TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Daniel Vasser 205864IADB

Pokkeriturniiride jälgimise veebirakenduse prototüüp

Bakalaureusetöö

Juhendaja: Kristiina Hakk
PhD

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: Daniel Vasser

06.01.2025

Annotatsioon

Käesoleva bakalaureusetöö eesmärk on luua veebirakenduse prototüüp pokkeriturniiride jälgimiseks ja haldamiseks. Praegune probleem valdkonnas seisneb selles, et turniiride tulemuste ja ajaloo jälgimine on tihti keeruline ning info on jaotatud erinevate platvormide vahel.

Töö teoreetilises osas analüüsitakse olemasolevaid lahendusi, määratakse rakenduse nõuded ning teostatakse põhjalik tehnoloogiate analüüs. Praktiline osa keskendub veebirakenduse arendamisele, kus luuakse vajalik funktsionaalsus nii taga- kui ka eesrakenduses.

Bakalaureusetöö praktilise osa tulemusena valmib veebirakenduse prototüüp, mis võimaldab turniiride registreerimist, reaalajas jälgimist, statistika kogumist ning turniiride administreerimist. Rakendus järgib kaasaegseid veebiarenduse tavasid ning on loodud edasise arenduse võimalusi silmas pidades.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 38 leheküljel, 7 peatükki, 17 joonist, 4 tabelit.

Abstract

Poker Tournaments Tracking Web Application Prototype

The aim of this thesis is to create a web application prototype for tracking and managing poker tournaments. The current problems in the field are that tracking tournament results and history is often complicated, important data is missing and information is often separated between multiple applications.

The theoretical part of the thesis analyzes existing solutions, defines theoretical requirements and performs a thorough technology analysis. The practical part focuses on web application development, creating the necessary functionality in frontend and backend.

As a result of the practical part of the thesis, a web application prototype will be developed that allows registration for tournaments, real-time tracking, statistics collection and tournament administration. The application follows modern web application development practices and is created with future development in mind.

The thesis is in Estonian and contains 38 pages of text, 7 chapters, 17 figures, 4 tables.

Lühendite ja mõistete sõnastik

ACID	*Atomicity, Consistency, Isolation, Durability*, atomaarsus, konsistentsus, isoleeritus, püsivus
Ante	Pokkeris mängijate poolt panustatav kohustuslik summa, mis lisatakse panka enne kaartide jagamist, et soodustada aktiivset mängimist
API	*Application Programming Interface,* rakenduse programmeerimisliides
Big Blind	Pokkeris kohustuslik panus, mille teeb diilerist teisele positsioonile jääv mängija enne kaartide jagamist
BLL	*Business Logic Layer,* äriloogika kiht
Claims	Väited või õigused, mis kirljeldavad kasutaja identiteeti ja õigusi
Context Provider	Komponendipuu kaudu andmete edastamise mehhanism React-i raamistikus
CRUD	*Create, Read, Update, Delete,* andmebaasi põhioperatsioonid: loomine, lugemine, värskendus, kustutus
CSRF	*Cross-Site Request Forgery,* veebirünnak, mille eesmärk on kasutaja volituste kuritarvitamine, tehes volitamata toiminguid
Dependency Injection	Sõltuvuste süstimine - disainimuster, kus klassile vajalikud sõltuvused edastatakse väljastpoolt, selle asemel, et klass neid ise luua
DTO	*Data Transfer Object,* andmeedastuse objekt
GUID	*Globally Unique Identifier,* globaalselt unikaalne identifikaator
Hook	React-i funktsioon olekute ja elutsükli haldamiseks
HTML	*HyperText Markup Language,* veebilehtede struktuuri ja sisu kirjeldamiseks kasutatav märgenduskeel
HTTPS	*HyperText Transfer Protocol Secure*, protokoll, mis võimaldab turvalist andmevahetust veebiserveri ja veebibrauseri vahel
JSON	*JavaScript Object Notation,* andmete vahetusvorming
JSONB	*JavaScript Object Notation Binary,* PostgreSQL-i binaarne JSON-andmetüüp
JWT	*JSON Web Token*, JSON-põhine turvaline andmevahetustoken
Lazy loading	Laadimise edasi lükkamine vastavalt vajadusele

Rake	Pokkeris mängu korraldaja poolt turniiri sisseostust või rahamängu panustest võetav komisjonitasu
Recognition Rather than Recall	Kasutajaliidese disaini põhimõte, mille kohaselt tuleks vähendada kasutaja vajadust meenutada teavet, pakkudes konteksti ja visuaalseid vihjeid, mis toetavad äratundmist
Refresh Token	Värskendustoken, mida kasutatakse uue juurdepääsutokeni saamiseks
Small Blind	Pokkeris kohustuslik panus, mille teeb diilerist vasakule jääv mängija enne kaartide jagamist
SOLID	Viis objektorienteeritud programmeerimise põhiprintsiipi
URL	*Uniform Resource Locator*, ühtne ressursi asukoha määraja
UTC	*Coordinated Universal Time*, koordineeritud maailmaaeg

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Pokker on strateegiline kaardimäng, mille turniiride populaarsus on viimastel aastatel jõudsalt kasvanud, mida kinnitavad kõrged osalejate arvud suursündmustel, nagu World Series of Poker [1]. Selle kasvuga kaasneb vajadus efektiivse turniiride jälgimise süsteemi järele. Praegu on pokkeriturniiride tulemuste ja ajaloo jälgimine tihti keeruline ning info on jaotatud erinevate platvormide vahel. Mängijatel ja korraldajatel puudub ühtne lahendus, mis võimaldaks lihtsalt ja mugavalt jälgida mängijate tulemusi, tulevasi turniire ning reaalajas seise.

Käesoleva lõputöö eesmärk on arendada veebirakenduse prototüüp, mis võimaldab pokkeriturniiride efektiivset haldamist ja jälgimist. Prototüübi arendamise lähtetingimuseks on luua rakendus, mis töötab veebibrauseris, võimaldab reaalajas andmete kuvamist ning toetab nii arvuti- kui ka mobiilivaateid.

Eesmärgi saavutamiseks on kavandatud järgmised tegevused:

- olemasolevate lahenduste ja nende puuduste analüüs;
- kasutajate vajaduste ja süsteemi nõuete määratlemine;
- sobivate tehnoloogiate valik ja süsteemi struktuuri disainimine;
- veebirakenduse prototüübi arendamine;
- tulemuste hindamine ja edasiste arendusvõimaluste tuvastamine.

Töö koosneb kahest põhiosast. Teoreetilises osas analüüsitakse olemasolevaid lahendusi, pannakse paika rakenduse nõuded ning teostatakse tehnoloogiate analüüs. Praktiline osa keskendub veebirakenduse arendamisele, kus luuakse vajalik funktsionaalsus ja kasutajasõbralik liides.

2 Probleemipüstitus ja rakenduse eesmärk

Pokkeriturniiride korraldamine ja jälgimine on muutumas aina keerulisemaks protsessiks kõrgete osalejate arvude tõttu, mis nõuab efektiivset infohaldust ja reaalajalist andmevahetust. Käesolevas peatükis käsitletakse pokkeriturniiride jälgimise rakendusega seotud peamisi väljakutseid ning antakse ülevaade, kuidas arendatav veebirakenduse prototüüp neid lahendama hakkab.

2.1 Taust

Pokker on strateegiline kaardimäng, millel on erinevaid formaate. Texas Hold'em formaadis panustavad mängijad žetoone eesmärgiga võita vastastelt, moodustades parima viiekaardilise kombinatsiooni. Iga mängija saab kaks kaarti, mis on nähtavad ainult talle. Mängu jooksul lisandub lauale kuni viis ühiselt kasutatavat kaarti, millest mängijad saavad kombineerida oma kahe isikliku kaardi abil parima viiekaardilise kombinatsiooni. Mängu strateegiline element väljendub ka bluffimises, kus mängija võib nõrga käega võita, kui suudab vastase panustamisest loobuma sundida. [2]

Pokkeriturniiride populaarsus kasvab nii rahvusvahelisel kui ka Eesti tasandil. World Series of Poker põhiturniir püstitas 2024. aastal osalejate rekordi 10112 mängijaga, mis näitab ala jätkuvat kasvu [3]. Eesti kontekstis on populaarsust mõjutanud rohkelt festivale nagu Kings of Tallinn ja World Series of Poker Circuit Tallinn, mis toovad osalejaid ka välisriikidest.

2.2 Eesmärk

Arendatava veebirakenduse peamine eesmärk on luua tervikliku lahenduse prototüüp pokkeriturniiride jälgimiseks ja haldamiseks, mis lahendab praegused väljakutsed nii korraldajate, mängijate kui ka kaasaelajate jaoks.

Praegune olukord valdkonnas on problemaatiline, kuna paljudel korraldajatel puudub ühtne platvorm turniiride haldamiseks ja jälgimiseks. Tihti jäävad puudulikuks ka reaalajas turniirijälgimise elemendid, näiteks mängijate istekohad, turniirikell või

auhindade jaotus. Pauside ajal on see eriti oluline, kui mängijad vajavad täpset infot turniiri jätkumise kohta. Samuti paljudel lahendustel puudub mängijatel võimalus jälgida oma tulemuste statistikat ja analüüsida sooritust pikema aja vältel.

Eesmärk on luua rakendus, mis lahendab need probleemid, luues platvormi, kus kõik turniiriandmed on kättesaadavad ühest kohast. Reaalajas jälgimise süsteem annab nii mängijatele kui ka pealtvaatajatele selge ülevaate turniiri kulgemisest, sealhulgas auhindade jaotusest, pimepanuste tasemetest ja turniirikellast.

Rakenduse kasutuselevõtt tooks efektiivsuse tõusu turniiride korraldusse. Korraldajad saavad keskenduda turniiri kvaliteedi tõstmisele tänu efektiivsematele tööprotsessidele. Mängijate jaoks muutub nii registreerimine kui ka turniiri jälgimine lihtsamaks. Statistika ja ajalooliste andmete parem kättesaadavus võimaldab teha informeeritumaid otsuseid ning analüüsida oma sooritust. Pealtvaatajate parem ligipääs turniiriinfole ja reaalajas jälgimise võimalused suurendaksid pokkeriturniiride levikut.

2.3 Metoodika

Käesoleva lõputöö eesmärk on luua pokkeriturniiride jälgimise veebirakenduse prototüüp. Töö keskendub neljale põhilisele osale: olemasolevate lahenduste analüüs, süsteemi disain, tehnoloogiate analüüs ja süsteemi arendus.

Olemasolevate lahenduste analüüs põhineb võrdlusmeetodil, kus uuritakse kolme aktiivses kasutuses olevat platvormi. Iga lahenduse puhul analüüsitakse nende tugevusi ja nõrkusi, keskendudes reaalajas jälgimise funktsionaalsusele ja kasutajaliidese lahendustele. Analüüs loob aluse uue rakenduse nõuete määramiseks.

Süsteemi disain määrab arendatava rakenduse funktsionaalsed ja mitte funktsionaalsed nõuded, mis on tugipunkt lahenduse arendusel. Samuti luuakse vaadete näidised ja uuritakse kasutajaliidese parimaid tavasid.

Tehnoloogiate analüüsis võrreldakse erinevaid tehnoloogiaid rakenduse anbmebaasile, tagarakendusele ja eesrakendusele. Põhikriteeriumid on tehnoloogia sobivus pokkeriturniiride jälgimise veebirakenduse kontekstis, tehnoloogia populaarsus ja autori kogemus.

Süsteemi arendus annab ülevaate, kuidas lahendus luuakse. Kirjeldatakse disaini ja vajalikke funktsionaalsusi taga- ja eesrakenduses, toetudes määratud nõuetele.

Rakenduse arendamisel järgitakse iteratiivset lähenemist, kus iga komponendi arendamisel läbitakse analüüsi, rakendamist ja testimist. See võimaldab tagada iga komponendi kvaliteedi ning vastavuse nõuetele.

2.4 Olemasolevate lahenduste analüüs

Turu-uuringu käigus analüüsitakse erinevaid pokkeriturniiride jälgimise lahendusi. Analüüs lähtub kasutajate vajadustest ja rakenduste terviklikusest.

