Saavutettavan reseptimobiilisovelluksen kehittäminen

Mari Luostarinen

Haaga-Helia ammattikorkeakoulu

Tradenomi

Opinnäytetyö

2024

Tiivistelmä

Tekijä
Mari Luostarinen

Tutkinto
Tradenomi

Opinnäytetyön nimi
Saavutettavan reseptimobiilisovelluksen kehittäminen

Sivu- ja liitesivumäärä
36 + 3

Opinnäytetyön tarkoituksena oli tuottaa saavutettava reseptimobiilisovellus, tutustua saavutettavuuteen käsitteenä, testata sovellus saavutettavuuskriteereillä sekä testata osa sovelluksesta käyttäen apuna digitaalista apuvälinettä, ruudunlukijaa. Työ perustuu tekijän omaan mielenkiintoon eikä sillä ole ulkoista toimeksiantajaa. Mobiilisovelluksen lähdekoodi julkaistaan GitHubissa. Projekti toteutettiin kevään 2024 aikana.

Työ alkaa esiselvityksestä, jossa kartoitetaan olemassa olevia reseptimobiilisovelluksia ja niiden toimintoja. Lisäksi käydään läpi yleisimmät käyttöliittymien taustalla olevat tekniset ratkaisut, rajapinta (API) ja tietokanta. Esiselvityksen perusteella tekijä päätyi käyttämään käyttöliittymän taustalla avointa rajapintaa. Valmiiden reseptipalvelujen toiminnallisuuksista sai taustaa vaatimusmäärittelyihin.

Opinnäytetyön tietoperustassa esitellään digitaalisen palvelun saavutettavuuskriteerit ja niitä ohjaavaa lainsäädäntöä ja kansainvälistä kriteeristöä. Saavutettavuutta käsitellään lisäksi mobiililaitteen ja ruudunlukijan näkökulmasta. Tietopohja jatkuu React Native -sovelluskehyksen ja kehitysympäristön esittelyllä. Omissa alaluvuissaan on esitelty React Nativen saavutettavuusominaisuudet ja muita React Nativen saavutettavuutta tukevia komponentteja.

Mobiilisovelluksen kehitys on dokumentoitu vaiheittain. Vaatimusmäärittelyssä käydään läpi käyttötapaukset, suunnitteluvaiheessa käyttöliittymän ja testauksen suunnittelu ja toteutusvaiheessa mobiilisovelluksen rakentaminen. Sovelluksen testaus rajattiin saavutettavuuskriteereillä testaamiseen ja etusivun toimintojen testaamiseen ruudunlukijalla.

Projekti pysyi aikataulussa ja tuotos onnistui suunnitellusti. Tekijä olettaa, että huomattava osuus onnistumisessa oli ketterän kehityksen menetelmän ja pienin toimiva tuote -periaatteen (MVP) noudattamisella. Jatkokehittämiskohteina on laajentaa mobiilisovelluksen toiminnallisuuksia, parantaa entisestään sovelluksen saavutettavuutta ja varmistaa, että sovellus toimii myös ulkoista näppäimistöä käytettäessä.

Asiasanat
Mobiilisovellus, saavutettavuus, ruudunlukija, React Native, rajapinta

Sisällys

1 Johdanto

Opinnäytetyön tavoitteena on tuottaa Android-alustalle mobiilisovellus, jossa käyttäjä pystyy hakemaan erilaisia ruokaohjeita. Työn tarkoitus on vastata arkipäiväiseen haasteeseen ja helpottaa päivittäistä ruoanlaittoa sekä tutustua saavutettavuuteen niin käsitteenä kuin React Nativen tarjoaman kirjaston kautta. Saavutettavuudella haetaan yhdenvertaisuuden toteutumista digitaalisissa palveluissa ja Suomessa lainsäädäntö on laajenemassa vuonna 2025 edellyttäen, että kaikki verkkokaupat täyttävät WCAG-kriteerit (Web Content Accessibility Guidelines) (Aluehallintovirasto 2024a).

Mobiilisovelluksen kohderyhmä on ihmiset, jotka hyötyvät digitaalisesta avusta ruoanlaitossa. Mobiilisovellus on helposti ja nopeasti käyttöönotettavissa, onhan jokaisella yleensä puhelin käden ulottuvilla. Käyttäjän tarvitsee vain avata sovellus ja hakea jotain tiettyä ruokaohjetta tai selailla vaihtoehtoja yhdestä paikasta omassa sijainnissaan.

Työhön kuuluu esiselvitys, määrittelyt, suunnittelu ja käytettävien teknologioiden valitseminen, sovelluksen tekeminen ja testaus sekä lähdekoodin julkaisu julkisessa versiohallintapalvelussa GitHubissa. Työ on toteutettu kevään 2024 aikana.

Opinnäytetyön alussa tutustutaan olemassa oleviin mobiilisovelluksiin, tekniikoihin ja tietovarastoihin käyttäen verkkohakua ja kirjaston palveluja. Mobiilisovelluksen tekemiseen käytetään JavaScript-pohjaista React Native -kehystä ja sen perusteisiin tutustutaan tarkemmin teoriaosassa. React Nativen valintaa tukee se, että sovelluskehyksestä löytyy saavutettavuutta tukevia komponentteja ja ominaisuuksia.

Saavutettavuuden kriteerit on koottu WCAG 2 -ohjeistukseen ja niillä on neljä pääperiaatetta: havaittavuus, hallittavuus, ymmärrettävyys ja toimintavarmuus. Kriteerit parantavat kaikkien käyttäjien käyttökokemusta ja sovelluksen selkeyttä. (Aluehallintovirasto 2024a).

Mobiilisovelluksen tietovarastona hyödynnetään Rapid API:n avointa rajapintaa Tasty API:a, koska palvelussa on valmiiksi paljon ruokaohjeisiin liittyvää dataa. Mobiilisovelluksen toiminnallisuudet rajataan lista- ja reseptin tieto -näkymiin sekä haku-toiminallisuuteen. Koodieditorina käytetään Visual Studio Codea, koska se on monipuolinen ja ennestään tuttu työväline. Uudempi teknologia on myös tehokkaampi käytöltään, mikä tukee vastuullisuutta. Projektin hallinnan apuna käytetään ketterän ohjelmistokehityksen menetelmiä.

Toimin itse loppukäyttäjän asemassa, jolloin toiminallisuudet ja käyttäjän tarpeet ovat helpommin tunnistettavissa. Tarkoitukseni on tutustua saavutettavuuteen ja oppia siitä mobiilisovelluksen kehitystyössä.

Ruudunlukijan käyttö on itselleni vierasta, joten opinnäytetyö onkin hyvä mahdollisuus oppia katsomaan teknistä toteutusta sen käytön näkökulmasta. Koska ruudunlukija-sovelluksen käyttö vaati opettelua ja sillä tehtävä testaus on aikaa vievää, rajataan saavutettavuustestaus ruudunlukijalla sovelluksen etusivuun, jossa on vieritettäviä komponentteja ja painikkeita. Muut saavutettavuuskriteerit testataan koko sovelluksessa. Sovellus testataan käyttäjän näkökulmasta eikä tässä vaiheessa toteuteta kattavampia testauksia tai käytetä testaustyökaluja.

Ajatus verkkopalvelujen saattamisesta kaikille ryhmille saavutettavaksi on innostavaa. Lisäksi sen tuoma hyöty kaikille käyttäjille antaa työlle lisäarvoa. Opinnäytetyössä laajentuu, syventyy ja samalla nivoutuu yhteen ohjelmointiosaaminen, tiedonhaku ja tekniikoiden käyttö sekä projektinhallinnan ja dokumentoinnin taidot.

Käsitteet

React Native	Avoimen lähdekoodin sovelluskehys
API	Ohjelmointirajapinta, jonka avulla sovellukset voivat keskustella keskenään – Application Programming Interface
Tietokanta	Tietovarasto, joka varastoi dataa palvelimelle ja pitää sen käyttäjille saatavilla
Saavutettavuus	Verkkopalveluiden käytön mahdollistaminen henkilöille, joilla on vammoja tai rajoitteita
WCAG-kriteerit	Kansainvälinen standardi verkkopalveluiden saavutettavuudesta – Web Content Accessibility Guideline
Ruudunlukija	Teknologia, joka tutkii ruudulla näkyvää tekstiä ja rakennetta sekä välittää tiedon kuunneltavassa muodossa tai pistekirjoituksena pistekirjoituspäätteelle
TalkBack	Android-järjestelmän ruudunlukijasovellus

2 Ennakkoselvitys

Tässä luvussa käydään läpi olemassa olevia ruoka-aiheisia mobiilisovelluksia ja niiden toiminnallisuuksia. Lisäksi tarkastellaan vaihtoehtoja mobiilisovelluksen käyttöliittymän taustaksi. Kehittäjien käyttöön on saatavilla avoimia rajapintoja, mutta toisaalta kehittäjällä on mahdollisuus rakentaa oma, räätälöity SQL- tai NoSQL-pohjainen tietovarasto.

2.1 Olemassa olevien reseptimobiilisovellusten kartoitus

Ohjelmistoa, joka on suunniteltu toimimaan juuri mobiililaitteilla, kutsutaan mobiilisovellukseksi. Mobiilisovelluksilla on lukematon määrä eri käyttötarkoituksia ja yli puolet mobiililaitteiden käyttäjistä käyttää verkkoselaimen sijaan mobiilisovelluksia. Mobiili alustana tarjoaa laajan valikoiman tekniikoita hyödynnettäväksi. Esimerkkinä näistä on paikannus, kosketustoiminnot ja puheohjaus. Mobiilisovelluksia pääsee lataamaan maksutta tai maksullisena mobiilisovelluskaupasta. (itewiki s.a..)

Tehdessäni taustaselvitystä, millaisia reseptisovelluksia on olemassa, käytin hakusanoina Google Play -kaupassa ruoka, resepti, recipes sekä erikoisruokavalioihin liittyviä termejä gluteeniton, ibs, ja fodmap. Mobiilisovellukset ja niiden toiminnallisuudet ovat koottuna luvun lopussa esitettävään taulukkoon 1.

K-ruoka-sovelluksessa pystyy rajaamaan reseptejä gluteenin, laktoosin, vegaanisuuden, maidon, lakto-ovon ja kananmunan mukaan. Sovelluksessa pääsee samalla kokoamaan ostoslistan ja tekemään mobiilissa ruokakauppatilauksen. K-ruoka-sovelluksessa ei pysty lisäämään omia ruokaohjeita, mutta käyttäjällä on mahdollisuus tehdä suosikkilistoja ja arvostella reseptejä lisäämällä niille tähtiä. Sovellus vaatii sisäänkirjautumisen. (Kesko 2024.)

Sisäänkirjautumisen vaatii myös Valion mobiilisovellus. Sillä on mahdollisuus hakea reseptejä Valion omasta reseptivalikoimasta. Käyttäjä pystyy luomaan oman keittokirjan, jonne tallentaa tietokannassa olevia reseptejä, mutta hän ei pysty lisäämään omia reseptejä. Käyttäjällä on mahdollisuus skaalata reseptin ainesosat oman toiveen mukaiselle annoskoolle. (Valio Oy 2024.)

Recipe Keeper -sovelluksessa käyttäjä pääsee lisäämään omia reseptejä kuvien kera, tekemään suosikkilistoja sekä luomaan ostoslistan ja suunnittelemaan ruokalistoja (Tudorspan Limited 2024). COOKmate- ja SuperCook-sovellukset ovat vastaavanlaisia mobiilisovelluksia (Maadinfo Services 2024; Google Play 2023).

Sovelluskaupasta löytyy myös keittokirjoja eri teemoilla, kuten Kakkureseptit, AirFryer Recipes, Gluteenittomat reseptit ja Gluteeniton: ruoka resepti (Google Play 2024a; Google Play 2024b;

Google Play 2024c; Google Play 2024d; Google Play 2024e). Sovellukset eroavat ulkoasuiltaan ja käytettävyydeltään.

FODMAP-ruokavaliolla löytyy sovelluksia, joista voi tarkistaa kuuluuko tietty ruoka-aine välteltävien listalle, kuten FODMAP Helper ja Low FODMAP diet A to Z foods. Maksullinen ja englanninkielinen Monash University FODMAP diet -sovellus kertoo kattavasti ruoka-aineiden soveltuvuuden ja sieltä löytyy myös reseptihaku-toiminnallisuus (Monash University ABN 2019). Tämä oli ainut FODMAP-teemainen reseptikirja.

Taulukko 1. Reseptisovellusten toiminnallisuudet (kaikissa on reseptinhakutoiminnallisuus)

Mobiilisovellus	**Ostoslista**	**Verkkoostokset**	**Reseptien lisääminen**	**Suosikkilista**	**Arvioiden antaminen**	**Ruokalistan suunnittelu**	**Sisäänkirjautuminen**	**Maksullinen käyttö**
K-ruoka	X	X	-	X	X	-	X	-
Valio	-	-	-	X	X	-	X	-
Recipe Keeper	X	-	X	X	X	X	X	X
COOKmate	X	-	X	X	X	X	X	X
SuperCook	X	-	X	X	X	-	X	-
Kakkureseptit	-	-	-	-	-	-	X	X
AirFryer Recipes	-	-	-	X	-	-	-	-
Gluteenittomat reseptit	-	-	-	-	-	-	-	-
Gluteeniton: ruoka resepti	X	-	-	X	-	-	X	-
Monash University FODMAP diet	-	-	-	X	X	X	X	X

2.2 Olemassa olevien avoimien rajapintojen kartoitus

Mobiilisovelluksen käyttöliittymä voi rakentua valmiin rajapinnan päälle, jolloin muun muassa erillistä tietokantaratkaisua ei tarvitse miettiä. Käsite rajapinta (API) tarkoittaa ohjelmistojen välistä rajapintaa. Sen takana on tekninen toteutus, jota ohjelma voi rajapinnan kautta kutsua ja kommunikoida. (Auburn, Bryant & Gough 2022, A Brief Introduction to APIs.) Rajapintaan liittyvän tiedon ja ominaisuuksien ollessa julkisia sekä kaikkien käytössä ja tarkasteltavissa sitä kutsutaan avoimeksi (Open API-työryhmä s.a.).

