

TEKOÄLYASIAKKAAN LAUSEIDEN KÄSITTELY

Karvinen Markku

Opinnäytetyö

Tieto- ja viestintäteknikka
Insinööri (AMK)

2021

Tieto- ja viestintäteknikka
Insinööri (AMK)

Tekijä	Markku Karvinen	Vuosi	2021
Ohjaaja	Petri Hannula		
Toimeksiantaja	OmaDigi-hanke		
Työn nimi	Tekoälyasiakkaan lauseiden käsittely		
Sivumäärä	36		

Opinnäytetyön tarkoituksena on toteuttaa OmaDigi-hankkeen pohjalle selainpohjainen Healthcare Customer Service -peli, jossa pelaaja toimii terveyskeskuksen puhelinvastaanotossa hoitajan roolissa. Pelaajan tehtävänä on vastaanottaa puheluita tekoälyasiakkailta sekä tehdä hoidon tarpeen arviointia.

Työn tavoitteena oli kehittää tarkoitukseen soveltuva tekoäly, joka kykenee keskustelemaan pelaajan kanssa avainsanoihin perustuvan lausegeneroinnin avulla. Työssä käsitellään myös testaamista ja sen palautetta.

Työ toteutettiin JavaScript-ohjelmointikieltä ja sen kirjastoja käyttäen. Työssä käytetty teoriaosuus koostuu lähinnä tekoälyyn ja käytettyihin teknologioihin liittyvistä tiedeartikkeleista, kirjoista ja tutkimuksista.

Lopputulokseksi saatiin yksinkertainen prototyyppi selainpelistä, jossa pelaaja toimii puhelinvastaanotossa ja tekoäly kykenee keskustelemaan itseensä liittyvistä asioista. Satunnaiset testaajat sekä oma työryhmä testasivat pelin. Testamalla kartoitettiin ja korjattiin pelin ongelmakohtia. Testaus myös synnytti uusia jatkokehitysideoita, joita ei kuitenkaan rajallisen aikataulun ja laajuuden johdosta toteutettu. Haasteena oli toteuttaa tavallisesta poikkeava tekoäly, joka kykenee toimimaan asiakkaan roolissa ja keskustelemaan omasta voinnistaan terveydenhuoltoalaan liittyvien avainsanojen perusteella.

Pidemmällä jatkokehityksellä pelistä on mahdollista saada laadukas opetuskäyttöön soveltuva työkalu, jota voidaan käyttää myös muissakin kohteissa modulaarisuuden ansiosta.

Avainsanat

avainsana, JavaScript, lausegenerointi, selainpeli, tekoäly, testaus, verkko-ohjelmointi

Degree Programme in Information
and Communication Technology
Bachelor of Engineering

Author	Markku Karvinen	Year	2021
Supervisor	Petri Hannula		
Commissioned by	OmaDigi-project		
Subject of thesis	Processing of AI Customer Sentences		
Number of pages	36		

The purpose of this thesis was to implement browser-based Healthcare Customer Service game based on the OmaDigi-project in which the player acts as a nurse at a health center's customer service. The aim was to develop artificial intelligence that can communicate with the player through keyword-based sentence generation.

The study was carried out using the JavaScript programming language and its libraries. Testing of the game was performed by creators of the game and random testers. The theoretical part used in the study consisted mainly of scientific articles, books and studies related to artificial intelligence and the technologies used.

The end result was a simple prototype of a browser game in which the player acts as a nurse at healthcare customer service and the artificial intelligence is able to discuss matters related to itself and make assessment of the need for treatment. The testing identified and corrected problem areas in the game. Testing also gave rise to new ideas for further development, which, however, were not implemented due to limited time and scope. The challenge was to implement unusual artificial intelligence that is able to act in the role of the customer and discuss its own well-being on the basis of keywords related to the healthcare sector. With further development of the game, it is possible to get a high-quality tool suitable for educational use, which can also be used in other locations thanks to its modularity.

Key words artificial intelligence, browser game, JavaScript, keyword, network programming, sentence generation, testing

SISÄLLYS

1	JOHDANTO	6
2	TEKOÄLY JA LAUSEEN KÄSITTELY	7
2.1	Tekoäly	7
2.2	Lauseen käsittely	7
3	KÄYTETYT TEKNIIKAT JA TYÖKALUT	9
3.1	Tekniikat	9
3.2	Työkalut	10
4	SOVELLUKSEN TOTEUTUS	11
4.1	Suunnittelu ja kehitysympäristö	11
4.2	Rakenne	13
4.3	Tekoälyasiakas	15
4.4	Lauseen käsittely	19
4.5	Pelinäkymä	22
5	TESTAUS	26
5.1	Alkuvaiheen testaus	26
5.2	Testikierroksien palautteen antaminen	28
5.3	Palautteen käsittely DbViewissä	29
5.4	Ensimmäinen testikierros	31
5.5	Toinen testikierros	32
6	POHDINTA	33
	LÄHTEET	35

KÄYTETYT LYHENTEET JA TERMIT

CSS	Cascading Style Sheets, WWW-dokumentin tyyliohjekieli
data	tieto, aineisto
front end	verkkoselaimessa suoritettava koodi
funktio	aliohjelma, joka suorittaa osan koodista
generaattori	ohjelman tai koodin osan, joka pitää sisällään yhden tai useamman funktion
HTML	Hypertext Markup Language, hypertekstin merkintäkieli
JavaScript	yleinen skriptipohjainen ohjelmointikieli selaimissa
JSON	JavaScript Object Notation, avoimen standardin tiedostomuoto tiedonvälitykseen
luokka	elementti, joka sulkee sisälleen olion tarvitseman tiedon ja tarjoaa metodeja, joiden avulla luokan sisäistä tietoa päästään muuttamaan
metodi	luokkaan kirjoitettu lähdekoodista koostuva kokonaisuus, jolle on annettu nimi, ja jota voidaan kutsua
objekti	ajonaikainen elementti, joka sisältää erilaista tietoa
RegExp	Regular Expression (säännöllinen lauseke)
taulukko	alkeellinen tietorakenne, jonka sisällä monta peräkkäistä muuttujaa
tekoäly	kone tai ohjelma, joka pyrkii jäljittelemään ihmisen toimintaa

1 JOHDANTO

Digitalisaation merkitys kasvaa nykyisessä maailmassa. Tuottamalla hyviä etäpalveluita saadaan vähennettyä julkisen palvelun kuormitusta, parannettua hyvinvointia ja säästettyä verorahoja. Keinoja näihin on tuottaa hyviä, helppokäyttöisiä ja tehokkaita virtuaaliympäristöjä älykkäiden terveyspalveluiden käyttöön, opiskelun tueksi ja muiden julkisten palveluiden käyttöön. Tähän tarpeeseen voidaan vastata nykyään nopeasti kehittyvän teknologian, tekoälyn avulla. (Valtiovarainministeriö 2020.)

Opinnäytetyö on toteutettu Lapin ammattikorkeakoulussa, OmaDigi-hankkeen pohjalta. Opinnäytetyön tarkoituksena on tutkia tekoälyn lauseen käsittelyä ja rakentaa prototyyppi pelille. Tavoitteena on terveyskeskuksen puhelinpäivystykseen liittyvä selainpeli, jossa pelaaja toimii hoitajan roolissa. Pelissä vastaanottoon tulee puheluita erilaisilta asiakkailta, joilla on monenlaisia vaivoja. Pelaajan tehtävänä on tehdä hoidon tarpeen arviointia asiakkaan kanssa keskustelemalla ja päättää ohjataanko asiakas hoitoon vai ei.

Opinnäytetyön tavoitteena on kehittää tekoäly, joka kykenee käsittelemään pelaajan kirjoittamat lauseet ja antamaan loogisia vastauksia avainsanoja ja aiheen-tarkistusta hyödyntäen. Sovelluksen testaaminen ja siitä saadun palautteen käsittely korostuvat paljon jatkokehityksen kannalta. Tavoitteena on, että sovellusta voisi myöhemmin käyttää terveydenhuollon opetuskäytössä yhtenä oppimisen työkaluna.

Rajauksena graafinen käyttöliittymä jätetään minimalistiseksi ja tekstipohjaiseksi. Asiakkaan yhteydenottosyyt rajoitetaan muutamiin erilaisiin sairauksiin. Puheluiden sisältö pyritään pitämään asiallisena ja neutraalina.

Tämä opinnäytetyö pitää sisällään tekoälyyn ja käytettyihin teknologioihin liittyvää teoriaa. Sen jälkeen opinnäytetyössä käydään läpi pelin toteutukseen ja testaamiseen liittyviä asioita.

2 TEKOÄLY JA LAUSEEN KÄSITTELY

2.1 Tekoäly

Tekoälyn AI (Artificial Intelligence) tarkoituksena on saada koneet tekemään ihmismäisiä toimintoja, jotka ajattelevat, ymmärtävät ja toimivat kuin normaali ihminen. Tavoitteena on saada tietokoneet tekemään asioita, joissa ihmiset ovat nykyään parempia. Käyttökohteita ovat esimerkiksi ongelmanratkaisut, haastavat laskennat, asioiden selittäminen ja käyttäjien neuvominen. Tekoäly vähentää ihmisen tekemää työtä ja se ei väsy. (Gupta & Mangla 2020, 4 – 7.)

Tekoälyä voidaan käyttää osana chatbotteja, joita eri yritykset käyttävät automatisoimaan ihmisten kanssa käytyjä keskusteluja. Se selvittää asiakkaan vaatimukset ja tarjoaa nopeita ja asianmukaisia vastauksia. (Kanagaraj, Navaneetha & Muhammed 2020, 94.)

Tekoälyllä on mahdollisuus olla merkittävä tekijä tulevaisuudessa lääketieteen ja kansanterveyden osa-alueella. Käsitteellisesti lääketieteellisen diagnoosin selvittäminen oireiden perusteella on helppoa, mutta käytännössä tekoälyn avulla sen toteuttaminen on haastavaa. (CACM Staff 2017, 11.)