2.4.1 Olympic Poker Club

Olympic Poker Club on Eesti suurimate pokkerifestivalide ametlik süsteem. Rakendus töötab eelkõige Olympic kasiinoketi turniiridel ning seda kasutavad ka Kings of Tallinn ja World Series of Poker Circuit festivalid. Olympic süsteemi tugevuseks on põhjalik turniiride kalender ning lihtne registreerimisprotsess. Samas ilmnevad puudused turniiride reaalajas jälgimisel, kus kasutajad peavad liikuma eraldi Pokerlens-i platvormile, mis tekitab kasutajatele ebamugavusi ja ajakulu. Lisaks puudub turniiritaimer, mille tõttu mängijatel pole pauside ajal selget ülevaadet turniiri jätkumise kohta. [4], [5]

2.4.2 LetsPoker

LetsPoker on kahe Eesti pokkeriklubi (Chesterfield ja Bombay) poolt kasutatav lahendus. Rakendus järgib terviklikku lähenemist, kus kõik funktsionaalsused on ühes süsteemis. Suurimateks tugevusteks on selge ülevaade tulevastest ja möödunud turniiridest ning efektiivne turniirikella ja tasemete kuvamine. See lihtsustab korraldajate tööd ja parandab mängijate kogemust. Puudulikuks jääb laudade vaade, mis ei võimalda mängijatel näha täieliku ülevaadet teiste mängijate paigutusest laudades. See info oleks kasulik strateegiliste otsuste tegemiseks ning muudaks pealtvaatajate kogemuse kaasahaaravamaks. Samuti puudub ka mängijate isiklik turniiride ajalugu, mis on kasulik mängijale enda tulemuste ja arengu jälgimiseks. [6]

2.4.3 Triton Poker

Triton festivalid on tuntud kui ühed kõige prestiižemad pokkerifestivalid maailmas, omades suurimaid sisseostusid, jõudes tuhandetest dollaritest kuni miljonini välja. Veebirakendus on visuaalselt professionaalne ja selge struktuuriga. Eriti kasulik on funktsionaalsus, kus kasutaja saab valida, milliseid ja mitut lauda ta korraga jälgida soovib. Lisaks on mängijad enamasti märgitud portreepiltidega, mis on visuaalselt kaasahaarav. Samuti on efektiivselt lahendatud mängijate ajaloo kuvamine - vajutades profiili peale, on näha varasemaid tulemusi ning tulusust. Inimestele, kellel puudub põhjalikum teadmine pokkerist ja kes soovivad lihtsalt jälgida ning kaasa elada, võib rakendus, mis on visuaalselt kaasahaarav, osutuda liiga koormavaks rohke informatsiooni ja keeruka terminoloogia tõttu. Mängijate ja haldajate vaated pole avalikult ligipääsetavad, ehk pole võimalik neid funktsionaalsusi hinnata, kuid olemasolev osa pakub head eeskuju turniiride jälgimise rakenduse arendamiseks. [7]

2.4.4 Lahenduste analüüsi järeldus

Olemasolevate lahenduste positiivseid ja negatiivseid külgi on näha järgnevas tabelis (Tabel 1).

Tabel 1. Olemasolevate lahenduste analüüs.

Lahendus	**Plussid**	**Miinused**
Olympic Poker [4]	▪ turniirikalender põhjalik ▪ registreerimine lihtne	▪ funktsionaalsus jaotatud mitme rakenduse vahel ▪ mängutaimeri puudus
LetsPoker [6]	▪ kõik funktsionaalsused ühtses süsteemis ▪ põhjalik turniiride ülevaade ▪ reaalajas turniirikell ja tasemed	▪ mängijate ajalugu puudub ▪ vaade mängijate paigutusest laudades puudub
Triton Poker [7]	▪ mängijate ajalugu ▪ laudade vaade ▪ visuaalselt kaasahaarav	▪ mängu tundmatutele võib terminoloogia rohkus olla koormav

Analüüsitud lahendustest selgub, et efektiivne pokkeriturniiride jälgimise süsteem peab ühendama registreerimise lihtsuse, reaalajas jälgimise funktsionaalsuse ning põhjaliku statistika kuvamise võimalused. Erilist tähelepanu tuleb pöörata kasutajaliidese intuitiivsusele ning info kättesaadavusele ühe rakenduse piires.

3 Rakenduse nõuded ja disain

Käesolevas peatükis autor analüüsib ja määratleb rakenduse nõuded ja disainielemendid. Tuuakse ka arendatava rakenduse põhivaadete prototüübid.

3.1 Funktsionaalsed nõuded

Funktsionaalsed nõuded määravad kasutajale vajalike funktsioone, mida rakendus võimaldama peab. Alljärgnevalt on kirjeldatud pokkeriturniiride jälgimise veebirakenduse prototüübi peamised funktsionaalsed nõuded:

1. Kasutajate registreerimine – Kasutajad peavad saama rakendusse registreerida ja sisse logida.
2. Turniiride tabel – Kasutajale peab olema nähtav turniiride tabel, kus saab filtreerida nii tulevaste, jooksvate kui ka möödunud turniiride vahel. Turniiridel peab olema tabelis nähtav info sisseostu, toimumisaja, formaadi ja asukoha kohta.
3. Turniiridele registreerimine – Kasutaja peab olema võimeline rakenduse kaudu ennast turniiridele eelregistreerida.
4. Reaalajas turniiri tulemused – Turniiride ajal peab olema nähtav jooksev info struktuuri, turniirikella ja auhindade jaotusest.
5. Turniiride haldamine – Turniirihaldajad peavad olema võimelised lisama tulevasi turniire ja jooksvaid muudatusi sisestama rakenduses.

3.2 Mittefunktsionaalsed nõuded

Mittefunktsionaalsete nõuetega määrab autor süsteemi omadused, mis on seotud rakenduse toimivuse ja kasutajasõbralikkusega. Alljärgnevalt on kirjeldatud pokkeriturniiride jälgimise veebirakenduse mittefunktsionaalsed nõuded:

1. Kasutusmugavus – Rakendus peab olema lihtsa ja intuitiivse struktuuriga. Navigeerimine peab olema loogiline ja kasutajaliides ühtse kujundusega, et vältida segadust ja tagada mugav kasutajakogemus.
2. Jõudlus ja kiirus – Rakendus peab suutma reaalajas andmeid töödelda, tagades sujuva toimimise.

3. Turvalisus – Kuna rakendus käsitleb kasutajate kontosid ja turniiriandmeid, peab see tagama andmete konfidentsiaalsuse ja terviklikkuse.
4. Skaaleeritavus – Rakendus peab olema võimeline toetama kasutajate ja andmete mahu kasvu, eriti kui on suured turniirid toimumas, et vältida probleeme suure kasutuskoormuse korral.
5. Hooldatavus – Rakendus peab olema lihtsa ülesehitusega, et võimaldada kerget hooldust ja uute funktsionaalsuste lisamist tulevikus.

3.3 Kasutajaliidese disain

Kasutajaliidese disainimisel lähtutakse kaasaegsete veebirakenduste põhimõtetest ja Jakob Nielseni 10 heuristikast, mis on laialdaselt tunnustatud kasutajaliidese disaini põhiprintsiipidena [8]. Disaini loomisel arvestatakse nii mängijate, turniiri korraldajate kui ka turniiride jälgijate vajadustega, et pakkuda igale kasutajagrupile meeldivat kasutajakogemust.

3.3.1 Kasutajaliidese põhimõtted

Rakenduse kasutajaliides järgib järgmisi põhiprintsiipe:

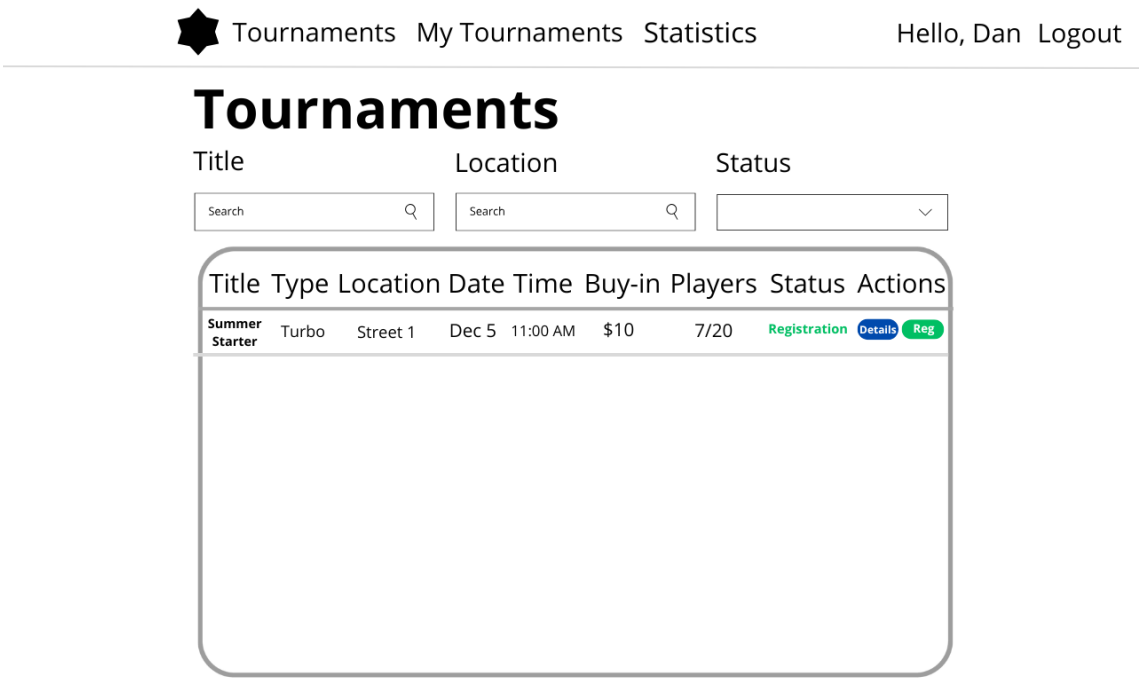
1. Süsteemi oleku nähtavus [8] - Turniiride hetkeseisu kuvatakse reaalajas, tagades kasutajatele pideva ülevaate toimuvast. See loob usaldust rakenduse vastu ja võimaldab kasutajatel teha informeeritud otsuseid.
2. Minimalistlik ja esteetiline disain [8] – Kasutajaliideses võetakse kasutusele minimalistlikku ja selget disainikeelt, kus iga element täidab konkreetset eesmärki.
3. Süsteemi ja reaalmaailma vastavus [8] - Kasutatakse terminoloogiat, mis on pokkerimängijatele tuttav.
4. Järjepidevus ja standardid [8] - Navigeerimine rakenduses lahendatakse peamenüü abil, mis on pidevalt nähtav lehe ülaosas. See võimaldab kasutajal kiiresti liikuda erinevate funktsionaalsuste vahel. Menüü grupeerib seotud funktsioonid loogilistesse kategooriatesse.

3.3.2 Põhilised vaated

Rakenduse põhilised vaated disainitakse vastavalt kasutaja rollile ja nõuetele, järgides hierarhilist navigatsioonimustrit, mis Microsoft UI Architecture Guidelines kohaselt

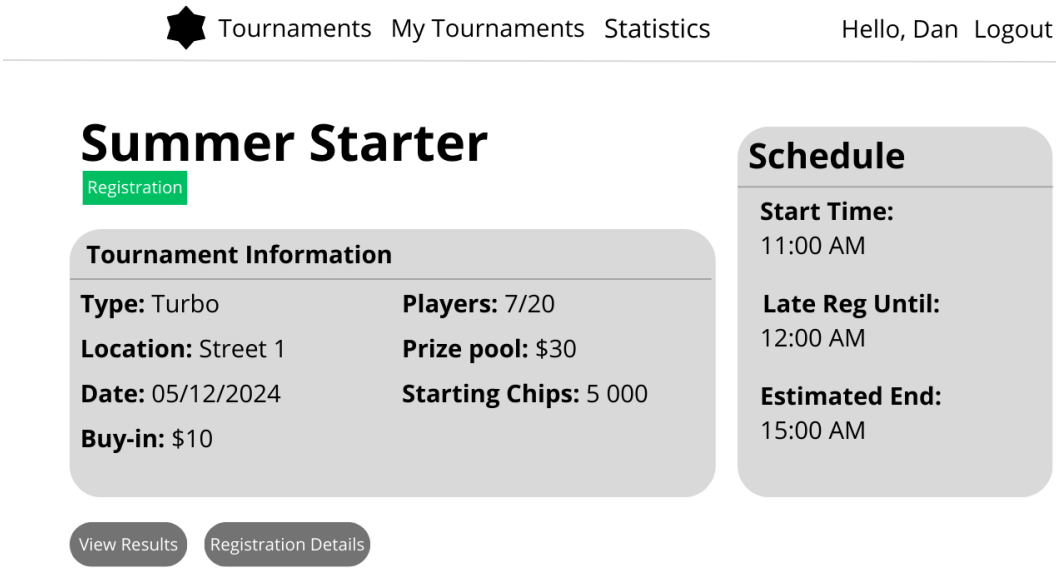
tagab kasutajatele selge ja etteaimatava liikumistee läbi rakenduse [9]. Luuakse vaadete prototüübid, kasutades Canva veebirakendust [10].

