Veera-Kerttu Kamula

Saavutettavuusvaatimusten soveltaminen design systemin kehityksessä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Insinöörityö

26.11.2024

Tiivistelmä

Tekijä:	Veera-Kerttu Kamula
Otsikko:	Saavutettavuusvaatimusten soveltaminen design systemin kehityksessä
Sivumäärä:	59 sivua
Aika:	26.11.2024
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Mediatekniikka
Ohjaajat:	Lehtori Ulla Sederlöf

Insinöörityössä selvitettiin digitaalisten palveluiden saavutettavuusvaatimusten, erityisesti WCAG 2.2 A- ja AA-tason vaatimusten, ja alan parhaimmaksi todettujen käytänteiden toteuttamisesta design systemin suunnittelussa ja kehityksessä. Työn taustalla oli digitaalisten palveluiden esteettömyysdirektiivin laajeneminen ja siitä johtuva tavoite syvemmän saavutettavuusosaamisen kasvattaminen ja ymmärrys, millaisin käytäntein saavutettavuus voidaan integroida osaksi nykyaikaista verkkokehitystä design systemeillä.

Työssä selvitettiin saavutettavuusvaatimusten pääperiaatteet sekä, mikä design systemi on riittävän tietopohjan luomiseksi aiheesta. Työn aikana tutkittiin, miten saavutettavuusintegraatio design systemissä voi parantaa käyttäjäkokemusta ja luoda tasavertaisia kokemuksia kaikille käyttäjille. Lisäksi selvitettiin, millaisia saavutettavuuspuutteita verkkopalvelut kohtaavat. Yleisiksi ongelmiksi voitiin päätellä värien heikko saavutettavuus, näppäimistöllä navigoinnin puutteet sekä lomakkeisiin liittyvät puutteelliset virheidenkäsittelyt.

Insinöörityön aikana kehitettiin oma design systemi pilottiprojektina, jonka saavutettavuuden tavoitteeksi otettiin WCAG 2.2 AA-taso. Saavutettavan design systemin kehittämisen keskeiset havainnot osoittivat, että saavutettavuuden ja design systemien yhdistäminen vaatii huolellista suunnittelua ja teknistä toteutusta.

Havainnoista voitiin päätellä, että saavutettavuuden huomioiminen design systemin suunnittelussa ja kehityksessä alusta alkaen tehostaa laadunvarmistamista ja myöhemmin saavutettavuuden auditonnista johtuvaa työkuormaa. Havaittiin, että vaikka yksittäiset käyttöliittymäkomponentit olisivat saavutettavia, niiden käytännön sovellukset voivat silti aiheuttaa saavutettavuuspuutteita. Työn tuloksena voitiin myös todeta, että saavutettavuus design systemeissä vaatii iterointia ja johdonmukaista dokumentaatiota, jotta design systemiä käyttävillä työryhmillä on kaikilla yhteiset tavoitteet ja ohjeet saavutettavuuden toteutumiselle.

Avainsanat: Saavutettavuus, WCAG, design system, käyttöliittymäsuunnittelu, verkkokehitys

---

Abstract

Author:	Veera-Kerttu Kamula
Title:	Applying Accessibility Standards to Design System Development
Number of Pages:	59 pages
Date:	26 November 2024
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technology
Professional Major:	Media Technology
Supervisors:	Ulla Sederlöf, Senior Lecturer

In this final year project, accessibility requirements were investigated for designing and developing an accessible design system. The purpose of the final year project was to develop deeper understanding of how accessibility can be integrated into modern web development through design systems as accessible products are more in demand because of the European Accessibility Act's goal to make life easier for people with disabilities.

The study examined the main principles of accessibility requirements and what a design system is to gain sufficient knowledge on the topic. Moreover, the thesis investigated how integrating accessibility into a design system can enhance user experience and what some of the most common accessibility issues that web services face are. The issues included poor colour accessibility, lack of keyboard navigation and inadequate error handling in forms. In addition, a small accessible design system was developed as a pilot project.

The key results showed that the development of an accessible design system requires careful designing and technical implementation from start to finish to create accessible products. In conclusion, developing an accessible design system needs iteration and careful accessibility auditing to ensure that both individual user interface components and more complex patterns are accessible.

Keywords: Accessibility, WCAG, design system, user interface design, web development

Sisällys

Lyhenteet

WCAG: *Web Content Accessibility Guidelines.* Saavutettavuusohjeistus, jota kehittää ja ylläpitää *World Wide Web* -konsortti (W3C). Saavutettavuusohjeistusta noudattamalla voidaan varmistaa verkkopalvelun ja -sisällön minimitavoite saavutettavuudessa.

HTML: *Hypertext Markup Language*. Standardoitu merkintäkieli asiakirjoille, jotka on suunniteltu näytettäväksi verkkoselaimissa.

CSS: *Cascading Style Sheets*. Verkkokehityksessä käytetty merkintäkieli, jolla tyylitellään HTML-asiakirjoja.

1 Johdanto

Saavutettavuuden ja esteettömyyden tarve digitaalisessa ympäristössä on erityisen tärkeää, kun otetaan huomioon ikääntyvä väestö sekä erilaisten toimintarajoitteiden kanssa elävät käyttäjät. Saavutettavuus tarkoittaa, että digitaaliset palvelut ja sovellukset ovat kaikkien saatavilla ja käytettävissä riippumatta fyysisistä tai kognitiivisista rajoitteista. Lisäksi esteettömyys on EU-lakisääteinen direktiivi, joka velvoittaa, että viranomaisten ja yksityisen sektorin tietyt tuotteet ja palvelut kehitetään esteettömiksi.

Tässä insinöörityössä tutkitaan saavutettavuusvaatimusten, erityisesti WCAG 2.2 (*Web Content Accessibility Guidelines*) A- ja AA-tason vaatimusten ja alan parhaiden käytänteiden toteuttamista sekä saavutettavuuden soveltamista design systemin suunnittelussa ja kehityksessä. WCAG on World Wide Web -konsortin (W3C) kehittämä ja ylläpitämä saavutettavuusohjeistus, jota noudattamalla voidaan varmistaa verkkopalvelun ja -sisällön minimitavoite saavutettavuudessa. Aihetta tarkastellaan sekä suunnittelijan että kehittäjän näkökulmasta. Design systemit tarjoavat johdonmukaisen ja skaalautuvan lähestymistavan käyttöliittymien suunnitteluun ja kehittämiseen, minkä vuoksi ne ovat nykyaikaisen kehitystyön kulmakivi.