Tekoälyn avulla voidaan järjestää sama opetuksen sisältö ja rakenne kaikille oppijoille. Se on helposti saatavissa aina haluttuna aikana ja tarjoaa oppijalle henkilökohtaisen opetussisällön. Tekoälyteknologian jatkokehityksen myötä erityisesti etäopetuksen tulisi ottaa positiivinen asenne testialueeksi tekoälyn perusteelliselle soveltamiselle. Sillä voitaisiin tehostaa oppimista. (Shuguang, Zheng & Lin 2020, 132.)

2.2 Lauseen käsittely

NLP (Natural Language Processing) on kokoelma prosesseja, algoritmeja ja työkaluja, joita tekoäly käyttää tulkitsemaan ihmiskielellä kirjoitettua tekstiä. NLP tarkoittaa strukturoimattoman tiedon tulkintaa. Se käyttää erilaisia menetelmiä, kuten mielipiteiden analysointia, asiakirjojen luokittelua ja tekstien yhteenvetoa.

(Deshpande & Kumar 2018, 163.) Kielen käsittelyssä on kyse lauseiden tunnistamisesta, niiden rakenteen määrittämisestä, merkityksen purkamisesta ja näiden tietojen käsittelystä (Clark, Fox & Lappin 2013, 40).

Päätavoitteena NLP:n kanssa on, että käyttäjä kysyy tietokoneelta tietoa omalla äidinkielellään ja tietokone osaa tulkita tämän syötteen. Tämän seurauksena kone antaa vastauksen takaisin käyttäjän kielellä. Tavoitteena on tehdä tietokoneista niin älykkäitä, että ne voisivat ymmärtää normaalia kieltä. NLP voidaankin jakaa kahteen tapaukseen. Ensimmäisenä tapauksena on luonnollisen kielen ymmärtäminen, jonka tavoitteena on, että tietokone ymmärtää sille annetut syötteet käyttäjän kielellä. Toisena on luonnollisen kielen generointi, jonka tarkoituksena on saada tietokone tuottamaan normaalia käyttäjän ymmärtämää kieltä. (Gupta & Mangla 2020, 8.)

3 KÄYTETYT TEKNIIKAT JA TYÖKALUT

3.1 Tekniikat

JavaScriptiä käytetään käyttäjän ja palvelimen puolella tekemään verkkosivuista interaktiivisia. HTML ja CSS antavat rakenteen ja tyylin sivulle ja JavaScriptin avulla sivulle saadaan interaktiivisia elementtejä parantamaan ja rikastuttamaan sivun käyttämistä. (Jeansoulin 2018, 14 – 15.)

React on käyttöliittymien rakentamiseen koottu JavaScript-kirjasto. Reactin avulla verkkosivun pystyy pilkkomaan palasiksi komponentteihin, jolloin jokaista eri palasta voidaan kutsua erikseen uudelleen käytettäväksi. (Chiarelli 2018, 6 – 7.)

JSON (JavaScript Object Notation) on tiedonsiirtämiseen käytetty formaatti käyttäjän ja palvelimen välillä verkkosovelluksissa. Se on johdannainen JavaScriptistä, mutta kuitenkin täysin itsenäisesti toimiva. JSON on kevyempi ja helppo-käyttöisempi kuin aiemmin käytössä ollut XML-formaatti. Useimmat yleisimmät ohjelmointikielet tukevat JSON:n käyttöä, kuten C#, C++, Java, PHP, Python ja Ruby. (Sriparasa & D'mello 2018, 6 – 7.)

Node.js on avoimen lähdekoodin pohjalle tehty ajonaikainen ympäristö, joka soveltuu palvelinpuolen verkkosovellusten kehittämiseen. Node.js on käytännössä JavaScript-palvelimen puolella. Se on toiminnallisuudeltaan asynkroninen. (Solutions 2018, 1.)

MongoDB on avoimen lähdekoodin asiakirjoille tarkoitettu tietokantaohjelma, joka tarjoaa sisäänrakennetusti tuen JSON-formaatille. Ohjelma skaalautuu hyvin korkean kuormituksen tilanteisiin. Se tarjoaa täyden hakemistotuen, joka perustuu mihin tahansa asiakirjan käytettävissä olevaan määritteeseen. (Bojinov 2018, 67.)

3.2 Työkalut

Git on suosittu ja laajasti käytössä oleva versionhallintajärjestelmä. Sen avulla käyttäjä voi tehdä useita kehityshaaroja, joissa voidaan luoda ja muokata tiedostojen välillä. Käyttäjä voi näin testata ja hienosäätää uuden ohjelmointiosuuden omassa ympäristössään, ennen tämän jakamista muiden käyttöön. (Laster 2016, 4.)

GitLab on Ruby-kielellä rakennettu käyttäjäystävällinen versionhallintajärjestelmä Git-arkistojen käsittelyyn ja versionhallintaan. Käyttäjä voi hallita käyttöoikeuksia ja ohjelman versioita helpommin ohjelman verkkosivulla kuin pelkästään pitkiä komentoja syöttämällä terminaalin Gitin avulla. (Hethy 2013, 5 – 7.) GitLab antaa luvat ja oikeudet vain tämän tiimin jäsenille, kun taas GitHubin arkistoihin ja koodeihin on vapaat pääsyt kaikilla (Vats 2020).

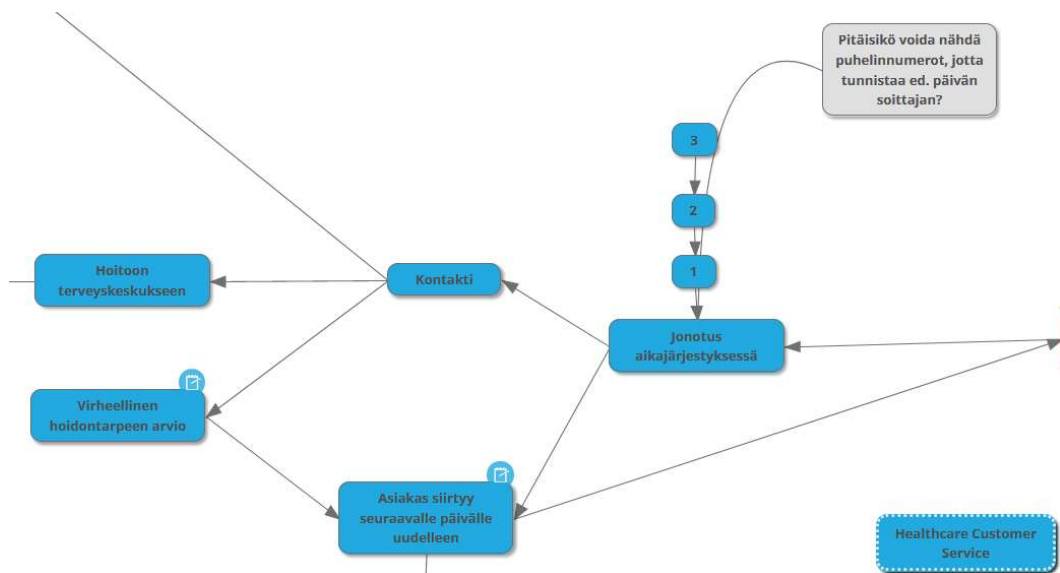
Visual Studio Code on Microsoftin kehittämä kevyt ja tehokas avoimen lähdekoodin ohjelmointityökalu, joka tarjoaa tuen monille ohjelmointikielille, kuten Java, C++, Python, CSS, Go ja Dockerfile. Ohjelma tarjoaa myös tuen eri laajennuksille debuggaukseen ja verkkosivujen kehittämiseen. (Mustafeez 2021.)

Discord on alun perin pelaajille ryhmäkeskusteluun kehitetty ohjelma, joka tarjoaa mahdollisuuden ääni- ja videopuheluihin sekä normaaliin viestittelyyn käyttäjien kesken. Käyttäjät pystyvät myös jakamaan oman työpöytänäkönsä muiden nähtäville. Myöhemmin alusta on nostanut suosiotaan myös yleisessä käytössä kommunikaation työkaluna. Discord jakaantuu omiin servereihinsä, joissa on omat käyttäjät, säännöt, aiheet ja kanavat. (Delfina & Dean 2021.)

4 SOVELLUKSEN TOTEUTUS

4.1 Suunnittelu ja kehitysympäristö

Projektia varten suunniteltiin nettiselaimessa toimiva pelimäinen prototyyppi potilasvastaanotosta, jossa pelaaja toimii terveyskeskuksen puhelinvastaanotossa hoitajan roolissa. Pelaajan tehtävänä oli käsitellä tulevia virtuaalipuheluita sekä tehdä hoidon tarpeen arviointia oireiden perusteella. Pelin tarkoitus oli toimia eri käyttöjärjestelmillä, selaimilla ja älypuhelimilla. Projektin karkea suunnitelma laadittiin mindmapia hyödyntäen (Kuvio 1).



Kuvio 1. Ote Mindmapista

Pelin laajuutta päätettiin rajata tiukan aikataulun vuoksi. Sen johdosta käyttöliittymä suunniteltiin yksinkertaiseksi ja selkeäksi. Pelin ulkoasu on minimaalinen, eikä peli sisällä ääniä. Käyttöliittymä toimii vapaamuotoisella tekstipohjaisella syötteellä, eli pelaajalla ei ole valmiita kysymysvaihtoehtoja.