Avalehe vaade toimib rakenduse põhipunktina, andes ülevaate aktiivsetest, möödunud ja tulevastest turniiridest. See lähenemine järgib *Recognition Rather than Recall* heuristikat, kus kõik oluline info on koheselt nähtav, mitte peidetud menüüde taha [8]. Iga turniiri kohta kuvatakse põhiinformatsioon nagu nimi, formaat, asukoht, toimumisaeg, sisseost, mängijate arv ja staatus. Turniirid järjestatakse kronoloogiliselt, tuues esile lähiajal toimuvad üritused. (Joonis 1)



Joonis 1. Avalehe vaate disain.

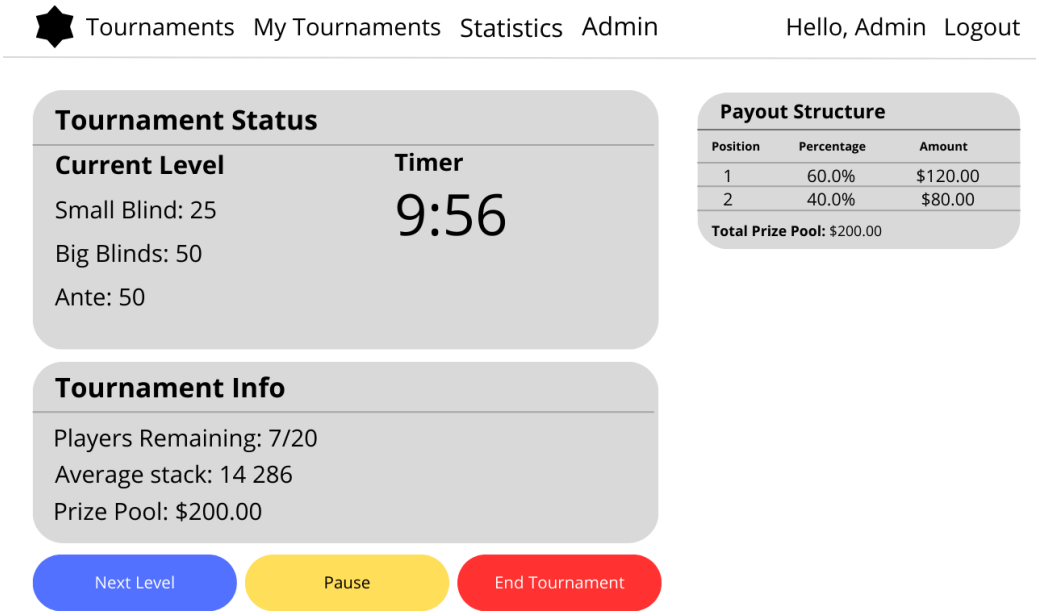
Turniiri detailvaates kuvatakse põhjalikku informatsiooni valitud turniiri kohta. Leht jaotatakse sektsioonideks, kus vasakul on turniiri üldinfo nagu pimepanuste struktuur, asukoht, toimumisaeg, registreerunud mängijate arv ja auhinnafond. Paremal on ajakavaga seotud info, nagu turniiri algus, registreerimisperioodi lõpp ja umbkaudne turniiri lõpp. (Joonis 2)



Joonis 2. Turniiri detailvaate disain.

Reaalajas jälgimise vaade (Joonis 3) disainitakse võimalikult lihtsa ja informatiivsena. Ekraan jagatakse kolmeks peamiseks osaks:

- Vasakul üleval kuvatakse turniirikell ja pimepanuste tasemed.
- Vasakul all asuvad alles olevate mängijate arv, keskmine žetoonide kogus mängijal ja auhinnafond.
- Paremal kuvatakse auhindade jaotust kohtade vahel.



Joonis 3. Turniiri reaalajas vaate disain.

3.3.3 Interaktiivsed elemendid

Kasutajaliidese interaktiivsed elemendid disainitakse intuitiivselt kasutatavaks. Nupud ja lingid on selgelt eristatavad ning annavad visuaalset tagasisidet kasutaja tegevustele. Vormielemendid nagu sisestusväljad ja valikmenüüd järgivad ühtset stiili, muutes rakenduse kasutamise loogiliseks.

Registreerimisprotsess muudetakse võimalikult lihtsaks, nõudes kasutajalt minimaalset sisestust. Vormid hakkavad andma kohest tagasisidet sisestatud andmete korrektsuse kohta.

4 Tehnoloogiate valik

Käesolevas peatükis analüüsitakse põhjalikult erinevaid tehnoloogiaid. Hindamisel lähtutakse kolmest peamisest kriteeriumist: tehnoloogia sobivus pokkeriturniiride jälgimise veebirakenduse kontekstis, populaarsus ja autori varasem kogemus antud tehnoloogiaga.

4.1 Tagarakenduse tehnoloogiad

Tagarakenduse tehnoloogiate valimisel on oluline leida raamistik, mis toetab efektiivselt pokkeriturniiride jälgimise rakenduse põhifunktsionaalsusi. Tagarakenduse jaoks analüüsitakse kolme raamistikku: ASP.NET Core, Node.js Express ja Spring Boot. Iga tehnoloogia puhul hinnatakse selle võimekust pakkuda vajalikke funktsionaalsusi, integratsioonivõimalusi ning arendusmugavust.

Oluline on tagarakenduse võimekus teostada reaalajas andmevahetust, pakkuda turvalist kasutajate autentimist ning toetada efektiivset andmebaasi integreerimist. Need on kõrge tähtsusega pokkeriturniiride jälgimise rakenduse kontekstis, kus on vaja tagada kiire ja usaldusväärne andmevahetus kõigi osalejate vahel.

4.1.1 ASP.NET Core

ASP.NET Core on Microsofti poolt arendatud avatud lähtekoodiga raamistik veebirakenduste loomiseks. Microsofti dokumentatsiooni kohaselt on raamistik üles ehitatud modulaarselt, võimaldades arendajatel valida täpselt need komponendid, mida nad vajavad [11].

Reaalajas funktsionaalsuse võimaldab ASP.NET Core SignalR tehnoloogiaga, mis on sisseehitatud lahendus serveri ja kliendi vaheliseks suhtluseks [12]. See võimaldab kahepoolset andmevahetust, mis on vajalik pokkeriturniiride jooksva seisu edastamiseks kõigile osalejatele.

Autentimise osas sisaldab raamistik Identity Framework-i, mis pakub terviklikku lahendust kasutajate registreerimiseks, autentimiseks ja autoriseerimiseks [13]. See võimaldab lihtsalt rakendada erinevad kasutajarollid nagu turniirikorraldajad, mängijad ja administraatorid.

Andmebaasi integratsiooniks pakub ASP.NET Core Entity Framework Core-i, mis võimaldab lihtsustatud andmebaasiga suhtlemist [14]. See võimaldab hallata keerukaid andmestruktuure, mis on vajalikud turniiriandmete jälgimiseks.

Stack Overflow 2024. aasta arendajate uuring näitab, et ASP.NET Core on professionaalsete arendajate seas üks enim kasutatavaid serveripoolseid raamistike, olles 19.1% vastanute valik [15]. See lai kasutajaskond tagab aktiivse kogukonna, regulaarsed uuendused ja rohked kolmanda osapoole teegid.

Autori kogemus ASP.NET Core-ga väljendub mitmes kooliprojektis. See tähendab, et arendusprotsess saab olla tõhus, kuna puudub vajadus tehnoloogia põhitõdede õppimiseks.

4.1.2 Node.js Express

Express on Node.js platvormil põhinev paindlik veebiraamistik. NPM dokumentatsiooni kohaselt järgib Express "minimaalse raamistiku" filosoofiat, võimaldades arendajatel ehitada rakendusi täpselt vastavalt oma vajadustele, lisamata ebavajalikku kompleksust [16].

Reaalajas funktsionaalsuseks vajab Express täiendavaid teeke nagu Socket.IO või WebSocket [17]. Need võimaldavad luua kahepoolse reaalajas suhtluse serveri ja kliendi vahel, mis on vajalik turniiride seisu jälgimiseks. Kuigi lahendused on töökindlad, nõuab nende integreerimine täiendavat konfigureerimist.

Autentimise funktsionaalsuse jaoks tuleb Expressile lisada eraldi teek, nagu Passport.js [18]. See pakub paindlikku autentimissüsteemi, kuid nõuab põhjalikku seadistamist ja täiendavat arendusaega, et rakendada turniirirakenduse jaoks vajalik rollipõhine ligipääsusüsteem.

Andmebaasi integratsiooniks vajab Express eraldi andmebaasi moodulit, kuna raamistikul puudub sisseehitatud andmebaasi suhtluse lahendus [19]. See tähendab, et autor peab valima sobiva raamistiku ja selle integreerimisele aega kulutama, et tagada efektiivne andmete haldamine.

Stack Overflow 2024. aasta statistika näitab, et Express on professionaalsete arendajate seas populaarne raamistik, olles 18.2% vastanute valik [15]. NPM Trends andmetel on

Express-il üle 20 miljoni igakuise allalaadimise, mis näitab raamistiku laialdast kasutust [20].

Autori kogemus Express-ga on puudulik, mis tähendaks märkimisväärset ajakulu keerulisemate funktsionaalsuste rakendamisel.

4.1.3 Spring Boot

Spring Boot on Java ökosüsteemil põhinev raamistik, mis on disainitud ettevõttetaseme rakenduste arendamiseks. Spring Boot ametliku dokumentatsiooni kohaselt on raamistiku eesmärk võimaldada iseseisvate, tootmisvalmis Spring-põhiste rakenduste loomine minimaalse konfiguratsiooniga [21].

Reaalajas funktsionaalsuse jaoks pakub Spring Boot sisseehitatud WebSocket tuge [22]. See võimaldab rakendada kahepoolse suhtluse serveri ja kliendi vahel, mis on vajalik turniiride reaalajas jälgimiseks.

Autentimise funktsionaalsus on tagatud Spring Security raamistikuga, mis pakub põhjalikku lahendust kasutajate autentimiseks ja autoriseerimiseks [23]. See võimaldab rakendada keerukaid ligipääsusüsteeme ja rollipõhist autoriseerimist.

Andmebaasi integratsiooniks pakub Spring Boot JPA ja Hibernate tuge, mis võimaldab efektiivset andmebaasiga suhtlust ja keerukate andmestruktuuride haldamist [24]. Lahendus toetab mitmekülgseid päringuvõimalusi ja andmete valideerimist.

Stack Overflow 2024. aasta statistika näitab, et Spring Boot on professionaalsete arendajate seas populaarne raamistik, olles 14.2% vastanute valik [15]. JetBrains-i 2023. aasta arendajate uuring näitab, et Spring Boot on juhtiv Java veebiarendajate seas, olles kasutusel 72% vastanutest [25].

Autori kogemus Java platvormiga on rahuldav, kuid Spring Boot-iga minimaalne. See on kerge risk projekti kontekstis, kuna raamistiku õppimine nõuaks täiendavat ajainvesteeringut.

4.1.4 Tagarakenduse tehnoloogia valik

Põhjaliku analüüsi tulemusena on koostatud võrdlustabel, mis võrdleb tehnoloogiate põhiomadusi vastavalt teostatud analüüsile (Tabel 2).

Tabel 2. Tagarakenduse tehnoloogiate võrdlus.

Kriteerium	**ASP.NET Core**	**Node.js Express**	**Spring Boot**
Reaalajas suhtluse tugi	SignalR (sisseehitatud) [12]	Socket.IO (eraldi teek) [17]	WebSocket (sisseehitatud, vajab põhjaliku seadistamist) [22]
Autentimise lahendus	Identity Framework (sisseehitatud) [13]	Passport.js (eraldi teek) [18]	Spring Security (sisseehitatud) [23]
Andmebaasi integratsioon	Entity Framework (sisseehitatud) [14]	Eraldi teek vajalik [19]	JPA/Hibernate (sisseehitatud) [24]
Populaarsus [15]	19.1%	18.2%	14.2%
Autori kogemus	Kõrge	Madal	Keskmine

Analüüsi tulemusena valitakse rakenduse arendamiseks ASP.NET Core. Valik põhineb järgmistel faktoritel:

- Terviklik lahendus sisseehitatud komponentidega, mis vähendab arendusaega ja võimalikke konflikte.
- Kõrge populaarsus tagab hea dokumentatsiooni ja kogukonna toe.
- Autori varasem kogemus ASP.NET Core-ga võimaldab keskenduda äriloogika arendamisele.

