse tüübid ja tulemuste hindamise meetodid. Näiteks on küsimuste keerukuse adekvaatne klassifitseerimine oluline kandidaadi teadmiste taseme õigeks hindamiseks, ning küsimuste tüüpide valik (näiteks valikvastustega või avatud vastusega küsimused) määrab, kui täpsed ja objektiivsed on tulemused. Testimine peab jääma piisavalt odavaks nii loomise kui ka läbiviimise osas ning olema rakendamisel kasutajasõbralik. Praktikas on sageli keeruline leida tasakaalu kulude ja kvaliteedi, objektiivsuse ja paindlikkuse vahel, mis võimaldab kohaneda testi koostaja vajadustega. Samuti on üheks suuremaks väljakutseks tagada tulemuste usaldusväärsus ja täpsus, minimeerides tegurite mõju, mis ei ole seotud kandidaadi professionaalsete oskustega.

**Süsteemi kasutajad**: Peamisteks loodava süsteemi kasutajateks on tööle kandideerijad, kes läbivad testimise, ning personaliosakonna spetsialistid ja tehnilised juhid, kes hindavad tulemusi. Kandidaatide jaoks peab liides olema intuitiivselt arusaadav, et tagada minimaalsed raskused testi läbimisel. Personaliosakonna spetsialistide ja tehniliste juhtide jaoks peab liides olema funktsionaalne, pakkudes paindlikke võimalusi testide koostamiseks, tulemuste jälgimiseks ja testimisaruannete saamiseks. Täpsem kirjeldus kasutajate nõudmiste kohta on esitatud peatükis 1.3 "Eesmärk".

**Ülevaade veebitestides enim kasutatavatest testimisliikidest**:

1. **Staatilised testid**: Kõige sagedamini kasutatakse staatilisi teste, mis koosnevad kindlast hulgast küsimustest või juhuslikult valitud küsimustest ettevalmistatud küsimustepangast. Staatiliste testide tulemuseks on tavaliselt kvantitatiivne hinnang, mis põhineb õigesti vastatud küsimuste arvul, pakkudes lihtsat ja kiiret viisi testitava

teadmiste taseme hindamiseks. Selline lähenemine on laialdaselt kasutusel, kuna see on lihtne realiseerida ja ei nõua keerulist seadistamist.

2. **Adaptiivsed testid**: Harvem kasutatakse adaptiivseid teste, mille loomine on oluliselt keerulisem. Need testid kohanduvad sõltuvalt testitava teadmiste tasemest, mis nõuab iga küsimuse suhtelise keerukuse määramist. Igal küsimusel on määratud omadus — keerukuse tase. Kui testitäitja vastab küsimusele õigesti, järgneb järgmine küsimus kõrgema keerukusega; kui vastus on vale, järgneb lihtsama keerukusega küsimus. Adaptiivsete testide eesmärgiks on leida keerukuse tase, millele testitäitja on võimeline vastama. Adaptiivsed testid võivad olla täpsemad, kuid nende väljatöötamine nõuab rohkem ressursse ja aega, et luua süsteem, mis suudaks kohanduda iga kandidaadiga.
3. **Diagnostilised testid**: Kõige täpsemad tulemused on võimalik saada diagnostiliste testide abil, mis suudavad tuvastada nii kandidaadi tugevad kui ka nõrgad küljed. Kuid diagnostiliste testide loomine ja seadistamine on väga keeruline, kuna need nõuavad süvitsi mõistmist testitavast teemast ja keerulist testi seadistamist. Näiteks tuleb määrata küsimuste järjestus sõltuvalt eelmistest vastustest ning see, kuidas sõnastatakse lõplik hinnang kandidaadi oskustele. Diagnostiliste testide tulemuseks on kvalitatiivne hinnang, mis annab detailse ülevaate testitava teadmiste tugevustest ja nõrkustest, pakkudes seeläbi põhjalikumat ja mitmekülgsemat arusaama tema teadmistest ja oskustest.

Tabelis 1 on toodud valitud testitüüpide võrdlus, mis aitab paremini mõista iga tüübi eeliseid ja puudusi.

| Testitüüp | Eeliseid | Puudused |
|---|---|---|
| **Staatilised testid** | - Lihtne realiseerimine<br>- Madalad loomiskulud<br>- Kiire rakendamine<br>- Lihtne mõista ja kasutada | - Madal täpsus teadmiste taseme hindamisel<br>- Piiratud paindlikkus<br>- Kõik kandidaadid vastavad samadele küsimustele, mis muudab testi vähem kohandatavaks |
| **Adaptiivsed testid** | - Hea teadmiste taseme hindamise täpsus<br>- Individuaalne lähenemine igale kandidaadile<br>- Kohandatav küsimuste keerukus sõltuvalt kandidaadi tasemest | - Keeruline realiseerimine<br>- Kõrged arenduskulud<br>- Vajab märkimisväärset aega ja ressursse küsimuste keerukuse määramiseks |
| **Diagnostilised testid** | - Väga kõrge teadmiste hindamise täpsus<br>- Võimalus hinnata nii kandidaadi tugevaid kui ka nõrku külgi<br>- Üksikasjalikud aruanded testitulemustest<br>- Võimalus arvestada keerulisi stsenaariume | - Väga kõrged kulud<br>- Keeruline arendus ja seadistamine<br>- Vajab ekspertide sügavat teadmist küsimuste koostamisel<br>- Vajab keerukamat tehnoloogiat ja algoritme küsimuste vaheliseks liikumiseks ning lõpparuande koostamiseks |

Tabel 1. Valitud testitüüpide võrdlus.

## 1.3 Eesmärk

Käesoleva bakalaureusetöö eesmärgiks on luua rakenduse prototüüp, mis pakub võimalust koostada, läbi viia ja kuvada nii adaptiivsete kui ka diagnostiliste testide tulemusi, mille abil saaks parandada kandidaatide teadmiste objektiivset hindamist. Põhiülesanne on välja töötada testimissüsteem, mis on kohanduv ja kasutajasõbralik.

Projekt on suunatud järgmiste eesmärkide saavutamisele erinevatele kasutajagruppidele:

1. **Tööle kandideerijad**: Pakutakse mugavat ja arusaadavat liidest testide sooritamiseks ning testitulemuste vaatamiseks.
2. **Personaliosakonna spetsialistid ja tehnilised juhid**: Pakutakse paindlikke võimalusi testide koostamiseks, tulemuste analüüsimiseks ja kandidaatide objektiivseks hindamiseks.

Eesmärk ei ole ainult prototüübi loomine, vaid ka selle valmiduse tagamine edasiseks täiustamiseks, mis võimaldaks selle rakendamist reaalses olukorras ja täisväärtusliku kaubandusliku lahenduse väljatöötamist.

Projekt koosneb järgmistest etappidest:

1. **Konkurentide analüüs**: Turul olevate olemasolevate lahenduste analüüs, et tuvastada nende tugevad ja nõrgad küljed, funktsionaalsus ja eripärad. See aitab täpsemalt kindlaks teha, millised võimalused tuleb süsteemis realiseerida, et saavutada konkurentsieelis.
2. **Rakenduse arhitektuuri valik**: Määrata projekti vajadustele ja võimalikele kasutusstsenaariumitele vastav rakenduse arhitektuur. Arhitektuuri valimisel arvestatakse süsteemi skaleeritavuse ja kohanemisvõime vajadust, mis tagab edasise arendamise ja täiustamise võimaluse.
3. **Andmebaasi väljatöötamine**: Andmebaasi kavandamine ja loomine, mis suudab tõhusalt salvestada teavet testide, tulemuste ja kasutajate kohta. Andmebaasi arendamine mängib võtmerolli andmete paindliku ja usaldusväärse säilitamise ning süsteemi laiendatavuse tagamisel.
4. **Kasutajaliidese arendamine testide koostamiseks ja tulemuste vaatamiseks**: Intuitiivse ja arusaadava liidese loomine, mis võimaldab administraatoritel või personaliosakonna spetsialistidel mugavalt teste koostada ja tulemusi analüüsida. Liides peab toetama testide keerukuse ja sisu kohandamise funktsioone, mis tagab kasutamise paindlikkuse.
5. **Rakenduse serveriosa loomine**: Vajalikku tagaplaaniloogikat arendatakse süsteemi kõigi vajalike funktsioonide tagamiseks, nagu näiteks testide töötlemine, kasutajate autoriseerimine ja andmete salvestamine. Samuti tagatakse süsteemi skaleeritavus

kasutajate ja kandidaatide arvu suurenemise korral.

6. **Testide läbiviimise lehe arendamine**: Liidese loomine, mida testitavad kasutavad testide läbiviimiseks ja tulemuste saamiseks. Oluline on tagada kasutusmugavus ja minimaalne ajakulu liidese kasutamiseks.
7. **Esialgne testimine ja tagasiside kogumine**: Rakenduse esialgne testimine väikese hulga reaalsete kasutajate peal, et tuvastada võimalikke probleeme ja hinnata selle mugavust ning tõhusust. Kasutajate, nagu personaliosakonna spetsialistide ja kandidaatide, tagasiside kogumine aitab saada väärtuslikku teavet võimalike täiustuste kohta.