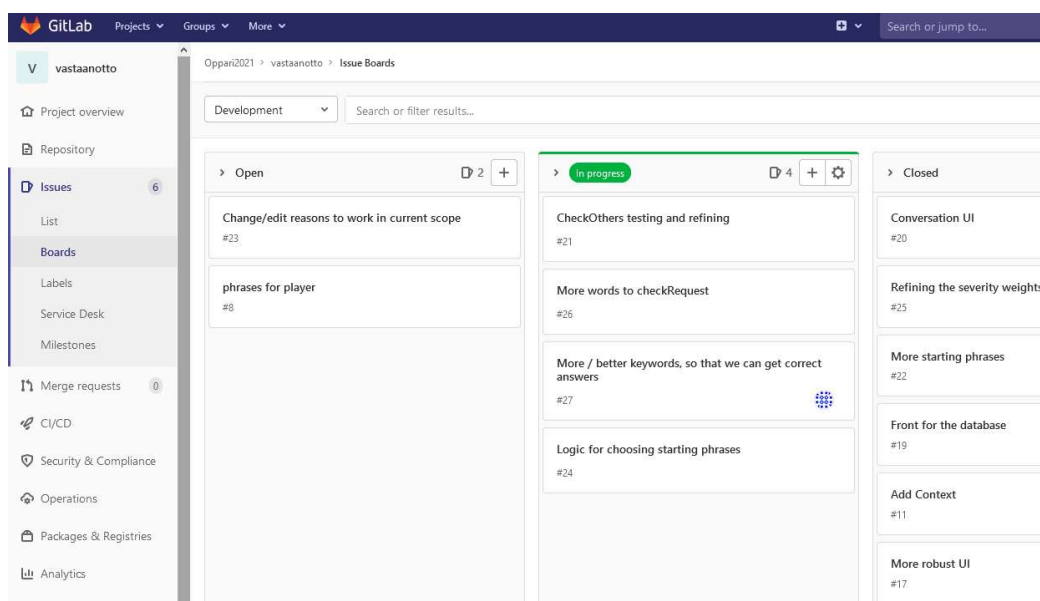
Yhteydenottosyyt rajattiin kuuteen sairauteen: arrhythmia (rytmihäiriö), gastroenteritis (suolistotulehdus), hypertension (verenpainetauti), flu (flunssa), meningitis (aivokalvotulehdus), panic disorder (paniikkihäiriö). Koettiin, että nämä sairaudet ovat riittävän erilaisia, mutta sisältävät useita samanlaisia oireita keskustelunaiheiksi lausegenerointiin.

Sairauksille ja oireille ei laadittu vaikeustasoja tai pisteytysjärjestelmää. Suunniteltiin, että pelaajan tavoitteena on onnistunut hoidon tarpeen arviointi ja useampien virtuaalipuheluiden käsittely. Myös erilaiset asiakastyypit (puhelias, ärtyisä, hiljainen) rajattiin peliprojektin ulkopuolelle jatkokehitettäviin ominaisuuksiin.

Lausegeneroinnin puolella päätettiin lähteä liikkeille perusteista ja rakentaa oma muista alustoista riippumaton lauseen käsittely. Prototyyppi pyrittiin rakentamaan siten, että tekoälyn tukena voitaisiin myös käyttää valmiita NLP-ratkaisuja.

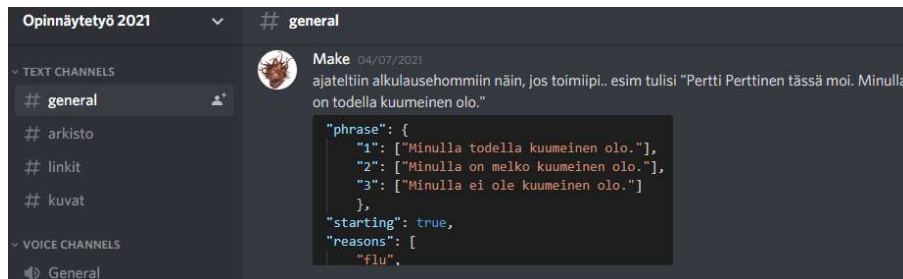
Pelin kehitykseen päädyttiin käyttämään JavaScript-kieltä, joka soveltuu hyvin selainpelien tekemiseen ja joka oli tekijöille jo ennalta tutumpi ohjelmointikieli. JavaScript-kirjastosta päätettiin käyttää Reactia, joka soveltuu hyvin modulaarisen käyttöliittymän rakentamiseen. Taustalle rakennettu testauksen tietokanta tehtiin MongoDB-tietokantaa hyödyntäen.

Pelin rakentamisessa päädyttiin käyttämään Visual Studio Code -ohjelmointityökalua. Versiohallintana lokaalisti käytettiin Git-versiohallintatyökalua VS Coden terminalia käyttäen. Projektin Git-kansio linkitettiin työryhmän käytettäväksi GitLab-versionhallintajärjestelmään. GitLabissa tiimin jäsenille pystyi määrittämään, mitä osuuksia kukin pääsääntöisesti tekee projektiin (Kuvio 2). Heikkoutena GitLab-ympäristö tuntui välillä hidastelevan usean käyttäjän tehdessä samanaikaisesti lisäyksiä pääkoodiin.



Kuvio 2. GitLab Issue Board

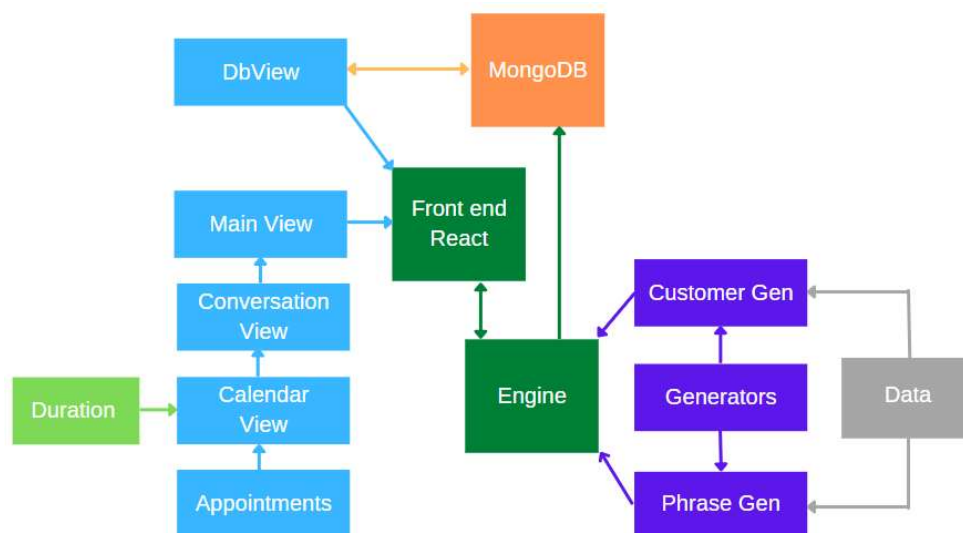
Viestintä työryhmän kesken toteutettiin Discord-keskusteluohjelman välityksellä (Kuvio 3). Ohjelma oli helppokäyttöinen, ja sen avulla pystyi nopeasti vaihtamaan viestejä, ottamaan puheluita ryhmän kesken ja myös jakamaan oman ruudun toiselle nähtäväksi ongelmakohtien kohdalla.



Kuvio 3. Osa Discord-keskustelua 7.4.2021

4.2 Rakenne

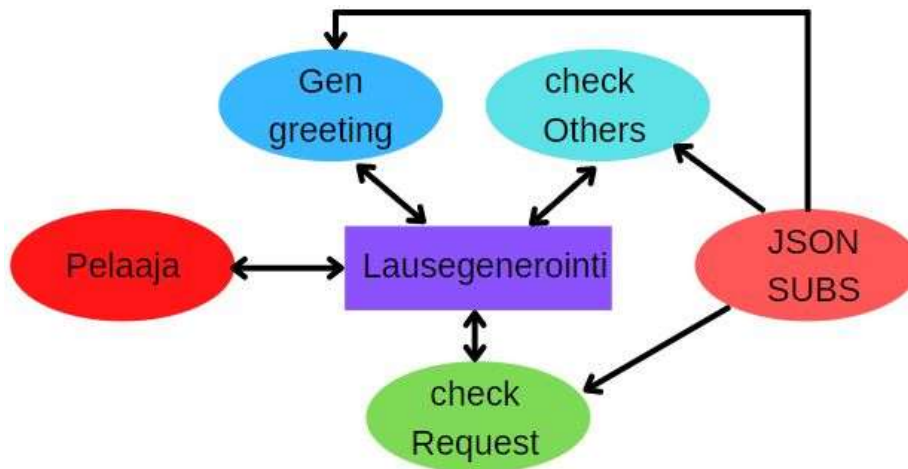
Sovellus on rakenteeltaan modulaarinen (Kuvio 4). Sen käyttöliittymä ja pelimoottori ovat erillään toisistaan. Käyttöliittymä on rakennettu Reactia käyttäen, jossa sen näkymät ovat omina komponentteinaan. Käyttöliittymästä lähtee pelaajan antamat käsiteltävät syötteet pelimoottorille käsiteltäväksi. Taustalla testausta varten on MongoDB-tietokanta, ja sillä on oma näkymä käyttöliittymässä.



Kuvio 4. Sovelluksen modulaarinen rakenne

Pelimoottori ottaa syötteen vastaan käsiteltäväksi. Se pitää sisällään lausegeneraattorin, asiakasgeneraattorin, yleisiä generaattoreita ja myös JSON-tiedostot, joissa säilötään esimerkiksi erilaisia aiheita, lauseita, täytesanoja ja

syyobjekteja. Kuviossa 5 näkyy esimerkkinä pelimoottorin lausegeneroinnin rakenne ja kuinka se hyödyntää ympärillä olevia generaattoreita ja funktioita.



Kuvio 5. Lausegeneroinnin rakenne

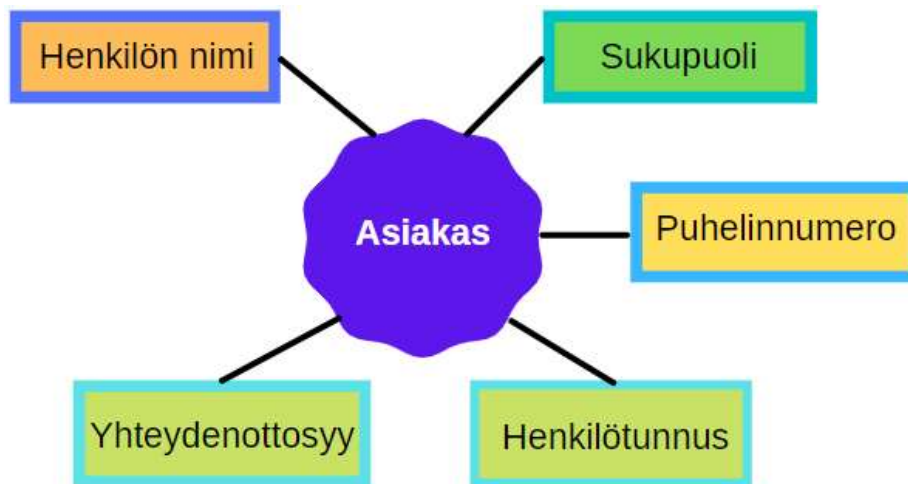
Sovellukselle luotu tiedostorakenne on esitetty kuviossa 6. Käyttöliittymän eri näkymät ja merkittävät ominaisuudet ovat omia komponenttejaan, ja ne on jaettu eri tiedostoihin. Pelimoottorin tiedostot ovat Engine-kansiossa, joka pitää sisällään omat tiedostonsa asiakas-, lause- ja muille generaattoreille. Lisäksi Data-kansio pitää sisällään kaikkien eri syiden, aiheiden ja täytelauseiden tiedostot JSON-muodossa.