Työn tavoitteena on tutkia, kuinka saavutettavuusvaatimukset voidaan integroida design systemin suunnitteluun sekä kehittämiseen ja miten näiden vaatimusten noudattaminen parantaa digitaalisten palveluiden käytettävyyttä. Lisäksi työssä tunnistetaan yleisiä saavutettavuuspuutteita, jotka voivat estää saavutettavuuden täysimääräisen toteutumisen.

Tarkoituksena on selvittää ja kehittää konkreettisia saavutettavia ratkaisuja design systemin eri osa-alueilla, kuten värikontrastien, lomake-elementtien ja virheen käsittelyn toteutumisessa käytännön projektissa. Työssä kehitetään oma design systemi pilottiprojektina, jonka avulla pyritään toteuttamaan lomakepatternit kehitetyn design systemin elementeillä siten, että ne täyttävät saavutettavuusvaatimukset. Työ keskittyy saavutettavuuden kannalta erityisesti

teknisiin ratkaisuihin, jotka ovat mitattavissa erilaisilla auditointityökaluilla, eivätkä syvenny esimerkiksi kognitiivisiin haasteisiin ja kielen saavutettavuuteen.

Insinöörityö tarjoaa katsauksen saavutettavuuden ja design systemien väliseen suhteeseen. Työllä pyritään edesauttamaan digitaalisten palvelujen saavutettavuuden parantamista ja tukemaan sekä suunnittelijoita että kehittäjiä rakentamaan kaikille tasavertaisen pääsyn digitaaliseen ympäristöön.

2 Saavutettavuuden periaatteet verkkopalveluissa

2.1 Saavutettavuus verkkopalveluissa ja -sisällöissä

Saavutettavassa verkkopalvelussa ja -sisällössä varmistetaan, että palvelut ja niiden sisältö ovat kaikkien käyttäjien käytettävissä ja havaittavissa mahdollisimman helposti (1). Saavutettavia palveluita toteutettaessa on huomioitava niiden helppokäyttöisyys, tekninen toteutus sekä sisältöjen selkeys ja ymmärrettävyys. Verkkosisällön tekninen saavutettavuus tarkoittaa sitä, että lähdekoodi on virheetöntä, loogista ja noudattaa verkkosisältöjen merkkauskielen, HTML-merkistön (*Hypertext Markup Language*) standardeja sekä WCAG-ohjeistusta. Lisäksi teknisesti hyvin toteutettu palvelu toimii erikokoisilla päätelaitteilla sekä avustavilla teknologioilla. Helppokäyttöinen palvelu on helppo hahmottaa, ja sen navigointi on vaivatonta. Tämä tarkoittaa esimerkiksi sitä, että navigaatio ei ole liian monitasoinen, pääsisältö erottuu muista elementeistä ja palvelussa on helppoa suorittaa haluttu toiminto. Ymmärrettävyydellä tarkoitetaan, että sisältö on kaikkien käyttäjien helposti ymmärrettävissä. Tämä saavutetaan selkeän kielen käytöllä ja riittävällä sisällön jäsentelyllä. (2.)

Verkkosisältöjen saavutettavuusongelmat voivat johtua erilaisista tekijöistä, kuten huonosta teknisestä toteutuksesta, joka estää palvelussa navigoinnin näppäimistön avulla, riittämättömästä kontrastista värien välillä tai sisällön skaalaamattomuudesta, joka hankaloittaa sisällön havainnointia

heikkonäköisille käyttäjille. On tärkeä muistaa, että saavutettavuus on aina tilanteen ja käyttäjän tarpeista riippuvaista. Tämän vuoksi saavutettavuuden perusajatuksena voidaan pitää sitä, että palvelut tehdään mahdollisimman monen käyttäjän tarpeet huomioiviksi, jotta sisältöä voidaan käyttää sujuvasti. (3.) Tästä syystä on tärkeää tarjota sisältö useammalle kuin yhdelle aistille, jotta palveluiden käyttö on yhdenvertaista kaikille, riippumatta käyttäjän rajoitteista (1). Eri käyttäjillä voi olla monia erilaisia vammoja ja toimintarajoitteita. Yleisiä verkkosisältöjen käyttöön vaikuttavia toimintarajoitteita ja vammoja ovat näkö- ja kuulorajoitteet, fyysiset ja motoriset rajoitteet, puhevammat, kognitiiviset haasteet sekä neurologiset sairaudet. Lisäksi iällä on suuri vaikutus tietoteknisiin taitoihin, sillä ikääntyessä näkö-, kuulo-, kognitiiviset ja motoriset rajoitteet yleensä lisääntyvät. (4.)

Saavutettavuuden voidaan ajatella myös edistävän käyttäjän kokemusta riippumatta siitä, onko hänellä rajoitteita vai ei. Saavutettavuuden periaatteilla voidaan esimerkiksi edistää tilanteita, joissa käyttäjä käyttää vanhaa päätelaitetta tai selainta, tai jossa tietoliikenneyhteys on hidas. (4.)

Suomessa on otettu käyttöön digipalvelulaki, joka velvoittaa julkisen sektorin toimijat sekä osan yksityisen ja kolmannen sektorin toimijoista tarjoamaan digitaaliset palvelunsa ja sisältönsä saavutettavuusvaatimusten mukaisesti. Digipalvelulain mukaan digitaalisten palveluiden ja sisältöjen on täytettävä eurooppalaisen standardin EN 301 549 tekniset vaatimukset, jotka viittaavat WCAG-ohjeistukseen ja velvoittavat noudattamaan sen A- ja AA-tason vaatimuksia. Lisäksi palveluista täytyy löytyä saavutettavuusseloste, jossa kerrotaan, missä määrin palvelu ja sen sisältö ovat saavutettavia ja mitä puutteita saavutettavuudessa on. Laki myös edellyttää, että käyttäjillä on mahdollisuus antaa palautetta saavutettavuudesta. (5.)

Digipalvelulaki laajenee Euroopan unionin esteettömyysdirektiivin myötä. Esteettömyysdirektiivi on merkittävä askel kohti yhdenvertaista digitaalista ympäristöä, sillä sen tavoitteena on varmistaa, että yhä useammat viranomaisten ja yksityisen sektorin tuotteet ja palvelut ovat esteettömiä.

Direktiivi koskee monia erilaisia tuotteita ja palveluita, kuten tietokoneita ja niiden käyttöjärjestelmiä, itsepalvelupäätteitä, kuluttajan pankkipalveluita sekä verkkokauppoja. Direktiivi on tullut kansallisesti toimeen viimeistään 28.6.2022 jokaisessa EU-maassa, ja vaatimusten soveltaminen alkaa 28.6.2025. Tämän jälkeen uusien tuotteiden ja palveluiden, joita direktiivi koskee, on vastattava esteettömyysvaatimuksia. (6.)