# 2 Analüüs

Selles peatükis viiakse läbi konkurentide omaduste ja puuduste analüüs, et tuvastada parimad praktikad, mida on võimalik integreerida välja töötatavasse süsteemi. Lisaks määratletakse arendatava rakenduse jaoks olulised nõuded, mis tagavad selle vastavuse sihtrühmade ootustele ja praktilistele vajadustele.

## 2.1 Konkurendid

Peatükk annab ülevaate konkurentidest, kes pakuvad lahendusi testimise ja hindamise valdkonnas. Iga konkurendi eeliseid ja puudusi on analüüsitud, et leida võimalusi nende omaduste integreerimiseks loodavasse süsteemi.

### 2.1.1 Google Forms

Google Forms [1] on tasuta platvorm, mis pakub funktsionaalsust staatiliste testide loomiseks.

**Eelised:** Kasutamise lihtsus, mis muudab selle sobivaks kiireks testide koostamiseks; platvormi populaarsus, mis tagab selle laialdase tunnustuse ja kasutajate kogukonna; tasuta kasutamine, mis sobib hästi väiksemate projektide ja eelarvepiirangutega.

**Puudused:** Piiratud funktsionaalsus, mis ei võimalda luua keerukamaid teste, näiteks adaptiivseid või diagnostilisi teste; ei paku tööriistade integreerimise võimalust keerukate andmeanalüüside tegemiseks; sobimatus süsteemidele, mis nõuavad suuremat paindlikkust ja kohandatavust.

### 2.1.2 Moodle

Moodle [2] on veebipõhine õppehaldussüsteem, mis on avatud lähtekoodiga ja litsentsitud GNU GPL-i alusel. Moodle on laialdaselt kasutusel Euroopas, sealhulgas näiteks Tartu Ülikoolis ja Tallinna Tehnikaülikoolis (TalTech). Moodle'i ökosüsteem sisaldab palju

erinevaid pistikprogramme, mida kogukond on loonud. Üheks selliseks sobivaks mooduliks on *Adaptive Quiz* [3], mis võimaldab luua adaptiivseid teste.

**Eelised:** Moodle on kasutusel olnud alates 2002. aastast ning selle arendamine jätkub aktiivselt. Platvorm on väga paindlik ja võimaldab kohandada funktsioone vastavalt kasutajate vajadustele. Lisaks toetab Moodle kolmandate osapoolte teenustega integreerimist ning võimaldab luua kohandatud mooduleid, muutes süsteemi sobivaks keerukate õppehaldustööde jaoks.

**Puudused:** Moodle sisaldab sageli liigset funktsionaalsust, mis võib muuta selle kasutajate jaoks keeruliseks. Süsteemi keerukus teeb selle vähem sobivaks kandidaatide testimiseks, kuna Moodle on peamiselt loodud hariduslikuks otstarbeks. Arendajate seisukohalt on puuduseks vananenud programmeerimiskeele PHP kasutamine, mis võib raskendada kaasaegsete arendusmeetodite rakendamist.

### 2.1.3 EIS

Eksamite Infosüsteem (EIS) [4] [5] on üldhariduskoolidele suunatud keskkond, kus saab läbi viia ja sooritada teste (sh eksameid), vaadata oma tulemusi ja sooritatud eksamite tunnistusi.

**Eelised:** süsteem sisaldab laia valikut erinevat tüüpi teste, mis muudab selle hariduslike eesmärkide saavutamiseks universaalseks. Platvorm on spetsiaalselt loodud testimise läbiviimiseks, mis tagab selle funktsionaalse optimeerimise. Saadaval on diagnostilised testid, mis aitavad õpetajaid ja õpilasi õppeprotsessis ja eksamiteks valmistumisel [6] [7].

**Puudused:** süsteemi piiratus koolitestidega ei võimalda selle kasutamist teistes valdkondades. Puudub integratsioon väliste toodetega. Samuti ei sobi platvorm äriliseks kasutamiseks, kuna see on mõeldud ainult hariduslikuks otstarbeks.

### 2.1.4 iMocha

iMocha [8] on veebipõhine hindamisplatvorm, mis on suunatud peamiselt ettevõtetele ja organisatsioonidele töötajate hindamiseks ja oskuste testimiseks.

**Eelised:** Platvorm pakub suurt hulka valmisolevaid teste, sealhulgas oskuspõhiseid ja

tehnilisi hindamisi. iMocha toetab kohandatud testide loomist, mis vastavad täpselt organisatsiooni vajadustele. Lisaks sisaldab see ulatuslikke analüüsi- ja aruandlustööriistu, mis võimaldavad hinnata tulemusi süvitsi. Platvorm on integreeritav mitmete ettevõtte kasutustarkvaradega, muutes selle sobivaks ärilises kontekstis.

**Puudused:** iMocha on tasuline teenus, mis võib väiksemate organisatsioonide või eelarvepiirangutega projektide jaoks olla kulukas. Halvasti dokumenteeritud: analüüsietapis tuli süsteemi funktsionaalsusega tutvumiseks vaadata iMocha tootevideote ülevaateid.

### 2.1.5 Diagnostic Questions

Diagnostic Questions [9] on veebipõhine keskkond, mis on loodud haridusasutustele ja õpetajatele, et aidata kaasa õpilaste mõistmise ja oskuste hindamisele läbi küsimuste põhise diagnostika.

**Eelised:** Platvorm toetab interaktiivsete küsimuste ja vastuste loomist, mis võimaldavad õpetajatel kiiresti tuvastada õpilaste puudujäägid teadmistes. Diagnostic Questions sisaldab mitmekülgseid aruandlustööriistu, mis pakuvad põhjalikku ülevaadet õppijate sooritustest. Lisaks on see tasuta kättesaadav õpetajatele ja haridusasutustele, mis muudab selle laialdaselt kättesaadavaks.

**Puudused:** Keskkond on suunatud peamiselt haridusasutustele, mistõttu ei sobi see ettevõtete või muude organisatsioonide vajaduste jaoks. Piiratud kohandamisvõimalused võivad olla takistuseks spetsiifiliste testide loomisel. Lisaks võib puududa integratsioon keerukamate analüüsi- ja hindamissüsteemidega.

### 2.1.6 Konkurentide võrdlus

Konkurentide analüüsi põhjal võib järeldada, et igal vaadeldud tootel on oma tugevad ja nõrgad küljed. Enamiku konkurentide peamine sihtrühm on haridussektoris tegutsevad kasutajad, samas kui ainult iMocha positsioneerib end ärirakendustele suunatud tootena. Selle projekti raames tuleks proovida kasutada konkurentide parimaid omadusi: Google Forms'ilt võib üle võtta intuitiivse kasutajaliidese testide koostamiseks, Moodle'ilt ja EIS'ilt testide seadistamise paindlikkuse ning iMocha'lt ärirakendustele kohandatud testimispõhimõtted. Tabelis 2 on toodud valitud konkurentide võrdlus, mis aitab paremini

mõista iga tüübi eeliseid ja puudusi.

| Konkurent | Eeliseid | Puudused |
|---|---|---|
| **Google Forms** | - Lihtne kasutada<br>- Tasuta<br>- Sobib kiireks testide koostamiseks | - Puuduvad adaptiivsed testid<br>- Piiratud analüüsivõimalused<br>- Ei sobi keerukatele süsteemidele |
| **Moodle** | - Paindlik ja kohandatav<br>- Toetab mooduleid, sealhulgas adaptiivseid teste<br>- Integreeritav kolmandate osapoolte teenustega | - Liigne funktsionaalsus<br>- Keeruline kasutada<br>- Aegunud PHP kasutamine |
| **EIS** | - Lai valik teste<br>- Optimeeritud hariduslike eesmärkide jaoks<br>- Toetab diagnostilisi teste | - Piiratud haridusega seotud valdkonnaga<br>- Ei ole integreeritav teiste süsteemidega<br>- Ei sobi äriliseks kasutamiseks |
| **iMocha** | - Suur valik valmis teste<br>- Analüüsi- ja aruandlusvõimalused<br>- Integreeritav ettevõtte tarkvaradega | - Kõrge hind<br>- Ebapiisav dokumentatsioon |
| **Diagnostic Questions** | - Sobib hariduslikeks eesmärkideks<br>- Tasuta kasutamine<br>- Tõhus aruandlusvahend | - Ei sobi äriliseks kasutamiseks<br>- Piiratud kohandamisvõimalused<br>- Puuduvad keerukad analüüsifunktsioonid |

Tabel 2. Valitud konkurentide võrdlus.