Kuvio 6. Sovelluksen tiedostorakenne

4.3 Tekoälyasiakas

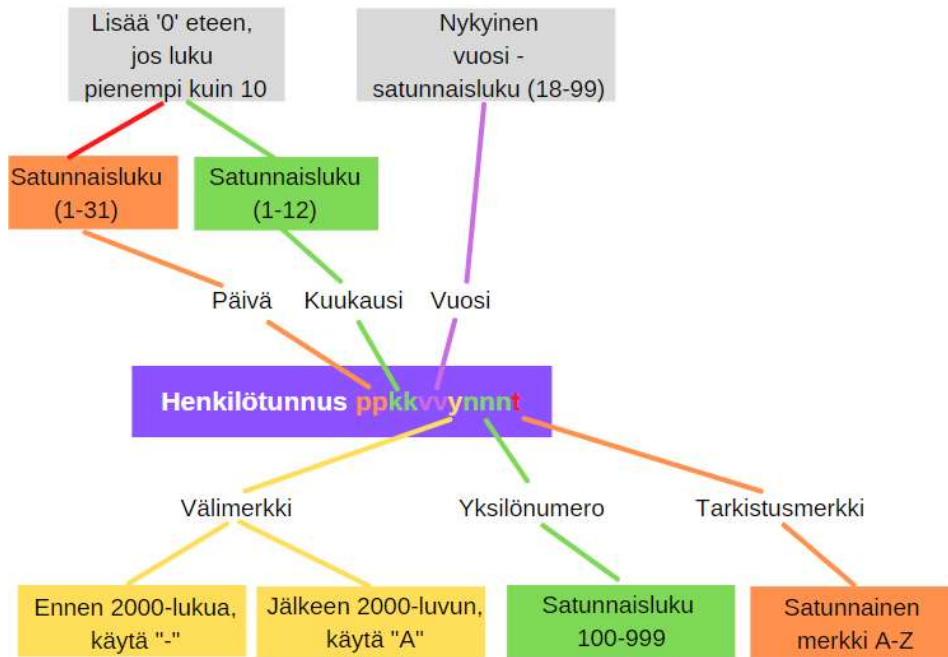
Tekoälyasiakas muodostetaan luokittelun avulla siten, että sille annetaan erilaisia muuttujia ja metodeja. Satunnaisgenerointien avulla jokaisesta asiakkaasta saadaan luotua yksilöllinen (Kuvio 7). Asiakkaalle annetaan satunnaisesti generoidut etu- ja sukunimi, sukupuoli, puhelinnumero, henkilötunnus ja yhteydenottosyy. Pelin aikana tekoälyasiakkaita voidaan luoda useampia kymmeniä, joista jokaisella on yksilölliset tiedot ja ominaisuudet.



Kuvio 7. Asiakkaaseen liittyvät generoinnit

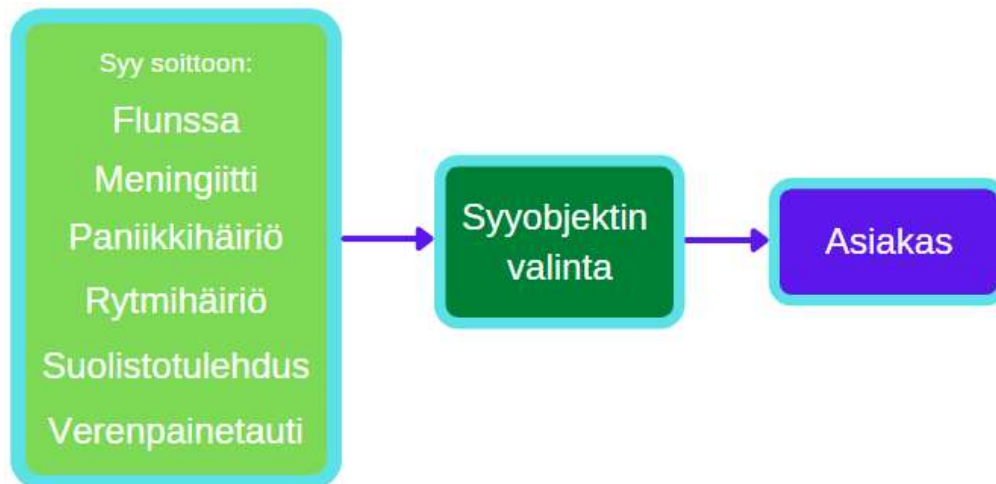
Henkilötunnuksen perusteella tekoälyasiakas saa muut henkilötiedot. Lisäksi sitä päädyttiin käyttämään tekoälyasiakkaan tunnistevaimena. Henkilötunnuksesta haluttiin luoda aidon tuntuinen, jotta se välittäisi realistisen tunteen soittavasta asiakkaasta pelaajalle.

Henkilötunnuksen generointi on kuviossa 8 purettu yksinkertaisempaan muotoon, jossa yhdistelemällä erilaisia numeroita, merkkejä ja kirjaimia saadaan aikaan asiakkaalle yksilöllinen henkilötunnus. Tässä päädyttiin luomaan lähes oikean tyylinen henkilötunnus, mutta yksilönumeron ja tarkistusmerkin kanssa jätettiin enemmän satunnaisuutta peliin, joten täysin oikeita henkilötunnuksia ei kovin helpolla tapahdu. Asiakkaan sukupuoli selviää myös normaaliin tapaan yksilönumeron miehellä ollessa pariton ja naisella parillinen.



Kuvio 8. Henkilötunnuksen generointi

Asiakkaan luontivaiheessa asiakkaalle generoidaan yhteydenottosyy. Periaatteessa yhteydenottosyy voi olla mikä tahansa, mutta pelin liittyessä hoitoalan terveyskeskuksen vastaanottoon, ne ovat erilaisia sairauksia. Opinnäytetyön kirjoitushetkellä erilaisia yhteydenottoja oli yhteensä kuusi kappaletta (Kuvio 9).



Kuvio 9. Asiakkaan yhteydenottosyyt

Yhteydenottosyyt pitävät sisällään erilaisia keskusteluaiheita, jotka ovat esimerkiksi erilaiset oireet. Kuviossa 10 on esitetty flunssaan liittyviä aiheita. Näiden aiheiden perusteella määritellään se, mistä aiheista asiakastekoäly pystyy tarkemmin keskustelemaan.



Kuvio 10. Flunssan sisältämät yleiset keskusteluaiheet

Yhteydenottosyyt on kuvattu allReasons.json-tiedostossa, joka pitää sisällään erilaisia syyobjekteja. Kuviossa 11 on otettu tarkempaan tarkasteluun flunssan syyobjekti, jonka Topics-taulukko pitää sisällään erilaisia aiheita. Aiheet ovat yleisesti yhteydenottosyyhyyn liittyviä oireita.

Jokaisella aiheella on kolme eritasoista vakavuusastetta, jotka on asetettu oman aiheessa olevan taulukon sisään. Taulukon solujen numerot kuvastavat prosentuaalisia mahdollisuuksia, jotka vaikuttavat siihen mikä vakavuusaste tähän oireeseen arvotaan generaattoreissa. Taulukon ensimmäinen solu vasemmalta luettuna on vakavin tapaus. Aiheiden vakavuusasteita pystytään näin tasapainotamaan ja säätämään aiheita paremmin jokaisen sairauden kohdalla erilaiseksi. Kuvion 11 flunssaesimerkissä nähdään, kuinka sillä on aika korkeat mahdollisuudet kuumeelle ja muille flunssalle tyypillisille oireille.

```

{
  "name": "flu",
  "topics": [
    ["exposure", [10,60,30]],
    ["fever", [40,55,5]],
    ["neckPain", [3,5,92]],
    ["stomachache", [5,10,85]],
    ["headache", [35, 55,10]],
    ["cough", [35, 55,10]],
    ["medicalConditions", [10,10,80]],
    ["dyspnea", [5,10,70]],
    ["pulse", [2,5,93]],
    ["rash", [2,5,93]],
    ["wounds", [2,5,93]],
    ["petechiae", [2,5,93]],
    ["urinary", [2,5,93]],
    ["soreThroat", [40,50,10]],
    ["muscleAche", [30,60,10]],
    ["congestion", [40,50,10]],
    ["nausea", [3,6,91]],
    ["shivering", [33,33,34]],
    ["soreThroat", [15,60,25]],
    ["medicine", [33,33,34]],
    ["consume", [0,5,95]]
  ]
},

```

Kuvio 11. Flunssa ja sen aiheet

Asiakkaalle yhteydenottosyyin aiheiden vakavuudet generoidaan tämän jälkeen. Joissain aiheissa voi olla jo asiakkaan luomisvaiheessa joitain lisäominaisuuksia. Näiden ominaisuuksien luomiseen käytetään generaattoreita, joista saadaan tarkempaa ja yksilöllistä tietoa aiheeseen liittyen. Esimerkiksi kuumeelle arvotaan useita tietoja, kuten lämpötila, kesto, onko mitannut, kuumeen tila.