Internetissä on tarjolla avoimia rajapintalistoja, joita on muun muassa GitHubin Public-apis, API list […], </> ApisList ja avoindata.fi (GitHub 2022; ApiList.fun 2019; Massound Samy 2024; Digi- ja väestötietovirasto 2023). Näitä hyödyntäen etsin ruoka-aiheisia avoimia rajapintoja. Esimerkiksi The Meal DD:llä ja Tasty:lla on kattavat tietokannat ja niiden rajapinnat on tehty REST API -tekniikalla, jolloin voidaan hyödyntää http-protokollaa tiedon hakemisessa rajapinnan yli. Nämä rajapinnat ovat tosin vain kehitysvaiheessa ilmaisia ja julkaisun jälkeen maksullisia. (TheMealDB 2024; RapidAPI 2024.)

Löytämieni rajapintojen tieto on englannin kielellä, mikä ei suoraan sovellu suomenkielisen mobiilisovelluksen sisällöksi. Sovelluksen saa käännettyä suomen kielelle React Nativea käytettäessä esimerkiksi react-i18next kirjastolla (i18next 2024), mutta dynaamisen datan, esimerkiksi chat-keskustelun, kääntämiseen tarvitaan erillinen kääntäjä (Medium 2023). Google Translate, Amazon Translate ja DeepL ovat esimerkkejä tällaisesta palvelusta (Cozmoslabs 2024). Käännöskoneita ja paikallista tallentamista tehokkaampi tapa on pitää eri käännökset tallennettuina tietokannan puolella (Medium 2023).

2.3 Mobiilisovelluksissa käytettyjen tietokantojen vertailu

Valmiin rajapinnan sijaan mobiilisovelluksen kehityksessä voidaan käyttää tietokantaa, jonka perustehtävänä on varastoida dataa ja pitää se käyttäjille saatavilla. Käytän tässä yhteydessä tietokantaa ja tietokantajärjestelmää synonyymeinä. Sovellus käyttää sen eri näkymissä ja toiminnallisuuksissa tietokantaan tallennettua dataa. Jotta joka kerta, kun luodaan uusi sovellus, ei tarvitse keksiä uutta tapaa, miten järjestää ja säilöä tietoa, käytetään tietokantoja. Tietokannat ovat modulaarisia järjestelmiä ja koostuvat useista osista. On pyyntöjä vastaanottava siirtokerros, tehokkaaksi määritelty kyselykerros, toimintojen suorituskerros ja tallennuskerros. (Petrov 2019, Part 1. Storage Engines.)

Vielä vuonna 2000 eri tietokannoilla oli käytännössä vain vähän eroja ja niillä oli samanlaiset toiminnallisuudet ja käyttötapaukset. Vuoden 2010 tienoilla syntyi uusi termi NoSQL (Not Only SQL) ja avoin lähdekoodiyhteisö, internet-yhtiöt ja tietokantojen tarjoajat ovat luoneet hurjan määrän erilaisia tietokantasysteemejä. Kehitys jatkuu edelleen avain-arvo-tietovarastoista ja NoSQL-tietokannoista entistä skaalautuvampiin ja tehokkaampiin ratkaisuihin, joissa pystytään suorittamaan monimutkaisia kyselyjä johdonmukaisesti. (Petrov 2019, Preface.)

Aalphan ylläpitäminen verkkosivujen mukaan vuoden 2024 suosituimmat ja parhaat tietovarastot mobiilisovellukselle ovat Cloud- based data storage (Google ja Firebase), Postgres, Back4app, MySQL, Couchbase ja AWS DynamoDB. Muita paljon käytettyjä edellisten lisäksi on MongoDB (TechRev Block 2023).

Vanha ja pitkään käytössä ollut tietorakenne perustuu relaatiomalliin ja vakiintuneeseen SQL-kieleen. Näitä tietokantajärjestelmiä ovat muun muassa MySQL, MariaDB ja Postgres. Relaatiomallissa kaikki tieto tallennetaan riveinä tauluihin ja nämä voivat viitata toisiinsa. Tauluihin tehdään lisäykset, poistot ja muokkaukset SQL-kielellä, jolloin käyttäjältä peittyy tietokannan sisäinen toiminta. (Helsingin yliopisto 2021.)

MySQL, MariaDB ja Postgres ovat maksuttomia, avoimen lähdekoodin tietokantasysteemejä. Kaikki saavat kaupallista tukea tahoiltaan ja ovat tehokkaita tietokantoja. Niissä on omia eroja suorituskyvyissä, tietomalleissa ja laajennettavuuksissa. Postgres soveltuu erityisen hyvin monimutkaisten kyselyjen tekemiseen, koska se käyttää tehokkaita indeksointeja. (Stackscale B.V. 2024.)

NoSQL-tietokantoihin lukeutuu MongoDB, Couchbase, AWS DynamoDB, Firebase ja Back4app. Ei-relaatiotietokannoissa voidaan säilöä laajoja tietomääriä vähemmän strukturoidusti. Tietoon pääsee käsiksi rajapinnan kautta tehtävillä kyselyillä. Tieto on taulukoiden ja rivien sijaan järjestetty avain-arvo-parien, dokumenttien tai kaavioiden muodossa tai sarakepohjaisesti. (GeeksforGeeks 2024.)

MongoDB on avoimen lähdekoodin tietokantasysteemi, joka luokitellaan NoSQL-kannaksi. Se tallentaa tiedot joustavissa, JSON-tyylisissä dokumenteissa, jotka voivat sisältää monimutkaisia rakenteita ja erilaisia kenttiä. MongoDB:tä voidaan käyttää myös pilvessä, eli verkon välityksellä palveluntarjoajan palvelimelta. MongoDB on monipuolinen tietokanta, joka tarjoaa joustavan tietomallin ja soveltuu erityisesti tilanteisiin, joissa on tarve tallentaa suuria datamääriä dokumenttien muodossa. Lisäksi sen kehittäjä tarjoaa kaupallista tukea käyttäjilleen. (Stackscale B.V. 2024; MongoDB 2024.)

Palvelimeton-teknologia (Serverless) on nopeasti yleistynyt ja tilankäyttöä ei tarvitse arvioida etukäteen. Serverless-pohjaiset tietokannat ovat hyvin skaalautuvia, reaaliaikaisia ja mukana on vahva tietoturva. (Amazon Web Services 2024.) Vaikka termi serverless antaa mielikuvan, ettei käytössä ole palvelinta, sellainen on taustalla, mutta kehittäjän ei tarvitse huolehtia palvelimesta ja sen rakenteesta tai skaalautuvuudesta (MongoDB 2024). Esimerkiksi AWS DynamoDB, Firebase sekä Microsoft Azure SQL Database ovat serverless-teknologialla toimivia (Amazon Web Services 2024; Google for Developers 2024; Microsoft 2024b).

2.4 Johtopäätökset esiselvityksestä

Ennen työn alkua ajatuksenani oli tehdä reseptivihko-mobiilisovellus, jonne käyttäjä pääsee lisäämään omia reseptejään ja muodostamaan omia suosikkilistojaan. Ajatuksena oli käyttää React Nativen kanssa usein käytettyä Firebase-tietokantaa. Esiselvitysvaiheessa tutustuin useisiin valmiisiin reseptisovelluksiin ja aloin pohtimaan niiden käytettävyyttä ja ulkoasua. Osasta sovelluksia

selkeästi huomasi, että ruudulle on asetettu liikaa elementtejä tai sovellus ei visuaalisesti miellyttänyt silmää. Aloin tutkimaan saavutettavuutta ja ymmärsin sen hyödyt kaikille sovelluksen käyttäjille. Mielenkiintoni aihetta kohtaan kasvoi ja päätin lähteä rakentamaan omaa sovellustani saavutettavuuden näkökulmasta.

Pitkään pohdin, miten ratkaisen käyttöliittymän tietovarastopuolen. Oman tietovaraston rakentaminen tuntui luontevalta ja siitä on myös kokemusta opinnoissa. Myös vapaus luoda oma struktuuri säilytettävälle tiedolle antaa joustavuutta sovelluksen toimintoihin. Näen, että NoSQL-tietokannat haastavat perinteisen relaatiotietokannan mallin tehokkuudessaan ja joustavuudessa nykytarpeita ajatellen. Esimerkiksi kuvien ja chat-viestien tallennukseen nämä soveltuvat hienosti. Joskin itse pidän jossain määrin relaatiotietokantojen selkeästä rakenteesta.

Koska tavoitteena oli tutustua saavutettavuuden ja oppia hyödyntämään siihen käytettäviä ohjelmistokehyksen ominaisuuksia, päädyin kuitenkin ottamaan käyttöön valmiin rajapinnan. Valmiissa tietovarastossa on paljon ruokaohjeita, jolloin pääsen suoraan hyödyntämään niitä, eikä minun tarvitse itse lisätä tietokantaan reseptien tietoja. Toinen syy valintaan oli oma mielenkiintoni monimutkaisen JSON-muotoisen datan käsittelyyn. Halusin oppia syvemmin käyttämään REST-rajapintaa ja tässä siihen tarjoutui oiva tilaisuus.

3 Digitaalisen palvelun saavutettavuus

Digitaalisten palvelujen yleistyessä on alettu korostaa palvelujen esteettömyyttä. Jokaisen, oli hänellä jokin fyysinen tai kognitiivinen rajoite tai ei, kuuluu yhdenvertaisuuden mukaisesti pystyä hoitamaan pankkiasiat ja muut päivittäiset toimet verkossa. Maailman terveysjärjestö WHO on arvioinut, että maailmanlaajuisesti 16 prosentilla ihmisellä on jonkin tasoinen kehitysvamma tai muu rajoite. Tämä luku tulee kasvamaan reilusti väestön ikääntymisen ja kroonisten sairauksien lisääntymisen seurauksena. (World Health Organization 2023.)

Aluehallintoviraston selvityksen mukaan yli miljoonalla suomalaisella on haasteita digitaalisen palvelun käytössä jossain elämänsä vaiheessa. Rajoitus voi olla pysyvä tai hetkellinen, kuten ranteen murtuminen. Saavutettavuudesta on myös hyötyä kaikille, koska se parantaa digitaalisen palvelun käytettävyyttä. (Aluehallintovirasto 2024c.) Saavutettavat digitaaliset palvelut parantavat ihmisten yhdenvertaisuutta, kun ne ovat kaikille esteettömästi saavutettavissa (Kehitysvammaliitto 2024).

Luvussa käsitellään saavutettavuuden taustalla olevaa lainsäädäntöä, kansainväliseen standardiin perustuvaa saavutettavuusohjeistusta ja mitä se tarkoittaa mobiilisovelluksissa. Lopuksi käydään läpi ruudunlukijan käyttöä Android-pohjaisessa laitteessa.

3.1 Lainsäädäntö ohjaa saavutettavuutta

Digipalvelulaki edellyttää viranomaisen asemassa toimivia organisaatioita tekemään verkkopalveluistaan saavutettavia. Lisäksi lain piiriin kuuluu julkisoikeudelliset laitokset, osa järjestöistä ja osa yksityistä sektoria. (Oikeusministeriö 2019.) Laki astui voimaan vuonna 2019 ja sen siirtymäaika oli 2019–2021 (Aluehallintovirasto 2024a). Liikenne- ja viestintäministeriön julkaisussa käy ilmi, että tavoitteena on ottaa huomioon väestön ikääntyminen, yhdenvertaisuuden vahvistaminen ja esteettömyyden parantaminen (Valtioneuvosto 2023).

Vaikka saavutettavuutta ei vielä vaadita esimerkiksi kaupanalan verkkopalveluilta, se nähdään merkittävänä kilpailuetuna ja markkinaosuuden kasvamisena. EU:n esteettömyysdirektiivin kirvoittamana vuonna 2025 tulee voimaan uudet lisäsäännökset. Saavutettavuusvaatimukset laajenevat koskemaan verkkokauppoja, sähkökirjoja ja sähkökirjapalveluja, oli palveluntarjoaja sitten julkinen tai yksityinen. (Aluehallintovirasto 2024a; Kaupanliitto 2024.)

Saavutettavuusohjeistusta noudattamalla verkkosisältö tulee saavutettavammaksi ihmisille, joilla on vammoja tai rajoitteita näkökyvyn, valoherkkyyden, kuulon tai puheentuoton kanssa. Lisäksi se mahdollistaa parannuksia henkilöille, joilla on kognitiivisia haasteita, oppimisen ja ymmärtämisen kanssa ongelmia tai liikuntarajoitteita. Ohjeet tekevät myös yleisesti verkkosisällöstä käytettävämpää niin vanhemmille ihmisille kuin ihan kaikille. (Aluehallintovirasto 2024c.)