## 2.2 Nõuete analüüs

Selles peatükis analüüsitakse süsteemi funktsionaalseid ja mittefunktsionaalseid nõudeid. Tuuakse esile testide ja küsimuste haldamise, adaptiivse ja diagnostilise testimise ning turvalisuse tagamise aspektid.

### 2.2.1 Funktsionaalsed nõuded

- **Testide haldamine:**
  - Uute testide loomise võimalus.
  - Olemasolevate testide redigeerimine.
  - Aegunud või mittevajalike testide kustutamine.
  - Testide klassifitseerimine kategooriate, teemade või raskusastmete järgi.
- **Küsimuste haldamine:**
  - Küsimuste lisamine, redigeerimine ja kustutamine testides.
  - Erinevate küsimuste tüüpide tugi.
  - Võimalus määrata küsimuste raskusaste.
- **Adaptiivne testimine:**
  - Küsimuste automaatne valik kasutaja varasemate vastuste põhjal.
  - Kasutaja teadmiste taseme hindamine testimise käigus.
- **Diagnostiline testimine:**
  - Testitava tugevate ja nõrkade külgede määramine.
  - Üksikasjaliku aruande genereerimine testi tulemustest.
- **Testide sooritamine:**
  - Võimalus valida test saadaval olevate testide nimekirjast.
  - Vahepealsete tulemuste salvestamine, et testi hiljem jätkata.
  - Ajapiirang testi sooritamiseks taimeri abil.
- **Aruandlus ja analüütika:**
  - Testimise tulemuste põhjal aruannete loomine.
  - Andmete visualiseerimine (graafikud, diagrammid) kasutajate edusammude kohta.
  - Erinevate kasutajate tulemuste võrdlemine.

### 2.2.2 Mittefunktsionaalsed nõuded

- **Kasutajate samaaegne töö:** Süsteem peab toetama testide samaaegset sooritamist vähemalt 10 kasutaja poolt ilma jõudluse vähenemise või päringute töötlemise viivitusteta.
- **Hästi seadistatud logimine:** Kõik süsteemi päringud (sisselogimine, registreerimine, testide sooritamine, testide redigeerimine jne) peavad olema logides korrektselt fikseeritud edasiseks analüüsiks. Logid peavad sisaldama minimaalselt vajalikke andmeid (aeg, päringu tüüp, tulemus, IP-aadress jne) ning välistama konfidentsiaalse teabe salvestamise.
- **Turvalisus (Security):** Kasutajate autentimiseks tuleb rakendada tokenipõhist mehhanismi (JWT).
- **Virtuaalse serveri kasutamine:** Süsteem peab olema paigaldatud ühele virtuaalsele serverile operatsioonisüsteemiga Ubuntu, minimaalsete süsteeminõuetega: 2 CPU tuuma, 4 GB RAM, 50 GB vaba kettaruumi.
- **Skaleeritavus:** Süsteem peab olema projekteeritud nii, et võimaldada kasutajate arvu suurenemist ja toetada horisontaalset või vertikaalset skaleerimist (andmebaasi viimine eraldi serverisse või rakenduse jagamine mikroteenusteks).

# 3 Süsteemi arhitektuur

Arendatav rakendus koosneb kahest osast: esirakendusest ja tagarakendusest. Esirakendus hõlmab halduspaneeli testide loomiseks ja muutmiseks ning kandidaadile suunatud kasutajaliidest. Tagarakendus tagab andmete töötlemise ja äriloogika rakendamise. Rakenduse loomisel võeti arvesse võimalikku kasvu ja laienemist ning selle arhitektuuri planeerimisel pandi alus mikroteenuste arhitektuurile. Lisaks on serveril paigaldatud nginx, mis täidab pöördproksi rolli. Käesolevas peatükis käsitletakse kasutatavaid tehnoloogiaid ja nende valiku põhjendusi.

## 3.1 Esirakendus

Esirakendus arendati vastavalt Rapid Prototyping [10] ja Minimum Viable Product (MVP) [11] põhimõtetele, et kiirendada arendusprotsessi ja keskenduda oluliste funktsioonide realiseerimisele. Valitud tehnoloogiad, nagu Vue.js, Vite ja Tailwind CSS, võimaldavad kiiret prototüüpide loomist ja liidese lihtsat kohandamist kasutajate vajadustele. Eriti pöörati tähelepanu kasutajaliidese lihtsusele ja funktsionaalsusele, et tagada kasutusmugavus ning luua paindlik alus süsteemi edasiseks arendamiseks ja täiustamiseks.

## 3.2 Tagarakendus

Tagarakenduse arendamiseks valiti kaasaegsed tehnoloogiad, mida laialdaselt kasutatakse ärisektoris: Java, Spring Boot ja PostgreSQL. Need tööriistad tagavad arendamise mugavuse, töökindluse ja kõrge jõudluse. Rakenduse koostamiseks kasutati Gradle'i, mis võimaldab tõhusat sõltuvuste haldamist ja arenduse paindlikkust. Lisaks integreeriti süsteemi Swagger, et lihtsustada arendusprotsessi ja pakkuda automaatselt genereeritud REST API dokumentatsiooni. Peamine tähelepanu oli suunatud lihtsale integreerimisele süsteemi teiste komponentidega, näiteks esirakenduse ja andmebaasiga, ning skaleeritavusele, et rakendus saaks kohaneda muutuvate nõuetega ja jätkata arengut.

## 3.3 Andmebaas

PostgreSQL — see on tasuta relatsiooniline andmebaas, millel on avatud lähtekood. Projektis kasutatakse PostgreSQL versiooni 16. PostgreSQL on tuntud oma usaldusväärsuse, stabiilsuse ja laiaulatusliku kogukonna toetuse poolest. PostgreSQL on hästi dokumenteeritud ja aktiivselt arendatav. Arendajal on olemas kogemus selle andmebaasiga töötamisel.

## 3.4 CI/CD

CI/CD [12] kasutamise eesmärk projektis on arenduse, testimise ja juurutamise protsesside automatiseerimine, mis võimaldab uute versioonide väljalaset kiirendada ja arenduse stabiilsust suurendada. Rakenduse lähtekoodi ja CI/CD protsesside haldamiseks kasutatakse ülikooli GitLab. Rakenduse konteineriseerimiseks kasutatakse Docker-it, mis lihtsustab rakenduse testimist ja juurutamist prognoositavates keskkondades. DockerHub on kasutusel Docker-piltide hoidmiseks ja jagamiseks, võimaldades kiiresti juurutada uusi versioone ning toetades rakenduse arenduse ja halduse järjepidevust.

# 4 Kasutajavoog

Parema arusaamise huvides on allpool kirjeldatud kasutajavoog nii testi lahendajale (SOLVER) kui ka testi loojale (CREATOR).

## 4.1 Üldine

- **Registreerimine:**
  - Kasutaja vajutab nuppu *REGISTREERIMINE*.
  - Kasutaja täidab väljad: nimi, perekonnanimi, parool, e-mail ning valib rolli *CREATOR* (testide loomiseks) või *SOLVER* (testide lahendamiseks).
  - Päring saadetakse serverisse, kus kontrollitakse vajalike väljade olemasolu ja e-maili unikaalsust:
    - Kui e-mail pole varem registreeritud, teavitatakse kasutajat registreerimise edukast lõpetamisest.
    - Kui e-mail on juba süsteemis olemas, kuvatakse kasutajale veateade e-maili olemasolust.
- **Sisse logimine:**
  - Kasutaja vajutab nuppu *LOGI SISSE*.
  - Kasutaja sisestab e-maili ja parooli.
  - Andmed saadetakse serverisse, kus kontrollitakse e-maili olemasolu ja parooli õigsust:
    - Kui kontroll on edukas, genereerib server JWT ja saadab selle kasutajale.
    - Kui kontroll ebaõnnestub, kuvatakse kasutajale veateade.

## 4.2 Testi lahendaja (SOLVER)

- **Testi lahendamine:**
  - Eeltingimus: kasutaja on olemas ja sisse loginud rolliga *SOLVER*.
  - Kasutaja avab testide nimekirja.

  - Kasutaja valib sobiva testi ja tutvub testi reeglitega.
  - Kasutaja vastab küsimustele seni, kuni server saadab küsimusi.
  - Testi lõppedes saab kasutaja vaadata tulemusi või naasta avalehele.
- **Tulemuste vaatamine:**
  - Kasutaja vajutab nupule *Minu testid*.
  - Kasutaja valib nimekirjast teda huvitava testi.
  - Kasutaja saab vaadata oma testide tulemusi.