Teiste tehnoloogiate peamised puudused antud projekti kontekstis olid:

- Express nõuab mitme täiendava mooduli rakendamist põhifunktsionaalsuse saavutamiseks.
- Spring Boot eeldaks pikka õppimisaega.

Valitud tehnoloogia tagab hea tasakaalu funktsionaalsuse ja arenduskiiruse vahel, mis võimaldab luua kvaliteetse kasutajakogemuse pokkeriturniiride jälgimiseks mõeldud veebirakenduses.

4.2 Eesrakenduse tehnoloogiad

Eesrakenduse tehnoloogia valikul analüüsiti kolme populaarset raamistikku: Next.js, Angular ja Vue.js. Iga raamistiku puhul hinnati nende sobivust pokkeriturniiride jälgimise rakenduse kontekstis, pöörates erilist tähelepanu komponentide taaskasutamisele ja arenduse efektiivsusele. Lisaks arvestati raamistike kogukonna suurust ning autori varasemat kogemust.

4.2.1 Next.js

Next.js on React-põhine raamistik, mis on disainitud pakkuma optimeeritud arenduskogemust ja jõudlust. Next.js dokumentatsiooni kohaselt pakub raamistik sisseehitatud serveripõhist renderdamist, automaatset koodi jagamist ja tõhusat marsruutimissüsteemi [26].

Next.js pakub tugevat komponendipõhist arhitektuuri React-i baasil [27], mis sobib hästi turniiride jälgimise rakenduse modulaarseks ülesehituseks. Komponentide taaskasutamine ja jagamine erinevate vaadete vahel on lihtne ja loogiline.

TypeScript-i tugi on Next.js raamistikus sisseehitatud, võimaldades tüübiohutut arendust ilma täiendava konfiguratsioonita [28]. See on eriti oluline rakendus kasvamise puhul, kus tüübitugi aitab vältida potentsiaalseid vigu.

Stack Overflow 2024. aasta statistika näitab, et Next.js on populaarseim React-põhine raamistik, olles 18.6% professionaalsete arendajate valik veebirakenduste arenduses [15].

Autori kogemus Next.js-ga on keskmine, ulatudes ühe suuremahulise kooliprojektini, võimaldades keskenduda äriloogika arendamisele tehnoloogia rohke õppimise asemel.

4.2.2 Angular

Angular on Google-i poolt arendatud TypeScript-põhine raamistik veebirakenduste loomiseks. Angulari dokumentatsiooni kohaselt võimaldab see luua kiireid ja usaldusväärseid rakendusi, pakkudes laialdast tööriistade, API-de (Application Programming Interface) ja teekide komplekti arendusprotsessi lihtsustamiseks [29].

Raamistik pakub põhjalikku komponendipõhist arhitektuuri, kus komponentide, teenuste ja moodulite struktuur on rangelt määratletud [30]. See arhitektuuriline lähenemine sobib

hästi keerukate rakenduste arendamiseks, võimaldades selget koodi struktureerimist ja komponentide taaskasutust.

Angular kasutab vaikimisi TypeScript-i, pakkudes tugevat tüübisüsteemi ja kompileerimisaegset vigade tuvastamist [31]. See tagab rakenduse parema töökindluse ja lihtsustab koodi hooldamist.

Stack Overflow 2024. aasta statistika näitab, et Angular on populaarne valik, olles kasutusel 19.4% professionaalsetest arendajatest [15]. Google-i tugi tagab raamistiku pideva arengu ja stabiilsuse.

Autori kogemus Angulariga puudub täielikult, mis põhjustaks rohket ajakulu raamistiku õppimisele.

4.2.3 Vue.js

Vue.js on progressiivne JavaScript raamistik, mis on disainitud olema järk-järgult kasutuselevõetav. Vue dokumentatsiooni kohaselt on raamistik üles ehitatud paindlikult, võimaldades selle kasutamist alates lihtsast HTML-i (HyperText Markup Language) täiendamisest kuni täisfunktsionaalsete serveripoolse renderdamisega rakendusteni [32].

Vue.js pakub selget ja intuitiivset komponendipõhist arhitektuuri, kus komponendid koosnevad mallist, skriptist ja stiilidest [33]. Selline struktuur lihtsustab komponentide loomist ja haldamist, võimaldades modulaarset arendust.

TypeScript-i tugi on Vue.js-is olemas, kuid nõuab täiendavat seadistamist ja konfiguratsiooni [34]. See pole nii sujuv kui Angularis või Next.js-is kuna nõuab täiendavat tööd tüüpide defineerimisel ja komponentide tüübiturvalisuse tagamisel.

Stack Overflow 2024. aasta statistika näitab, et Vue.js on kasutusel 16.6% professionaalsetest arendajatest [15]. Kuigi see on vähem kui Next.js-il või Angular-il, on Vue.js kogukond aktiivne ja kasvav.

Autori kogemus Vue.js-ga on minimaalne, piirdudes ühe kooliainega, mis tähendaks täiendavat ajakulu raamistiku põhjalikumaks tundmaõppimiseks.

4.2.4 Eesrakenduse tehnoloogia valik

Analüüsi põhjal koostati võrdlustabel erinevate eesrakenduse tehnoloogiate põhiomaduste hindamiseks (Tabel 3).

Tabel 3. Eesrakenduse tehnoloogiate võrdlus.

Kriteerium	**Next.js**	**Angular**	**Vue.js**
Komponendipõhine struktuur	React-põhine, paindlik struktuur [27]	Rangelt struktureeritud, moodulipõhine [30]	Intuitiivne, lihtne struktuur [33]
Tüübitugi	Sisseehitatud TypeScript [28]	Vaikimisi TypeScript [31].	Täiendav seadistamine vajalik [34]
Populaarsus [15]	18.6%	19.4%	16.6%
Autori kogemus	Keskmine	Puudub	Madal

Läbiviidud analüüsi tulemusena valiti eesrakenduse arendamiseks Next.js. Valik põhineb mitmel olulisel faktoril:

- Komponendipõhine struktuur sobib hästi turniiride jälgimise rakenduse modulaarseks ülesehituseks.
- Sisseehitatud TypeScript tugi tagab tüübiohutu arenduse ilma täiendava konfiguratsioonita.
- Autori keskmine kogemustase võimaldab keskenduda äriloogika arendamisele.

Teiste tehnoloogiate peamised puudused projekti kontekstis:

- Angular nõuaks rohket õppimisaega autori puuduva kogemuse tõttu.
- Vue.js vajaks täiendavat seadistamist TypeScript-i rakendamiseks.

Valitud tehnoloogia tagab arenduskiiruse, funktsionaalsuse ja jätkusuutlikkuse, võimaldades luua kvaliteetse kasutajakogemuse pokkeriturniiride jälgimise veebirakenduse jaoks.

4.3 Andmebaasi tehnoloogiad

Andmebaasi tehnoloogiate analüüsil keskendutakse andmetüüpide ja päringute paindlikkusele ning võimekusele hallata suurt hulka samaaegseid ühendusi. Tähtis on ka populaarsus ja autori varasem kogemus vastava tehnoloogiaga.

4.3.1 PostgreSQL

PostgreSQL on avatud lähtekoodiga objekt-relatsiooniline andmebaasisüsteem, mis on tuntud oma töökindluse ja andmetervikluse poolest. PostgreSQL dokumentatsiooni kohaselt on süsteem olnud ACID (Atomicity, Consistency, Isolation, Durability) ühilduv alates 2001. aastast, tagades andmete usaldusväärsuse ka keerukate operatsioonide korral [35].

PostgreSQL pakub laiendatud andmetüüpide tuge, sealhulgas JSONB (JavaScript Object Notation Binary) andmetüüpi [36]. See pakub paindlikku andmestruktuuri turniiride erinevate formaatide jaoks ning on kasulik pokkeriturniiride süsteemis, kus turniiri struktuur ja reeglid võivad olla väga erinevad.

Paralleeltöötluse võimekus PostgreSQL-is on kõrgel tasemel, võimaldades efektiivselt hallata suurt hulka samaaegseid ühendusi [37]. See on oluline aktiivse turniiri ajal, kui mitmed kasutajad üheaegselt süsteemi kasutavad.

Stack Overflow 2024. aasta statistika näitab, et PostgreSQL on professionaalsete arendajate seas kõige populaarsem andmebaas, olles 51.9% vastanute valik [15]. See lai levik tagab hea dokumentatsiooni ja kogukonna toe.

Autori kogemus PostgreSQL-iga on kõrge, mis tuleneb kahest suuremahulisest kooliprojektist.

4.3.2 MySQL

MySQL on Oracle Corporation-i poolt hallatav avatud lähtekoodiga relatsiooniline andmebaasisüsteem. MySQL dokumentatsiooni kohaselt toetab süsteem suuri andmebaase ning on optimeeritud skaleeruvuseks, suutes hallata andmebaase, mis sisaldavad üle 200 000 tabeli ja 5 miljardi rida [38].

MySQL pakub standardset SQL andmetüüpide tuge, sealhulgas JSON-andmetüüpi (JavaScript Object Notation) [39]. Selle funktsionaalsus on piiratum võrreldes

PostgreSQL-i JSONB-ga, mis võib tekitada piiranguid turniiridele paindlike andmestruktuuride loomisel.

Paralleeltöötluse osas pakub MySQL head võimekust tänu mitmelõimelisele arhitektuurile, mis on saadaval MySQL Enterprise Edition-is [40]. Süsteem suudab efektiivselt hallata suurt hulka samaaegseid ühendusi, mis on oluline aktiivsete turniiride ajal.

Stack Overflow 2024. aasta statistika kohaselt on MySQL teine enim kasutatav andmebaas, olles kasutusel 39.4% professionaalsetest arendajatest [15].

Autori kogemus MySQL-iga on minimaalne, piirdudes läbitud andmebaaside ainega.

4.3.3 SQLite

SQLite on kompaktne, serverita relatsiooniline andmebaasisüsteem, mis töötab ühe failina. SQLite ametliku dokumentatsiooni kohaselt on süsteem disainitud olema iseseisvalt töötav, nullkonfiguratsiooniga SQL andmebaasimootor, mis ei vaja eraldi serverprotsessi [41].

SQLite pakub põhilisi SQL andmetüüpe ja JSON-andmetega töötamist läbi JSON1 laienduse [42]. Selle funktsionaalsus on piiratud teiste tehnoloogiatega võrreldes, mis toetavad binaarset JSON-andmetüüpi (nt PostgreSQL-i JSONB). See võib olla probleemiks pokkeriturniiride süsteemis, kus on vaja töödelda keerukaid andmestruktuure.

SQLite toetab mitut samaaegset lugemisoperatsiooni, kuid kirjutamisoperatsioonide ajal lukustatakse andmebaasifail, mis piirab paralleeltöötlust [43]. See võib tekitada jõudlusprobleeme olukorras, kus mitu kasutajat üritab samaaegselt andmeid uuendada.

Stack Overflow 2024. aasta statistika näitab, et SQLite on populaarne, olles kasutusel 32.1% professionaalsetest arendajatest [15].

Autori kogemus SQLite-iga on keskmine, olles seda kasutanud ühes kooliprojektis.

4.3.4 Andmebaasi tehnoloogia valik

Lähtudes analüüsist, koostati võrdlustabel andmebaasi tehnoloogiate hindamiseks pokkeriturniiride jälgimise rakenduse kontekstis (Tabel 4).

Tabel 4. Andmebaasi tehnoloogiate võrdlus.