Valitettavasti saavutettavuuskriteeritkään eivät ole aukottomia, koska esimerkiksi Näkövammaliiton tekemän kyselyn perusteella suurin osa näkövammaisista kokee digipalvelut haastaviksi käyttää, vaikka ne täyttäisivätkin saavutettavuuskriteerit. Samaisessa tutkimuksessa ilmenee myös, että saavutettavuus voi olla yrityksille kilpailuetu, koska saavutettavat palvelut palvelevat paremmin laajempaa asiakasjoukkoa ja huonosti toimiva palvelu aiheuttaa yritykselle mainehaittaa. Käyttäjät eivät yleensä anna palautetta huonosti toimivasta palvelusta, vaan siirtyvät toisen palveluntarjoajan palveluun. (Näkövammaisten liitto 2021.)

Avaava on tutkinut yritysten kilpailuetua ja toteaa saavutettavan digisisällön olevan osa asiakaskokemusta ja merkittävä markkinaetu. Myönteiset käyttökokemukset saavat asiakkaat palaamaan uudelleen palveluiden pariin ja kasvattavat kohderyhmiä. Yli puolet ihmisistä on valmiita käyttämään enemmän rahaa palveluihin, joissa heidät on huomioitu ja kolmeneljäsosaa ihmisistä lähtee pois palvelusta, mikäli se ei ole esteetön. (Tamminen 2024.)

3.2 Saavutettavuusohjeistus – WCAG 2.1

Etelä-Suomen aluehallintoviraston ylläpitämillä verkkosivuilla on listattuna suomennetut digitaalisen palvelun saavutettavuuskriteerit (WCAG 2), joita on 49. Kriteerit pohjautuvat kansainväliseen standardiin, jonka parissa työskentelee järjestö World Wide Web Consortium (W3C). Suomessa laki velvoittaa noudattamaan ohjeistuksen tasojen A ja AA kriteerejä. AAA-tason kriteerejä ei velvoiteta täyttämään. (Aluehallintovirasto 2024b.)

WCAG 2 -ohjeistuksessa on neljä pääperiaatetta (taulukko 2), jotka ovat perusta verkkosaavutettavuudelle. Nämä ovat havaittavuus, hallittavuus, ymmärrettävyys ja toimintavarmuus. Periaatetason alapuolelle on luotu puitteet, yhteensä 13 ohjetta, jotka määrittelevät sisällöntuottajien työskentelyn yleiset tavoitteet ja auttavat soveltamaan tekniikoita paremmin ja ymmärtämään onnistumiskriteereitä. Jokaiselle ohjeelle on määritelty testattavat onnistumiskriteerit. Lisäksi jokaiselle ohjeelle ja onnistumiskriteerille on luotu riittäviä ja neuvoa-antavia tekniikoita. (Aluehallintovirasto 2024c.)

Taulukko 2. WCAG 2 -ohjeistuksen perusperiaatteet ja ohjeet (Aluehallintovirasto 2024c)

Havaittava	Hallittava	Ymmärrettävä	Toimintavarma
1.1 Aikasidonnainen media	2.1 Käytettävissä näppäimistöltä	3.1 Luettava	4.1 Yhteensopiva
1.2 Mukautettava	2.2 Tarpeeksi aikaa	3.2 Ennakoitava	
1.3 Erottuva	2.3 Sairauskohtaukset	3.3 Syötteen avustaminen	
	2.4 Navigoitava		
	2.5 Syötetavat		

3.3 Saavutettavuus mobiilisovelluksessa

Verkkopalveluja koskevat ohjeistukset pätevät myös mobiilisisältöön, mobiilisovelluksiin, natiivisovelluksiin ja hybridisovelluksiin. Niiden erityispiirteitä varten W3C on laatinut tarkennuksia ja lisäohjeistuksia. (W3C 2015.)

Saavutettavuus käytänteitä haastaa mobiililaitteissa niiden näytön pieni koko, jolloin saavutettavuuden parantamiseksi ruudulle ohjattavaa tiedon määrää tulee rajoittaa ja kontrastin tulee olla riittävä. Lisäksi käyttäjän on pystyttävä suurentamaan ja loitontamaan sisältöä. (W3C 2015.)

Kosketusnäyttöisissä laitteissa ja sovelluksissa on tarpeen tehdä mahdolliseksi käyttää ulkoista näppäimistöä. Näytöllä olevien kosketuselementtien tulee olla riittävän suuret ja erillään toisistaan. Interaktiivisten elementtien tulee olla sijoitettuna helppokäyttöisesti ja niin, ettei ole merkitystä kummalla kädellä niitä käyttää. Mikäli sovelluksessa on käytössä muokattuja kosketuseleitä, niistä on informoitava käyttäjää. Esimerkiksi johonkin toimintoon liittyvä ravistaminen tai kallistaminen on pystyttävä toteuttamaan tavanomaisella kosketuseleellä tai näppäimistöltä. (W3C 2015.)

Jotkut käyttäjät pystyvät käyttämään laitetta vain vaakatasossa, jolloin on tärkeää, että sisältö näkyy oikein. Samat toiminnot sovelluksen eri sivuilla tulisi olla johdonmukaisia. Jos näytöllä on esimerkiksi teksti- ja kuvalinkki, jotka vievät samaan lopputulokseen, ne olisi hyvä ryhmitellä saman toiminnon alle. Tärkeiden elementtien sijoittaminen vieritystoiminnon ulkopuolelle varmistaa sen, että suurennusta käyttävä löytää ne helpommin. (W3C 2015.)

Toimintavarmuutta tukee virtuaalisen näppäimistön asettaminen vaaditun datatyypin mukaan. Myös helppokäyttöisiä tiedon syöttötapoja tulisi suosia. Esimerkiksi valintaruudut, valintapainikkeet ja automaattinen päivämäärän syöttö ovat hyviä tapoja tukea saavutettavuutta. Kriteereissä kehotetaan tukemaan alustakohtaisia ominaisuuksia, esimerkiksi tekstin suurentaminen tulisi huomioida mobiilisovelluksen sisällön käyttäytymisessä. (W3C 2015.)

Taulukkoon 3 on koottuna otsikkotasolla tiivistelmä ohjeistuksista. Perusperiaatteiden alle on listattuna ohjetason kriteerit. Listausta käytetään myös sovelluksen testaukseen.

Taulukko 3. Perusperiaatteet ja mobiilisisältöön liittyvät ohjeet (W3C 2015)

1. Havaittava	2. Hallittava	3. Ymmärrettävä	4. Toimintavarma
1.1 Pieni ruudun koko	2.1 Ulkoisen näppäimistön käyttö kosketusnäytön kanssa	3.1 Näytön suunnan muuttaminen pysty- tai vaakasuuntaan	4.1 Virtuaalinen näppäimistö asetetaan vaaditun datatyypin mukaan
1.2 Suurennus ja loitonnus	2.2 Kosketuselementtien koko ja etäisyys toisistaan	3.2 Toimintojen johdonmukainen asettelu	4.2 Tiedonsyöttötavoista tehdään helppoja
1.3 Kontrasti	2.3 Kosketusnäytön ohjauseleet	3.3 Tärkeiden elementtien asettelu niin, ettei tarvitse vierittää	4.3 Alustakohtaisia ominaisuuksia tuetaan
	2.4 Laitteen manipuloinnin eleet	3.4 Toimintoelementtien ryhmittely saman toiminnon mukaan	
	2.5 Painikkeiden sijoittaminen helppokäyttöisesti	3.5 Elementin toiminto osoitetaan selkeästi	
		3.6 Muokattavien kosketuseleiden käyttöön annetaan ohjeet	

3.4 Ruudunlukija ja sen käyttö

Ruudunlukija on ohjelmistosovellus, jonka avulla näkövammaiset voivat käyttää tietokonetta ja mobiililaitteita. Sovellus kääntää Text-To-Speech (TTS) teknologialla ruudun visuaalisen sisällön käyttäjälle kuuluvaksi puheeksi tai tuottaa pistekirjoitusta ulkoiselle pistekirjoituslaitteelle. Ruudunlukijasovelluksia on useita eri käyttöjärjestelmille. (Nomensa 2005.)

Sen lisäksi, että ruudunlukija käy sivun läpi ylhäältä alaspäin aloittaen ylätunnisteesta otsikkoon, alaotsikkoon ja tekstisisältöön, se antaa käyttäjälle mahdollisuuden liikkua elementtien välillä. Käyttäjä navigoi mobiililaitteella tietyillä eleillä ja ruudunlukija ilmoittaa käyttäjälle näytön sisällöstä. Android-laitteelle on saatavana TalkBack- ja Voice Assistant -ruudunlukijat ja Applelle VoiceOver. (Sapega 2021.)

Sovelluksen ohjelmointikielessä täytyy käyttää semanttisia elementtejä, jotta sisällöstä tulee ymmärrettävää ruudunlukijalle ja käyttäjälle. Ruudunlukija-sovellus lukee ohjelmointikielestä komponentin roolin (role) ja nimen (name/label) ja esimerkiksi tekstisyöttökentistä arvon (value) tai mikäli kyseessä on valikko, niin tilan (state). (Dodson 2018.) Lisää elementeistä kerrotaan luvussa 4.3.1 React Nativen saavutettavuusominaisuudet.

Mobiililaitteissa on yleensä valmiiksi ruudunlukijasovellus. Androidissa TalkBackin saa käyttöön laitteen asetuksista, pikanäppäimellä (painamalla äänenvoimakkuutta pitkään) tai Google

Assistanten kautta ääniohjauksella (Google 2024a). TalkBackiä käytetään erilaisilla kosketuseleillä. Ele voi olla yhden tai kahden sormen pyyhkäisyele tai yhden tai useamman naputuksen ele. Näytöllä voi myös liikkua sormella kohteelta toiselle, jolloin kohde luetaan ääneen.

TalkBackin perustoimintoja:

- Näytön vieritys: Pyyhkäistään kahdella sormella ylös tai alas/ vasemmalle tai oikealle
- Näytöllä siirtyminen seuraavaan kohteeseen: Yhden sormen ele oikealle
- Näytöllä siirtyminen edelliseen kohteeseen: Yhden sormen ele vasemmalle
- Suurennusele: Kolmoisnapautus
- TalkBackin puheen keskeytys tai jatkaminen: Kahden sormen napautus

TalkBackin asetuksista pääsee myös muokkaamaan useimmille toiminnoille käytettäviä eleitä omaan käyttöön sopivaksi. Asetuksiin pääsee napauttamalla kolmella sormella näyttöä. (Google 2024b.)

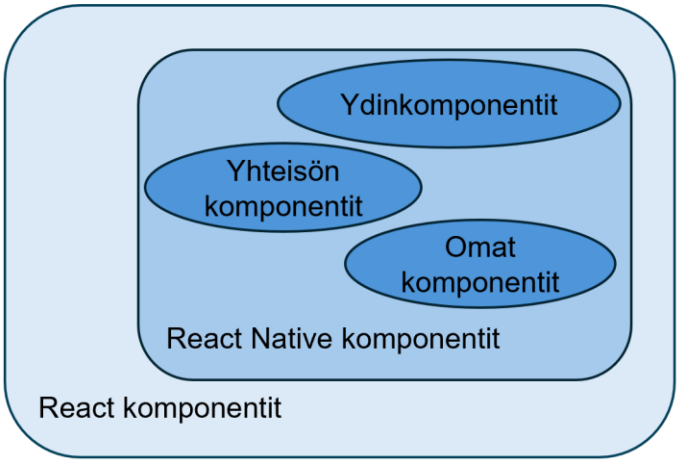
4 Käytettävät teknologiat ja työvälineet

Luvussa käydään läpi React Native -kehys ja sen yleisimmin käytetyt komponentit, puhutaan Expo Go:sta ja neuvotaan, miten sovelluksen tekeminen aloitetaan. Kaksi alalukua käsittelee React Nativen saavutettavuusominaisuuksia ja muita sen saavutettavuutta tukevia tekniikoita. Lopuksi vielä otetaan esille käytettävä kehitysympäristö ja versionhallinta.

4.1 React Native -kehys ja Expo

JavaScript-pohjaista React Native -kehystä pidetään tehokkaana tapana tuottaa koodia ja se mahdollistaa sovelluksen osien kirjoittamisen eri ohjelmointikielillä. React Native -pohjaiset sovellukset lataavat sivut nopeasti ja luotettavasti. Sovelluskehys mahdollistaa Android- ja iOS-sovelluksen rakentamisen yhteen sovellukseen sen sijaan, että joutuisi käyttämään eri ohjelmointikieliä molemmille käyttöjärjestelmille. (Dabit 2019, 4; TechMagic 2022.) Vaihtoehtona React Nativelle mobiiliohjelmoinissa on muun muassa Googlen kehittämä Flutter, Microsoftin Xamarin ja iOS ympäristöön käytettävä SwiftUI. (Meta Platforms, Inc. 2024a.)

React Native on Facebookin kehittämä avoimen lähdekoodin kehys natiivi, eli alustakohtaisten, mobiilisovellusten käyttöliittymien kehittämiseen. Se perustuu web-ohjelmoinnissa suosiota saaneeseen JavaScript-pohjaiseen Reactiin. (Meta Platforms, Inc. 2024a.) Kuvassa 1 on havainnollistettu komponenttiarkkitehtuuria.



Kuva 1. React Nativella rakennetun mobiilisovelluksen käyttöliittymä koostuu erilaisista komponenteista (mukaillen Meta Platforms, Inc. 2024a)

Sovelluskehyksessä on käyttövalmiita natiiveja ydinkomponentteja (taulukko 4). Näitä voi myös muokata tarpeeseen sopivaksi käyttämällä muuttujaa (property) ja poimimalla siihen esimerkiksi

tietyn nimen, jonka funktio palauttaa näytölle. Ydinkomponenttien lisäksi voi rakentaa omia natiivikomponentteja. (Meta Platforms, Inc. 2024a.)