## 4.3 Testi looja (CREATOR)

- **Küsimuse lisamine:**
  - Kasutaja vajutab nupule *Testide haldamine*.
  - Kasutaja valib menüüst *Küsimuste haldamine*.
  - Kasutaja valib *Lisa küsimus*.
  - Kasutaja täidab küsimuse vormi: tekst, kategooria, tüüp, raskusaste.
  - Kasutaja lisab küsimusele vastused.
  - Kasutaja vajutab nuppu *Salvesta*.
- **Küsimuse muutmine:**
  - Kasutaja vajutab nupule *Testide haldamine*.
  - Kasutaja valib menüüst *Küsimuste haldamine*.
  - Kasutaja valib *Muuda küsimust*.
  - Kasutaja uuendab küsimuse vormi: tekst, kategooria, tüüp, raskusaste.
  - Kasutaja saab muuta olemasolevaid vastuseid või lisada uusi vastuseid.
  - Kasutaja vajutab nuppu *Salvesta*.
- **Küsimuse kustutamine:**
  - Kasutaja vajutab nupule *Testide haldamine*.
  - Kasutaja valib menüüst *Küsimuste haldamine*.
  - Kasutaja valib *Kustuta küsimus*.
  - Kasutaja kinnitab küsimuse kustutamise.
- **Testi lisamine:**
  - Kasutaja vajutab nupule *Testide haldamine*.
  - Kasutaja valib *Lisa test*.
  - Kasutaja lisab testi kirjelduse, kategooriad ja valib küsimused, mis testi

kuuluvad.

- **Testi muutmine:**
  - Kasutaja vajutab nupule *Testide haldamine*.
  - Kasutaja valib *Muuda testi*.
  - Kasutaja uuendab testi kirjeldust, kategooriaid ja küsimusi.
  - Kasutaja vajutab nuppu *Salvesta*.
- **Testi kustutamine:**
  - Kasutaja vajutab nupule *Testide haldamine*.
  - Kasutaja valib *Kustuta test*.
  - Kasutaja kinnitab testi kustutamise.
- **Testi tulemuste vaatamine:**
  - Kasutaja vajutab nupule *Testide haldamine*.
  - Kasutaja valib *Vaata tulemusi*.

# 5 Andmemudel

Andmemudel mängib projektis võtmerolli. See kujutab objekte, nende omadusi ja omavahelisi seoseid, millega töö toimub. Kuna projekti arendamine toimub vastavalt agiilse metoodika põhimõtetele, luuakse andmemudel iteratiivselt. Arenduse käigus, kui arendajate arusaam domeenist süveneb, võidakse üksusi ja nende omadusi lisada, muuta või eemaldada [13]. See võimaldab projekti paremaks muuta, täpsustada andmemudelit ning muuta see lihtsamaks ja arusaadavamaks.

Antud projektis viidi andmemudelisse üks kord sisse oluline muudatus, kuna algselt ei olnud arvesse võetud olulisi aspekte, mis on seotud kasutaja vastuste salvestamisega. Samuti väärib märkimist, et andmemudel muutus uue funktsionaalsuse lisamisega. Näiteks lisati tabelisse *test* veerg *std_error*, et salvestada määratud standardvea väärtust. Pärast testserverisse juurutamist viidi andmemudeli muudatused sisse andmebaasi migratsioonide abil.

Allpool on esitatud andmemudel seisuga 13.12.2024.

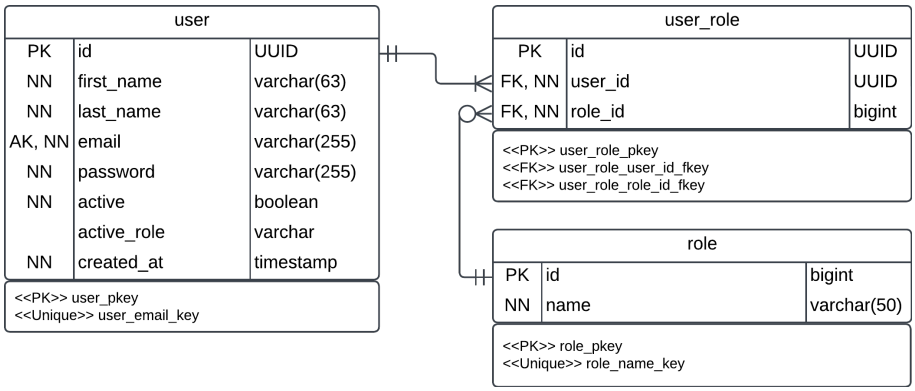
## 5.1 Kasutaja

Kasutaja on isik, kes kasutab antud rakendust. Kasutajat iseloomustab unikaalne e-posti aadress, mis peab süsteemis olema kordumatu. Kasutajal võib olla üks või mitu rolli, mille vahel ta saab vajadusel vahetada. Käesolevas bakalaureusetöös käsitletakse ainult kahte rolli:

- **CREATOR** — testi looja.
- **SOLVER** — testi lahendaja.

Kasutaja paroolid on hashitud algoritmiga **BCrypt** ning neid säilitatakse ainult räsi kujul, et vähendada võimalikke kahjusid andmebaasiandmete lekkimise korral.

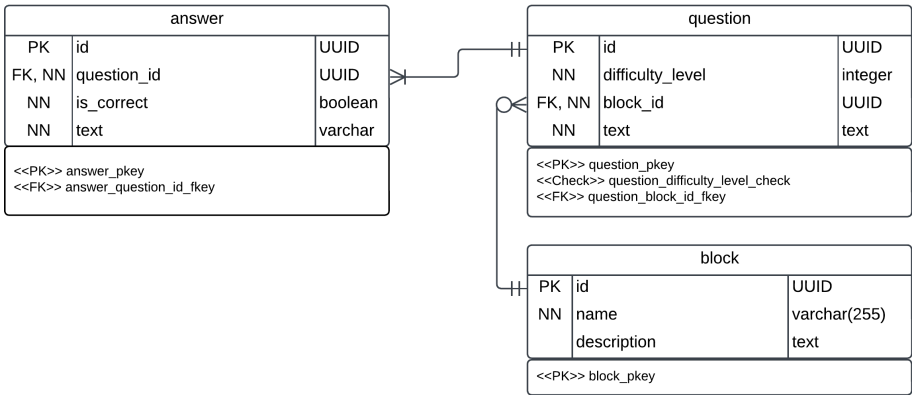
Allpool oleval Joonisel 1 on esitatud kasutaja üksuse struktuur ning selle seosed rollidega.




Joonis 1. Kasutaja andmestruktuur.

## 5.2 Küsimuste plokk

Blokk on küsimuste kogum, mis on ühendatud ühise teemaga. Igal küsimusel võib olla mitu vastusevarianti. Küsimuste blokk kujutab endast minimaalset ühikut testi koostamiseks. Autor peab seda lahendust prototüübi jaoks piisavalt sobivaks, kuna antud realiseering ei takista testimist. Igal küsimusel on raskusastme omadus, mis võimaldab viia läbi adaptiivseid teste. Järgmistes arendusetappides on plaanis võimaldada kasutajale kuvada ainult osa vastustest, et suurendada testi varieeruvust. See vähendab märgatavalt võimalust meelde jätta vastusemalle ning muudab testi usaldusväärsemaks teadmiste hindamise seisukohalt.




Joonis 2. Küsimuse ja vastuste andmestruktuur.

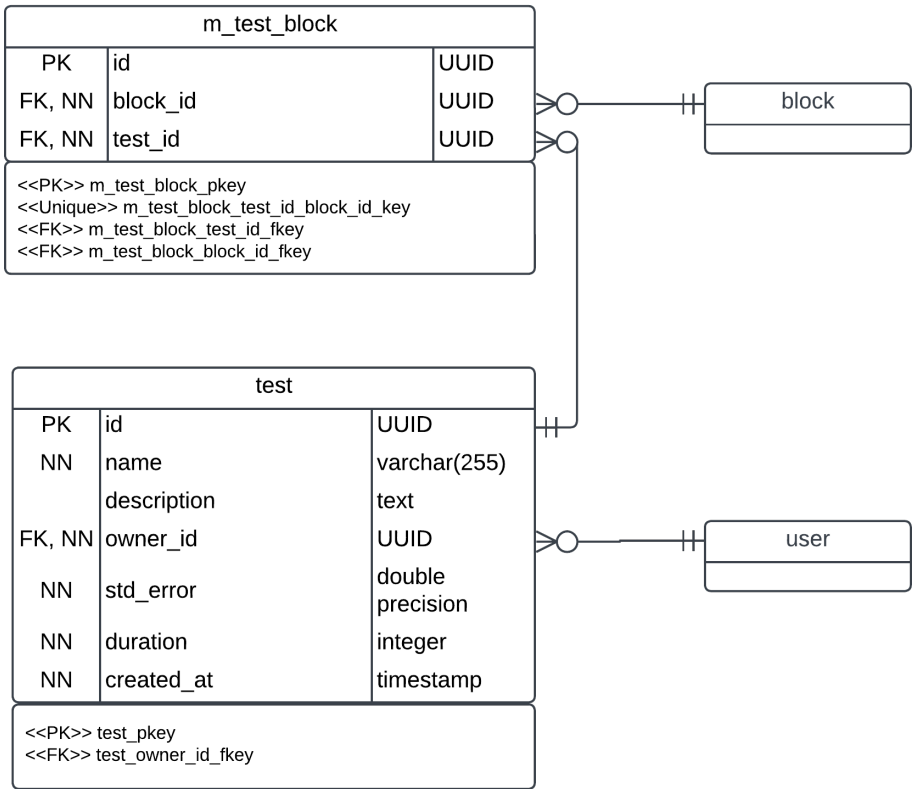
## 5.3 Test

Test võib koosneda ühest või mitmest plokist. Iga plokk võib omakorda kuuluda mitme testi koosseisu. Selline lähenemine võimaldab luua teste erinevate teadmiste valdkondade jaoks, taaskasutades juba valmis plokke. Näiteks saab edasijõudnute programmeerimise

testimiseks kasutada plokke, mis pärinevad algtaseme programmeerimise testidest.