2.2 WCAG 2 -ohjeistuksen ymmärtäminen ja periaatteet

WCAG 2 -ohjeistus on kehitetty tarjoamaan yksilöille, organisaatioille ja valtioille yhteinen jaettu standardi verkkosisällön saavutettavuudelle kansainvälisellä tasolla. WCAG-ohjeistuksessa selitetään, miten verkkosisältö voidaan toteuttaa saavutettavammin käyttäjille, joilla on rajoitteita. WCAG:n uusin versio, 2.2, julkaistiin 05.10.2023. (7.)

Ohjeistuksessa on neljä periaatetta: havaittava, hallittava, ymmärrettävä ja toimintavarma. Jokaisen periaatteen alla on lista ohjeista, jotka koskettavat kyseistä periaatetta. (8.) Ohjeistuksessa käytetään kolmea eritasoista testattavaa onnistumiskriteeriä: A, AA ja AAA (7). Jokaisessa ohjeessa annetaan tarkka kuvaus siitä, miten kyseinen standardi saavutetaan. Osa testauksista voidaan automatisoida ohjelmallisesti, mutta osa vaatii manuaalista testausta osittain tai kokonaan. (8.)

WCAG 2 on tarkoitettu tekniseksi standardiksi esimerkiksi verkkosisällön kehittäjille ja verkon työkalujen kehittäjille sekä työkaluksi verkkopalvelun ja -sisältöjen saavutettavuuden arviointiin (7). WCAG 2 -ohjeistuksen A- ja AA-tason vaatimukset eivät kuitenkaan ota kantaa verkkosisällön ymmärrettävyyteen tai palvelun käytettävyyteen, jotka ovat tärkeitä saavutettavuuden kannalta. Tästä syystä voidaankin todeta, että WCAG-vaatimusten noudattaminen varmistaa saavutettavuuden minimitason, ja A- ja AA-tason vaatimusten noudattaminen parantaa etenkin verkkopalvelun teknistä saavutettavuutta. Lisäksi on hyvä huomioida, että jotkin ohjeistukset ovat

tulkinnanvaraisia, ja ammattilaisilla voi olla eriäviä mielipiteitä niiden toteutustavoista. (9.)

Havaittava

Yksi WCAG 2 -ohjeistuksen periaatteista on, että verkkopalvelun ja -sisällön tulee olla havaittavia. Tämä tarkoittaa sitä, että tieto ja käyttöliittymäkomponentit on esitettävä käyttäjälle siten, että hän voi havaita ne, eikä niiden tule olla näkymättömiä kaikille käyttäjän aisteille. (8.) WCAG-ohjeistuksen mukaan havaittavuuden periaatteen mukaisia ohjeita ovat korvaavien tekstivastineiden tarjoaminen, korvaavien esitystapojen tarjoaminen aikasidonnaiselle medialle sekä sisällön mukautettavuus ja erotettavuus (10).

Hallittava

Hallittavuuden periaate WCAG-ohjeistuksessa tarkoittaa, että verkkopalvelun käyttöliittymäkomponenttien ja navigoinnin tulee olla helposti käytettävissä. Käyttäjän on pystyttävä käyttämään käyttöliittymää ilman, että palvelu edellyttää vuorovaikutusta, jota käyttäjä ei voi suorittaa. (8.) WCAG:n hallitavuusperiaatteen mukaiset ohjeet edellyttävät, että palvelua voidaan käyttää näppäimistöllä, käyttäjille annetaan riittävästi aikaa toimintoihin, sisältö ei saa aiheuttaa sairauskohtauksia, palvelun on oltava navigoitavissa, ja käyttäjälle on tarjottava helppoja syötetapoja näppäimistön lisäksi (10).

Ymmärrettävä

Käyttäjälle esitettävän tiedon ja käyttöliittymän toiminnan tulee olla ymmärrettävää. Tämä tarkoittaa sitä, että käyttäjän on kyettävä ymmärtämään sekä tieto että käyttöliittymän toiminta, eivätkä ne voi olla hänen ymmärryksensä ulkopuolella. (8.) Ymmärrettävyyden periaatteeseen liittyvät ohjeet koskevat sisällön luettavuutta ja ymmärrettävyyttä, verkkopalvelun ennakoitavuutta sekä riittävää avustusta syötteille (10).

Toimintavarma

Toimintavarmalla verkkopalvelulla ja -sisällöllä varmistetaan, että ne on tuotettu kestävällä tavalla, jotta useimmat käyttäjäagentit ja avustavat teknologiat voivat tulkita palveluita ja niiden sisältöä luotettavasti. Käyttäjän on siis voitava päästä käsiksi sisältöön, vaikka tekniikka kehittyisi. (8.)

Toimintavarmuuden ohjeen tarkoituksena on tukea yhteensopivuutta nykyisten ja tulevien käyttäjäagenttien, erityisesti avustavien teknologioiden kanssa. Tämä saavutetaan varmistamalla, että kehittäjät käyttävät oikeaa merkintäkieltä ja välttävät huonosti muodostettua merkintäkieltä tai koodia, joka voisi estää avustavan teknologian toiminnan. (11.)