Taulukko 4. Yleisimmin käytettävät ydinkomponentit (mukaillen Meta Platforms, Inc. 2024a)

React Native komponentti	**Web suunnittelu**	**Kuvaus**
`<View>`	`<div>`	Sisältö, joka tukee asettelua flexboxilla, tyyleillä, kosketustapahtumilla ja esteettömyysominaisuuksilla
`<Text>`	`<p>`	Näyttää, tyylittelee, asettelee sisäkkäin string-muotoista tekstiä ja käsittelee kosketustapahtumia
`<Image>`	`<img>`	Näyttää erityyppisiä kuvia
`<ScrollView>`	`<div>`	Geneerinen vierityssisältö, joka voi sisältää useita komponentteja ja näkymiä
`<TextInput>`	`<input type="text">`	Sallii käyttäjän lisätä tekstiä

4.2 Expo Go -kehittäjätyökalu

Mobiilisovelluksen kehitysympäristöön tarvitaan myös Expo Go tai React Native CLI (Command Line Interface). React Nativen dokumentaatiossa suositellaan tuoreelle mobiiliohjelmoijalle aloittamaan Expo Go:n käytöllä, koska sen sisältämät työkalut saa nopeasti ja helposti käyttöön. Lisäksi kehittäjä tarvitsee puhelimen tai emulaattorin. React Native CLI sisältää laajempia toimintoja ja vaatii enemmän kehittäjän laitteistolta. Puhelimeen asennetaan Expo Go -sovellus, ja laite yhdistetään samaan langattomaan internetiin, jossa tietokone on. Projekti käynnistetään terminaalissa käskyllä ’npx expo start’ ja terminaaliin generoitu QR-koodi luetaan Android-puhelimen Expo Go -sovelluksella. Koneelle tarvitsee olla myös asennettuna Node.js, jotta saa Expon käyttöönsä. (Meta Platforms, Inc. 2024a.)

Opinnäytetyössä kehitettävä sovellus on sen verran suppea, että päädyn käyttämään suositeltua Expo Go:ta ja ainakin alkuun puhelimeen ladattavaa sovellusta tietokoneelle asennettavan emulaattorin sijaan.

4.3 React Native ja saavutettavuus

React Nativella on accessibility-ominaisuuksia, jotka tukevat saavutettavan mobiilisovelluksen rakentamista ja ruudunlukijan käyttöä. Ominaisuuksilla kerrotaan ruudunlukijalle, millainen komponentti on kyseessä ja voiko ruudunlukija ohittaa komponentin. Lisäksi React Nativen tekniikoilla saadaan parannettua muita mobiilisovelluksen saavutettavuusominaisuuksia. Elementit ja niiden arvot vaihtelevat riippuen käyttöjärjestelmästä. Tässä luvussa käydään läpi merkittävimmät Androidille tehdyt saavutettavuusominaisuudet. (Meta Platforms, Inc. 2024b.)

4.3.1 React Nativen saavutettavuusominaisuudet

Asettamalla komponentin, esimerkiksi <View>:n, sisälle accessible={true}-ominaisuuden, view-komponentista tulee saavutettava elementti ja sen sisältö ryhmittyy omaksi kokonaisuudeksi. Ruudunlukijan käyttäjiä voidaan auttaa myös kertomalla ruudunlukijalle, jos jokin komponentti ei ole tärkeä, jolloin sen voi ohittaa. (Meta Platforms, Inc. 2024b.)

Hyvän käytännön mukaista on lisätä komponentin sisälle myös ominaisuus accessibilityLabel ja sille kuvaava toiminto, esimerkiksi painikkeelle accessibilityLabel=”Tap me! Jolloin ruudunlukija kertoo käyttäjälle, mikä elementti on kyseessä. (Meta Platforms, Inc. 2024b.)

Kompleksisempia kokonaisuuksia saa saavutettavaksi käyttämällä accessibilityLabelledBy-ominaisuutta. Tekstikentän voi esimerkiksi linkittää input-kenttään lisäämällä TextInput-komponentin sisälle teksti-komponentin nativeID:n. Ominaisuudella accessibilityHint voidaan viestittää käyttäjälle enemmän tietoa komponentin käytöstä kuin mitä ominaisuudella accessibilityLabel pystytään. (Meta Platforms, Inc. 2024b.)

Ominaisuus accessibilityLiveRegion auttaa kertomaan loppukäyttäjälle ruudunlukijan kautta, kun komponentti dynaamisesti muuttuu. Sille asetettavia arvoja ovat:

- none - ei muutosilmoituksia
- polite - ilmoita muutoksista
- assertive – ruudunlukijan tulee välittömästi keskeyttää puhe ja ilmoittaa muutoksesta

(Meta Platforms, Inc. 2024b.)

Komponenteille on syytä myös asettaa käyttötarkoitus, jotta käyttäjä saa ruudunlukijalta riittävän informaation, kuten accessibilityRole=”header”. Muita accessibilityRole-arvoja on esimerkiksi text, button, grid, menu, scrollbar, search, image ja imagebutton. Pidempi lista arvoista löytyy Metan verkkosivulta. (Meta Platforms, Inc. 2024b.)

4.3.2 Muita saavutettavuutta tukevia tekniikoita

Mobiilisovelluksen saavutettavuutta voidaan myös parantaa muilla keinoilla. Navigaatio-elementtiin suositellaan lisättävän kunkin sivun otsikon nimi title-ominaisuuteen options-määritysten yhteyteen, jotta näyttö tulee identifioiduksi ruudunlukijalle. (Appt Foundation 2024.)

Myös muut käyttäjät kuin ruudunlukija-avusteiset tulee huomioida. Teksti-komponentin rivitys ehkäisee suurta fonttia laitteessaan käyttävän henkilön juuttumisen siihen, että teksti on yhdellä rivillä ja se jatkuu kauas näytön reunan ulkopuolelle. Teksti-komponenttiin voi määrittää numberOfLines={}-ominaisuudella tekstin rivityksen. (Appt Foundation 2024.)

ScrollView-komponentin käyttö tekee sovelluksesta käytettävämmän mahdollistamalla pidemmän sisällön näytölle mahtumisen näytön vierittämisellä. React Native tukee myös oletusarvoisesti pysty- ja vaakasuuntaisia näyttöjä. (Appt Foundation 2024.)

4.4 Kehitysympäristö ja versionhallinta

Opinnäytetyön toiminnallisen osan tekemisessä käytän koodieditorina Visual Studio Codea, joka on Microsoftin kehittämä avoin koodieditori sovellusten rakentamiseen ja virheenetsintään (Microsoft 2024a). VS Codea pystyy käyttämään sovelluskehitykseen eri ohjelmointikielillä ja sen saa kustomoitua asentamalla lisäosia. Esimerkiksi Debugger-lisäosia eri ohjelmointikielille sekä lisäominaisuus Dockerin käyttöön.

Koodieditorin käyttöä sujuvoittaa, kun oppii käyttämään tiettyjä pikanäppäinyhdistelmiä. Valitsin sen työvälineeksi, koska se on yleisesti hyväksi havaittu ja olen tottunut käyttämään sitä aiemmissa projekteissani.

Git on hajautettu versionhallintajärjestelmä ohjelmakehitysprojekteille, jotta tehdyt muutokset voidaan tarvittaessa myöhemmin palauttaa (git. s.a.). GitHub on avoimen lähdekoodin webpohjainen versionhallintapalvelu, jossa tietovarastot ovat usein julkisia ja kehittäjät voivat jakaa ja julkaista omia projekteja muille (GitHub 2024). Tässä projektissa käytän GitHubia, koska se on avoin, ennestään tuttu sekä muutenkin laajasti käytetty alusta.

Mobiilisovelluksen lähdekoodi tullaan julkaisemaan GitHubissa. Jatkokehityksenä sovelluksen voisi julkaista Google Play -sovelluskaupassa, joka tarjoaa Android-pohjaisille mobiililaitteille, televisioille ja älykelloille maksuttomia ja maksullisia sovelluksia (Google Store 2024). Mobiilisovelluksen julkaisun aion tässä vaiheessa rajata vain Android-laitteille, mutta käytettävä teknologia mahdollistaa sen laajentamisen myöhemmin myös iOS-alustalle.

5 Mobiilisovelluksen kehitys

Ohjelmistokehityksessä käydään yleensä läpi vaiheet: vaatimusmäärittely, suunnittelu, toteutus, testaus ja käyttöönotto. Seuraavassa esitellään havainnollisesti, miten nämä vaiheet toteutuivat tässä projektissa.

5.1 Vaatimusmäärittely

Projektissa suunniteltiin ja toteutettiin saavutettava reseptisovellus. Toiminnallisena vaatimuksena oli, että käyttäjän on pystyttävä selaamaan ruokaohjeita eri kategorioissa sekä hakemaan haluamaansa reseptiä. Käyttäjän on pystyttävä valitsemaan yksittäinen resepti, jolloin sovellus näyttää reseptin tiedot, ja loppukäyttäjällä on mahdollisuus tutustua reseptin sisältöön ja sen perusteella toteuttamaan ruoka tai leivonnainen. Sovelluksen on oltava helppokäyttöinen ja saavutettava. Ruudunlukijaa tulee pystyä käyttämään ainakin yhdellä sovelluksen näytöllä.

Mobiilisovellukselle ei tässä vaiheessa rakenneta omaa tietovarastoa, vaan JSON-muotoiset tiedot haetaan valmiin REST-rajapinnan kautta fetch()-funktiolla. Toimintavarmuutta heikentää avoimen rajapinnan rajoitteet, maksuton määrä kutsuja on 500 kuukaudessa, joten käyttö rajautuu kehitysympäristöön.

Sovellus kehitetään pelkästään Android-käyttöjärjestelmälle ja ohjelmointikielenä käytetään JavaScriptin React Native -kehystä. Sovelluksen toiminnot on pystyttävä testaamaan laitteelle asennetun Expo Go:n avulla sekä ruudunlukijalla. Projektin sisältöön kuuluu ruudunlukija-sovellukseen tutustuminen. Testitulokset kirjataan ylös ja niistä muodostetaan tarvittaessa jatkokehityskohteita.

Mobiilisovelluksen ensisijainen tavoitekäyttäjä on opinnäytetyön tekijä, mutta sovelluksen olisi oltava myös aisti- tai liikerajoittuneiden käyttäjien hyödynnettävissä. Työ perustuu omaan kiinnostukseen, eikä työllä ole toimeksiantajaa. Työ suunnitellaan ja toteutetaan Android-käyttöjärjestelmälle ja se julkaistaan GitHubissa. Työn lopuksi analysoidaan lopputulos ja listataan jatkokehityskohteet.

Projektin tekeminen rajoitettiin tapahtuvaksi kevään 2024 aikana ja käytössä oli viisipäiväinen työviikko kohtuullisilla tunneilla.

5.2 Vaatimusmäärittelyn käyttötapaukset

Keräsin ajatuksiani kasaan aivoriihimäisesti sovelluksen eri toiminnoista ja piirsin itselleni rönsyilevän miellekartan paperille. Näistä valitsin toteutettavaksi neljä päätoimintoa, joiden käyttötapaukset on kuvattu alla olevissa taulukoissa 5–8.

Taulukko 5. Käyttötapaus reseptin hakemisesta

Käyttötapaus:	Reseptin hakeminen
Yhteenveto:	Käyttäjä hakee reseptin nimellä reseptiä
Toimijat:	Käyttäjä
Ehdot:	Käyttäjän on kirjoitettava hakusana hakukenttään
Kuvaus:	Käyttäjä kirjoittaa hakusanan hakukenttään ja painaa näppäimistön etsi/enter-painiketta. Sovellus hakee kannasta reseptin tiedot
Poikkeukset:	Annetulla hakusanalla ei löydy tuloksia. Haulla löytyy useampi tulos.
Lopputulos:	Resepti löytyy onnistuneesti

Taulukko 6. Käyttötapaus reseptien selailu kategorioittain vertikaalisesti

Käyttötapaus:	Reseptien selailu kategorioittain
Yhteenveto:	Käyttäjä tarkastelee leivonta- tai kokkauskategorian tuotteita listalta
Toimijat:	Käyttäjä
Ehdot:	Käyttäjän on vieritettävä näyttöä alaspäin ja takaisin ylöspäin
Kuvaus:	Käyttäjä napauttaa sovelluksen alanavigaation Cooking- tai Baking-elementtiä, jolloin sovellus menee valitulle näytölle ja listaa valitun kategorian reseptien kuvat ja nimet. Käyttäjä pääsee tarkastelemaan listausta vierittämällä näyttöä alas- ja ylöspäin.
Poikkeukset:	
Lopputulos:	Kategorialistaus näkyy onnistuneesti.

Taulukko 7. Käyttötapaus reseption selailu kategorioittain horisontaalisesti

Käyttötapaus:	Reseptien selailu kategorioittain horisontaalisesti
Yhteenveto:	Käyttäjä tarkastelee tuotteita listalta
Toimijat:	Käyttäjä
Ehdot:	Käyttäjän on vieritettävä näyttöä vasemmalle sivulle ja takaisin oikealle sivulle
Kuvaus:	Sovellus listaa ns. näyteikkunakategorian reseptien kuvat ja nimet. Käyttäjä pääsee tarkastelemaan listausta vierittämällä näyttöä sivuille.
Poikkeukset:	
Lopputulos:	Kategorialistaus näkyy onnistuneesti.