Testi läbimise aega saab piirata omadusega *duration*, mis määrab lubatud sooritamise kestuse. Samuti määratakse testi täpsus kandidaadi teadmiste hindamisel, kasutades standardvea piirväärtust omaduses *std_error*. Kui standardviga muutub väiksemaks kui määratud väärtus, tähendab see, et kandidaadi teadmiste hindamine on piisavalt täpne, praegune plokk loetakse lõpetatuks ning testimine liigub järgmise ploki juurde või lõppeb.

Testi läbimise tingimustest räägitakse lähemalt peatükis 5.4 "Kasutaja test"




Joonis 3. Testi ja blokki andmestruktuur.

## 5.4 Kasutaja test

Kasutaja test on antud rakenduse andmemudeli kõige keerulisem osa. Süsteem salvestab kasutajale esitatud küsimused ja tema vastused. See võimaldab vajadusel taastada konkreetse kasutaja testimisprotsessi. Pärast igat vastust kontrollib süsteem selle õigsust. Kõik väljad, mille alusel tehakse otsinguid andmebaasis, on indekseeritud, mis tagab kiire otsingu.

Käesolevas töös on kasutusele võetud küsimused mitme valikuga vastustega, kus punktide arv jaotatakse proportsionaalselt õigete vastuste alusel ning valede vastuste eest punkte ei

vähendata. Selline küsimusetüüp on tehniliselt lihtsasti realiseeritav, kuid samas võimaldab paindlikult ja täpselt hinnata testi tulemusi. Ühe õige vastusega küsimused on mitme valikuga küsimuste alamkogum. Avatud vastustega küsimusi on keeruline masinloetavalt hinnata; järgmistes iteratsioonides on plaanis rakendada avatud vastuste hindamist kasutades LLM-i.

Õigesti vastatud ja küsimuste koguarv salvestatakse tabelisse *user_test_block*. See võimaldab:

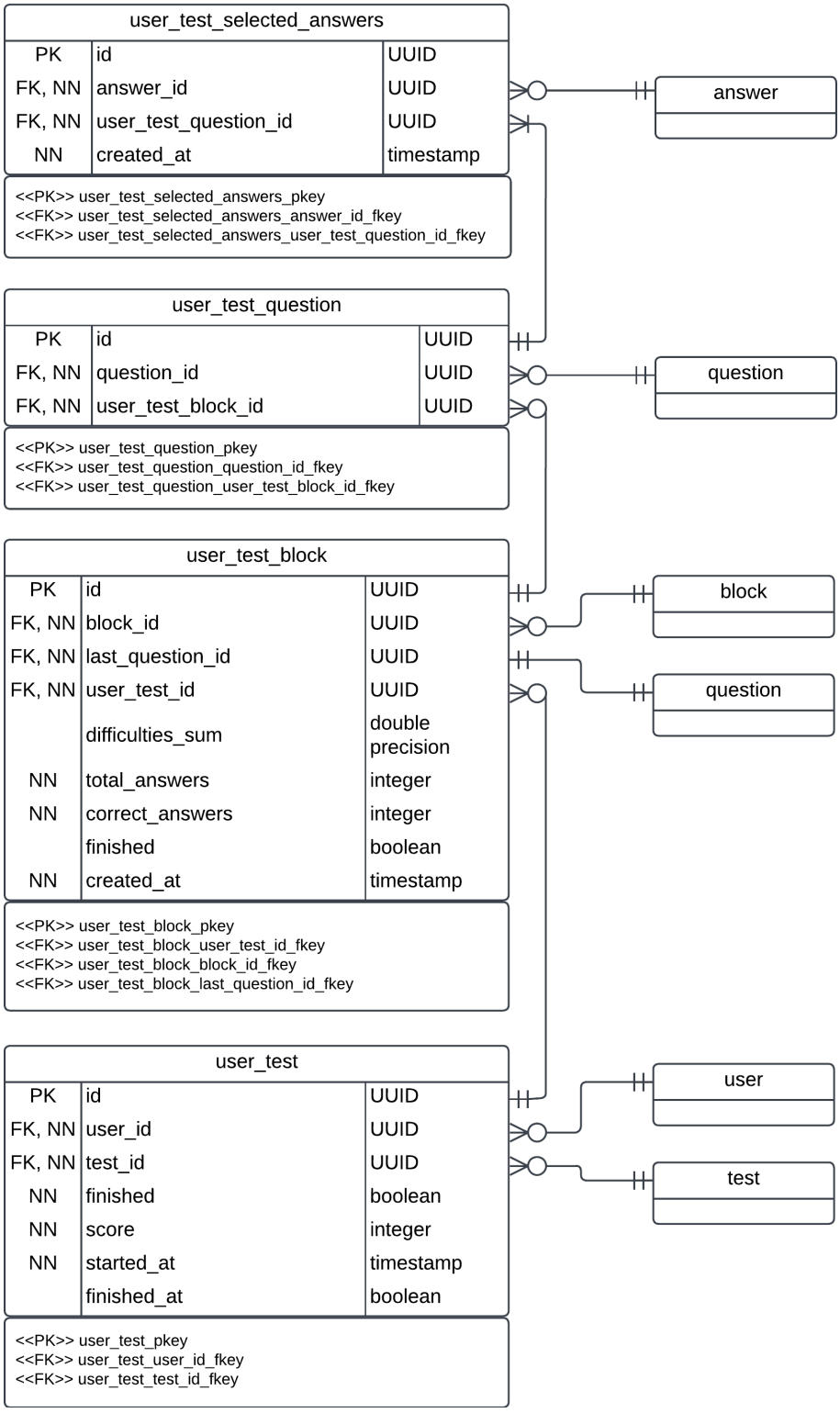
- arvutada punktisumma testi ploki kohta,
- valida järgmiseks sobiv küsimus,
- arvutada standardviga.

Punktide arvutamise põhimõtted, testimisprotsess ja järgmise sobiva küsimuse valik on põhjalikumalt käsitletud peatükis 6 "Adaptiivne testimine".

Kui plokk on lõpetatud, märgitakse tabelis *user_test_block* vastav kirje lõpetatuks.

Tabel *user_test* on üksus, mis salvestab andmed testi sooritava kasutaja ning testi enda kohta. Lisaks salvestatakse:

- testi koguskoor,
- testi algus- ja lõpuaeg.




Joonis 4. Kasutaja testi andmebaasiskeem.

# 6 Adaptiivne testimine

Pärast arenduse esimest iteratsiooni otsustati analüüsi tulemusel loobuda diagnostilisest testimisest, kuna praeguste teadmiste juures ei ole võimalik antud töö jaoks ettenähtud ajaraamidesse mahtuda. Käesolevas töös on realiseeritud adaptiivne testimine.

Aastal 1988 pakkus Wright välja lihtsustatud algoritmi adaptiivseks testimiseks, kasutades Raschi mudelit [14].

D - küsimuse raskusaste,

L - vastatud küsimuste arv,

H - esitatud küsimuste raskuste summa,

R - õigete vastuste arv.

```
1. Request next candidate. Set D=0, L=0, H=0, and R=0.
2. Find next item near difficulty, D.
3. Set D at the actual calibration of that item.
4. Administer that item.
5. Obtain a response.
6. Score that response.
7. Count the items taken: L = L + 1
8. Add the difficulties used: H = H + D
9. If response incorrect, update item difficulty: D = D - 2/L
10. If response correct, update item difficulty: D = D + 2/L
11. If response correct, count right answers: R = R + 1
12. If not ready to decide pass/fail, Go to step 2.
13. If ready to decide pass/fail, calculate wrong answers: W = L - R
14. Estimate measure: B = H/L + log(R/W)
15. Estimate standard error of the measure: S = sqrt([L/(R*W)])
16. Compare B with pass/fail standard T.
17. If (T - S) < B < (T + S), go to step 2.
18. If (B - S) > T, then pass.
19. If (B + S) < T, then fail.
20. Go to step 1.
```

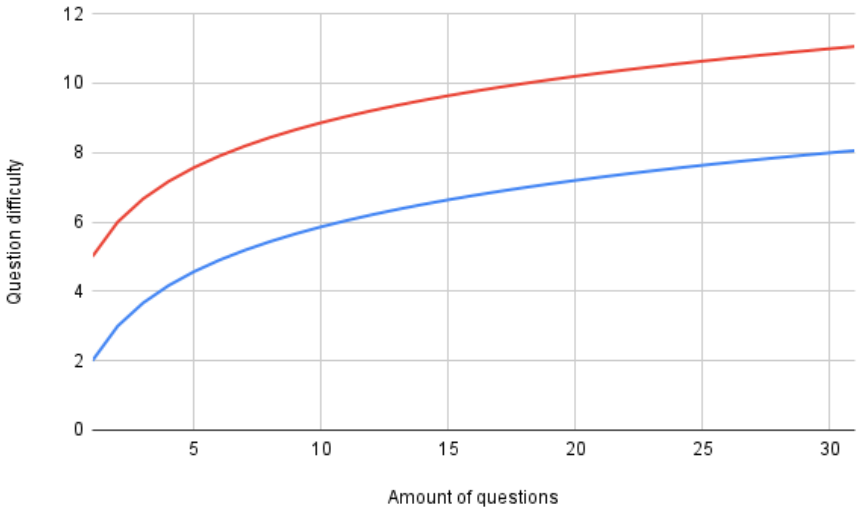
Antud algoritm sobib hästi testidele, kus küsimuste maksimaalne raskusaste ei ületa 10.