Kuviossa 12 on esitetty `genParalysis`-funktio. Se ottaa vastaan sille arvotun vakavuusasteen. Funktiossa on ensin halvaustaulukkoon lisätty eri sijainteja, missä halvaus voi kehossa sijaita, kuten tässä tapauksessa kasvoissa tai vartalossa. Tämän jälkeen luodaan `paralysis`-objekti, jolle annetaan vakavuusaste ja halvaustaulukosta arvottu sijainti. Tällä tavalla halvauksen voimakkuus ja sijainti voivat vaihdella eri asiakkailla.

```

const genParalysis = (severity) => {
  const paraLocation = ["kasvoissa", "vartalossa"];

  const paralysis = {
    severity: severity,
    location: randomElement(paraLocation)
  };
  return paralysis;
};

```

Kuvio 12. Halvaus-aiheen generointi

4.4 Lauseen käsittely

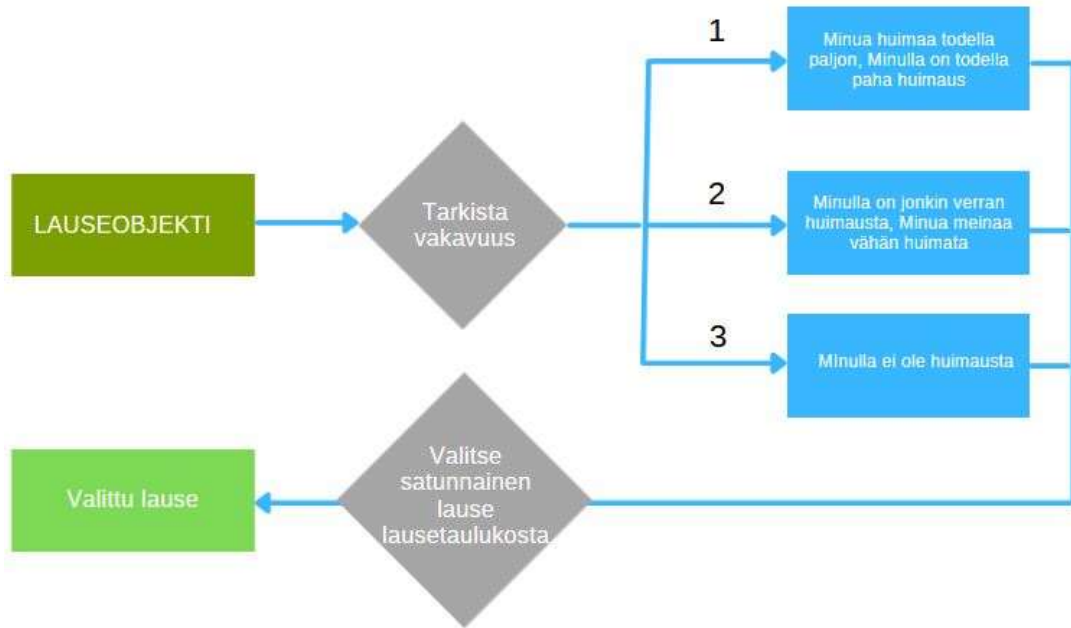
Aiheet on kuvattu Data\Topics-kansion alla olevissa JSON-tiedostoissa. Näiden aiheiden sisällä on erilaisia lauseobjekteja, joilla kullakin on erilaisia ominaisuuksia, käyttötarkoituksia ja avainsanoja. Kuviossa 13 näkyy esimerkki koodissa olevasta Huimaus-aiheen lauseobjektista.

```
{
  "phrase": {
    "1": ["Minua huimaa todella paljon", "Minulla on todella paha huimaus"],
    "2": ["Minulla on jonkin verran huimausta", "Minua meinaa vähän huimata"],
    "3": ["Minulla ei ole huimausta"]
  },
  "canStart": true,
  "reasons": [
    "arrhythmia",
    "motionSickness",
    "hypertension"
  ],
  "keywords": ["huimaa", "huima", "huimaus", "heijaa"]
},
```

Kuvio 13. Huimaus-aiheen lauseobjekti

Jokaiselle lauseobjektille määriteltiin kolme vakavuusastetta, jotka pitävät sisällään lausetaulukoita. Vakavuusasteissa numero 1 on vakavin, 2 on keskiasteen ja 3 on juuri alkanut tai aiheen kanssa ei ole ongelmaa. Lausetaulukon sisällä voi olla useita eri lauseita ja vastaus arvotaan näistä. Näin saadaan tuotua satunnaisuutta annettuihin vastauksiin.

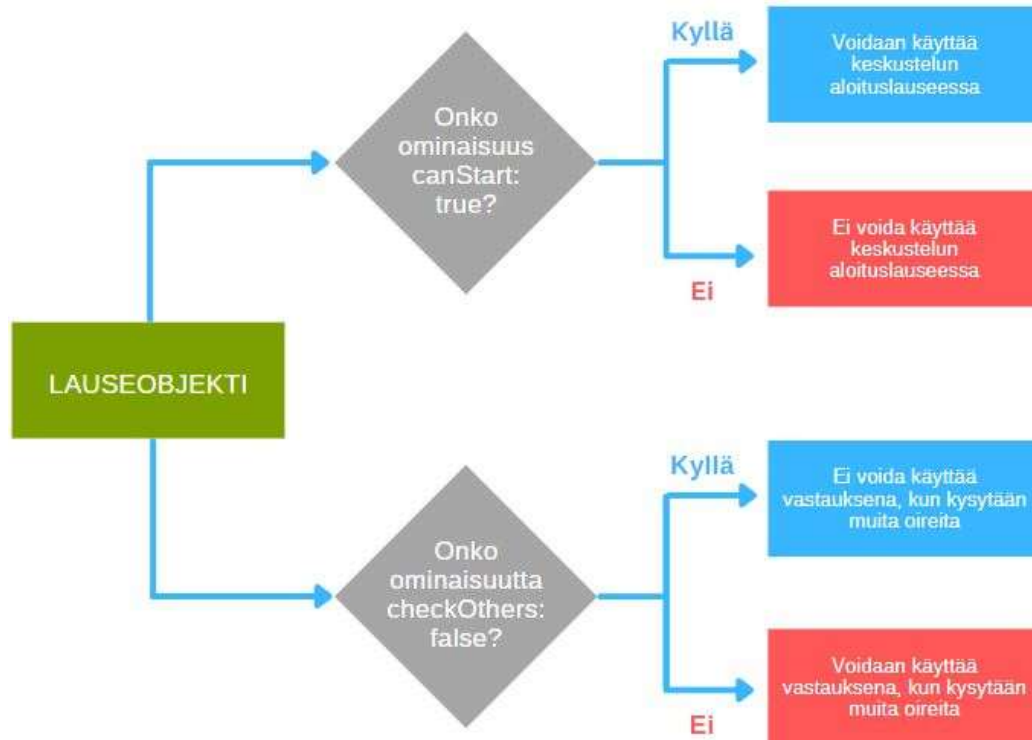
Valittu lause tarkastetaan checkRequest-funktiossa. Reasons-taulukkoon on merkitty yhteydenottosyyt, joissa lauseobjektia voidaan käyttää. Jos yhteydenottosyyntaulukossa on vain "all"-arvo, lauseobjektia voidaan käyttää kaikkien yhteydenottoyhteyksien yhteydessä. Kuviossa 14 näkyy lauseobjektin valinta vakavuusasteen perusteella.



Kuvio 14. Lauseobjektin valinta vakavuuden perusteella

Lauseobjektit sisältävät erilaisia käyttötarkoitusta määritteleviä ominaisuuksia, kuten `canStart: true` ja `checkOthers: false`. Kuviossa 15 on esitetty, kuinka sovelluksen algoritmi käsittelee ja tekee johtopäätöksen valitun lauseobjektin suhteen.

Kun lauseobjektilla on ominaisuus `canStart: true`, niin tällöin keskustelun aloituslauseeseen pystytään lisäämään tämä lauseobjekti. Jos lauseobjektilla löytyy ominaisuutena `checkOthers: false`, niin algoritmi jättää tämän lauseen pois käytökelpoisista lauseobjekteista pelaajan kysyessä muista oireista. Tällä tavalla pyrittiin rajoittamaan epäloogisia vastauksia.



Kuvio 15. Lauseobjektin ominaisuuksien tarkastaminen

Joitakin lauseobjekteja ei haluta vastaukseksi pelaajan kysyessä muista oireista. Kuviossa 16 on esimerkki vain tiettyyn tilanteeseen sopivasta lauseobjektista, joka sisältää `checkOthers: false` -ominaisuuden.

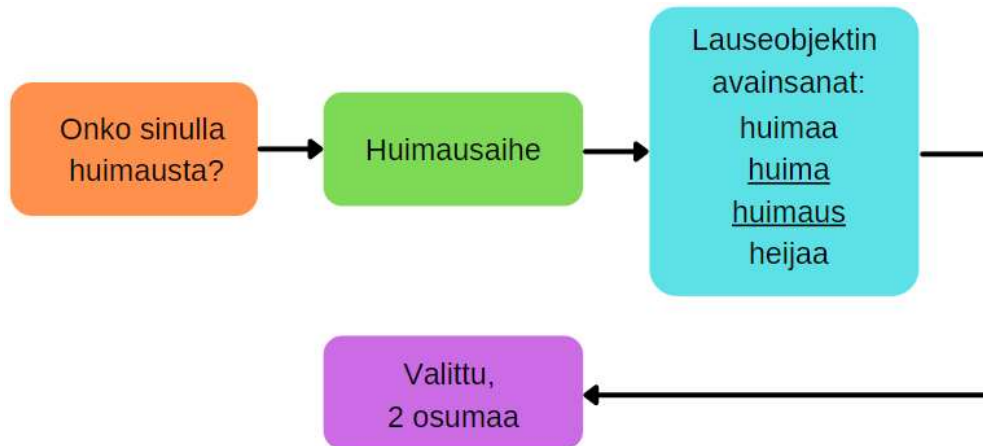
```

{
  "phrase": {
    "1": ["Oireet alkoivat {flu.started}"],
    "2": ["Oireet alkoivat {flu.started}"],
    "3": ["Ihan vasta alkoi"]
  },
  "checkOthers": false,
  "reasons": ["flu"],
  "keywords": ["milloin", "alkoi", "alanut"]
},