3 Nykyaikainen webkehitys design systemeillä

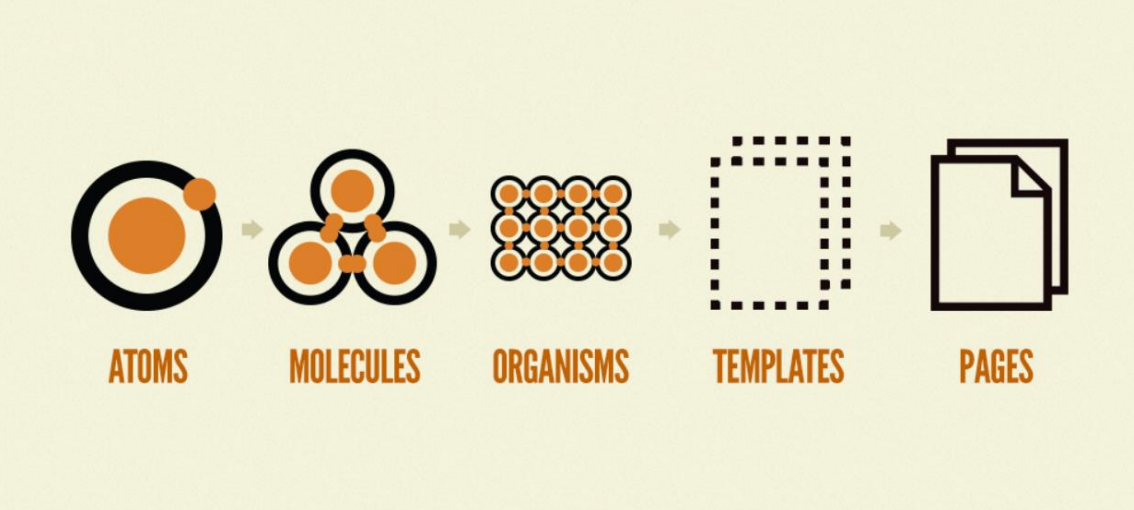
3.1 Mikä on design systemi

Design systemi on joukko standardeja, jotka on luotu hallitsemaan ja skaalaamaan käyttöliittymäsuunnittelua ja -kehitystä. Design systemit kokoavat yhteen dokumentaatioon suunnitteluperiaatteet, uudelleenkäytettävät käyttöliittymäkomponentit sekä ohjeistukset, joiden avulla pyritään saavuttamaan johdonmukaisuutta ja tehokkuutta digitaalisissa tuotteissa. Ne parantavat työnkulkua, helpottavat yhteistyötä ja auttavat ylläpitämään brändi-identiteettiä. (12.) Design systemit vastaavat tarpeeseen, jossa organisaatioiden on pitänyt tehostaa suunnitteluprosessia kasvavan käyttöliittymäsuunnittelun tarpeen vuoksi (13).

Design systemit tarjoavat yhtenäisen tavan toteuttaa käyttöliittymiä, ja ne voivat sisältää esimerkiksi käyttöliittymän värit ja typografian, käyttöliittymän ikoniikan, yleisimpiä käyttöliittymäkomponentteja, kuten painikkeita ja lomake-elementtejä, ja asetteluohjeistuksia. Design systemin sisältö riippuu kuitenkin työryhmän ja digitaalisten tuotteiden tarpeista. Joissakin tapauksissa design systemi voi laajentua visuaalisten elementtien ulkopuolelle ja ohjeistaa myös sisällön tuotantoon ja kielenkäyttöön liittyvissä asioissa. (12.)

3.2 Design systemien arkkitehtuuri

Design systemin perustana voidaan pitää atomista suunnittelua (eng. atomic design), jossa käyttöliittymäsuunnittelu puretaan osiksi, ja yksittäiset osat muodostavat komponenteja ja laajempia kokonaisuuksia. Atominen suunnittelu jakaa käyttöliittymän atomeihin, molekyyleihin, organismeihin, malleihin ja sivuihin (ks. kuva 1). (14.)



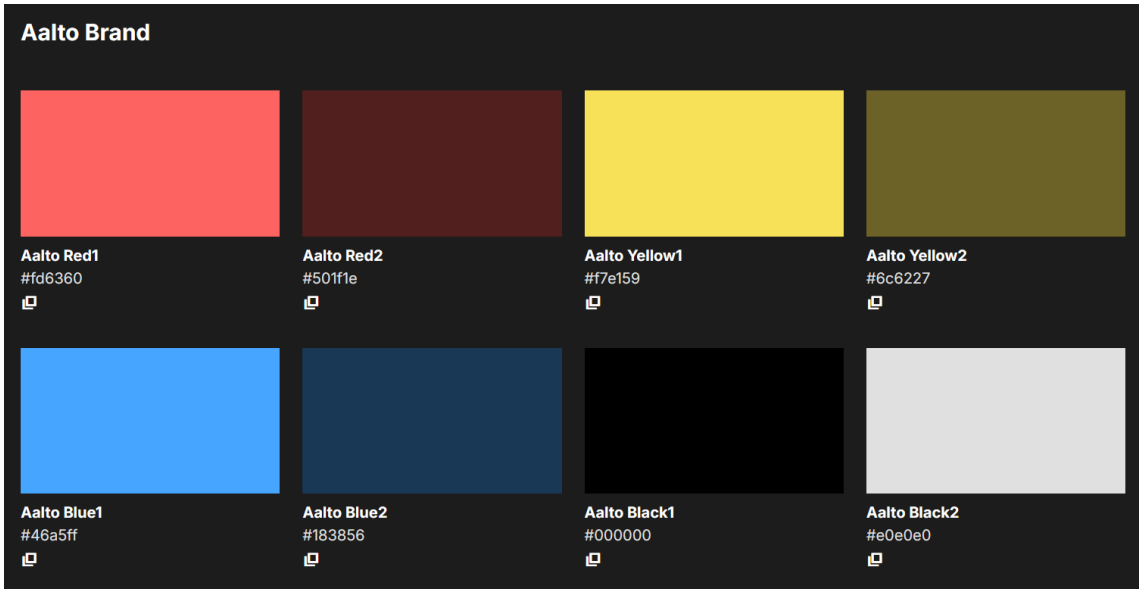
Kuva 1. Atominen suunnittelu perustuu siihen, että käyttöliittymäsuunnittelu koostuu atomeista, molekyyleistä, organismeista, malleista ja sivuista (14).

Yksinkertaistetusti atomit ovat HTML-merkintäkielen perusosia, kuten painikkeet ja tekstisyötekentät. Lisäksi atomit voivat sisältää suunnitteluperiaatteita, kuten värejä ja typografiaa. Atomeista muodostuvat molekyylit voivat olla esimerkiksi käyttöliittymän hakukenttiä. Molekyylien nyrkkisääntönä on, että niiden tarkoitus on toteuttaa yksi asia, eli ne ovat suhteellisen yksinkertaisia kokonaisuuksia. Molekyylit muodostavat edelleen uudelleenkäytettäviä rakenteita, joita yhdistämällä luodaan organismeja. Organismit ovat monimutkaisempia kokonaisuuksia, jotka toimivat erillisinä käyttöliittymän osina. Esimerkiksi verkkopalvelun ylätunnistetta voidaan pitää organismina, koska se saattaa sisältää useita molekyylejä, kuten päänavigoinnin, hakupalkin ja logon. (14.)

Mallit ovat joukko organismeja, jotka muodostavat esimerkkikokonaisuuksia sivuista. Mallit voivat olla esimerkiksi HTML-sivujen rautalankaversioita, joista saa hyvän käsityksen sivun kokonaisuudesta. Sivut puolestaan ovat tiettyjä malleja, joissa esimerkkisisältö korvataan oikealla sisällöllä, jotta saadaan tarkka kuva siitä, mitä käyttäjä näkee. (14.)