Kui kasutaja nõustub testi läbiviimise reeglitega, luuakse tabelisse *user_test* uus kirje, milles salvestatakse testi algusaeg. Seejärel valitakse juhuslikult küsimuste plokk nende plokkide seast, mis kuuluvad valitud testi. Esialgne raskusaste määratakse väärtusega, mis on $\frac{1}{3}$ maksimaalsest küsimuste raskusastmest selles plokis. Selline algväärtus võimaldab

kandidaadil saavutada positiivse kogemuse tänu küsimuste suhtelisele lihtsusele, mis vähendab testi sooritamise stressi [15]. Lisaks aitab see vähendada testi läbimise aega, kuna piisavate teadmiste korral alustatakse kohe keerukuse tasemelt $\frac{1}{3}$.

Tabelis 5 on toodud raskusastme kasvu kiirus. Oletame, et meil on test, kus küsimuste raskusaste on vahemikus $[0, 10]$, ja kaks identset kasutajat, kelle teadmised hõlmavad kõiki küsimusi raskusastmega kuni 6. - Esimene kasutaja alustab baasraskusastmega $0$, nagu pakkus välja Wright. - Teine kasutaja alustab raskusastmega $\left\lfloor \frac{10}{3} \right\rfloor = 3$.

Selline lähenemine võimaldab vähendada testi sooritamise aega ning tagab täpsema tulemuse kandidaatide jaoks, kelle teadmised vastavad algtasemele või kõrgemale.




Joonis 5. Küsimuse keerukuse kasvu maksimaalne kiirus.

Testibloki lõpetamise kriteeriumid

- Küsimused on otsas: Testiblokk lõpetatakse, kui kasutaja on vastanud kõigile konkreetse testi raames ette nähtud küsimustele.
- Kasutaja on saavutanud nõutud arvu küsimusi: Süsteem määrab minimaalse küsimuste arvu, mis on vajalik valideeritud tulemuse saamiseks (näiteks 10 küsimust).
- Piisav täpsus: Kui kasutaja saavutab määratud täpsuse taseme (näiteks 85%), võib testibloki ennetähtaegselt lõpetada.
- Sobiva raskusastmega küsimuste puudumine: Kohanduva testimise algoritm võib ammendada küsimused, mis vastavad kasutaja oskuste tasemele (näiteks liiga lihtsad

või liiga keerulised küsimused jäetakse välja).

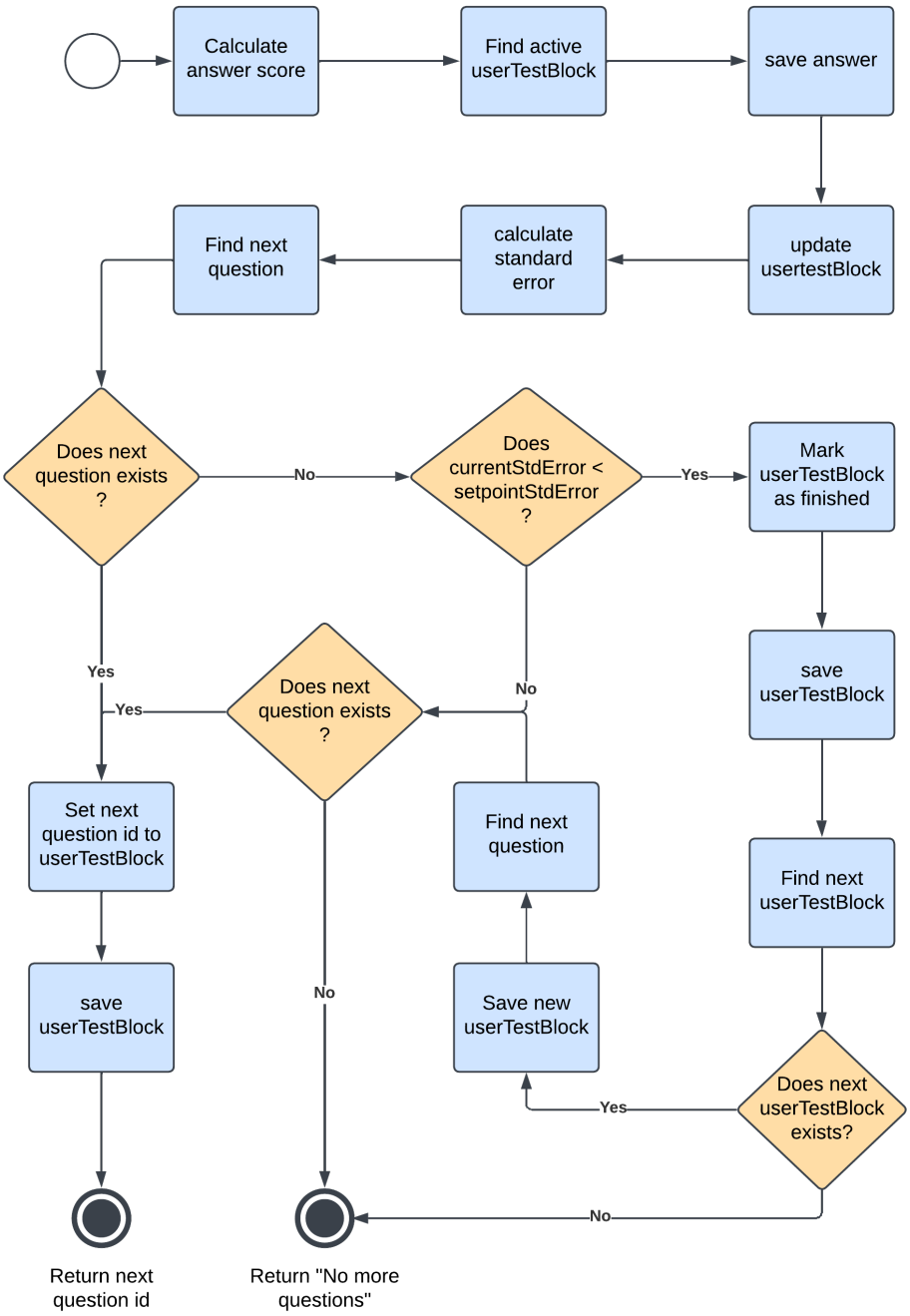
- Ajapiirangu täitumine: Kasutaja on ära kasutanud kogu konkreetsele testile eraldatud aja.

## Kasutaja vastuste salvestamine

Pärast seda, kui kasutaja valib enda arvates õiged vastused, saadab esirakendus need tagarakendusele.

Joonisel 6 on kujutatud kasutaja vastuste töötlemise protsessi. Algoritm hõlmab järgmisi samme:

1. Punktide arvutamine vastuse eest: Süsteem arvutab kasutaja vastuse tulemuse ja uuendab aktiivset testiblokki (*userTestBlock*).
2. Vastuse salvestamine: Vastus salvestatakse andmebaasi ning seejärel uuendatakse aktiivse testibloki andmeid.
3. Standardhälbe arvutamine: Süsteem arvutab praeguse standardhälbe (*currentStdError*), et kontrollida, kas on saavutatud nõutud täpsus.
4. Aktiivse testibloki lõpetamise kontroll:Süsteem otsib järgmist küsimust aktiivses testiblokis. Kui küsimust ei leidu või standardhälve on väiksem määratud väärtusest, märgitakse testiblokk lõpetatuks ning kontrollitakse uue testibloki loomise võimalust. Vastasel juhul salvestatakse järgmise küsimuse identifikaator aktiivsesse testiblokki.




Joonis 6. Kasutaja vastuse töötlemise protsess.