Kriteerium	**PostgreSQL**	**MySQL**	**SQLite**
Andmetüüpide tugi	Laiendatud (JSONB) [36]	Põhiline JSON tugi [39]	Piiratud JSON tugi [42]
Paralleeltöötlus	Kõrge võimekus [37]	Hea võimekus [40]	Piiratud [43]
Populaarsus [15]	51.9%	39.4%	32.1%
Autori kogemus	Kõrge	Madal	Keskmine

Analüüsi tulemusena valiti rakenduse andmebaasiks PostgreSQL. Valik põhineb järgmistel faktoritel:

- JSONB andmetüübi tugi tagab paindlikkuse erinevate turniiriformaatide andmestruktuuride haldamisel.
- PostgreSQL-i suur populaarsus tagab hea dokumentatsiooni kättesaadavuse ja aktiivse kogukonna toe.
- Autori kogemus PostgreSQL-ga võimaldab keskenduda rakenduse äriloogikale.

Teiste tehnoloogiate peamised puudused projekti kontekstis:

- MySQL-i ja SQLite-i piiratud JSON andmetüübi tugi võrreldes PostgreSQL-i JSONB-ga raskendaks turniiride paindlike andmestruktuuride haldamist.
- SQLite omab piiranguid selle võimekuses hallata samaaegseid ühendusi, mis on aktiivse turniiri kontekstis vajalikud.

Valitud tehnoloogia tagab tasakaalu jõudluse, funktsionaalsuse ja töökindluse vahel, vastates pokkeriturniiride jälgimise süsteemi nõuetele.

5 Lahenduse arendus

Rakenduse arendus viiakse läbi kahes peamises osas, keskendudes eraldi taga- ja eesrakenduse arendamisele. Arenduse käigus järgitakse kaasaegseid tarkvaraarenduse praktikaid, keskendudes koodi modulaarsusele, taaskasutatavusele ning efektiivsele ressursikasutusele. Põhiprioriteediks on korraliku reaalajas andmevahetuse tagamine, mis on pokkeriturniiride jälgimise rakenduse puhul üks põhielemente.

5.1 Tagarakenduse arendus

Tagarakenduse arendamisel kasutatakse ASP.NET Core raamistiku. Rakenduse konteineriseerimiseks kasutatakse Docker-it, mis tagab järjepideva arendus- ja tootmiskeskkonna. Docker Compose abil ühendatakse kõik vajalikud komponendid (tagarakendus, eesrakendus ja andmebaas) ühtseks terviklikuks süsteemiks, lihtsustades rakenduse käivitamist ja testimist erinevates keskkondades.

Rakendus on üles ehitatud mitmekihiliselt, kus on selgelt eristatud vastutusvaldkonnad:

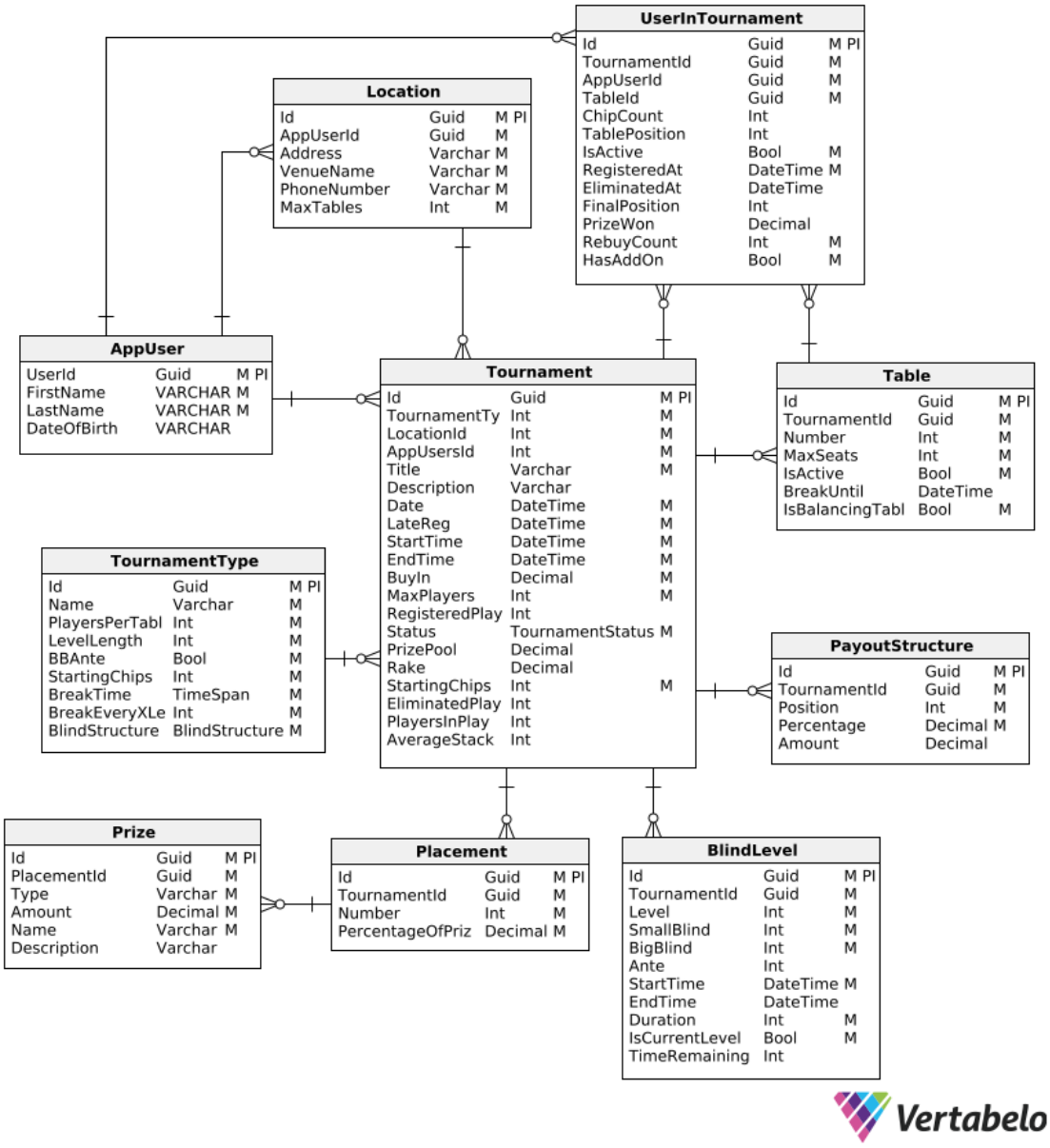
- Presentatsioonikiht - töötleb kasutajaliideselt tulevaid päringuid.
- Äriloogikakiht - koordineerib ärireeglite rakendamist ja suhtlemist andmekihiga.
- Andmekiht - tagab andmete terviklikkuse ja suhtleb otseselt andmebaasiga.

Dependency Injection vähendab erinevate kihtide vahelist tugevat seotust ja võimaldab paindlikkust. Uute funktsioonide lisamine on lihtne, piisab nende registreerimisest vastavas konteineris ja vastava kihi loogikas rakendamisest.

Arendus keskendub koodi puhtusele ja modulaarsusele, et tagada lihtsam testimine ja süsteemi skaleeritavus. Selge struktuur lihtsustab erinevate rakenduse osade koostööd ja süsteemi efektiivset arendamist.

5.1.1 Andmemudeli disain ja andmebaasi optimeerimine

Andmemudeli disain luuakse pokkeriturniiride haldamise ja jälgimise põhinõuetel, kus keskne üksus on turniiri tabel *Tournament*, mis ühendab endas kõiki turniiriga seotud andmeid. Mudel on üles ehitatud paindlikult, et toetada erinevaid turniiriformaate ja võimaldada efektiivset andmevahetust reaalajas (Joonis 4).



Joonis 4. Anbmebaasi olemi-suhte diagramm.

Diagramm esitab olulised tabelid, sealhulgas:

- *Tournament* - Haldab põhiandmeid, nagu turniiri pealkiri, toimumisaeg, osalejate piirangud ja auhinnafond.
- *UserInTournament* - Jälgib mängijate osalemist, säilitades žetoonide arvu, positsiooni ja lõpliku tulemuse. Sobib nii reaalajas jälgimiseks kui ka hilisemaks analüüsiks.
- *Table* - Haldab mängijate paigutust laudadesse, võimaldades turniirikorraldajatel jälgida laudu ja mängijate liikumist.
- *BlindLevel* - Määrab pimepanuste struktuuri ja kestvuse.

Andmebaasi optimeerimiseks on rakendatud järgmised meetmed:

- Indekseerimine - Kiirendab sagedasi päringuid, nagu turniiride otsimine kuupäeva järgi või mängijate seisundi leidmine.
- Andmete terviklikkus - Piirangud ja relatsioonid tagavad andmete korrektsuse (nt mängija saab olla korraga ainult ühes lauas).
- Ajatemplid - Andmed salvestatakse UTC (Coordinated Universal Time) formaadis, tagades korrektse ajakäsitluse rahvusvahelistel turniiridel.

Rakendus kasutab PostgreSQL andmebaasi. Selle konfiguratsioon on tehtud paindlikuks läbi keskkonna muutujate, võimaldades lihtsat seadistamist ja rakenduse skaleeritavust.

5.1.2 Äriloogika

Äriloogika kihis teostatakse kõik turniiride haldamiseks vajalikud toimingud. See kiht järgib mitmekihilist arhitektuuri ja SOLID disainiprintsiipe, kus iga teenus vastutab konkreetse äriprotsessi eest ning uute funktsionaalsuste lisamine ei nõua olemasoleva koodi muutmist. Selline lähenemine võimaldab pokkeriturniiride haldamisel tagada nii paindlikkuse erinevate turniiriformaatide toetamiseks kui ka süsteemi lihtsa laiendatavuse.

Äriloogika kekseks komponendiks on AppBll klass, mis koordineerib erinevate teenuste koostööd (Joonis 5).

```
public class AppBll: BaseBLL<AppDbContext>, IappBll
{
  private readonly IMapper _mapper;
  private readonly IAppUnitOfWork _uow;

  public AppBll(IAppUnitOfWork uow, IMapper mapper) : base(uow)
  {
    _mapper = mapper;
    _uow = uow;
  }

  // Lazy loading vähendab mälukasutust, luues teenuse instantsi
  // vastavalt vajadusele
  private ITournamentService? _tournaments;
  public ITournamentService Tournaments =>
  _tournaments ??= new TournamentService(_uow, _uow.Tournaments, _mapper);
}
```

Joonis 5. AppBll klassi koodinäide.

Pokkeriturniiride spetsiifilised ärireeglid on rakendatud vastavates teenustes. Turniiri alustamise funktsioon sisaldab nii staatuse muutmist kui ka auhinnafondi lukustamist (Joonis 6).

```
public async Task UpdateTournamentStart(Guid tournamentId, decimal prizePool)
{
  var tournament = await Repository.FirstOrDefaultAsync(tournamentId);
  if (tournament == null) throw new ArgumentException("Tournament not
  found");

  // Turniiri alustamisel lukustatakse auhinnafond ja registreerimine
  tournament.Status = Domain.TournamentStatus.Running;
  tournament.PrizePool = prizePool;
  tournament.EliminatedPlayers = 0;
  tournament.UpdatedAt = DateTime.UtcNow;

  Repository.Update(tournament);
  await UoW.SaveChangesAsync();
}
```

Joonis 6. Turniiri alustamise funktsiooni koodinäide.

Auhinnafondi arvutamine järgib pokkeriturniiride standardset arvutuskäiku, arvestades sisseostusummat, mängijate arvu ja vahendustasu (*rake*) (Joonis 7).

```
public async Task<decimal> CalculatePrizePool(Guid tournamentId)
{
  var tournament = await _bll.Tournaments.FirstOrDefaultAsync(tournamentId);
  if (tournament == null) return 0;

  // Auhinnafond arvutatakse mängijate sisseostu summast,
  // millest arvestatakse maha rake (vahendustasu)
  return tournament.BuyIn * tournament.RegisteredPlayers * (1 –
  (tournament.Rake ?? 0));
}
```

Joonis 7. Auhinnafondi arvutamise funktsiooni koodinäide.

Äriloogika kiht tagab kolm põhilist funktsionaalsust pokkeriturniiride haldamisel:

1. Turniiri elutsükkel

   - Turniiri alustamine koos auhinnafondide arvutamisega.
   - Pimepanuste tasemete automaatne genereerimine vastavalt turniiri formaadile.
   - Mängijate registreerimise ja väljalangemise jälgimine koos koha määramise automatiseerimisega.

2. Andmete terviklikkus

    - Turniirireeglite valideerimine (nt maksimaalne mängijate arv või sobiv pimepanuste struktuur).
    - UTC ajavööndi kasutamine rahvusvaheliste turniiride toetamiseks.
    - Mängijate paigutuse kontroll laudades.