Atomisen suunnittelun tavoin design systemin arkkitehtuuri perustuu uudelleenkäytettävistä tyyleistä, komponenteista ja kokonaisuuksista. Design systemi koostuu tyypillisesti tyylikirjastosta, komponenttikirjastosta sekä patternikirjastosta. Tyylikirjasto sisältää design systemin suunnitteluperiaatteet ja peruselementit, kuten värit, typografia ja logot. Lisäksi tyylikirjasto voi sisältää ohjeita sisällöntuotantoon, kuten äänensävy- ja kielenkäyttösuosituksia sekä ohjeita visuaalisesta ja vuorovaikutteisesta suunnittelusta. (13.)

Aalto-yliopiston design systemin tyylikirjasto perustuu design tokenien käyttöön. Design tokenit määrittävät visuaaliset vakioarvot, kuten värit, typografian, ikonit ja välistykset (ks. kuva 2, jossa on esitetty Aalto-yliopiston design systemin värimuuttujia). (15.)

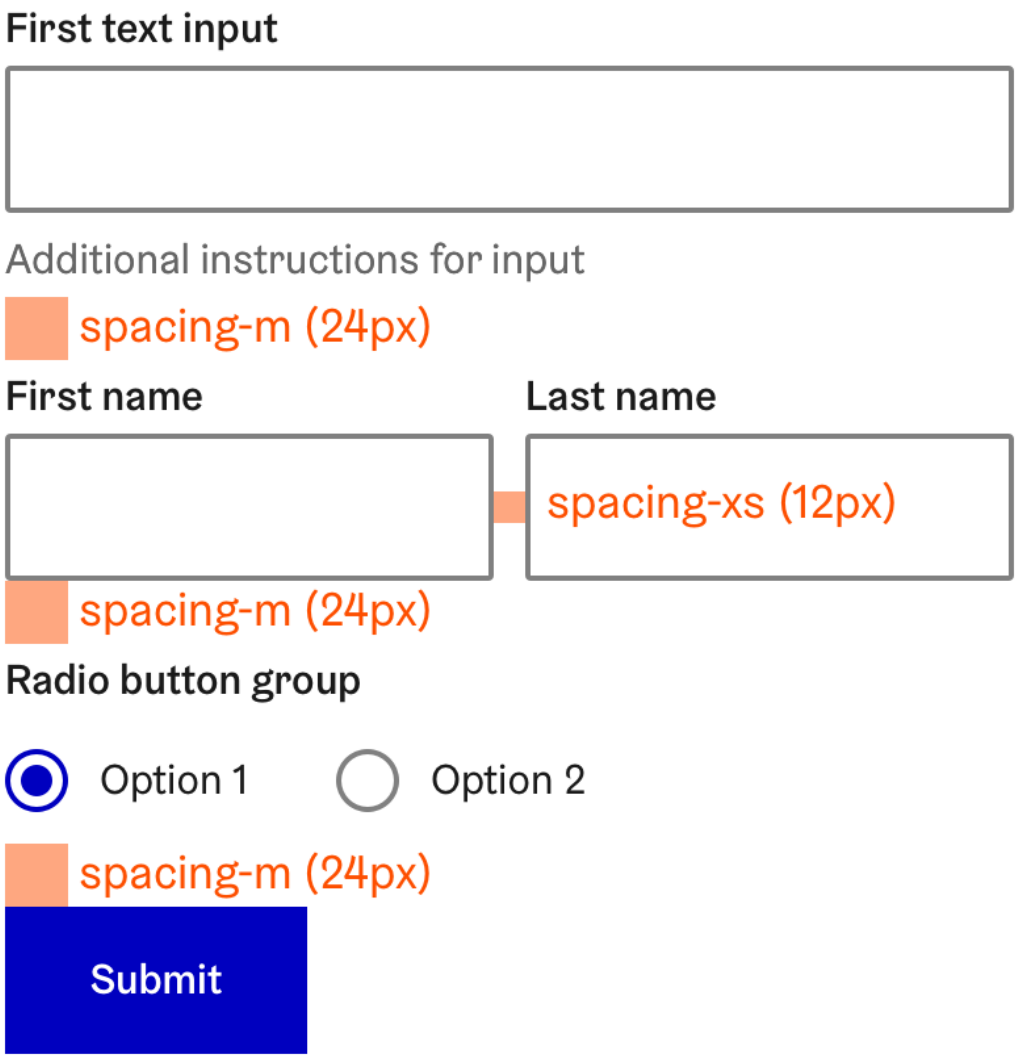


Kuva 2. Kuvakaappaus Aalto-yliopiston design systemin tyylikirjaston värimuuttujista (15).

Komponenttikirjastossa on koottu design systemin uudelleenkäytettävät käyttöliittymäkomponentit, kuten painikkeen ja syötekentät. (12.) Käyttöliittymäkomponenttien visuaalisen esimerkkien lisäksi ohjeet voivat sisältää komponentin nimen, kuvauksen, attribuutit, tilat ja koodiesimerkin sen käytöstä (13). Kuvassa 3 on esitetty Suomi.fi design systemin painikkeen visuaalinen esimerkki ja koodiesimerkki komponentin käytöstä. Koodiesimerkillä helpotetaan ohjelmistokehittäjän työtä uudelleenkäytettävän koodinpätkän avulla, jonka kehittäjä voi kopioida helposti omaan käyttöönsä. (16.)

Kuva 3. Kuvakaappaus Suomi.fi design systemin painikkeen visuaalisesta esimerkistä ja koodiesimerkikstä komponentin käytöstä (16).

Design systemin patternikirjasto on käyttöliittymäkomponenteista koostettuja kokonaisuuksia, kuten sisällön rakenteita, leiskoja ja valmiita ratkaisuja yleisiin suunnitteluhaasteisiin, esimerkiksi lomakkeisiin (12). Patternit ovat käyttöliittymäkomponenttien tapaan uudelleenkäytettäviä ja muokattavissa (13). Esimerkiksi Helsingin kaupungin desgin systemin patternikirjasto sisältää seikkaperäisiä ohjeita ja esimerkkejä lomakkeiden luomiseen, asetteluun ja validointiin. Kuva 4 esittää Helsingin kaupungin design systemin patternikirjaston ohjeistuksen siitä, miten lomakkeissa tulee käyttää tyhjää tilaa komponenttien ryhmittämiseen. Pienempää väliä käytetään komponenttien välillä, jotka halutaan ryhmittää yhteen, ja isommalla välillä erotetaan komponentit toisistaan. (17.)



Kuva 4. Kuvakaappaus Helsingin kaupungin design systemin patternikirjaston ohjeistuksesta tyhjäntilan käyttämisestä lomakkeissa (17).