# 7 Valideerimine

Algsetes plaanides oli ette nähtud kaheetapiline valideerimine:

1. Testimine beetatestijatega — väike grupp (10–15 inimest), kes olid autori tuttavad. Peamiseks eesmärgiks oli suuremate puuduste avastamine, peamiste algoritmide töökindluse kontrollimine ning esialgse tagasiside kogumine.
2. Testimine tudengitega — mõõdukalt väike testijate hulk (umbes 50–100 inimest). Peamine eesmärk oli kogu süsteemi töökindluse testimine, süsteemi käitumise kontroll koormuse all ning tagasiside kogumine halduskeskkonna kasutajasõbralikkuse kohta.

Ajaplaneerimise puudulikkuse tõttu kulus vigade parandamisele rohkem ressursse, kui algselt eeldati. See tõi kaasa olulisi viivitusi arenduses ja testimises. Selle tulemusel polnud projekt tudengite testimiseks täielikult valmis ning kogu fookus oli suunatud beetatestimisele. Siiski pakkus läbi viidud testimine olulisi andmeid analüüsiks. 27.12.2024 seisuga oli testimissüsteemis registreeritud 23 kasutajat, sooritatud 27 testi ning 12 inimest täitis küsimustiku.

## 7.1 Testimise tulemused

Küsimustiku vastustest võib välja tuua järgmised andmed:

- 25% vastajatest märkis, et nad osalesid intervjuudes intervjueerijate rollis.
- 66% neist kinnitas, et see testimisviis aitab tõenäoliselt aega säästa.
- Lisaks märkisid 66%, et selliste testide rakendamine on õigustatud ja tõhus kandidaatide teadmiste hindamisel.

Need andmed näitavad, et kasutajad näevad testide praktilist väärtust, eriti tööintervjuude efektiivsuse tõstmise kontekstis. Samas on oluline tähele panna, et vaid 25% vastajatest eelistaks kasutada veebipõhist testi traditsioonilise intervjuu asemel. See fakt viitab

vajadusele selle tulemuse põhjuste täiendavale uurimisele ning võimalike lahenduste leidmisele.

## 7.2 Järeldused testi raskusastme kohta

Üle poole vastajatest (63%) märkisid, et test oli nende jaoks liiga lihtne ja ei peegeldanud nende tegelikke teadmisi (55%). See võib olla tingitud mitmest tegurist:

1. Piiratud küsimustepank: küsimuste arvu ja mitmekesisuse piiratus vähendab testi kohandatavust erinevatele kasutajate tasemetele.
2. Tasakaalustamata test: mõned küsimused võivad olla liiga lihtsad, mis viitab vajadusele küsimuste raskusastme jaotuse süsteemi parandamiseks.

Esimeseks sammuks nende probleemide lahendamisel võiks olla küsimustepanga suurendamine ning adaptiivse testimisalgoritmi rakendamine, mis valib küsimusi vastavalt kasutaja teadmiste tasemele. Täpsemate põhjuste väljaselgitamiseks plaanitakse läbi viia laiendatud küsimustikuanalüüs. See aitab täpsustada, kas vastajad olid kõrgema kvalifikatsiooniga kui testi sihttase või test ise oli liiga lihtne sihtgrupi jaoks.

#### Tagasiside vastajatelt

Vastajatelt küsiti ka, mis neile testimissüsteemis meeldis, mis ei meeldinud ning milliseid elemente saaks parandada.

**Positiivsed aspektid:**

- Lihtne ja arusaadav liides (3 mainimist).
- Selged ja arusaadavad küsimused (2 mainimist).

**Puudused:**

- Testi progressi puudumine (3 mainimist).
- Tulemuste puudumine testi lõpus (3 mainimist).
- Mõnede küsimuste kehv vormistus (2 mainimist).

**Parandamisettepanekud:**

- Lisada lõplikud tulemused (2 mainimist).
- Lisada progressiriba (4 mainimist).
- Lisada koodiplokid küsimustele, mis sisaldavad koodinäiteid (1 mainimine).

# 8 Tulemused

Selles peatükis antakse ülevaade rakenduse arenduse tulemustest, käsitletakse lahenduse väljatöötamise käigus tekkinud probleeme ning esitatakse ettepanekud süsteemi edasiseks täiustamiseks.

## 8.1 Arendus

Loodi minimaalselt elujõuline rakendus (*MVP*), mis vastab projekti eesmärkidele. Kuna tegemist oli prototüübiga ning autoril puudusid piisavad teadmised domeenist, tuli andmemudelit tihti ümber töötada. Need muudatused tõid mõnikord kaasa äriloogika korrigeerimise. Näiteks viidi läbi küsimuste raskusastme tasakaalustamine: esialgu hinnati raskusastet skaalal 1 kuni 100, kuid Raschi mudel ei toeta nii laia vahemikku. Arenduse varasematel etappidel leidis autor testimise käigus vigu testiloogikas, mis viis uute objektide *test_block* ja *user_test_block* lisamiseni andmemudelisse.

Protsesside haldamiseks kasutati paindlikku metoodikat (*agile*). Kuna projekt arendati ühe inimese poolt, valiti kahe metoodika hübriid:

- *Kanban* – arendusprotsessi organiseerimiseks ja ülesannete visualiseerimiseks;
- *Feature-Driven Development (FDD)* – ülesannete planeerimiseks, mis võimaldas keskenduda konkreetsetele funktsionaalsetele võimalustele.

Ülesanded valiti suuremahulisteks, et säästa aega. Nendes kirjeldati ainult üldist funktsionaalsust, mida tuli rakendada, ilma väiksemateks alamülesanneteks jaotamata. Seda põhjendas asjaolu, et funktsionaalsuse konkreetne realiseerimine selgus sageli alles koodi kirjutamise ja esmaste testide käigus.

Seati üles *CI/CD*, mis võimaldas automatiseerida rakenduse juurutamise ja testimise protsesse. Rakendus juurutati autori serverisse, mis võimaldab edasist täiustamist ja arendust.

Arenduse käigus kasutati aktiivselt OpenAI keelemudelit *ChatGPT*, mis aitas optimeerida koodi, lahendada keerukaid ülesandeid ning testida rakendust. Lisainfot selle kohta võib leida Lisa 3 – ChatGpt kasutamine.

## 8.2 Raskused

Üheks peamiseks raskuseks oli domeeniala (testimise ja teadmiste hindamise valdkonna) keerukuse alahindamine. See tõi kaasa olulise lahknevuse oodatavate ja tegelike tulemuste vahel, mis avaldas samuti negatiivset mõju motivatsioonile. Tulevikus soovitatakse selliste olukordade vältimiseks rohkem aega pühendada domeeniala analüüsile projektietapis.

## 8.3 Tagasivaade

**Mis oli hästi:**

- Minimaalselt elujõuline rakendus (*MVP*) loodi edukalt.
- Ei olnud piiranguid tehnoloogiate valikul projekti realiseerimiseks, mis võimaldas valida kaasaegsed tehnoloogiad.
- Regulaarsete juhendajaga kohtumiste abil lahendati õigeaegselt tekkinud küsimused.
- Edukalt beetatestimise läbinud.

**Mis oli mitte nii hästi:**

- CI/CD seadistamise vigade tõttu kaotati üks kord tootmiskeskkonna andmebaas, mis nõudis selle taastamiseks ja protsesside täiustamiseks aega.
- Rakenduse arendamine oleks olnud oluliselt produktiivsem meeskonnas, mitte üksinda, kuna töömahukus osutus väljakutsuvaks.

## 8.4 Edasised sammud

Edasised tegevused võib jagada kaheks: olemasoleva funktsionaalsuse parandamine ja uue funktsionaalsuse lisamine.

Parandused:

- Luua mugav ja funktsionaalne halduskeskkond, mis võimaldab süsteemi tõhusalt hallata, sealhulgas jälgida kasutajate aktiivsust ja analüüsida andmeid.
- Täiendada testimismooduleid *Spring Boot* platvormil, et parandada jõudlust ja kasutatavust.

Uus funktsionaalsus:

- Treenida närvivõrk avatud küsimuste hindamiseks. See võimaldab automatiseerida vastuste kontrollimist ja muuta süsteem paindlikumaks ja tõhusamaks.
- Lisada adaptiivne testimine, mis kohandab küsimusi kasutaja teadmiste taseme järgi. See parandab kasutajakogemust ja suurendab testimise tulemuste täpsust.

# 9 Kokkuvõte

Käesoleva bakalaureusetöö eesmärgiks oli luua prototüüp adaptiivse testimissüsteemi jaoks, mis aitaks kandidaatide teadmiste taseme alusel neid tööle võtmisel sorteerida.

Töö raames loodi prototüüp, mis koosneb serveripoolsest rakendusest, mis on kirjutatud *Spring Boot*-is, *esirakendus*-est, mis on arendatud *Vue 3* abil, ning andmebaasist *Postgres*. Autor uuris adaptiivse testimise domeeni, valis sobiva testimismudeli (Raschi mudel), seadis üles *CI/CD*-protsessi, juurutas testserveri ning viis läbi beetatestimise.