3. Andmete teisendamine

    - DTO-de (Data Transfer Object) kasutamine andmevahetuseks erinevate kihtide vahel.

Teenused pärivad BaseEntityService klassist, mis tagab standardse CRUD (Create, Read, Update, Delete) funktsionaalsuse, võimaldades keskenduda pokkeriturniiridele spetsiifilise äriloogika lisamisele. Äriloogika kiht suhtleb tihedalt andmekihiga ja presentatsioonikihiga (läbi DTO-de), tagades efektiivse andmevahetuse ja ärireeglite järgimise kogu rakenduses.

Selline struktureeritud lähenemine äriloogikale võimaldab rakenduse lihtsa laiendamise tulevikus, näiteks uute turniiriformaatide lisamisega, ilma olemasolevat koodi ulatuslikult muutmata.

5.1.3 Turvalahendused

Rakenduse turvalisus on tagatud kasutajate autentimise, autoriseerimise ja andmete turvalise edastamisega. Turvalahenduste aluseks on ASP.NET Core Identity raamistik koos JWT (JSON Web Token) põhise autentimisega. JWT-autentimine seadistatakse rakenduse käivitamisel, kus määratakse tokeni valideerimise parameetrid ja turvapoliitikad (Joonis 8).

```
services.AddAuthentication(options =>
{
  options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
  options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
})
.AddJwtBearer(options =>
{
  options.TokenValidationParameters = new TokenValidationParameters()
  {
    ValidIssuer = builder.Configuration.GetValue<string>("JWT:issuer"),
    ValidAudience = builder.Configuration.GetValue<string>("JWT:audience"),
    IssuerSigningKey = new SymmetricSecurityKey(
      Encoding.UTF8.GetBytes(
        builder.Configuration.GetValue<string>("JWT:key")
      )
    ),
    ClockSkew = TimeSpan.Zero,
  };
});
```

Joonis 8. JWT autentimise konfigureerimine.

Autentimine toimub kahes osas: JWT tokenid igapäevaseks autentimiseks ning *refresh* tokenid pikemaajaliseks sessiooni haldamiseks. JWT tokenid genereeritakse kasutaja sisselogimisel ning need sisaldavad kasutaja põhiandmeid ja õigusi. Rakendus kasutab Identity raamistikku kasutajate ja rollide haldamiseks. Kui kasutaja registreerib, siis luuakse uus AppUser konto, mis seotakse vastava rolliga (User või Admin). See tagab ligipääsu kontrolli rakenduse eri osadele.

Refresh tokenid võimaldavad kasutajatel säilitada oma sessiooni ilma pideva uuesti sisselogimiseta. Need on pikema eluajaga kui JWT tokenid ning neid hoitakse andmebaasis, võimaldades vajadusel tühistamist. Väljalogimisel kustutatakse refresh token, mis tagab sessiooni turvalise lõpetamise.

Reaalajas andmevahetus toimub kehtiva JWT tokeni alusel. Enne ühenduse loomist valideeritakse kasutaja õigused, et lubada juurdepääs ainult autoriseeritud kasutajatele. Tänu sellele on reaalajas uuendused kättesaadavad ainult neile, kellel on vastav luba.

Rakendus kasutab CORS (Cross-Origin Resource Sharing) poliitikat, mis piirab ligipääsu ainult lubatud domeenidele. See on eriti oluline, kuna rakenduse esi- ja tagarakendus töötavad erinevatel portidel. CORS on seadistatud lubama ainult vajalikke päringutüüpe ja päiseid, et vähendada turvaohtusid.

Turbeseadistused on hallatavad läbi konfiguratsiooni, võimaldades lihtsat kohandamist erinevate keskkondade jaoks. JWT seadistused, nagu token-i kehtivusaeg ja krüpteerimisvõti on defineeritud keskkonna seadistustes, mis võimaldab nende turvalist muutmist koodi muutmata.

Parooliturvalisuse tagavad Identity raamistiku sisseehitatud funktsioonid, mis nõuavad piisavalt tugevaid paroole ja hoiavad neid turvaliselt krüpteerituna. Samuti rakendatakse kaitset OWASP 2021 Top 10 turvaohtude vastu, nagu CSRF (Cross-Site Request Forgery), token-põhise valideerimise abil [44].

Need meetmed tagavad, et rakenduse turvalisus vastab kaasaegsetele nõuetele ning kaitseb kasutajaid ja andmeid rünnakute eest.

5.1.4 Reaalajas andmevahetus

Rakenduse reaalajas andmevahetus on rakendatud kasutades SignalR tehnoloogiat, mis võimaldab kahepoolset suhtlust serveri ja kliendi vahel. See on eriti oluline pokkeriturniiride kontekstis, kus mängijad ja pealtvaatajad vajavad kiireid uuendusi reaalajas jooksva turniiri seisu kohta.

Serveri poolel on loodud TournamentHub, mis vastutab ühenduse ja sõnumite edastamise eest (Joonis 9).

```
[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]
public class TournamentHub : Hub, ITournamentHub
{
  public async Task JoinTournamentGroup(string tournamentId)
  {
    var groupName = $"tournament_{tournamentId}";
    await Groups.AddToGroupAsync(Context.ConnectionId, groupName);
  }

  [Authorize(Roles = "Admin")]
  public async Task StartTournament(string tournamentId)
  {
    await Clients.Group($"tournament_{tournamentId}")
    .SendAsync("TournamentStatusUpdated", tournament);
  }
}
```

Joonis 9. TournamentHub klassi koodinäide.

Kasutatakse grupipõhist lähenemist, kus iga turniiri jaoks luuakse eraldi grupp. See tagab, et uuendused saadetakse ainult nendele kasutajatele, kes antud turniiri jälgivad.

TournamentHub võimaldab nii tavakasutajatel kui ka administraatoritel liituda turniirigruppidega, kuid teatud toimingud (nt turniiri alustamine) on piiratud ainult administraatoritele.

Turniiride haldamiseks on rakendatud meetodid erinevate sündmuste edastamiseks (Joonis 10).

```
public async Task NotifyBlindLevelChange(Guid tournamentId, BlindLevelDto
blindLevel)
{
  await hubContext.Clients.Group($"tournament{tournamentId}")
  .SendAsync("BlindLevelUpdated", blindLevel);
}

public async Task NotifyPlayerElimination(Guid tournamentId,
UserInTournamentDto player)
{
  await hubContext.Clients.Group($"tournament{tournamentId}")
  .SendAsync("PlayerEliminated", player);
}
```

Joonis 10. Sündmuste edastamise koodinäited.

Need meetodid võimaldavad teavitada kõiki turniiri jälgivaid kasutajaid olulistest muutustest, nagu pimepanuste tasemete muutused või mängijate elimineerimised.

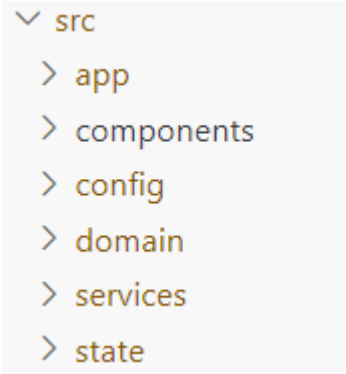
Andmete värskendamine toimub automaatselt läbi SignalR teadete. Lisaks kasutatakse regulaarseid päringuid varundusmeetodina, et tagada süsteemi töökindlus ka ühendustõrgete korral. Reaalajas andmevahetuse süsteem tagab skaleeruvuse ja võimaldab mitme kasutaja korraga ühendumist.

5.2 Eesrakenduse arendus

Eesrakenduse arendamisel kasutatakse Next.js raamistikku, järgides kaasaegseid veebiarenduse tavasid. Arendus keskendub kasutajakogemuse täiendamisele, sujuvale navigeerimisele ja kiirele andmete töötlusele. Suurt rõhku pannakse komponentide taaskasutatavusele ja koodi kvaliteedile, et tagada süsteemi laiendatavus ja hooldatavus.

5.2.1 Eesrakenduse struktuur

Next.js-i app-kataloogi lähenemine võimaldab luua loogilise ja kergelt muudetava failistruktuuri, kus iga komponent ja funktsioon paikneb selgelt määratud kohas (Joonis 11).



Joonis 11. Src kataloogi sisu.

Struktuur toetab kasutajate ja administraatorite vaateid, olles jaotatud järgmistesse üksustesse:

- App - Peamine kaust, mis sisaldab Next.js-i marsruute ja nende lehtede loogikat. Näiteks:
  - /tournaments/details/[id] vastab turniiri detailvaatele.
  - /admin/locations/create/page.tsx on administraatorile asukohtade haldamise leht.
- Components - Taaskasutatavad kasutajaliidese elemendid, nagu Header, Footer ja lehe paigutused.
- Config - API konfiguratsioonifailid.
- Domain - Andmemudelid, mis kirjeldavad süsteemi põhifunktsionaalsusi. Näiteks:
  - ITournament – Turniiri kirjeldav andmemudel.
  - ITable – Lauda kirjeldav andmemudel.
- Services - API päringute ja äriloogika teenused. Näiteks:
  - TournamentService – Töötleb turniiridega seotud API päringuid.
  - StatisticsService – Töötleb turniiride ja mängijate statistikat.
- State - Rakenduse globaalse oleku haldamine. AppContext võimaldab jagada erinevate rakenduse osade vahel andmeid, nagu autentimisinfo ja kasutajaõigused, vähendades nii vajadust korduvate päringute järele.

Marsruutimine järgib Next.js-i konventsioone, kus iga URL (Uniform Resource Locator) vastab rakenduse kataloogistruktuurile, mis vähendab käsitsi konfiguratsiooni vajadust.

Näiteks:

- Turniiri detailvaade - tournaments/details/[id], kus [id] tähistab dünaamilist identifikaatorit.
- Turniiri registreerimine – tournaments/details/[id]/registration.

Selline struktuur tagab modulaarsuse ja paindlikkuse, võimaldades autoril lihtsasti lisada uusi funktsionaalsusi ja muuta olemasolevaid komponente, et see ei mõjutaks teisi rakenduse osi.

5.2.2 Andme- ja olekuhaldus

Rakenduse andmehaldus põhineb kahel põhimõttel: globaalne olek kasutajainfo ja põhiseisundi jaoks ning lokaalne olek komponentide andmete jaoks. See tagab tugeva andmete kättesaadavuse ja jõudluse.

Globaalne olek on rakendatud React Context API abil (Joonis 12).

```
export interface IUserContext {
  userInfo: IUserInfo | null;
  setUserInfo: (userInfo: IUserInfo | null) => void;
  userRole: string | null;
  setUserRole: (role: string | null) => void;
}
export const AppContext = createContext<IUserContext | null>(null);

export default function AppState({ children }: { children: React.ReactNode })
{
  const [userInfo, setUserInfo] = useState<IUserInfo | null>(null);
  const [userRole, setUserRole] = useState<string | null>(null);

  useEffect(() => {
    const storedUserInfo = localStorage.getItem("userInfo");
    if (storedUserInfo) {
      setUserInfo(JSON.parse(storedUserInfo));
    }
  }, []);

  return (
    <AppContext.Provider value={{ userInfo, setUserInfo, userRole,
    setUserRole }}>
      {children}
    </AppContext.Provider>
  );
}
```

Joonis 12. Globaalse oleku rakendamise koodinäide.

Reaalajas andmevahetus tagarakendusega toimub SignalR kaudu, võimaldades kahepoolset suhtlust (Joonis 13).

```
export class TournamentHubService {
  private static connection: signalR.HubConnection | null = null;

  static async initialize(jwt: string) {
    const baseUrl = "http://localhost:5001";
    this.connection = new signalR.HubConnectionBuilder()
      .withUrl(`${baseUrl}/api/v1/tournamentHub`, {
        accessTokenFactory: () => jwt,
      })
      .withAutomaticReconnect()
      .build();
    await this.connection.start();
  }

  static async joinTournament(tournamentId: string)
  {
    if (!this.connection) throw new Error('Connection not initialized');
    await this.connection.invoke('JoinTournamentGroup', tournamentId);
  }

  static onTournamentUpdate(callback: (tournament: ITournament) => void)
  {
    if (!this.connection) return;
    this.connection.on('TournamentStatusUpdated', callback);
  }
}
```

Joonis 13. SignalR-i eesrakenduses kasutuse koodinäide.