Taulukko 8. Käyttötapaus yksittäisen reseptin tietojen näkeminen

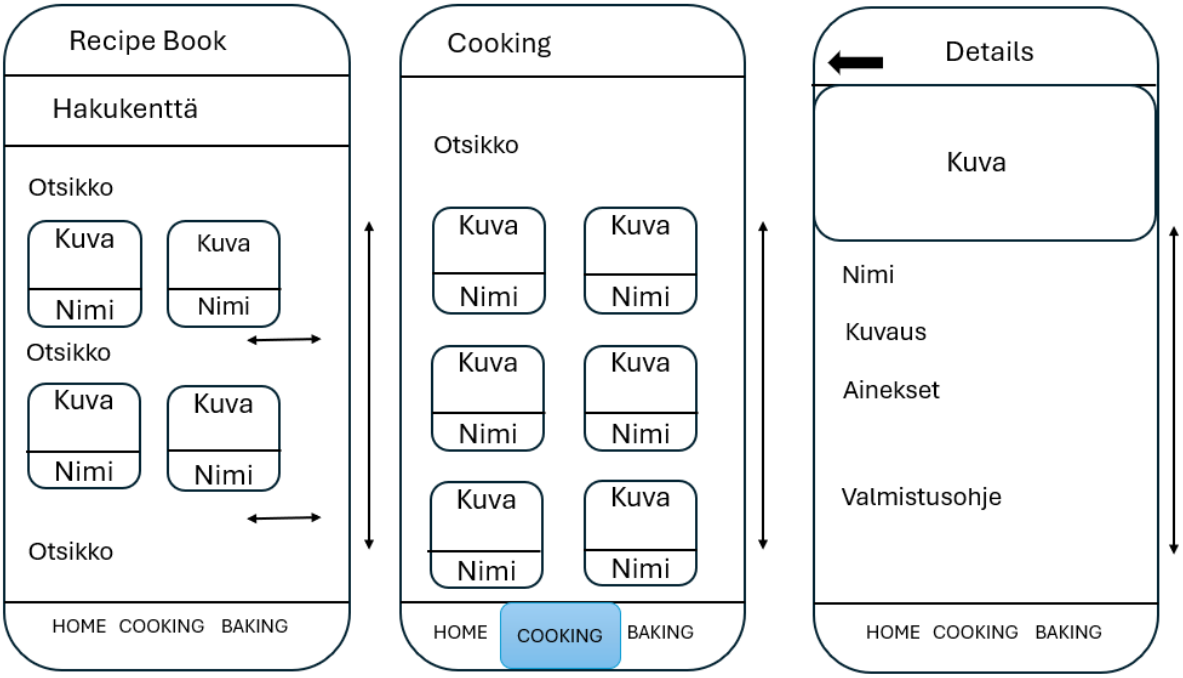
Käyttötapaus:	Yksittäisen reseptin tietojen näkeminen
Yhteenveto:	Käyttäjä valitsee listalta yhden reseptin painamalla sitä ja pääsee tarkastelemaan reseptin sisältöä
Toimijat:	Käyttäjä
Ehdot:	Käyttäjän on napautettava haluttua reseptiä ruudulla
Kuvaus:	Käyttäjä valitsee reseptin, josta haluaa nähdä tarkemmat tiedot, napauttamalla reseptin kuva- ja nimielementtiä. Sovellus vie käyttäjän Details-näytölle, jossa reseptin kuvan ja nimen lisäksi on reseptin kuvaus, ainesosat ja valmistusohje. Käyttäjä näkee lisää sisältöä vierittämällä näyttöä alaspäin. Käyttäjä pääsee takaisin listaukseen napauttamalla navigaation takaisinnuolta.
Poikkeukset:	
Lopputulos:	Reseptin tiedot näkyvät Details-näytöllä.

5.3 Käyttöliittymän suunnittelu

Käyttöliittymän suunnittelussa otettiin huomioon saavutettavuuden kriteerit. Värityksessä käytetään riittävää kontrastia ja painikkeet tehdään riittävän suuriksi, jotta käyttäjä paremmin osuu oikeaan toimintoon.

Näkymiin ei haluta liikaa sisältöä ja välkkyviä kuvia vältetään. Navigaatio rakennetaan selkeäksi yhdistelemällä React Nativen Navigator kirjastosta Stack- ja Tab-navigaattorit. Varmistetaan, että käyttäjällä on aina pääsy takaisin etusivulle.

Reseptin tiedot ovat selkeästi näkyvillä ja sivu liukuu alaspäin vierittämällä sormea ylöspäin. Havainnekuvat sovelluksesta on esiteltynä kuvassa 2.



Kuva 2. Sovelluksen rautalankamallit

5.4 Testauksen suunnittelu

Jokainen yksittäinen toiminnallisuus ja uusi kehitys testataan ennen seuraavaa kehityskohdetta, jotta mahdolliset virheet saadaan heti kiinni. Laajempi käyttäjätestaus tehdään, kun sovellus on lähes valmis. Testauksessa käytetään omalle laitteelle asennettavaa EXPO Go -sovellusta. Testaus keskittyy komponenttien toimintaan ja loppukäyttäjän toimiin. Sovellukselle ei kirjoiteta erillisiä automaattitestejä tai käytetä testaustyökaluja, kuten Jest-työkalua. Työn tavoitteena on, että testit läpäisevät määritellyt käyttötapaukset.

Jo koodia kirjoittaessa tavoitteena on käyttää hyvää ohjelmointikieltä ja lisätä selitteitä, jotta myöhemmin koodiin on helpompi palata ja myös mahdollisten virheiden ja poikkeamien korjaus sekä jatkokehitys helpottuvat.

Sovelluksen etusivun komponentit testataan ruudunlukijalla. Koko mobiilisovellus testataan saavutettavuuskriteerien osalta ja tulokset dokumentoidaan.

5.5 Ketterällä kehityksellä tuloksia

Sovelluksen suunnitelma ja määrittelyt saattavat kasvaa laajoiksi, jolloin vaarana on sovelluksen keskeneräisyys ja käyttökelvottomuus, jos kaikkia toiminnallisuuksia ei saada toteutettua ja testattua. Riskin pystyy minimoimaan käyttämällä ketterää ohjelmistokehityksen menetelmää, jolloin tuotos saadaan käyttöön suppeammillakin ominaisuuksilla ja loput ominaisuudet voidaan raportoida

kehityskohteiksi. Jokaisen aikajakson lopussa arvioidaan tuotos ja seuraavan aikajakson alussa suunnitellaan tehtävät toiminnallisuudet ja priorisoidaan.

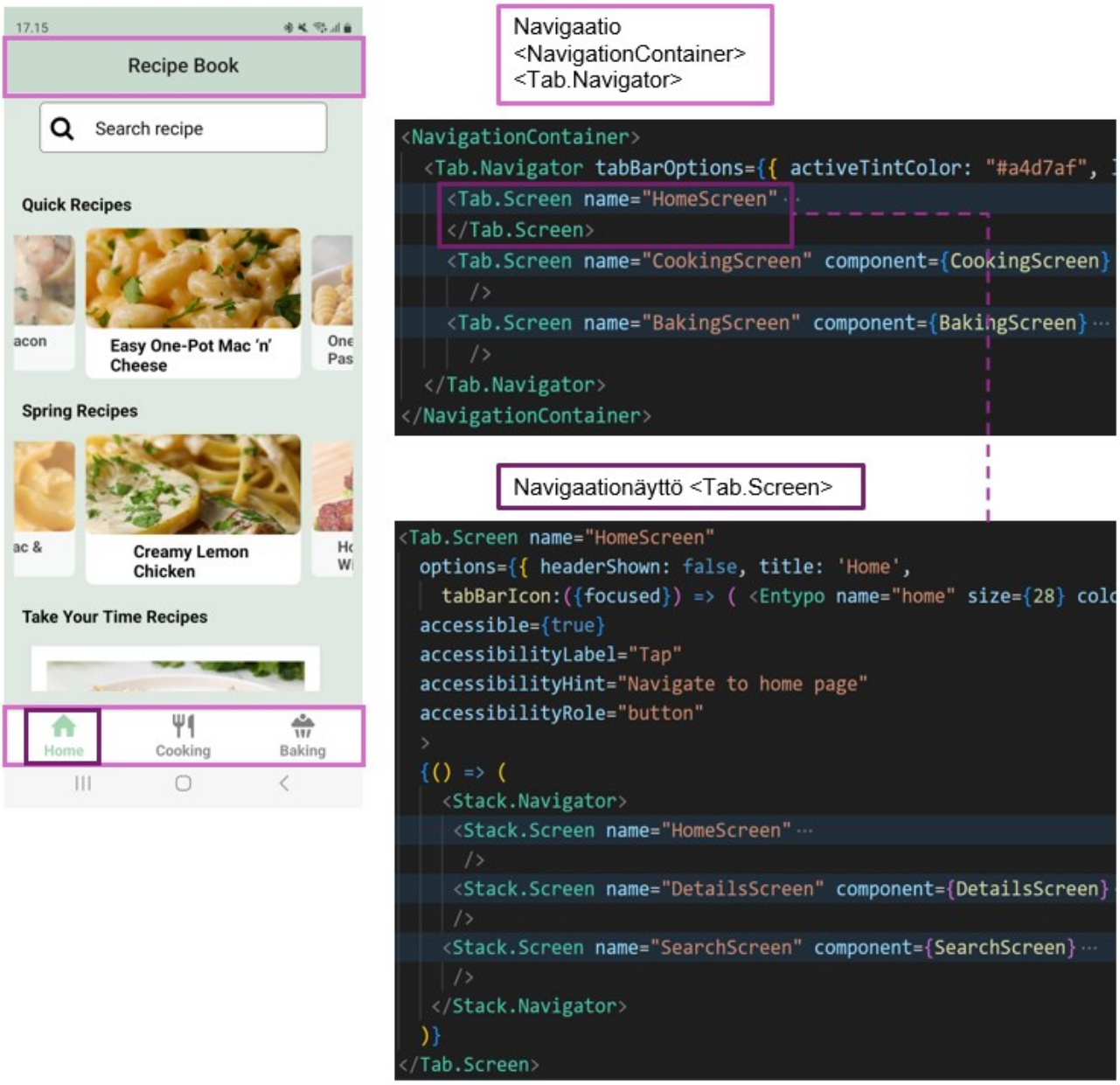
Projektin lähtöajatuksena on pienin toimiva tuote eli Minimum Viable Product (MVP). Suunnittelun pohjana käytetään vaatimusmäärittelyä ja tehtävät aikataulutetaan jokaiselle viikolle.

Tässä projektissa sprintin pituus on yksi viikko, jonka lopussa pysähdytään tarkastelemaan mennyttä viikkoa ja pohtimaan, mikä meni hyvin? Missä olisi kehitettävää? Näin seuraavalla viikolla on mahdollisuus parantaa toimintaa. Myös jokaisena iltapäivänä otetaan hetki katsaukseen, mitä on saatu aikaiseksi ja mistä seuraavana aamuna jatketaan. Näin aamuksi suunnitelma on valmiina ja päivä alkaa ketterästi.

5.6 Toteutus

Aloitin sovelluksen luomisen kirjoittamalla Windows-terminaaliin käskyn ’npx create-expo-app’ ja nimeämällä projektin. Expo generoi aloitusprojektin peruskomponentilla, jonka pystyy heti ajamaan käskyllä ’npx expo start’. Loin myös GitHubiin uuden repositoryn https://github.com/maellu8/saavutettava_reseptimobiilisovellus.git ja lisäsin toiminnan, jossa commitoidessani muutokset siirtyvät suoraan GitHubiin eikä niitä tarvitse paikallisesta Git:stä siirtää erikseen.

Päätin aluksi tehdä navigaation ja edetä siitä etusivun korttikomponenttien luontiin. Navigaatioon rakensin Tab-navigaation sisälle Stack-navigaation, jossa etusivulta yksittäisen reseptin tietoihin tai hakusivulle mennessä näytön vasempaan yläreunaan ilmestyy takaisinnuoli, jota painamalla palautuu takaisin lähtösivulle. Navigaation komponentteja ja lähdekoodia on havainnollistettu kuvassa 3.