Atominen suunnittelu tarjoaa selkeän lähestymistavan design systemien rakentamiseen, jossa konsepti ja systeemi rakentuu vaiheittain. Atominen suunnittelu edistää johdonmukaisuutta ja skaalautuvuutta, samalla kun käyttöliittymä näytetään lopullisessa kontekstissa. (14.)

3.3 Design systemien hyödyt ja haasteet

Design systemit tarjoavat yhtenäisen lähestymistavan käyttöliittymien luomiseen. Design systemin avulla voidaan saavuttaa useita hyötyjä suunnittelussa ja kehittämisessä. (12.)

Yksi keskeisimmistä eduista on, että design systemi kehittää käyttöliittymien johdonmukaisuutta. Se yhtenäistää esimerkiksi typografian ja käyttöliittymien välistykset eri alustojen ja laitteiden välillä. Lisäksi uudelleenkäytettävät komponentit tehostavat suunnittelu- ja kehitysprosessia. (12.) Suunnittelijat ja kehittäjät voivat käyttää samoja komponentteja ja pohjia, mikä vähentää tarvetta kehittää samoja ratkaisuja uudelleen ja pienentää epäjohdonmukaisuuksien riskiä (13).

Projektien laajentuessa design systemi auttaa skaalaamaan käyttöliittymää vastaamaan kasvavia tarpeita. Design systemi auttaa ylläpitämään designin eheyttä ja sen laatua säännöllisellä päivittämisellä ja ylläpidon avulla. (12.)

Yhteinen design systemi luo myös yhteisen kielen työryhmien välillä. Design systemi parantaa työryhmien välistä vuorovaikutusta ja vähentää väärinymmärryksiä, kun kaikki ymmärtävät samat käyttöliittymäkomponenttien periaatteet ja toiminnallisuudet. (13.)

Lisäksi design systemit voivat edistää inklusiivisuutta. Jos design systemiin sisällytetään saavutettavuusohjeita, se varmistaa, että työryhmät noudattavat niitä lopputuotteessa. Tämä parantaa osallisuutta ja mahdollistaa kaikille käyttäjille yhtenvertaisen käyttömahdollisuuden. (12.)

Toisaalta tarkasti määritelty design systemi voi rajoittaa suunnittelijan luovaa työtä, erityisesti tilanteissa, joissa projekti vaatii kokeellisempaa ja taiteellisempaa otetta. Jos johdonmukaiselle periaatteille ei ole tarvetta, design systemi ei välttämättä vastaa projektin tarpeita. Tällaisissa tilanteissa on lähtökohtaisesti tärkeä harkita, onko laaja design systemi tarpeen vai riittäisikö projektin tarpeisiin yksinkertaisempi tyylikirjasto. (12.)

Design systemin opettaminen työryhmälle voi viedä aikaa ja vaatii selkeät ohjeet. Ilman perusteellista koulutusta ja ohjeistusta on riski, että design systemiä käytetään epäjohdonmukaisesti eri käyttöliittymissä ja työryhmissä. (13.)

Onnistunut design systemin kehittäminen vaatii työryhmän sitoutumista sekä pitkäjänteistä kehitystä ja päivittämistä, jotta se pystyy vastaamaan työryhmän ja tuotteiden tarpeisiin. Design systemiä luotaessa on aina otettava huomioon projektin laajuus sekä käytettävissä oleva aika ja resurssit. Huonosti toteutettuna ja ylläpidettynä design systemi voi jäädä vajaaksi komponenttikirjastoksi, mutta hyvin toteutettuna se voi tehostaa prosesseja ja parantaa työryhmien välistä vuorovaikutusta. (13.)

4 Saavutettavuus design systemeissä

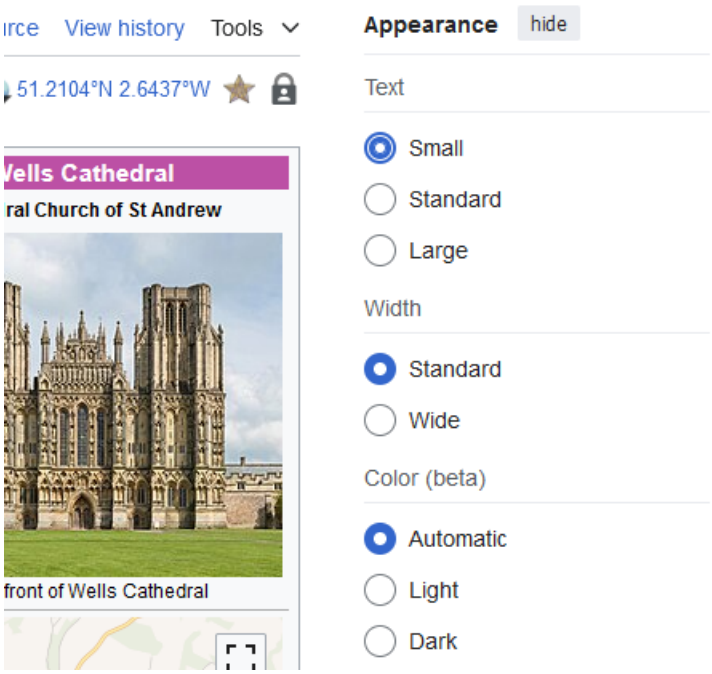
4.1 Saavutettavuus osana käyttäjäkeskeistä suunnittelua

Käyttäjäkeskeisen suunnittelun periaatteen ytimenä on suunnitteluprosessi, jossa suunnittelijat keskittyvät käyttäjiin ja heidän tarpeisiinsa suunnittelun jokaisessa vaiheessa. Käyttäjäkeskeisessä suunnittelussa työryhmät ottavat käyttäjät mukaan suunnitteluprosessiin erilaisten menetelmien ja työkalujen kautta, jotka voidaan jakaa tutkimus- ja generatiivisiin menetelmiin. Tutkimusmenetelmiä voivat olla esimerkiksi kyselyt ja haastattelut, ja generatiiviset menetelmät voivat olla esimerkiksi erilaiset työpajat, joissa ideoidaan yhdessä. Erilaisia menetelmiä hyödyntäen rakennetaan käytettäviä tuotteita pitäen käyttäjät prosessin keskiössä läpi jokaisen vaiheen. (18.)