```

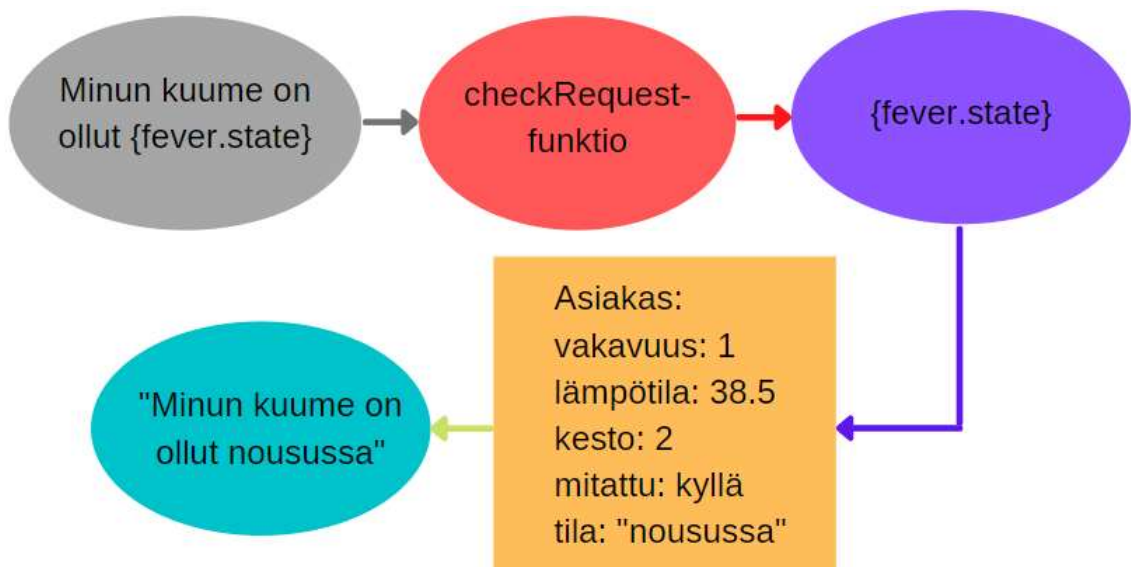
Kuvio 16. Lauseobjekti `checkOthers: false` -ominaisuudella

Lauseobjekti pitää sisällään Keywords-avainsanataulukon. Sen sisällä voi olla useita erilaisia avainsanoja tai lauseita, jotka liittyvät lauseobjektin käsittelemään aiheeseen. Algoritmi käy lävitse jokaisen lauseobjektin hakusanat ja laskee avainsanojen osumien lukumäärät kustakin lauseobjektista. Jos lauseobjekti täyttää kaikki muut hakukriteerit ja sillä on korkein määrä osumia avainsanoissa, niin tällöin se tulee valituksi lauseobjektiksi. Kuviossa 17 on esimerkkinä kuvattu lauseobjektin valinta, kun pelaaja kysyy huimauksesta tietoa.



Kuvio 17. Lauseobjektin valinta avainsanojen perusteella

Lauseet voivat pitää sisällään aaltosulkeiden sisään kirjoitettuja kutsuja, joiden avulla asiakkaalta voidaan checkRequest-funktiolla hakea tietoja tai korvata se täytelauseilla. Esimerkiksi lauseessa *"Minun kuume on ollut {fever.state}"* aaltosulkeet sisältöineen korvataan asiakkaan kuume-aiheen tiedoilla kuvion 18 mukaisesti.



Kuvio 18. Lauseen kutsun käsittely

4.5 Pelinäköymä

Pelin alunäköymä (Kuvio 19) on rakenteeltaan yksinkertainen ja pitää sisällään näppäimen, jolla pelaaja voi ottaa vastaan seuraavan puhelun. Näppäimen pe-

rässä oleva numero kertoo jonossa olevien asiakkaiden määrän. Ruudun yläosassa näkyy senhetkinen viikonpäivä. Kymmenen asiakkaan käsittelyn jälkeen vuorokausi vaihtuu ja uudet asiakkaat generoidaan.



Kuvio 19. Pelin alkunäkymä

Puheluun vastatessa peli siirtyy keskustelunäkymään. Aloituslauseet generoidaan automaattisesti, minkä jälkeen pelaaja voi keskustella tekoälyasiakkaan kanssa kirjoittamalla alhaalla olevaan syötekenttään.

Kuviossa 20 on esitetty esimerkkikeskustelu asiakkaan kanssa. Pelaajan valitessa Hoida kotona -näppäimen pelin keskustelutapahtuma loppuisi ja asiakkaalle annetaan loppulauseena automaattisesti viesti *"Kuulostaa siltä, että pärjät vielä kotona. Soita huomenna uudelleen, jos olo ei rupea helpottamaan."* Kuvan tapauksessa asiakkaan oireet kuulostavat vakavilta, joten on järkevintä antaa vastaanottoaika. Ajanvarauskalenteriin siirrytään painamalla Anna aika -näppäintä.

HEALTHCARE CUSTOMER SERVICE

Maanantai

Puh: 0505983014
Kesto: 0:02:46

Terveyskeskuksen ajanvaraus, kuinka voin auttaa?

Terve, minun nimi on Anneli Kokko ja minun HETU on 280148-688E. Minulla on todella paha pääkipu.

Onko pitkään ollut pää kipeänä?

Pääkipua on kestänyt 1 päivää.

Onko muita oireita?

Minulla on todella paha niskasärky ja kirkas valo häikäisee jonkin verran.

Onko kuumetta?

Joo, on minulla 39.7 astetta lämpöä.

Onko kuume noussut nopeasti?

Kuume on noussut korkealle nopeasti ja yllättäen.

Kirjoita tähän, esim. "Onko sinulla kuumetta?"

LÄHETÄ

ANNA AIKA

HOIDA KOTONA

Kuvio 20. Keskustelunäkymä

Pelaaja voi antaa asiakkaalle ajan kalenterista jollekin viikonpäivälle ja kellonajalle. Kuviossa 21 näkyy ajanvarauskalenteri, jossa maanantaipäivän kello 10 ja kello 12 ajat ovat varattuja. Pelaaja on parhaillaan valitsemassa kello 14 aikaa. Valinnan jälkeen peli siirtyy takaisin keskustelunäkymään ja siinä asiakkaalle kerrotaan lopuksi päätetty vastaanottoaika ja puhelutapahtuma päättyy.

Maanantai	Tiistai	Keskiviikko	Torstai	Perjantai
08:00		vapaa		
09:00		vapaa		
10:00		141088-493X		
11:00		vapaa		
12:00		080454-288I		
13:00		vapaa		
14:00		vapaa		
15:00		vapaa		

SULJE

Kuvio 21. Ajanvarauskalenteri

5 TESTAUS

Testauksella pyrittiin kartoittamaan erilaisia pelaajan syöttämiä avainsanoja, jotka tuottavat potilailla erilaisia vastauksia sairaudesta ja aiheesta riippuen. Testausta tapahtui kolmessa eri vaiheessa. Ensimmäisessä vaiheessa testaaminen tapahtui projektin tekijöiden kesken. Toisessa ja kolmannessa vaiheessa eli varsinaisilla testikierroksilla pelistä jaettiin toimivaa versiota myös projektin ulkopuolisille jäsenille. Testaukseen osallistuminen oli vapaaehtoista, ja tietoa kerättiin anonyymisti. Testauksessa kävi myös ilmi, että peli oli vaikeampi terveydenhuoltoalan ulkopuolisille henkilöille, jotka eivät osanneet kartoittaa sairauksia ja oireita oikeilla terveydenhuoltoalan käyttöön soveltuvilla termeillä.

5.1 Alkuvaiheen testaus

Alkuvaiheessa tehtäviini kuului erityisesti pelin testaamiseen liittyviä asioita. Lauseobjektien rakenteet muuttuivat alkuvaiheessa moneen kertaan ja tähän vaikutti esimerkiksi se, että asiakkaiden persoonallisuus päätettiin rajata neutraaliksi ja pitää lauseiden ulkoasu eettisenä.

Yhteydenottosyiden ja lauseiden tiedostorakenne muutettiin JSON-muotoon, sillä se toimii paremmin tiedon säilömiseen ja käyttöön valittu MongoDB-tietokanta tukee suoraan JSON-rakennetta. Aiheet olivat pääasiassa oireita, jotka oli merkitty suomen kielellä. Jokaiselle lauseelle oli merkitty oma vakavuutensa. Lisäksi alkuvaiheessa kaikki lauseet olivat yhdessä tiedostossa saman taulukon sisällä. Kuviossa 21 näkyy alkuvaiheen rakenne lausetaulukosta.

```

allPhrases=[
  {
    phrase: "Olin viime viikolla ulkomailla ja nyt perse laulaa!",
    symptoms: "oksentelu",
    severity: 1,
    reasons: "ruokamyrkytys"
    "keywords": ["ulkomaa", "maassa", "toisessa", "käynyt"],
  }
  {
    phrase: "Olin viime viikolla ulkomailla ja nyt oksettaa koko ajan.",
    symptoms: "oksentelu",
    severity: 1,
    reasons: "ruokamyrkytys"
    "keywords": ["ulkomaa", "maassa", "toisessa", "käynyt"],
  }
  {
    phrase: "Olin viime viikolla ulkomailla ja nyt perse laulaa!",
    symptoms: "ripuli",
    severity: 1,
    reasons: "ruokamyrkytys"
    "keywords": ["ulkomaa", "maassa", "toisessa", "käynyt"],
  }
]

```

Kuvio 21. Alkuvaiheen lausetaulukon rakenne

Lauseiden sisälle oli liitetty erilaisia funktiokutsuja ja vakavuusasteita, joita hyödynnettiin tietojen hakemisessa. Kuviossa 22 näkyvät lauseen sisällä olevat ``${}`-merkit, joiden sisälle kutsuttava funktio oli merkitty.

```

{
  phrase: `Olen pystynyt syömään ${generateConsumeAmount(severity)}.`,
  symptoms: ["ripuli", "oksentelu"],
  severity: [2, 3],
  reasons: ["vatsatauti", "ruokamyrkytys", "ripuli"],
  keywords: ["pystynyt", "ruoka", "syödä", "syönyt", "syöminen"],
},