Kuva 3. Etusivun Tab-navigaatio, jonka sisälle on rakennettu Stack-navigaatio

Lisäsin etusivulle SafeAreaProvider- ja ScrollView-komponentit, joista ensimmäinen estää sisältöä menemästä puhelimen vapaan alueen ulkopuolelle ja toinen mahdollistaa sisällön vierittämällä näytöllä ylhäältä alaspäin ja takaisin. Otin karusellikomponentin seuraavaksi työn alle ja rakensin siitä oman komponentin ListHorizontal. Testasin sen toimivuuden pienestä paikallisesta tiedostosta parametreina otettavilla tiedoilla. Laitoin hiukan muotoilua, mutta päätin jättää ulkoasun myöhemmäksi. Edellisessä kuvassa 3 näkyy karusellikomponentit käytössä. Lisäksi etusivulle tuli vertikaalisesti vieritettävä lista, jonka lista-alkiot ovat kortteja. Seuraavassa kuvassa 4 on havainnollistettu etusivun komponenttihierarkia.

```
<SafeAreaProvider>
  <ButtonToSearch/>
  <ScrollView>
    <View>
      <Text/>
      <ListHorizontal/>
    </View>
    <View>
      <Text/>
      <ListHorizontal/>
    </View>
    <View>
      <Text/>
      <ListVertical/>
    </View>
  </ScrollView>
</SafeAreaProvider>
```

Kuva 4. HomeScreenin komponenttihierarkia

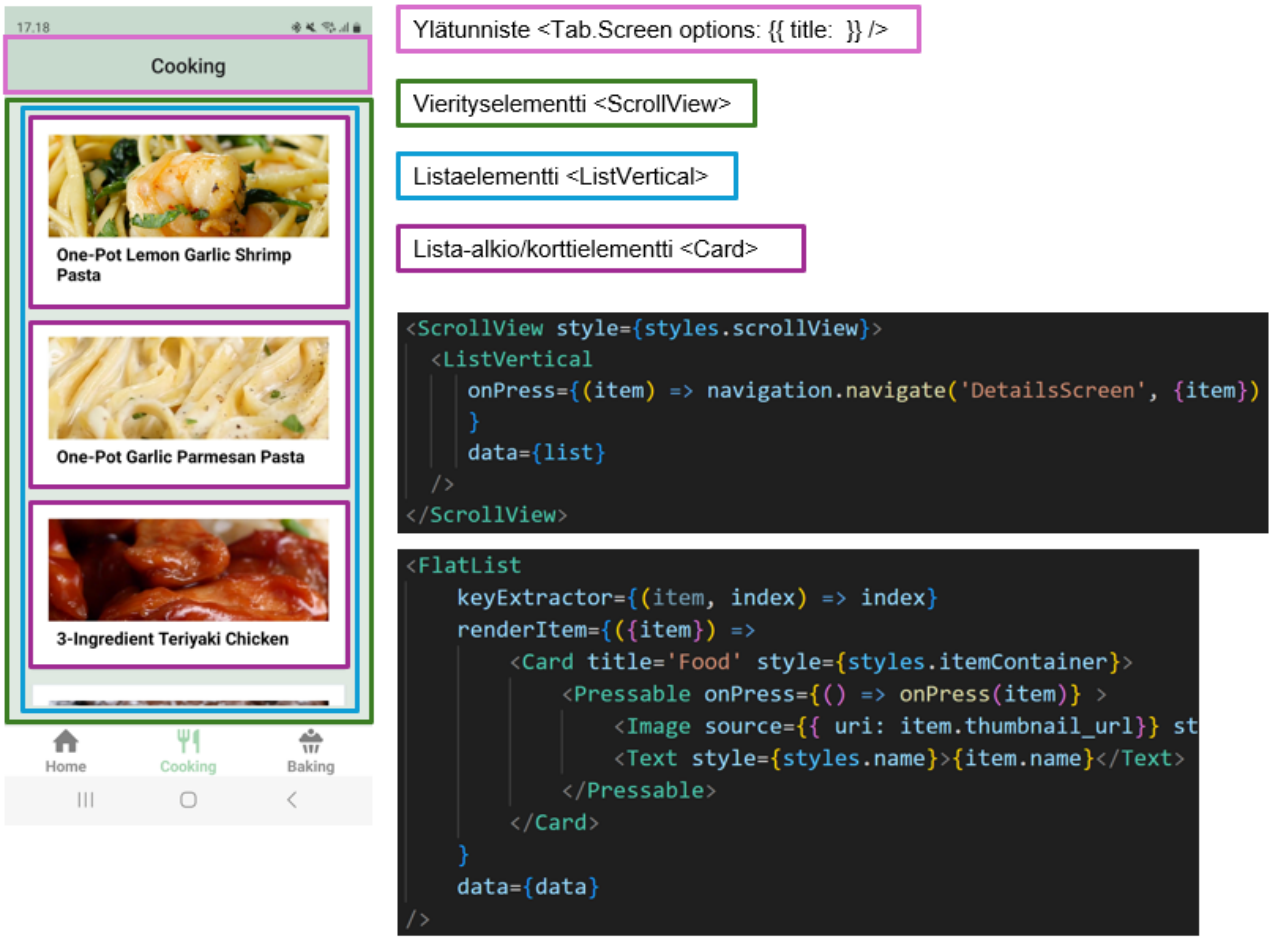
Saadakseen Tasty API:n rajapinnan toiminnallisuudet ja API key:n käyttöön, Rapid API -palveluun on kirjauduttava ja luotava käyttäjätunnus (Rapid API 2024). Lisäsin projektiin valitsemani rajapinnan käyttämällä tilamuuttujaa ja dotenv-tiedostoa, jotta API-key ei päätyisi versionhallintaan.

Fetch()-funktiolla hain http-rajapinnan kautta reseptin listauksen dinner- ja dessert-rajauksilla omille sivuilleen sekä etusivun reseptikarusellien kategoriat. Tasty API:n data on JSON-muotoista ja sisältää listauksia, joten siihen tutustuminen vaati paneutumista. Yhdellä reseptillä on valtavasti erilaista tietoa tallennettuna sisäkkäisiin listoihin. Rajapinnan hakupisteet (endpoints) on annettu palvelussa valmiina ja palvelu myös antaa esimerkkikoodin niiden käyttöön ja esimerkkidatarakenteen.

Valitsin sovelluksen värimaailmaksi vaaleanmurretunvihreän. Tekstillä ja näkyvillä elementeillä on vahva kontrasti. Hakukentän muotoilin riittävän suureksi (korkeus 50 px), jotta käyttäjän on helpompi osua kenttään. Vieritettävien elementtien viereen lisäsin vieritysviivan, joka mielestäni kertoo hyvin käyttäjälle vieritystilan.

Mobiilisovelluksen Cooking-näkymä on esitelty kuvassa 5, jossa sovelluksen komponentit on korostettu erivärisillä laatikoilla ja viereen on tuotu vastaava ote lähdekoodista. Käyttöliittymän ohjelmalogiikka kokonaisuudessaan funktioineen on nähtävillä GitHubissa. Baking-näytön ulkoasua ja komponentit ovat vastaavanlaisia kuin Cooking-näytön. Listaus on toteutettu vertikaalisesti

vieritettävinä korttielementteinä, joita napauttamalla pääsee reseptin tietoihin. Näkymien sisältö on eri, toiselle haetaan rajapinnan yli ruokaohjeita, toiseen leivontaohjeita.



Kuva 5. Sovelluksen Cooking-näkymän komponentit

Otin huomioon saavutettavuuden periaatteet jo mobiilisovelluksen suunnitteluvaiheessa ja käytin esimerkiksi näytönvierityspalkkia jokaisella sivulla. Lisäsin etusivun elementteihin accessibility-ominaisuudet, kun olin aluksi tehnyt toiminalliset elementit ja testannut niiden toimivuuden. Huomioin ruudunlukijan tarpeet myös navigaatiossa App.js-sivulla.

Details-näytölle pääsee näpäyttämällä listauksesta halutun reseptin kuvaa ja nimeä. Tiedot lähtevät parametrina Details-näkymään. Sivulta pääsee takaisin etusivulle näytön vasemman ylänurkan takaisinnuolella.

Päädyin rakentamaan reseptin haun omalle sivulleen. Etusivulla on reseptihaku-painike, joka näyttää tekstikentältä, mutta ohjaa käyttäjän uudelle sivulle. Sivulta pääsee takaisin etusivulle näytön vasemman ylänurkan takaisinnuolella. Jätin hakukentän vierityselementin ulkopuolelle, jotta se olisi vierittämisestä huolimatta näkyvillä ja nopeasti käytettävissä.

5.7 Testauksen toteutus

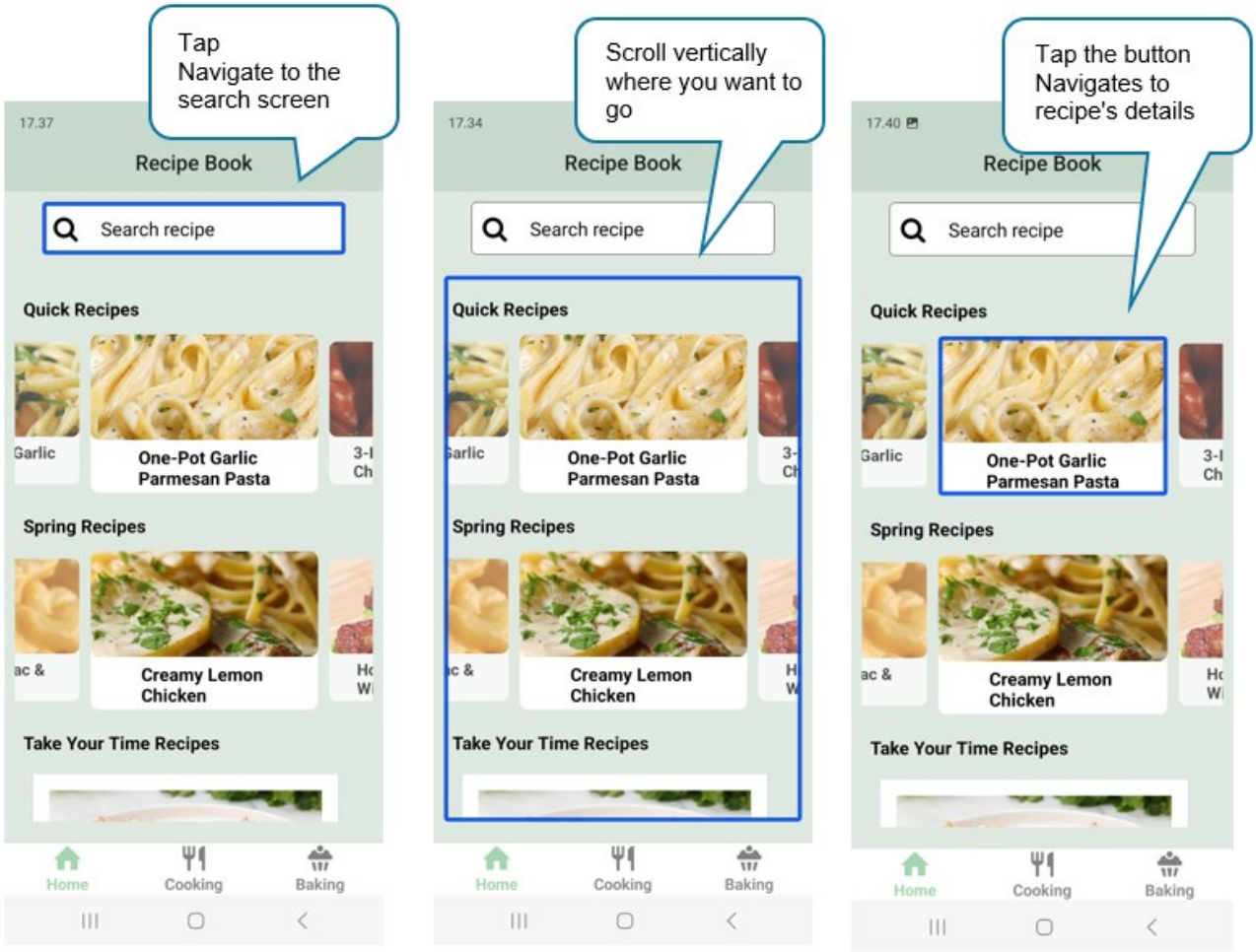
Toteutusvaiheessa lisäsin koodiin debuggausta varten console.log()-funktioita, jotka tulostivat halutun datan kehitysympäristön console-ikkunaan koodin virheettömyyden varmistamiseksi. Koodin kommentointia käytin myös väliaikaisten muistiinpanojen tekemiseen. Esimerkiksi tilanteessa, jossa en saanut jonkun toiminnallisuuden testausta loppuun asti tehtyä kertaheitolla, niin merkitsin selkeästi, että kyseinen komponentti on testaamaton.

Jokaisen komponentin ja toiminnallisuuden testasin työn edetessä. Seuraavissa alaluvuissa käydään läpi ruudunlukijalla testaus ja saavutettavuuskriteerien toteutuminen koko sovelluksessa.

5.7.1 Reseptisovelluksen etusivun testaus ruudunlukijalla

Opettelin käyttämään ruudunlukijan, TalkBackin, eleitä Googlen omasta ohjeistuksesta. Vaikuttaa, että ruudunlukija toimii hyvin intuitiivisesti, kunhan on aluksi ymmärtänyt sen toimintaperiaatteen ja toimintoja ohjaavat kosketuseleet. Olin lisännyt jokaisen etusivun elementtiin mahdollisimman paljon tietoa, jota käyttäjä voi hyödyntää ruudunlukijan avulla. Olin positiivisesti yllättynyt, että myös reseptikarusellin horisontaalinen vieritys toimi hyvin.

Kuvassa 6 on kolme kuvakaappausta ruudunlukijalla testauksesta. Ruudunlukija kohdistaa yhteen elementtiin kerralla ja esittää sen näytöllä sinisenä laatikkona. Puhekuplissa lukee ruudunlukijan ääneen lausuma sisältö. Aktiivisen kohteen, kuten painikkeen, saa toimimaan painamalla pitkään ja napauttamalla kahdella sormella. Vieritystoiminto toimii, kun liu’uttaa kahta sormea näyttöä pitkin.