Siinä missä käyttäjäkeskeinen suunnittelu pitää käyttäjän keskiössä, voidaan sen soveltavuusalaa pitää rajoittuneena, miten se ottaa huomioon riittävän monimuotoisesti käyttäjien tarpeita. Tähän tarpeeseen vastaa kaikille suunnittelun konsepti (eng. *design for all*), joka tarjoaa laajasti erilaisia lähtökohtia, metodeja, tekniikoita ja työkaluja suunnitteluun, jotta voitaisiin vastata monimuotoisesti käyttäjien tarpeisiin ja vaatimuksiin, jolloin

saavutettavuus erilaisille käyttäjille otetaan huomioon heti suunnitteluprosessin alussa. (19.)

Kaikille suunnittelu tarkastelee saavutettavuutta niin, että se otetaan huomioon alusta alkaen, ja tarkastelee ongelmaa siten, että käyttöliittymien on oltava mukautettavissa käyttäjän tarpeiden ja vaatimusten mukaan. Käytännössä tämä tarkoittaa, että käyttäjille tarjotaan vaihtoehtoisia tapoja toimia ja esittää tietoa ja tämä otetaan huomioon jo suunnitteluvaiheessa sen sijaan, että luotetaan avustavien teknologioiden pystyvän muuttamaan tieto käyttäjälle havaittavassa muodossa. Etenkin nopeasti kehittyvän teknologian vuoksi on haastavaa kehittää ajantasaisia lisävarusteita käyttäjille, jotka niitä tarvitsevat. Käyttöliittymä voi sisältää mukautettavat asetukset, säädettävät käyttöliittymäkomponentit ja mukautettavaa sisältöä. (19.) Käyttäjälle voidaan tarjota esimerkiksi toimintoja muuttaa verkkosivuston palstan leveyttä, tekstin kokoa tai teksti puhuttuna -ominaisuutta ilman, että käyttäjän tarvitsee tukeutua avustavaan teknologiaan tai selaimen asetuksiin muuttaakseen esimerkiksi tekstin kokoa isommaksi, jotta hän voi havainnoida sisällön paremmin. Esimerkiksi Wikipedia-verkkosivusto tarjoaa käyttäjälle mahdollisuuden muuttaa tekstin kokoa, palstan leveyttä ja sivuston väriprofiilia vaaleasta tummaksi suoraan sivuston käyttöliittymästä (ks. kuva 5) (20).



Kuva 5. Kuvakaappaus Wikipedia-verkkosivulta, jossa käyttäjä voi muuttaa sivun ulkoasun asetuksia suoraan käyttöliittymästä (20).

Nopeasti kehittyvä teknologia on myös johtanut ihmisten eriarvoistumisen ja riskin teknologiseen syrjäytymiseen. Teknologisen syrjäytymisen riskissä olevat käyttäjät, esimerkiksi suuret ikääntyvät sukupolvet, on tärkeä huomioida palveluita kehittäessä, jotta teknologian luomaa kuilua voidaan kuroa kiinni ja tarjota kaikille mahdollisuus toimia digitaalisessa ympäristössä. (19.)

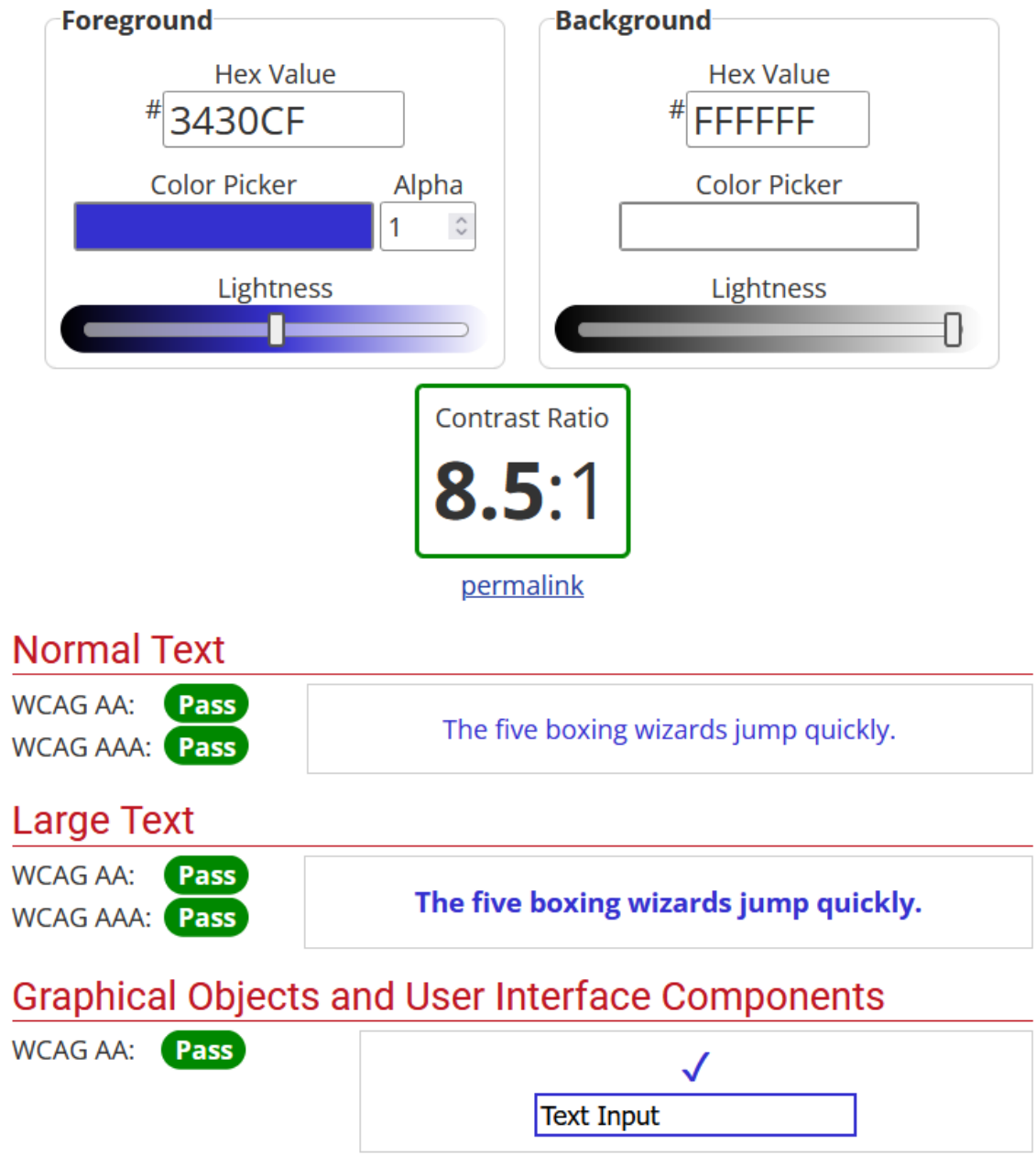
4.2 Yleisiä ongelmia saavutettavuudessa

Kun saavutettavuus unohdetaan design systemissä, voi sillä olla kauaskantoiset vaikutukset. Saavutettavuusongelmia voi olla vaikea korjata jälkikäteen, kun design systemi on jo kehitetty ja se on implementoitu useampaan verkkopalveluun. Atomisen suunnittelun periaatteen osia käyttämällä, kuten atomeita, molekyylejä ja organismeja, ilman saavutettavuuden huomioimista, syntyy saavutettavuuspuutteita suoraan systeemin perusosiin. Tämä johtaa haastavampaan saavutettavuuden auditoimiseen, kun ongelmat liittyvät käyttöliittymien periytyviin tyyleihin, miten käyttöliittymäkomponentit on rakennettu ja miten käyttöliittykomponentit toimivat saavutettavasti. Kun