Beetatestimise tulemused näitasid, et süsteemil on potentsiaali kasutamiseks, kuid see vajab veel täiustamist. Testimises osales 23 inimest, kes andsid tagasisidet. Nende tagasiside põhjal avastati süsteemi töös mitmeid vigu (näiteks lõpphinde puudumine testi lõpus ja probleemid kujundusega) ning tehti ettepanekuid uue funktsionaalsuse lisamiseks (näiteks edusammude riba lisamine), sealhulgas küsimuste laiendatud valiku ja kasutajaliidese täiustamine.

Arenduse käigus tekkis sageli olukordi, kus lõppeesmärk ei olnud piisavalt selge, ning paljud aspektid said selgeks alles testimise domeeni sügavamal mõistmisel. See tõi kaasa andmemudeli regulaarse täiustamise ja koodibaasi refaktoreerimise. Sellised iteratiivsed muudatused tegid koodi puhtamaks, stabiilsemaks ja selgemaks.

Loodud rakendus ei ole veel täielikult funktsionaalne lahendus. Kuid töö käigus viidi ellu kõik püstitatud eesmärgid. Välja töötatud süsteem on tugev alus edasisteks täiustusteks ja arendusteks.

# Kasutatud kirjandus


[1] Google. *About Google Forms*. [Kasutatud: 09-12-2024]. URL: https://www.google.com/forms/about/.

[2] Moodle. [Kasutatud: 09-12-2024]. URL: https://moodle.org/.

[3] Vitaly Potenko Adam Franco. *Adaptive Quiz*. [Kasutatud: 09-12-2024]. URL: https://moodle.org/plugins/mod_adaptivequiz.

[4] Eksamite Infosüsteem (EIS). *Eksamite ja testide loomise ning lahendamise keskkond*. [Kasutatud: 08-12-2024]. URL: https://eis.ekk.edu.ee/eis/.

[5] Eksamite Infosüsteem. *Keskkonna tutvustus*. [Kasutatud: 08-12-2024]. URL: https://projektid.edu.ee/display/EKAV/Keskkonna+tutvustus.

[6] Haridus- ja Noorteamet. *E-ülesanded ja d-testid on abiks õpetajale ja toeks õpilasele*. [Kasutatud: 08-12-2024]. URL: https://harno.ee/uudised/e-ulesanded-ja-d-testid-abiks-opetajale-ja-toeks-opilasele.

[7] EIS. *Testid EIS keskkonnas*. [Kasutatud: 08-12-2024]. URL: https://eis.ekk.edu.ee/eis/lahendamine.

[8] iMocha. *iMocha - Skills Intelligence and Assessment Platform*. [Kasutatud: 08-12-2024]. URL: https://www.imocha.io/.

[9] Diagnostic Questions. *Diagnostic Questions - Free Assessment Platform*. [Kasutatud: 08-12-2024]. URL: https://diagnosticquestions.com.

[10] Figma. *What is Rapid Prototyping?* [Kasutatud: 07-12-2024]. URL: https://www.figma.com/resource-library/what-is-rapid-prototyping/.

[11] Wikipedia. *Minimum Viable Product*. [Kasutatud: 07-12-2024]. URL: https://en.wikipedia.org/wiki/Minimum_viable_product.

[12] RedHat. *What is CI/CD?* [Kasutatud: 05/12/2024]. URL: https://www.redhat.com/en/topics/devops/what-is-ci-cd.

[13] Eric Evans. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. [Kasutatud: 13-12-2024]. Addison-Wesley Professional, 2003. ISBN: 0321125215.

[14] John Michael Linacre. „Computer-Adaptive Testing: A Methodology Whose Time Has Come. MESA Memorandum No. 69.“ (2000). [Kasutatud: 17-12-2024], lk. 14. URL: https://www.rasch.org/memo69.pdf.

[15] Vitaly Potenko Adam Franco. *Adaptive Quiz: CAT (Computer-Adaptive Testing) implementation for Moodle*. [Kasutatud: 17-12-2024]. URL: https://moodle.org/plugins/mod_adaptivequiz.

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

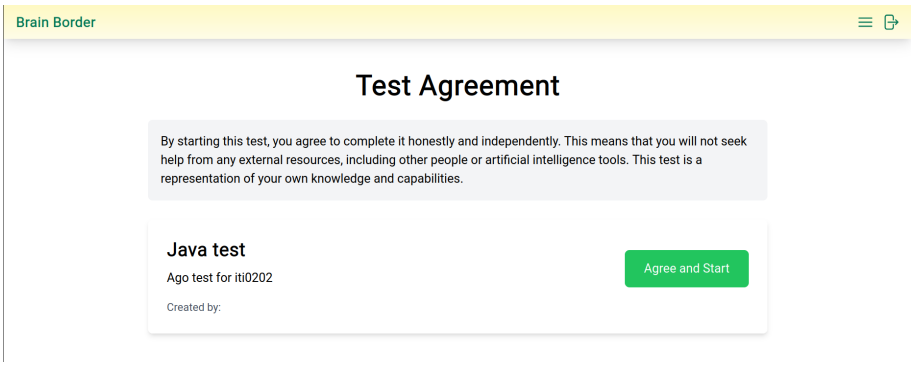
Mina, Aleksei Šohh

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Adaptiivse teadmiste hindamissüsteemi väljatöötamine IT-positsioonide kandidaatide esmaseks skaneerimiseks", mille juhendaja on Siim Rebane
    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.

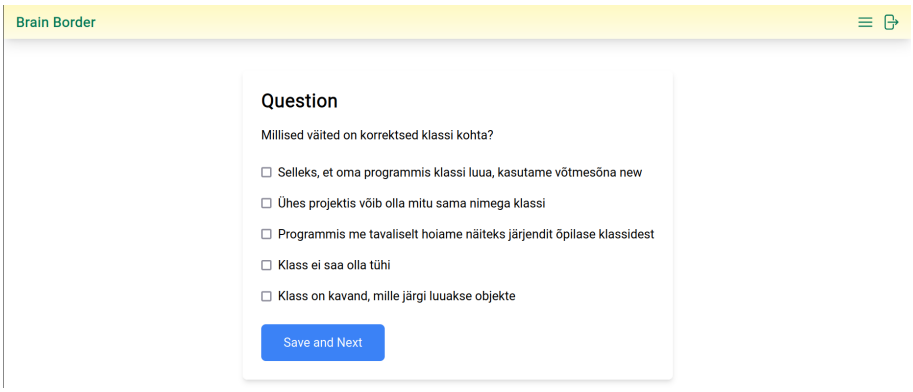
18.01.2025

[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 – Kasutajaliides




Testi alustamise vaade.




Küsimuse vaade.

# Lisa 3 – ChatGpt kasutamine

## LLM-i kasutamine käesolevas töös

LLM-id on juba saanud meie igapäevaelu lahutamatuks osaks, sealhulgas ka rakenduste arenduse valdkonnas. Need keelemudelid aitavad automatiseerida keerukaid protsesse ja kiirendada ülesannete täitmist, mis varem nõudsid märkimisväärset ajaressurssi. Käesolevas töös kasutas autor aktiivselt ühte OpenAI populaarseimat *LLM*-i — *ChatGPT*-d, mis oma paindlikkuse ja intellektuaalsete võimalustega lihtsustas oluliselt projekti loomise protsessi.

Kuna töö viidi läbi üksinda, oli *ChatGPT* sageli kaaslane, nõustaja, kriitik ning tõlkija. Lisaks konkreetsete ülesannete lahendamisele aitas *ChatGPT* parandada teksti, leida ebatäpsusi ning pakkuda alternatiivseid lahendusi, mis võimaldas autoril saavutada töö kõrgema kvaliteedi.

*ChatGPT*-d kasutati mitte ainult tekstide kirjutamiseks, vaid ka info analüüsimiseks, koodi kontrollimiseks ja keerukate ideede struktureerimiseks. See tegi temast asendamatu töövahendi kõigil töö etappidel.

Allpool on toodud näited vestlusaknasse esitatud päringutest (*promptidest*), mis illustreerivad selle mudeli abil lahendatud ülesannete mitmekesisust:

1. Teksti tõlkimine ja struktuuri täiustamine.

   ```
   Palun tõlkida eesti keelde, vältides kõnekeelt ja
   kasutades akadeemilist stiili:
   {tekst vene keeles}
   ```

2. Alternatiivsete lahenduste pakkumine teatud metoodilistele probleemidele.

   ```
   Antud andmebaasi disainis on andmebaas rakendusele, mis
   testib kandidaatide oskusi. Mida saaksime parandada ja miks?
   {andmebaasi DDL}
   ```

3. Peatükki valideerimine

   ```
   Vaata läbi antud peatükk, tee parandusettepanekuid ja esita
   kriitikat vastuvõtukomisjoni žürii liikme rollis.
   {peatükki tekst}
   ```