Kuva 6. Esimerkki ruudunlukijan toiminnasta

Liitteenä olevassa taulukossa, Ruudunlukijalla testauksen tulokset, esittelen komponenttikohtaiset testitulokset (Liite 1). Ruudunlukija ilmoitti halutun informaation, joskin luotettavin tulos saavutettavuuden kannalta olisi varmasti tullut, jos sovelluksen olisi testannut autenttinen ruudunlukijan käyttäjä. Itse kehittäjä tuntee sovelluksen liian hyvin, joten todennäköisesti jotain jää huomaamatta. Kohdistus toimi kaikkiin elementteihin ja sitä pystyi vaihtamaan sormea liu'uttamalla. Painikkeet toimivat myös hyvin ruudunlukijan kosketuseleillä.

Kehityskohteena ilmeni, että ruudunlukija kohdisti hakukentän <Pressable>-komponentin sisällä olevaan tekstikomponenttiin, vaikka se ei ollut tarkoituksenmukaista toiminnan kannalta. Olisi riittänyt, että ruudunlukija kertoo pelkästään tekstin sisällön. Jäin pohtimaan myös, oliko kaikki komponenttien kohdistukset välttämättömiä ruudunlukijaa käyttävän näkökulmasta. Kiinnitin huomiota myös hakupainikkeen kokoon, joka voisi olla suurempi.

5.7.2 Saavutettavuuskriteerien testaus perusperiaatteiden mukaan jaoteltuna

Saavutettavuuskriteerit on käyty tarkemmin läpi luvussa 3. Seuraavassa verrataan sovelluksen toimintoja näihin ja nostetaan kehityskohteita. Kriteerien toteutuminen kussakin näkymässä on koottu luvun lopussa olevaan taulukkoon (taulukko 9).

Havaittavuusperiaatteeseen liittyvä ruudun pieni koko on otettu huomioon sovelluksessa, eikä näytöille ole ahdettu liikaa elementtejä tai toimintoja. Käyttäjän asettama suuri fonttikoko toimii hyvin ja teksti rivittyy nätisti estäen käyttäjää jäämästä jumiin yhden rivin ansaan. Pienin sovelluksessa käytetty fonttikoko on 15 pt, joka katsotaan pieneksi, jolloin ohjeen mukaan kontrastin täytyy olla isompi. Reseptisovelluksessa onkin pääsääntöisesti käytetty mustaa fontinväriä valkoisella taustalla. Suurennus- tai loitonnustoimintoja ei estetä sovelluksen lähdekoodissa ja ne toimivat hyvin laitteen asetusten mukaisesti.

Ulkoisen näppäimistön käyttö on yksi **hallittavuusperiaatteen** ohjeesta. Tässä sovelluksessa ei rajauksen vuoksi testattu ulkoista näppäimistöä eikä sitä ole huomioitu erikseen lähdekoodissa. Kosketuselementtien koko on isompi kuin suositeltu 9 mm * 9 mm ja etäisyys toisistaan on riittävä virhenapautuksen välttämiseksi. Reseptisovelluksessa on käytetty perinteisiä kosketuseleitä, jolloin käyttö on sujuvaa. Painikkeet on sijoiteltu niin, että niitä on mahdollista käyttää kummalla kädellä tahansa. Ainoastaan reseptinhakupainike on virtuaalisessa näppäimistössä oikealla puolella, mikä nousee kehityskohteeksi.

Ymmärrettävyysperiaatteeseen kuuluu näytön suunnan muuttaminen pysty- ja vaakasuuntaan. Sovellusta ei ole lukittu tiettyyn suuntaan, vaan käyttäjä voi halutessaan vaihtaa vaakasuuntaan ja sovellus skaalautuu sen mukaisesti. Toiminnot on aseteltu johdonmukaisesti, esimerkiksi toisiaan muistuttavat Cooking- ja Baking-näytöt toimivat samalla tavalla. Johdonmukaisuutta lisää myös se, että käytetyn navigaation muoto on yleisesti käytössä oleva. Tärkeäksi elementiksi on katsottu reseptihaku, joten se on asetettu etusivulle niin, ettei se peity näyttöä vieritettäessä tai suurennusta käytettäessä. Toimintoelementit on ryhmitelty yhteen (esimerkiksi kuva ja teksti), jolloin niistä on tullut isompi painike. Etusivun elementeissä toiminto on ilmoitettu selkeästi ruudunlukijalle. Muissa näytöissä ruudunlukija nappaa tekstielementit ja ilmoittaa niiden roolin ja sisällön, mutta ennalta tehdyn rajauksen mukaisesti toiminnot eivät ole ruudunlukijalla luettavissa. Sovelluksessa ei ole käytössä muokattuja kosketuseleitä.

Mobiilisovelluksen **toimintavarmuusperiaatetta** tukee virtuaalisen näppäimistön asettaminen vaaditun datatyypin mukaan. Reseptihaun tekstikentässä on määritelty keyboardTypen arvoksi ’default’, jolloin laite antaa numeronäppäimistön ja returnKeyTypen arvoksi ’search’. Yksi kriteeri on tiedonsyöttötapojen helpoksi tekeminen. Ainut syötekenttä sovelluksessa on reseptihaku ja se on

jätetty vapaaksi kentäksi. Hakua voisi kehittää saavutettavammaksi esimerkiksi nostamalla yleisimmät hakusanat tai kategoriat valintaruutujen taakse ja mahdollistamalla tarkemmat hakukriteerit ilman kirjoittamista tai lisäämällä automaattitäydennyksen, kun käyttäjä alkaa kirjoittamaan hakusanaa. Alustakohtaisia ominaisuuksia tuetaan sallimalla käyttöjärjestelmän asetukset, mutta itse sovelluksessa ei ole suurennustoiminnallisuusta tai mahdollisuutta fontin muuttamiseen.

Taulukko 9. Saavutettavuustestauksen tulokset (X = Saavutettavuuskriteeri täyttyy)

Saavutettavuuskriteeri	Etusivu	Cooking/Baking	Details	Haku
1. Havaittavuus				
1.1 Pieni ruudun koko	X	X	X	X
1.2 Suurennus ja loitonnus	X	X	X	X
1.3 Kontrasti	X	X	X	X
2. Hallittavuus				
2.1 Ulkoisen näppäimistön käyttö kosketusnäytön kanssa	-	-	-	-
2.2 Kosketuselementtien koko ja etäisyys toisistaan	X	X	X	X
2.3 Kosketusnäytön ohjauseleet	X	X	X	X
2.4 Laitteen manipuloinnin eleet	X	X	X	X
2.5 Painikkeiden sijoittaminen helppokäyttöisesti	X	X	X	-
3. Ymmärrettävyys				
3.1 Näytön suunnan muuttaminen pysty- tai vaakasuuntaan	X	X	X	X
3.2 Toimintojen johdonmukainen asettelu	X	X	X	X
3.3 Tärkeiden elementtien asettelu niin, ettei tarvitse vierittää	X	X	X	X
3.4 Toimintoelementtien ryhmittely saman toiminnon mukaan	X	X	X	X
3.5 Elementin toiminto osoitetaan selkeästi	X	-	-	-
3.6 Muokattavien kosketuseleiden käyttöön annetaan ohjeet	X	-	-	-
4. Toimintavarma				
4.1 Virtuaalinen näppäimistö asetetaan vaaditun datatyypin mukaan	X	X	X	X
4.2 Tiedonsyöttötavoista tehdään helppoja	X	X	X	-
4.3 Alustakohtaisia ominaisuuksia tuetaan	X	X	X	X

Saavutettavuustestaus osoittaa, että sovellus täyttää lähes kaikki saavutettavuuskriteerit. Ruudunlukija toimii muillakin näytöillä kuin etusivulla ja kohdistaa tekstikomponentteihin, mutta ei anna käyttäjälle lisäohjeita tai ilmoita komponenttien toiminnoista. Loput saavutettavuuskriteerit täyttäisi ruudunlukijan käytön tukeminen kaikilla näytöillä ja ulkoisen näppäimistön käytön varmistaminen. Näyttäisi siltä, että mobiilisovelluksen hyvä ja saavutettavuuskriteerit huomioiva suunnittelu parantaa jo yksistään merkittävästi sovelluksen saavutettavuutta.

6 Pohdinta

Olen tyytyväinen projektinhallintaan ja periaatteeseen toteuttaa pienin toimiva tuote (MVP). Tein jatkuvaa rajausta, jolloin projekti pysyi paremmin kasassa. Kanban-tyyppinen to do -listan päivittäminen vihkoon ja tehdyn työn merkkaaminen valmiiksi toimi hyvin. Projektin hyvä suunnittelu ja pilkkominen pieniksi osatehtäviksi auttoi olennaisesti onnistumisessa. Mukailin työskentelyssäni ketterän kehityksen periaatteita ja se toi myös työn sisältöön hyvää vaihtelua. Päivittäisessä työskentelyssäni käytin myös Pomodoro-tekniikkaa, joka rytmitti tekemistä ja muistutti huolehtimaan mikrotauoista.

Tarkoituksenani oli hyödyntää React Nativen react-18next-kirjastoa, jotta saan englanninkielisen datan käännettyä suomeksi. Vasta tehdessäni huomasin, että kirjasto käyttää paikallisia kehittäjän itsensä luomia JSON-tiedostoja käännöksiin, eikä suoraan mitään käännöspalvelua. Jos backend-datan haluaa käännettävän, paras tapa olisi kääntää ja pitää käännös backendin puolella. Tutkin erilaisia kääntäjävaihtoehtoja, mutta translate API:t ovat vain tiettyyn merkkimäärään saakka ilmaisia, jolloin törmäsin tilanteen mahdottomuuteen. Lisäksi pohdin, miten hidasta datan kääntäminen ja lataaminen olisi. Ratkaisin pulman vaihtamalla sovellukseni kielen englanniksi.

Avointa rajapintaa käyttäessä törmäsin sen rajallisuuteen. Olin optimistisesti ajatellut, että 500 kutsua kuukaudessa riittää tarpeisiini, mutta ylitin sen, mikä sai pohtimaan kaivaako lompakkoa vai odottaako, että ilmaiskutsut ovat jälleen käytössä. Loppujen lopuksi oman tietovaraston luominen ja ylläpito olisi ollut varteenotettava vaihtoehto. Myös ajatellen sovelluksen tulevaisuutta, tuotantoympäristöön soveltuva tietovarasto olisi etu sovelluksen julkaisemisessa sovelluskaupassa ja käyttöön saamisessa. Toisaalta olen kiitollinen, että opin REST-rajapinnan käytöstä ja JSON-datan käsittelystä.

Mobiilisovellus jäi melko yksinkertaiseksi toiminnallisuuksien osalta, vaikkakin toimii hyvin näinkin. Hakuominaisuutta voisi kehittää käyttäjäystävällisemmäksi lisäämällä hakukriteerejä. Olisi hienoa, jos käyttäjä voisi itse lisätä omia reseptejään, luoda suosikkilistoja, arvostella tai jakaa reseptejä. Myös kieliversiot olisivat hyvä lisä. Näihin sovellus tietysti tarvitsisi tietokannan. Olen tyytyväinen, että ohjelmointiosaamiseni syventyi ja opin paremmin React Nativen kirjastojen käyttöä.

Tuleva kehityskohde on saavutettavuuden parantelu ja ruudunlukijan käytön mahdollistaminen kaikille näytöille. Nyt sain jonkinlaisen näkemyksen saavutettavuudesta ja sen pariin olisi kiinnostavaa syventyä vieläkin tarkemmin. Myös ulkoisen näppäimistön mahdollistaminen sovelluksen käyttöön ja näppäimistökomentojen opettelu olisi mieluista oppia. Tietoisesti rajasin uuden oppimisen ruudunlukijaan, sen kosketuseleisiin ja saavutettavuuskriteereihin.

Sovelluksen laadukkaaseen testaukseen olisi myös hyvä panostaa, mikäli se julkaistaan joskus sovelluskaupassa. Kehittäjä itse ei ole ehkä se paras testaaja eikä tässä testauksessa otettu mukaan esimerkiksi poikkeuksia. Saavutettavuustestauskin jää mielestäni vajaaksi, koska itselläni ei ole kokemusta näkörajoitteista tai digitaalisesta apuvälineiden käytöstä. Toki tämä projekti antoi hyvän kuvan siitä ja vahvan perustan jatkaa tällä saralla.

Opinnäytetyön toteutus syvensi oppimaani ja kokosi eri aiheet hienosti yhteen. Oppimani jäsentyi hajanaisista tekniikoista kokonaiseksi projektiksi ja sen kyljessä opin saavutettavuuden perusteet ja ruudunlukijan käytön.

Lähteet

Aalpha 2024. Best Database for Mobile Apps in 2024. Luettavissa: https://www.aalpha.net/articles/best-database-for-mobile-apps/. Luettu 6.3.2024.

Aluehallintovirasto 2024a. Soveltamisala: kuulummeko lain piiriin? Luettavissa: https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/soveltamisala-kuulummeko-lain-piiriin/. Luettu 8.4.2024.

Aluehallintovirasto 2024b. WCAG 2.1: lain vaatimukset. Luettavissa: https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/wcag-2-1/. Luettu 8.4.2024.

Aluehallintovirasto 2024c. Verkkosisällön saavutettavuusohjeet (WCAG) 2.1. Luettavissa: https://www.w3.org/Translations/WCAG21-fi/#background-on-wcag-2. Luettu 8.4.2024.

Amazon Web Services 2024. Amazon DynamoDB. Luettavissa: https://aws.amazon.com/dynamodb/?nc2=h_ql_prod_db_ddb. Luettu 6.3.2024.

ApiList.fun 2019. API list […]. Luettavissa: https://apilist.fun/category/foodLuettu. Luettu 29.2.2024.