Andmevahetus tagarakendusega toimub läbi spetsiifilise teenuste kihi, kus iga teenus pärib baaskonfiguratsiooni BaseApiService klassist (Joonis 14).

```
export abstract class BaseApiService {
  protected constructor(endpoint: string) {
    BaseApiService.httpClient = axios.create({
      baseURL: `http://localhost:5001/api/v1/${endpoint}`,
      headers: { "Content-Type": "application/json" }
    });
  }

  protected static handleResponse<T>(response: AxiosResponse<T>) {
    return response.status < 300 ? { data: response.data } : { errors:
    [response.statusText] };
  }
}
```

Joonis 14. BaseApiService klassi koodinäide.

See lähenemine võimaldab:

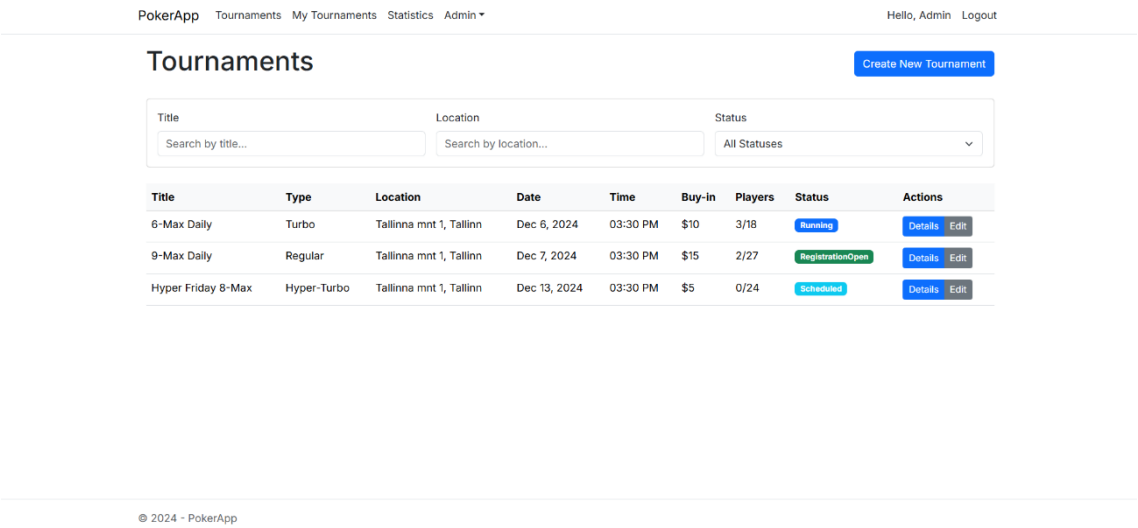
- Ühtse veatöötluse ja vastuste formaadi kõigile API päringutele.
- Turvalist autentimist JWT tokenite abil.
- Efektiivset andmete sünkroniseerimist serveri ja kliendi vahel.
- Sujuvat kasutajakogemust tänu reaalajas uuendustele.

Üksikud komponendid kasutavad lokaalset olekut useState *hook*-i abil, säilitades ainult enda tööks vajalikke andmeid. See vähendab globaalset olekukoormust ja parandab rakenduse jõudlust.

5.2.3 Kasutajaliidese komponendid ja vaated

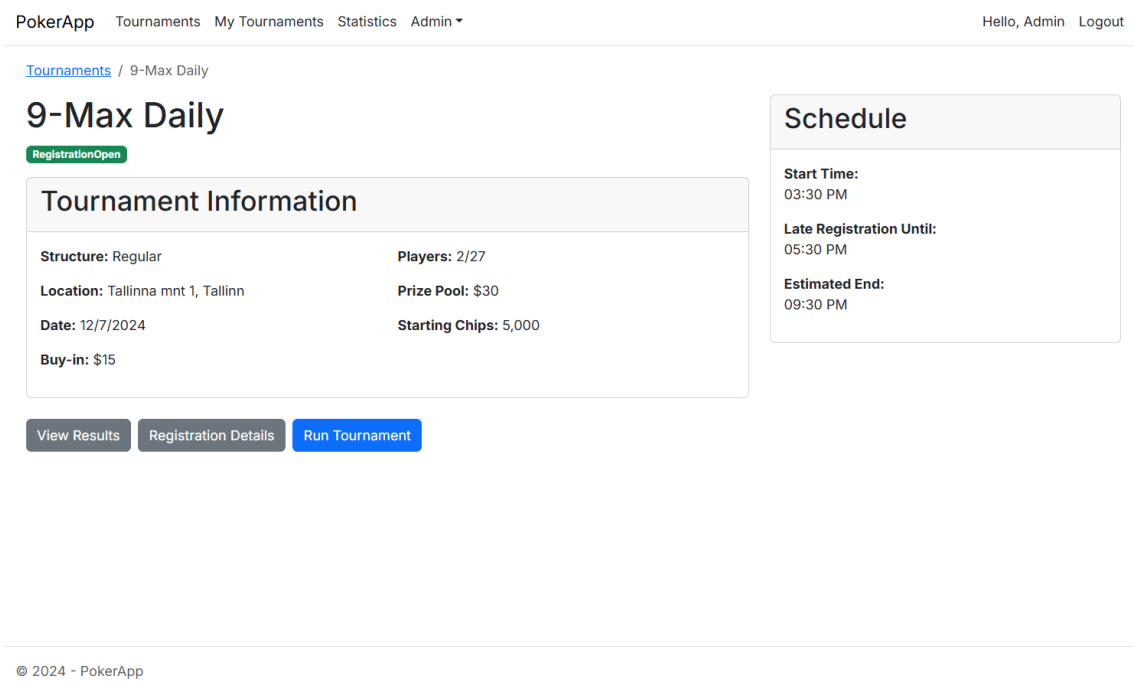
Rakenduse kasutajaliides on üles ehitatud modulaarselt. Põhivaated on turniiride ülevaade, turniiri detailvaade ja turniiri haldamise vaade. Iga vaade on disainitud kasutajakogemuse täiendamiseks, arvestades nii tavakasutajate kui ka administraatorite vajadusi.

Turniiride ülevaade pakub kasutajatele ülevaate kõikidest turniiridest. Vaade sisaldab filtreerimis- ja otsimisfunktsionaalsust, võimaldades kiiresti leida huvipakkuvaid turniire. Turniiride staatus on visualiseeritud värviliselt, kus "RegistrationOpen" on tähistatud rohelise ja "Running" sinise värviga. Tabelis kuvatakse kompaktselt turniiri põhiandmed, nagu formaat, toimumisaeg, asukoht, sisseost ja osalejate arv. Administraatoritele on lisatud nupp "Create New Tournament", mis võimaldab uusi turniire lisada. (Joonis 15)



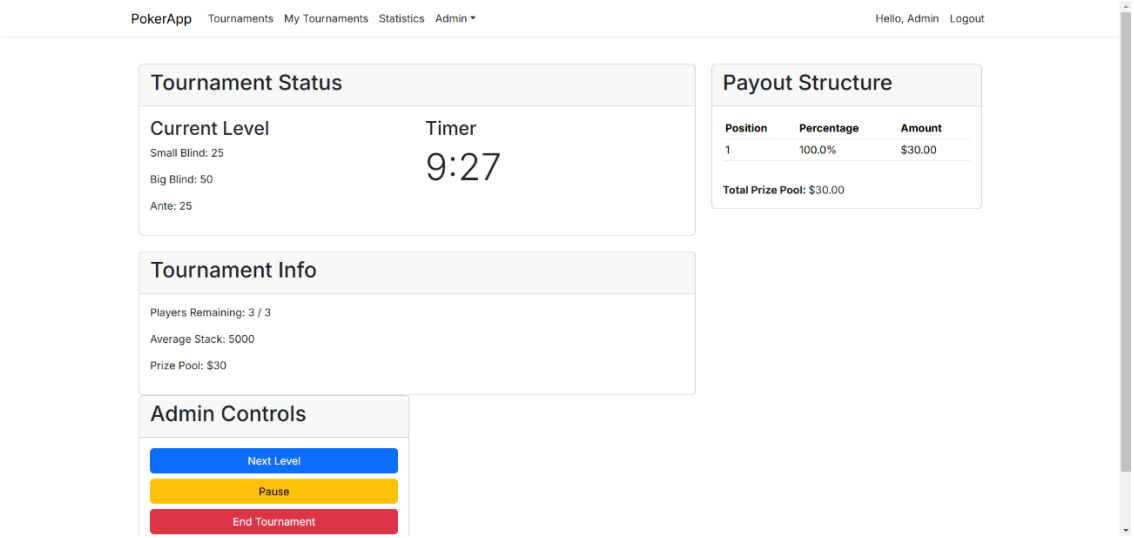
Joonis 15. Turniiride ülevaade administraatoritel.

Turniiri detailvaade kasutab kaheveeruga paigutust, andes selget ülevaadet turniiri põhiinfost ja ajakavast. Vasakul pool kuvatakse turniiri põhiandmed nagu sisseost, mängijate arv ja turniiri formaat, samas kui paremal pool on näha ajakava informatsiooni. Oluline info on esitatud card-komponentides, mis tagab hea loetavuse ja visuaalse haaravuse. Registreerimisnupp muudab oma olekut vastavalt kasutaja staatusele, näidates kas "Register Now" või "Unregister". Navigatsioonitee lehe ülaosas võimaldab kasutajatel lihtsalt mõista oma asukohta rakenduses. (Joonis 16)



Joonis 16. Turniiri detailvaade administraatoritel.

Turniiri vaade on reaalajas toimuvate turniiride haldamiseks või jälgimiseks. Vaade sisaldab reaalajas uuenevat turniirikella, mis näitab taseme kestvust, ning pimepanuste infot, mis kuvab hetkel kehtivad *small blind*-i, *big blind*-i ja *ante* väärtused. Lisaks on toodud alles olevate mängijate arv, keskmine žetoonide kogus mängija kohta, auhinnafond ning dünaamiliselt arvutatav auhinnafondi jaotus. Administraatorile on loodud juhtpaneel, mis võimaldab turniiri alustada, peatada, jätkata ja liikuda järgmisele tasemele. (Joonis 17)



Joonis 17. Turniiri vaade reaalajas administraatoritel.

Rakenduse navigatsioonisüsteem on loodud responsiivsena, kohanedes erinevate ekraanisuurustega. Menüüstruktuur on dünaamiline ja rollipõhine, kuvades kasutajatele ainult nende õigustele vastavaid funktsionaalsusi. Administraatoritel on ligipääs täiendavatele haldusfunktsioonidele läbi "Admin" rippmenüü, samas kui tavakasutajad näevad ainult neile vajalikke põhifunktsioone.

6 Tulemused ja edasiarenduse võimalused

Käesoleva lõputöö tulemusena valmis veebirakenduse prototüüp pokkeriturniiride jälgimiseks ja haldamiseks. Järgnevalt analüüsitakse, kuidas rakendus vastab püstitatud funktsionaalsetele ja mittefunktsionaalsetele nõuetele.

Funktsionaalsete nõuete täitmine:

- Kasutajate registreerimine - Rakendatud ASP.NET Core Identity raamistikuga, mis võimaldab kasutajatel luua kontosid ja neid turvaliselt hallata.
- Turniiride tabel – Loodud Next.js abil dünaamiline vaade, mis võimaldab turniiride filtreerimist ja sorteerimist.
- Turniiridele registreerimine - Teostatud turvalise API kaudu, võimaldades kasutajatel end mugavalt registreerida.
- Reaalajas turniiriinfo jälgimine - SignalR tehnoloogia tagab andmete kiire uuenemise kõigile jälgijatele.
- Turniiride haldamine – Administratiivsed funktsioonid võimaldavad turniiride loomist ja muutmist.

Mittefunktsionaalsete nõuete täitmine:

- Kasutusmugavus - Intuitiivne kasutajaliides ja selge navigatsioon, järgides kaasaegseid disainipõhimõtteid.
- Jõudlus ja kiirus - Optimeeritud andmevahetuse ja vahemälu kasutamine tagab kiire toimivuse.
- Turvalisus - JWT autentimine, HTTPS (HyperText Transfer Protocol Secure) protokoll ja turvalised andmebaasipäringud kaitsevad kasutaja andmeid.
- Skaaleeritavus - Docker konteineriseerimine võimaldab süsteemi lihtsat laiendamist.
- Hooldatavus - Selge koodi struktuur ja põhjalik dokumentatsioon lihtsustavad edasist arendust.

Rakenduse edasiarendamiseks on järgmised võimalused:

- Mobiilirakendus – Võimaldaks pakkuda täiendavaid funktsioone, näiteks teavitusi ja offline režiimi.