```

Kuvio 22. Alkuvaiheen funktiokutsut

Lauseiden lisääntyessä huomattiin, että lausetaulukon käsittely yhdessä tiedostossa alkoi käymään todella haastavaksi. Tästä johtuen päädyttiin siihen, että jokainen lause lajiteltiin eri tiedostoihin aiheiden perusteella ja niiden sisään rakennettiin eri lauseobjekteja. Lisäksi eri vakavuusasteiden alle rakennettiin omat lausetaulukonsa. Tällä tavalla yhteen lauseobjektiin saatiin lisättyä kaikki vakavuusasteet ja niihin liittyvät lausetaulukot. Aiheiden ja yhteydenottoesityden nimissä päädyttiin käyttämään lääketieteellisiä ja englanninkielisiä termejä, jotta ne

olisivat helposti standardisoitavissa eri kielille. Kuviossa 23 näkyy lauseobjektin uusi rakenne.

```
{
  "phrase": {
    "1": ["Minulla on vaikea olla kirkkaassa valossa", "Kirkas valo häikäisee pahasti", "Kirkas valo tuottaa kipua silmissä"],
    "2": ["Minulla on jonkin verran vaikeaa kirkkaan valon kanssa", "Kirkas valo häikäisee jonkin verran"],
    "3": ["Ei ole ongelmaa sen kanssa"]
  },
  "reasons": ["meningitis"],
  "keywords": [
    "valo",
    "valon",
    "arkuu",
    "arkuutta"
  ]
}
```

Kuvio 23. Uusi lauseobjektin rakenne

Lauseiden sisältämät funktiokutsut muuttuivat rakenteeltaan siten (Kuvio 24), että niillä viitattiin suoraan asiakkaalle määritettyyn arvoon tai haluttuun täytelauseeseen. Samalla sulkeiden edestä poistettiin tarpeettomana \$-merkki.

```
"phrase": {
  "1": [
    "Minulla on {fever.temp} astetta lämpöä",
    "Joo, on minulla {fever.temp} astetta lämpöä"
  ],
  "2": [
    "Minulla on {fever.temp} astetta lämpöä",
    "Joo, on minulla {fever.temp} astetta lämpöä"
  ],
}
```

Kuvio 24. Lauseiden uudet funktiokutsut

Alkuvaiheiden rakennemuutoksien jälkeen koodin ja lauseobjektien rakenne todettiin riittävän hyväksi. Testaaminen oli aikaa vievää, vaikka varsinainen korjauksien tekeminen tapahtui melko nopeasti. Kattavamman testidatan saamiseksi tarve ulkopuolisille testaajille lisääntyi ja heitä varten rakennettiin testinäkömä ja mahdollisuus antaa palautetta.

5.2 Testikierroksien palautteen antaminen

Pelin alunäkymässä pelaajalle esiteltiin testikierroksen tarkoitus ja tavoitteet (Kuvio 25). Pelaajalle kerrottiin testidatan tallentamiseen ja käyttöön liittyvät asiat. Alkunäkymässä oli mahdollista jättää myös vapaamuotoista palautetta.

HEALTHCARE CUSTOMER SERVICE

Maanantai

OTA SEURAAVA PUHELU (10)

HARJOITTELE PUHELINVASTAANOTTOA!

Tervetuloa pelin testikierrokselle! Olet terveyskeskuksen puhelinpäivystyksessä työntekijänä. Sinun tehtävänäsi on vastata puhelimeen ja arvioida soittavien asiakkaiden hoidon tarvetta. Jos koet, että asiakas tarvitsee ajan, niin voit antaa hänelle ajan ajanvarauskalenterista painamalla "Anna aika" nappia. Jos taas koet, että asiakas pärjää vielä oireiden kanssa kotona, voit painaa "Hoida kotona" nappia.

Tällä testikierroksella olemme erityisesti kiinnostuneita kuulemaan, että miten keskustelu asiakkaan kanssa on sujunut. Jokaisen keskustelun lopussa pyydämme sinua valitsemaan, että menikö keskustelu hyvin vai menikö jotain vikaan, eli oliko keskustelu jollain tapaa outo.

Emme tallenna mitään käyttäjätietoja, vaan pelkästään asiakkaan kanssa käydyt keskustelut ja nekin tuhoetaan viimeistään opinnäytetyöprojektin päättyttyä.

Voit myös jättää vapaasti palautetta kirjoittamalla alapuolella olevaan tekstikenttään ja sen jälkeen painamalla "Lähetä", kiitos!

Anna palautetta tähän ja paina lähetä.

LÄHETÄ

Kuvio 25. Pelin alkunäkymä

Jokaisen keskustelun päättyttyä pelaajalta kysyttiin palautetta juuri käydystä keskustelusta. Kuviossa 26 näkyvät käyttöliittymässä esitetyt palautepainikkeet, joilla pelaaja voi oman tuntemuksensa mukaan merkitä, miten hänen mielestään keskustelu sujui, eli oliko se looginen tai menikö jotain vikaan.

ANNA PALAUTETTA, MITEN KESKUSTELU MIELESTÄSI SUJUI?

KAIKKI MENI HYVIN

JOTAIN MENI VIKAA

Kuvio 26. Palautevalinta keskustelun päättyessä

5.3 Palautteen käsittely DbViewissä

Testausta ja palautteen käsittelyä varten oli rakennettu oma DbView-näkymä, johon normaalilla pelaajalla ei ollut pääsyä. DbView oli yhteydessä tietokantaan, johon kaikki keskustelut ja niiden palautteet tallentuivat. Kuviossa 27 on kuvattu DbView-näkymä, josta testaaja voi tarkastella kaikkien asiakkaiden kanssa käytyjä keskusteluita. Näkymä piti sisällään painikkeet, joilla keskusteluja pystyi lajittelemaan ongelman, päivämäärän tai yhteydenottosyyntä mukaan.

Healthcare Customer Service

SET FIXED
SHOW PROBLEMS
SHOW ALL

<input type="checkbox"/>	ID	Customer	Added	Problem	Reason
<input type="checkbox"/> >	181176-331P	Oliver	19.4.2021 kl...	false	panicDisor...
<input type="checkbox"/> >	041133-184H	Linda	19.4.2021 kl...	false	meningitis
<input type="checkbox"/> >	200832-587G	Pekka	19.4.2021 kl...	false	meningitis
<input type="checkbox"/> >	071187-771V	Santeri	19.4.2021 kl...	false	arrhythmia
<input type="checkbox"/> >	290863-249P	Miikka	19.4.2021 kl...	false	flu
<input type="checkbox"/> >	140774-420D	Helena	19.4.2021 kl...	false	arrhythmia
<input type="checkbox"/> >	220741-621F	Tomi	19.4.2021 kl...	false	hypertension
<input type="checkbox"/> >	240725-231P	Miika	19.4.2021 kl...	false	arrhythmia
<input type="checkbox"/> >	071163-734V	Päivi	19.4.2021 kl...	false	gastroenter...
<input type="checkbox"/> >	110593-622...	Helena	19.4.2021 kl...	false	hypertension

Rows per page: 10 ▾ 51-60 of 506 |< < > >|

Kuvio 27. DbView-hallintanäkymä

Jos testaaja oli löytänyt ongelman keskustelusta, näkyy se hallintanäkymän Problem-sarakkeessa arvona true. Tämä helpotti ongelmien löytämistä käydyistä keskusteluista. Yleisimmät pelaajan raportoimat syyt olivat ongelmat lauseen ymmärtämisessä, mikä johtui aiheiden ja avainsanojen puutteesta.

Kuviossa 28 näkyy DbViewissä avoinna oleva ongelmalliseksi merkitty keskustelu. Keskustelussa näkyy hyvänä esimerkkinä yleinen ongelma, jossa tarvittavaa keskusteluaihetta (nukkuminen) ja siihen viittaavia avainsanoja ei vielä löytynyt.

<input type="checkbox"/>	▼	040592-409N	Matti	26.4.2021 kl...	true	flu
requestStartingPhrase						
Tervehdys, minun nimi on Matti Nevalainen ja minun henkilötunnus on 040592-409N. Olen yskinyt paljon.						
Onko muita oireita?						
Minulla tulee jonkin verran kylmiä väreitä ja minun kuume on ollut laskussa.						
Milloin kuume alkoi?						
Yskää on kestänyt 5 päivää.						
Häiritseekö yskeminen nukkumista?						
Olen yskinyt paljon.						
Kuulostaa siltä, että pärjäilet vielä kotona. Soita huomenna uudelleen, jos olo ei rupea helpottamaan.						
Ok, tehdään niin. *click*.						

Kuvio 28. Ongelmalliseksi merkitty keskustelu

5.4 Ensimmäinen testikierros

Ensimmäisellä testikierroksella tekoälyn lausegenerointia testattiin noin viiden lähipiiristä löytyneen testaajan voimin. Testikierroksen loppupuolella keskusteluja oli DbViewissä noin 300. Tämän testikierroksen yhteydessä pelin käyttöliittymästä tehtiin responsiivisempi, jotta se toimisi paremmin myös älypuhelimilla ja eri kokoisilla näyttöpäätteillä.