Appt Foundation 2024. Luettavissa: https://appt.org/en/docs/react-native/samples. Luettu 9.4.2024.

Auburn, M., Bryant, D. & Gough, J. 2022. Mastering API Architecture. O'Reilly Media, Inc. E-kirja. Luettu 29.2.2024.

Tanninen, T. 2024. Saavutettavuus on kilpailuetu. Luettavissa: https://avaava.fi/saavutettavuus-on-kilpailuetu/. Luettu 3.5.2024.

Cozmoslabs 2024. Best Alternative for Google Translate? 4 Top Options Compared in 2024. Luettavissa: https://translatepress.com/best-alternative-google-translate/. Luettu 11.4.2024.

Dabit, N. 2019. React Native in Action. Manning. USA.

Digi- ja väestötietovirasto 2023. Avoindata.fi. Luettavissa: https://www.avoindata.fi/fi. Luettu 1.3.2024.

Dodson 2018. Sematic and screen readers. Luettavissa: https://web.dev/articles/semantics-and-screen-readers. Luettu 19.4.2024.

GeeksforGeeks 2024. Introduction of DBMS (Database Management System) – Set 1. Luettavissa: https://www.geeksforgeeks.org/introduction-of-dbms-database-management-system-set-1/. Luettu 8.3.2024.

git s.a. git--everything-is-local. Luettavissa: https://git-scm.com/. Luettu: 11.4.2024.

GitHub 2024. Let's build from here. Luettavissa: https://github.com/about. Luettu 6.3.2024.

GitHub 2022. Public-apis. Luettavissa: https://github.com/public-apis/public-apis?tab=readme-ov-file#food--drink. Luettu 29.2.2024.

Google 2024a. TalkBackin laittaminen päälle. Luettavissa: https://support.google.com/accessibility/android/answer/6007100?hl=fi&ref_topic=10601570&sjid=6243930959234929233-EU. Luettu 19.4.2024.

Google 2024b. TalkBack-eleiden käyttäminen. Luettavissa: https://support.google.com/accessibility/android/answer/6151827?hl=fi&ref_topic=10601570&sjid=6243930959234929233-EU. Luettu 19.4.2024.

Google for Developers 2024. Learn the fundamentals. Luettavissa: https://firebase.google.com/docs. Luettu 8.3.2024.

Google Play 2023. SuperCook – Recipe Generator. Luettavissa: https://play.google.com/store/apps/details?id=com.supercook.app&hl=en_US. Luettu 29.2.2024.

Google Play 2024a. Kakkureseptit -Helppo sekoitus. Luettavissa: https://play.google.com/store/apps/details?id=com.riatech.cakerecipes&hl=fi&gl=US. Luettu 29.2.2024.

Google Play 2024b. Jälkiruoka Reseptit. Luettavissa: https://play.google.com/store/apps/details?id=com.riatech.dessertrecipes&hl=fi&gl=US. Luettu: 29.2.2024.

Google Play 2024c. Air Fryer Recipes. Luettavissa: https://play.google.com/store/apps/details?id=easy.airfryer.recipes&hl=en_US. Luettu: 29.2.2024.

Google Play 2024d. Gluteenittomat reseptit. Luettavissa: https://play.google.com/store/apps/details?id=com.cookware.glutenfreerecipes&hl=fi&gl=US. Luettu 29.2.2024.

Google Play 2024e. Gluteeniton: ruoka resepti. Luettavissa: https://play.google.com/store/apps/details?id=com.uzairrecipeapp.recipe.glutenfreerecipes&hl=fi&gl=US. Luettu 29.2.2024.

Google Store 2024. Google Play. Luettavissa: https://play.google.com/store/apps?hl=fi&gl=US. Luettu 6.3.2024.

Helsingin yliopisto 2021. Tietokantojen perusteet kevät 2021. Luettavissa: https://tikape.mooc.fi/kevat-2021/content/osa-1/index.html#relaatiomalli-ja-sql-kieli. Luettu 7.3.2024.

itewiki s.a. Mobiilisovellus. Luettavissa: https://www.itewiki.fi/opas/mobiilisovellus/. Luettu 11.4.2024.

i18next 2024. react-i18next documentation. Luettavissa: https://react.i18next.com/. Luettu 29.2.2024.

Kaupanliitto 2024. Luettavissa: https://kauppa.fi/uutishuone/2023/12/07/saavutettavuus-on-yrityk-selle-merkittava-kilpailuetu-valmistaudu-lain-muutoksiin-jo-hyvissa-ajoin/. Luettu 8.4.2024.

Kehitysvammaliitto 2024. Luettavissa: https://papunet.net/saavutettavuus/miksi-saavutettava/. Luettu: 3.5.2024.

Kesko 2024. K-Ruoka-sovellus – paras kauppa- ja kokkauskaveri. Luettavissa: https://www.k-ruoka.fi/artikkelit/sovellus/mobiilisovellus. Luettu: 29.2.2024.

Maadinfo Services 2024. Hallinnoi suosikkireseptejäsi webissä. Luettavissa: https://www.cook-mate.online/fi/home/. Luettu: 29.2.2024.

Massound Samy 2024. </> ApisList. Luettavissa: https://apislist.com/category/23/food-and-drink. Luettu 29.2.2024.

Medium 2023. Complete Guide to Internationalization in React using React i18n. Luettavissa: https://muhammedcuma.medium.com/complete-guide-to-internationalization-in-react-using-react-i18n-15bfd7d736e8. Luettu 11.4.2024.

Meta Platforms, Inc 2024a. React Native. Luettavissa: https://reactnative.dev/. Luettu 6.3.2024.

Meta Platforms, Inc 2024b. Accessibility. Luettavissa: https://reactnative.dev/docs/accessibility. Luettu 8.4.2024.

Microsoft 2024a. Code editing. Redefined. Luettavissa: https://code.visualstudio.com/. Luettu 6.3.2024.

Microsoft 2024b. What is Azure SQL Database? Luettavissa: https://learn.microsoft.com/en-us/azure/azure-sql/database/sql-database-paas-overview?view=azuresql. Luettu 8.3.2024.

Monash University ABN 2019. Get the App. Luettavissa: https://www.monashfodmap.com/ibs-central/i-have-ibs/get-the-app/. Luettu: 29.2.2024.

MongoDB 2024. Serverless Databases Explained. Luettavissa: https://www.mongodb.com/databases/serverless-database. Luettu 8.3.2024.

Nomensa 2005. What is a screen reader? Luettavissa: https://www.nomensa.com/blog/what-screen-reader/. Luettu 19.4.2024.

Näkövammaisten liitto 2021. Suurin osa näkövammaisista kokee digipalvelut haastaviksi käyttää. Luettavissa: https://www.nakovammaistenliitto.fi/fi/artikkeli/suurin-osa-nakovammaisista-kokee-digipalvelut-haastaviksi-kayttaa. Luettu 3.5.2024.

Oikeusministeriö 2019. Laki digitaalisten palvelujen tarjoamisesta. Luettavissa: https://www.finlex.fi/fi/laki/alkup/2019/20190306. Luettu 3.5.2024.

Open API-työryhmä s.a. Avoin rajapinta. Luettavissa: http://avoinrajapinta.fi/. Luettu 29.2.2024.

Petrov, A. 2019. Database Internals. O'Reilly Media, Inc. E-kirja. Luettu 7.3.2024.

RapidAPI 2024. Tasty. Luettavissa: https://rapidapi.com/apidojo/api/tasty. Luettu 29.2.2024.

Sapega 2021. What Is a Screen Reader and Why Is It Important? Luettavissa: https://www.tpgi.com/what-is-a-screen-reader-and-why-are-they-important/. Luettu 19.4.2024.

Stackscale B.V. 2024. 10 popular database management systems (DBMS). Luettavissa: https://www.stackscale.com/blog/popular-database-management-systems/. Luettu 7.3.2024.

TechMagic 2022. JavaScript for Mobile Apps: Choose the Perfect Framework. Luettavissa: https://www.techmagic.co/blog/why-use-javascript-for-mobile-apps/. Luettu 6.3.2024.

TechRev Blog 2023. Top 5 Databases for Mobile App Development. Luettavissa: https://blog.techrev.us/top-5-databases-for-mobile-app-development/. Luettu 6.3.2024.

TheMealDB 2024. TheMealDB. Luettavissa: https://www.themealdb.com/api.php. Luettu 29.2.2024.

Tudorspan Limited 2024. Helpoin tapa järjestää reseptejäsi. Luettavissa: https://recipekeeperonline.com/. Luettu: 29.2.2024.

Valio Oy 2024. Lataa Valio sovellus, tuhansien reseptien koti. Luettavissa: https://www.valio.fi/sovellus/. Luettu: 29.2.2024.

Valtioneuvosto 2023. Tavoitteena yhdenvertainen ja tasa-arvoinen tulevaisuus. Luettavissa: https://valtioneuvosto.fi/-/1410829/tavoitteena-yhdenvertainen-ja-tasa-arvoinen-tulevaisuus. Luettu: 3.5.2024.

World Health Organization 2023. Disability. Luettavissa: https://www.who.int/news-room/fact-sheets/detail/disability-and-health. Luettu 3.5.2024.

W3C 2024. Mobile Accessibility: How WCAG 2.0 and Other W3C/WAI Guidelines Apply to Mobile. Luettavissa: https://www.w3.org/TR/mobile-accessibility-mapping/. Luettu 8.4.2024.

650 Industries, Inc. 2024. Develop an app with Expo. Luettavissa: https://docs.expo.dev/work-flow/overview/. Luettu 6.3.2024.

Liitteet

Liite 1. Taulukko: Ruudunlukijalla testauksen tulokset

Elementti	Saavutettavuusominaisuudet	Odotettu toiminta	Toiminta laitteessa
<Stack.Navigator>	accessible={true} accessibilityLabel="This is header" accessibilityRole="header title: 'Recipe Book'	Ruudunlukija ilmoittaa roolin ja sisällön.	Ruudunlukija aloittaa ylätunnisteesta. Toiminta odotettua.
<Pressable>	accessibilityRole="button" accessibilityLabel="Tap" accessibilityHint='Navigate to search screen'	Ruudunlukija ilmoittaa painikeroolin ja tarkoituksen	Toiminta odotettua muuten, paitsi ruudunlukija kohdistaa turhaan tekstin syöttökenttään. Muu huomio: Onko painike liian pieni?
<ScrollView>	accessibilityRole="grid" accessibilityLabel='Scroll vertical' accessibilityHint='Scroll the view to the top reviewing whole content'	Ruudunlukija ilmoittaa, että sisältöä pystyy vierittämään vertikaalisesti	Toiminta odotettua.
<View>, jonka sisällä <Text> ja <Carousel> <Carousel>:n sisällä listaelementtinä käytössä <Pressable>	<View>:n sisällä accessible={true} <Text>:n sisällä accessible={true} accessibilityRole='text' <Carousel>:n sisällä accessible={true} accessibilityLabel="Carousel" accessibilityHint="Scroll to the left to see recipies and back to the right" accessibilityRole="adjustable"	Ruudunlukija yhdistää tekstikentän karuselliin, ilmoittaa tekstikentän sisällön ja roolin sekä karusellin roolin ja käyttöohjeen. Jokaisesta listaelementistä ruudunlukija ilmoittaa painikkeen roolin ja ohjeistuksen, tekstikentän roolin sekä tekstisisällön.	Toiminta odotettua.

Elementti	Saavutettavuusominaisuudet	Odotettu toiminta	Toiminta laitteessa
	<Pressable>: sisällä accessible={true} accessibilityRole="imagebutton" accessibilityLabel="Tap the button" accessibilityHint="Navigates to recipe's details"		
<View>, jonka sisällä <Text> ja <Carousel> <Carousel>:n sisällä listaelementtinä käytössä <Pressable>	<View>:n sisällä accessible={true} <Text>:n sisällä accessible={true} accessibilityRole='text' <Carousel>:n sisällä accessible={true} accessibilityLabel="Carousel" accessibilityHint="Scroll to the left to see recipies and back to the right" accessibilityRole="adjustable" <Pressable>: sisällä accessible={true} accessibilityRole="imagebutton" accessibilityLabel="Tap the button" accessibilityHint="Navigates to recipe's details"	Ruudunlukija yhdistää tekstikentän karuselliin, ilmoittaa tekstikentän sisällön ja roolin sekä karusellin roolin ja käyttöohjeen. Jokaisesta listaelementistä ruudunlukija ilmoittaa painikkeen roolin ja ohjeistuksen, tekstikentän roolin sekä tekstisisällön.	Toiminta odotettua.
<View>, jonka sisällä <Text> ja <View>, jonka sisällä on <FlatList> Listaelementtinä on <Pressable>	<View>:n sisällä accessible={true} <Text>:n sisällä accessible={true} accessibilityRole='text' <FlatList>:n sisällä accessible={true}	Ruudunlukija yhdistää tekstikentän listaan, ilmoittaa tekstikentän sisällön ja roolin sekä listan roolin ja käyttöohjeen. Jokaisesta listaelementistä ruudunlukija ilmoittaa painikkeen roolin ja ohjeistuksen,	Toiminta odotettua.

Elementti	Saavutettavuusominaisuudet	Odotettu toiminta	Toiminta laitteessa
	accessibilityRole='grid' accessibilityLabel="List" accessibilityHint="Scroll vertically to review recipies" <Pressable>:n sisällä accessible={true} accessibilityRole="imagebutton" accessibilityLabel='Tap' accessibilityHint='Navigate to recipe details'	tekstikentän roolin sekä tekstisisällön	