- Statistikamooduli täiendamine - Detailsem andmeanalüüs aitaks paremini mõista tulemusi ja trende.
- Otseülekanded – Võimaldaks luua kaasahaaravama platvormi pokkeriturniiride jälgimiseks, eriti suurturniiridel.
- Maksesüsteemid - Lihtsustaks turniiridele registreerimist ja auhinnarahade väljamaksmist.
- Testimine reaalses keskkonnas – Kasutajate tagasiside kogumine funktsionaalsuse ja jõudluse täiustamiseks.

Loodud rakendus on tugev alus pokkeriturniiride jälgimise rakenduseks. Täidetud on esialgselt seatud tehnilised eesmärgid ning loodud on skaleeritav rakendus, mida on võimalik edasi arendada vastavalt kasutajate vajadustele. Edasised arendused keskenduksid funktsionaalsuse laiendamisele ja kasutajakogemuse täiustamisele reaalse tagasiside põhjal.

7 Kokkuvõte

Bakalaureusetöö keskendus pokkeriturniiride jälgimise veebirakenduse prototüübi arendamisele. Töö eesmärgiks oli luua lahendus, mis võimaldaks turniiride tõhusat jälgimist ja haldamist. See eesmärk saavutati läbi põhjaliku analüüsi, sobivate tehnoloogiate valiku ja süstemaatilise arendusprotsessi abil.

Projekti käigus lahendati mitmeid olulisi probleeme pokkeriturniiride jälgimise valdkonnas. Andmete killustatuse probleem lahendati tsentraliseeritud andmebaasi ja ühtse kasutajaliidese loomisega, mis viis kõik info ühte süsteemi. Reaalajas jälgimise funktsionaalsus lahendati SignalR tehnoloogia abil, mis võimaldab turniiriinfo kohest uuenemist kõigile jälgijatele. Turniiride haldamist lihtsustati läbi automatiseeritud protsesside ja intuitiivse administraatoritele suunatud liidese.

Arenduses kasutati rakenduseks sobivaid tehnoloogiaid. Tagarakenduse jaoks ASP.NET Core, eesrakenduse jaoks Next.js ning andmebaasiks PostgreSQL. Need valikud võimaldasid luua skaleeritava prototüübi, mis vastab kõigile püstitatud funktsionaalsetele ja mittefunktsionaalsetele nõuetele. Rakendus pakub turvalist autentimis- ja autoriseerimissüsteemi, kiiret reaalajas andmevahetust ning kohanduvat kasutajaliidest.

Tuvastati ka mitmeid edasiarendusvõimalusi. Nende hulka kuuluvad eraldiseisva mobiilirakenduse arendamine, statistika täiendamine, otseülekannete lisamine ja maksesüsteem. Need võimaldaksid luua veelgi terviklikuma lahenduse.

Loodud prototüüp pakub tugeva aluse täisfunktsionaalse pokkeriturniiride jälgimise ja haldamise platvormi edasiarendamiseks, võimaldades tulevikus lisada uusi funktsionaalsusi vastavalt kasutajate vajadustele.

Kasutatud kirjandus

[1] R. Rinkema, „A Numbers Game: WSOP Main Event Winners & Stats,“ PGT, 05 07 2018. [Võrgumaterjal]. Available: https://www.pgt.com/news/a-numbers-game-the-wsop-main-event. [Kasutatud 10 11 2024].

[2] „How To Play | Texas Holdem Rules,“ WSOP, [Võrgumaterjal]. Available: https://www.wsop.com/poker-games/texas-holdem/rules. [Kasutatud 16 11 2024].

[3] D. Holden, „WORLD SERIES OF POKER® MAIN EVENT® SETS ATTENDANCE RECORD FOR SECOND CONSECUTIVE YEAR,“ WSOP, 08 07 2024. [Võrgumaterjal]. Available: https://www.wsop.com/news/2024/Jul/14022/WORLD-SERIES-OF-POKER-MAIN-EVENT-SETS-ATTENDANCE-RECORD-FOR-SECOND-CONSECUTIVE-YEAR.html. [Kasutatud 15 10 2024].

[4] „Olympic Poker,“ Olympic Park Casino Poker, [Võrgumaterjal]. Available: https://www.olympic-poker.com/tournaments. [Kasutatud 30 10 2024].

[5] „Pokerlens,“ Pokerlens, [Võrgumaterjal]. Available: https://olympic-casino.pokerlens.net/en/venue/7dc5a1b3-9d81-4c74-ba96-c804b94c9f6b/tournaments. [Kasutatud 30 10 2024].

[6] „LETSPOKER,“ LetsPoker, [Võrgumaterjal]. Available: https://lets.poker. [Kasutatud 30 10 2024].

[7] „Triton Poker,“ Triton Poker, [Võrgumaterjal]. Available: https://tritonpoker.plus. [Kasutatud 30 10 2024].

[8] J. Nielsen, „10 Usability Heuristics for User Interface Design,“ Nielsen Norman Group, 24 04 1994. [Võrgumaterjal]. Available: https://www.nngroup.com/articles/ten-usability-heuristics. [Kasutatud 20 11 2024].

[9] „Navigation design basics for Windows apps,“ Microsoft, 04 12 2023. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/windows/apps/design/basics/navigation-basics. [Kasutatud 17 11 2024].

[10] „Canva,“ Canva, [Võrgumaterjal]. Available: https://www.canva.com/. [Kasutatud 10 12 2024].

[11] „Overview of ASP.NET Core,“ Microsoft, 18 06 2024. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-9.0. [Kasutatud 19 11 2024].

[12] „Overview of ASP.NET Core SignalR,“ Microsoft, 02 12 2024. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/aspnet/core/signalr/introduction?view=aspnetcore-9.0. [Kasutatud 18 11 2024].

[13] „Introduction to Identity on ASP.NET Core,“ Microsoft, 30 08 2024. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-9.0&tabs=visual-studio. [Kasutatud 19 11 2024].

[14] „Entity Framework Core,“ Microsoft, 12 11 2024. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/ef/core/. [Kasutatud 19 11 2024].

[15] „2024 Developer Survey,“ Stackoverflow, 2024. [Võrgumaterjal]. Available: https://survey.stackoverflow.co/2024/technology. [Kasutatud 19 11 2024].

[16] „express,“ npm, 2024. [Võrgumaterjal]. Available: https://www.npmjs.com/package/express. [Kasutatud 20 11 2024].

[17] „Introduction,“ Socket.IO, [Võrgumaterjal]. Available: https://socket.io/docs/v4/. [Kasutatud 19 11 2024].

[18] „Overview,“ Passport.js, [Võrgumaterjal]. Available: https://www.passportjs.org/concepts/authentication/. [Kasutatud 20 11 2024].

[19] „Database integration,“ OpenJS Foundation, [Võrgumaterjal]. Available: https://expressjs.com/en/guide/database-integration.html. [Kasutatud 09 12 2024].

[20] J. Potter, „express,“ npm, [Võrgumaterjal]. Available: https://npmtrends.com/express. [Kasutatud 19 11 2024].

[21] „Spring Boot,“ spring, [Võrgumaterjal]. Available: https://spring.io/projects/spring-boot. [Kasutatud 20 11 2024].

[22] „WebSockets,“ spring, [Võrgumaterjal]. Available: https://docs.spring.io/spring-framework/reference/web/websocket.html. [Kasutatud 20 11 2024].

[23] „Spring Security,“ spring, [Võrgumaterjal]. Available: https://spring.io/projects/spring-security. [Kasutatud 20 11 2024].

[24] „SQL Databases,“ spring, [Võrgumaterjal]. Available: https://docs.spring.io/spring-boot/reference/data/sql.html. [Kasutatud 20 11 2024].

[25] „Developer Ecosystem Java,“ Jetbrains, 2023. [Võrgumaterjal]. Available: https://www.jetbrains.com/lp/devecosystem-2023/java. [Kasutatud 20 11 2024].

[26] „Introduction,“ [Võrgumaterjal]. Available: https://nextjs.org/docs#main-features. [Kasutatud 20 11 2024].

[27] „Building UI with Components,“ Next.js, [Võrgumaterjal]. Available: https://nextjs.org/learn/react-foundations/building-ui-with-components. [Kasutatud 09 12 2024].

[28] „TypeScript,“ Next.js, [Võrgumaterjal]. Available: https://nextjs.org/docs/pages/api-reference/config/typescript. [Kasutatud 09 12 2024].

[29] „What is Angular?,“ Angular, [Võrgumaterjal]. Available: https://angular.dev/overview. [Kasutatud 20 11 2024].

[30] „Anatomy of a component,“ Angular, [Võrgumaterjal]. Available: https://angular.dev/guide/components. [Kasutatud 09 12 2024].

[31] „Angular,“ Microsoft, [Võrgumaterjal]. Available: https://www.typescriptlang.org/docs/handbook/angular.html. [Kasutatud 09 12 2024].

[32] „Introduction,“ Vue.js, [Võrgumaterjal]. Available: https://vuejs.org/guide/introduction.html. [Kasutatud 20 11 2024].

[33] „Components Basics,“ Vue.js, [Võrgumaterjal]. Available: https://vuejs.org/guide/essentials/component-basics. [Kasutatud 09 12 2024].

[34] „Using Vue with TypeScript,“ Vue.js, [Võrgumaterjal]. Available: https://vuejs.org/guide/typescript/overview. [Kasutatud 09 12 2024].

[35] „About,“ postgresql, [Võrgumaterjal]. Available: https://www.postgresql.org/about. [Kasutatud 20 11 2024].

[36] „JSON Types,“ PostgreSQL, [Võrgumaterjal]. Available: https://www.postgresql.org/docs/current/datatype-json.html. [Kasutatud 09 12 2024].

[37] „Chapter 15. Parallel Query,“ PostgreSQL, [Võrgumaterjal]. Available: https://www.postgresql.org/docs/current/parallel-query.html. [Kasutatud 09 12 2024].

[38] „The Main Features of MySQL,“ [Võrgumaterjal]. Available: https://docs.oracle.com/cd/E17952_01/mysql-9.1-en/features.html. [Kasutatud 20 11 2024].

[39] „13.5 The JSON Data Type,“ Oracle, [Võrgumaterjal]. Available: https://dev.mysql.com/doc/refman/8.4/en/json.html. [Kasutatud 09 12 2024].

[40] „7.6.3 MySQL Enterprise Thread Pool,“ Oracle, [Võrgumaterjal]. Available: https://dev.mysql.com/doc/refman/8.4/en/thread-pool.html. [Kasutatud 09 12 2024].

[41] „About SQLite,“ [Võrgumaterjal]. Available: https://www.sqlite.org/about.html. [Kasutatud 20 11 2024].

[42] „JSON Functions And Operators,“ SQLite, [Võrgumaterjal]. Available: https://www.sqlite.org/json1.html. [Kasutatud 09 12 2024].

[43] „Using SQLite In Multi-Threaded Applications,“ SQLite, [Võrgumaterjal]. Available: https://www.sqlite.org/threadsafe.html. [Kasutatud 09 12 2024].

[44] „OWASP Top Ten,“ OWASP, 09 2024. [Võrgumaterjal]. Available: https://owasp.org/www-project-top-ten. [Kasutatud 16 12 2024].

[45] L. S. Sterling, The Art of Agent-Oriented Modeling, London: The MIT Press, 2009.

[46] J. Nielsen, „Response Times: The 3 Important Limits,“ Nielsen Norman Group, 2014. [Võrgumaterjal]. Available: https://www.nngroup.com/articles/response-times-3-important-limits. [Kasutatud 16 11 2024].

[47] „React Quick Start,“ React, 2024. [Võrgumaterjal]. Available: https://react.dev/learn. [Kasutatud 10 11 2024].

[48] „Notify,“ postgresql, [Võrgumaterjal]. Available: https://www.postgresql.org/docs/current/sql-notify.html. [Kasutatud 20 11 2024].

[49] „Angular compiler options,“ Angular, [Võrgumaterjal]. Available: https://angular.dev/reference/configs/angular-compiler-options. [Kasutatud 09 12 2024].

[50] „Appropriate Uses For SQLite,“ SQLite, [Võrgumaterjal]. Available: https://www.sqlite.org/whentouse.html. [Kasutatud 09 12 2024].

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

Mina, Daniel Vasser

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose Pokkeriturniiride jälgimise veebirakenduse prototüüp, mille juhendaja on Kristiina Hakk
   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.

06.01.2025

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.