Testikierroksella nousi esiin ongelma, jossa vahingossa lauseeseen näppäily erikoismerkki aiheutti sovelluksen kaatumisen. Tämä ratkaistiin siten, että syötteen käsittelyn alkuun lisättiin replace-metodi (Kuvio 29), joka poistaa erikoismerkit pelaajan antamasta syötteestä.

```
const handleInput = (text, id) => {
  // Remove all special marks from input
  text.replace(/[^a-zA-Z0-9 ]/g, "");
}
```

Kuvio 29. Erikoismerkkien poistaminen

Toinen esille tullut haaste liittyi eri testaajien käyttämiin murteisiin ja puhekieleen. Tämä huomioitiin siten, että avainsanoihin lisättiin osittaisia sanoja ja synonyymejä parantamaan avainsanojen vastaavuuksia (Kuvio 30). Lauseen tarkastamisessa käytetty RegExp-menetelmä kykenee käsittelemään myös osittaisia sanoja.

```
"keywords": ["yskä", "kauanko", "kauan", "kaua", "pitkään", "pitkä", "jatkunut", "jatku", "milloin", "millo", "alkanut", "alkanu", "alkoi", "kestänyt", "kestäny"]
```

Kuvio 30. Osittaisia sanoja avainsanataulukossa

Testikierroksen aikana havaittiin myös tarve aiheille, jotka koskevat jokaista asiakasta. Lauseobjekteihin ja näiden käsittelyyn tehtiin muutoksia, missä lauseobjektin yhteydenottosyyksi kirjoitettiin "all". Tällä tavalla merkityt lauseobjektit ovat käyttökelpoisia kaikille tekoälyasiakkaille. Yleisiä aiheita olivat esimerkiksi syömiseen, juomiseen, lääkitykseen ja perussairauksiin liittyvät aiheet. Kuviossa 31 näkyy lääkitykseen liittyvä esimerkki.

```
{
  "phrase": {
    "1": ["Tuntuu että lääkkeitä ei ole ollut apua"],
    "2": ["Lääkkeistä on ollut jonkin verran apua"],
    "3": ["Kaikki on toiminut oikein hyvin"]
  },
  "checkOthers": false,
  "reasons": ["all"],
  "keywords": ["ottanut", "käyttänyt", "vaikutusta", "lääke", "lääki", "lääkke", "muistanut", "lääkitys", "särkylääke", "auttaako"]
}
```

Kuvio 31. Yleinen lääkitysausaihe

5.5 Toinen testikierros

Ensimmäisen testikierroksen korjauksien jälkeen peli annettiin testattavaksi toimeksiantaja Outi Tierannalle, joka vei sen testattavaksi verkostolleen. Toisen testikierroksen edettyä viikon verran, tarkasteltavia keskusteluja testinäkömässä oli noin 500. Tällä testikierroksella lisäyksen tarvetta tuli lähinnä puuttuviin aiheisiin ja näiden avainsanoihin.

Testikierroksen aikana tehdyissä korjauksissa moniin aiheisiin käytettiin väliaikaisia oletuslauseita, millä saatiin jonkinlainen looginen vastaus. Laadukkaampi aiheen käsittely olisi tarvinnut yksilöllisten generaattoreiden rakentamista, joka olisi vienyt enemmän aikaa. Puuttuvia aiheita olivat esimerkiksi valonherkkyys, migreeni ja näkeminen.

6 POHDINTA

Työn tarkoituksena oli kehittää terveyskeskuksen puhelinpäivystykseen prototyyppi selainpelistä osana OmaDigi-hanketta. Mielestäni työn tarkoitus ja tavoitteet saavutettiin onnistuneesti. Pelaaja pystyy toimimaan puhelimeen vastaavan hoitajan roolissa ja keskustelemaan asiakkaan kanssa. Tekoäly kykenee muodostamaan satunnaisgeneroinnin avulla pääosin loogisia ja järkeviä lauseita, joissa on hyödynnetty asiakkaaseen liitettyjä ominaisuuksia. Testaajilta saadun palautteen perusteella prototyyppiin ja lausegenerointiin oltiin pääosin tyytyväisiä.

Tiukkojen aikataulujen vuoksi testaamiseen käytetty aika oli melko rajallisia. Eriyisesti toiselta testikierrokselta saatiin määrällisesti suhteellisen vähän palautetta. Tämän huomioiden alussa tehdyt rajaukset yksinkertaisesta käyttöliittymästä, sairauksien rajaamisesta ja neutraalista linjasta olivat järkeviä.

Työn toteutukseen käytetyt työkalut soveltuivat tarkoitukseen hyvin. JavaScript ja React toimivat odotetusti hyvin selainpelin kehitykseen. Gitin ja GitLabin avulla yhdessä sovelluksen kehittäminen oli sujuvaa, vaikkakin GitLab välillä hidasteli paljon. Discord toimi pienen ryhmän kanssa hyvin yhteydenpitoon.

Testaamisen näkökulmasta lausegeneroinnissa on vielä jatkokehityksen tarvetta. Keskustelun aloituslause voisi pitää sisällään useita eri aiheita ja näiden tulisi olla järkevämmin priorisoitu. Esimerkiksi asiakas kertoisi korkeasta kuumeesta osana aloituslausetta. Samaa aiheen priorisointia voitaisiin hyödyntää myös muista oireista kysyttäessä. Testatessa selvisi myös, että ilman terveydenhuoltoalan kokemusta omaavilla testaajilla oli selvästi haastavampaa keksiä oikeanlaisia kysymyksiä hoidon tarpeen arvioimiseen. Toisaalta vapaamuotoinen syöte on opetuskäyttöön suunnatussa pelissä hyvä ratkaisu, mutta oppimista voitaisiin tukea kehittämällä peliin vaikeusasteita esimerkiksi tarjoamalla pelaajalle valmiita kysymysvaihtoehtoja.

Peliin voitaisiin lisätä enemmän pelillisiä elementtejä, kuten pelaajan menestystä arvioivia pisteytyksiä, rajallisia hoitoaikoja ja vaihtelevia pelitapahtumia. Opinnäytetyön kirjoitushetkellä pelaaja ei myöskään saa palautetta siitä, oliko hoidon tar-

peen arvio onnistunut. Näillä saataisiin lisättyä houkuttelevuutta pelin pelaamiseen ja puhelinpäivystyksen harjoitteluun. Terveystieteiden opetuskäyttöön suunnatussa pelissä voitaisiin myös huomioida vanhempien tekoälyasiakkaisen kasvavat riskit sairauksiin. Lisäksi peli voisi huomioida paremmin realistiset todennäköisyydet eri sairauksille ja niiden oireille.

Jatkokehityksessä voitaisiin myös hyödyntää erilaisia koneoppimisen menetelmiä ja valmiita NLP-ratkaisuja osana tekoälyn lauseen käsittelyä. Tällaisia menetelmiä voisi olla esimerkiksi TensorFlow ja Googlen Natural Processing API.

Opinnäytetyön aihe oli mielenkiintoinen, vaikka minulla ei ollut aikaisempaa kokemusta hoidon tarpeen arvioinnista. Pidemmällä kehitysajalla ja testaamisella pelistä saataisiin hyvä työkalu hoitoalan opetuskäyttöön ja harjoitteluun.

LÄHTEET

Bojinov, V. 2018. RESTful Web API Design with Node.js 10 - Third Edition. Packt Publishing. Viitattu 18.5.2021 <https://luc.finna.fi/lapinamk/>, ProQuest Ebook Central.

CACM Staff 2017. Artificial intelligence. Communications of the ACM, no. 2. Viitattu 20.5.2021 <https://doi.org/10.1145/3034429>.

Clark, A., Fox, C. & Lappin, S. 2013. The handbook of computational linguistics and natural language processing. Wiley-Blackwell. Viitattu 19.5.2021 <https://luc.finna.fi/lapinamk/>, ProQuest Ebook Central.

Chiarelli, A. 2018. Beginning React: Simplify Your Frontend Development Workflow and Enhance the User Experience of Your Applications with React. Packt Publishing. E-kirja. Viitattu 25.4.2021 <https://luc.finna.fi/lapinamk/>, ProQuest Ebook Central.

Delfino, D. & Dean, G. 2021. What is Discord? A guide to the popular group-chatting app. Viitattu 25.4.2021 <https://www.businessinsider.com/what-is-discord?r=US&IR=T>.

Deshpande, A. & Kumar, M. 2018. Artificial Intelligence for Big Data. Packt Publishing. E-kirja. Viitattu 18.5.2021 <https://luc.finna.fi/lapinamk/>, ProQuest Ebook Central.

Gupta, N. & Mangla, R. 2020. Artificial intelligence basics. Mercury Learning and Information. E-kirja. Viitattu 18.5.2021 <https://luc.finna.fi/lapinamk/>, ProQuest Ebook Central.

Hethey, J. 2013. GitLab Repository Management. Packt Publishing. E-kirja. Viitattu 25.4.2021 <https://luc.finna.fi/lapinamk/>, ProQuest Ebook Central.

Jeansoulin, R. 2018. JavaScript and Open Data. London, UK: Hoboken. E-kirja. Viitattu 24.4.2021 <https://luc.finna.fi/lapinamk/>, ProQuest Ebook Central.

Kanagaraj, V., Navaneetha, K. & Muhammed, Y. 2020. A study of Human Resources Development through Chatbots using Artificial Intelligence. Viitattu 18.5.2021 <https://doi.org/10.1145/3034429>.

Laster, B. 2016. Professional Git. John Wiley & Sons. E-kirja. Viitattu 25.4.2021 <https://luc.finna.fi/lapinamk/>, ProQuest Ebook Central.

Mustafeez, A. 2021. What is Visual Studio Code. Viitattu 25.4.2021 <https://www.educative.io/edpresso/what-is-visual-studio-code>.

Shuguang, L., Zheng, L. & Lin, B. 2020. Impact of Artificial Intelligence 2.0 on Teaching and Learning. Viitattu 20.5.2021 <https://doi.org/10.1145/3383923.3383928>.

Sriparasa, S. & D'mello, B. 2018. JavaScript and JSON Essentials. Packt Publishing. E-kirja. Viitattu 24.4.2021 <https://luc.finna.fi/lapinamk/>, ProQuest Ebook Central.

Solutions, E. 2018. Learn Node.js by Building 6 Projects. Packt Publishing. Viitattu 20.5.2021 <https://luc.finna.fi/lapinamk/>, ProQuest Ebook Central.

Valtiovarainministeriö 2020. Julkisen hallinnon digitalisaatio. Viitattu 27.4.2021 <https://vm.fi/digitalisaatio>.

Vats, R. 2020. GitHub vs GitLab: Difference Between GitHub and GitLab. Viitattu 25.4.2021 <https://www.upgrad.com/blog/github-vs-gitlab-difference-between-github-and-gitlab/>.