yhdessä käyttöliittymäkomponentissa on esimerkiksi ongelma värien saavutettavuudessa, voimme olla lähes varmoja, että ongelma esiintyy myös muualla, koska ongelma on perustavanlaatuinen. (21.)

Yli puolet saavutettavuusongelmista alkaa jo suunnitteluvaiheessa. Yleisiä saavutettavuuteen liittyviä ongelmia ovat värien käyttö, typografia ja otsikoiden hierarkia. Nämä design systemien tyylikirjastojen atomit asettavat perustan design systemille, ja unohtamalla saavutettavuuden jo näin varhaisessa vaiheessa design systemin suunnittelussa, vaikutetaan koko design systemin ja sillä luodun kokemuksen saavutettavuuden puutteeseen. (21.)

Yksi yleisimmistä saavutettavuuspuutteista liittyy väreihin ja niiden käyttöön verkkopalveluissa (22). Design systemissä on oleellista määrittää saavutettavat värit, jotka vastaavat WCAG 2.2 A-tason kriteeriä, 1.4.3 *Contrast (Minimum)*, jossa kriteerinä on, että tekstin värin ja taustavärin kontrasti on oltava vähintään 4.5:1, lukuunottamatta tiettyjä poikkeuksia. Tämä kriteeri varmistaa sen, että käyttäjät, joilla on heikompi näkö ja vaikeuksia hahmottaa matalakontrastista sisältöä, voivat havaita tiedon ilman avustavaa teknologiaa, joka nostaa kontrastia. (23.) Lisäksi design systemien kehityksen kannalta oleellisessa värien käyttöön liittyvässä AA-tason kriteerissä, 1.4.11 *Non-text Contrast*, määritetään, että käyttöliittymäkomponenttien eri tilojen tunnistamiseen, lukuunottamatta ei-aktiivisia komponentteja, kontrastisuhteen on oltava vähintään 3:1 viereisiin väreihin nähden. (24.) Kuvassa 6 on esitetty sinisen ja valkoisen värin kontrastin ratio ja miten se vastaa WCAG 2.2 1.4.3 *Contrast (Minimum)* ja 1.4.11 *Non-text Contrast* kriteereitä. Kyseisen sinisen ja valkoisen värin kontrasti on 8.5:1, joka on riittävän suuri vastaamaan AA ja AAA-tasojen värikriteereitä. (25.)



Kuva 6. Kuvakaappaus WebAIM-sivustolta, jossa vertaillaan sinisen ja valkoisen värin kontrastia ja miten se onnistuu WCAG 2.2 -vaatimuksissa (25).

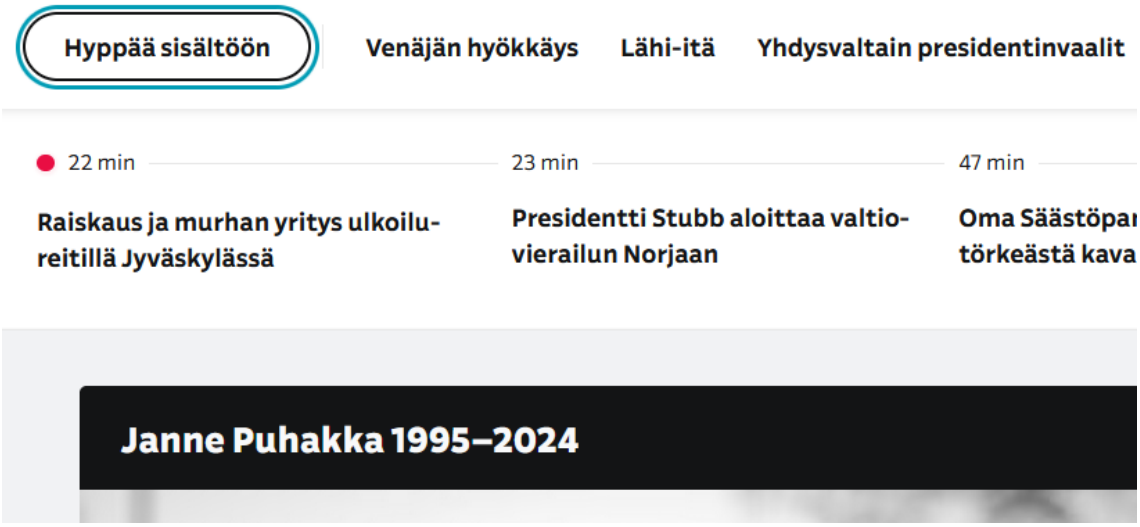
Monet saavutettavuuden ongelmista liittyvät myös verkkopalvelun typografiaan ja puuttellisiin otsikko- ja tekstirakenteisiin (21). Saavutettavan kirjasintyypin valitseminen ei ole yksiselitteinen asia, eikä ole yhtä oikeaa vastausta siitä, mikä kirjasintyyppi olisi kaikille saavutettava. Saavutettavaan kirjasintyyppiin vaikuttavat monet seikat, kuten yksittäisten kirjainten sisällä olevat tyhjät tilat. Isompi tyhjätila kirjaimessa tai merkissä parantaa sen luettavuutta. Lisäksi

yksittäisten kirjainten koetaan vaikeaksi lukea, kun kirjaimen peilaa ja se muistuttaa toista kirjainta. Esimerkkinä voidaan pitää “d”- ja “b”-kirjaimia, kun ne kirjoitetaan pienellä. Kirjasintyypin luettavuuden optisoiminen ei kuitenkaan takaa tekstin saavutettavuutta. (26.) Saavutettavuuden kannalta tekstin koolla on suuri merkitys erityisesti käyttäjille, joilla on heikko näkö tai kognitiivisia vaikeuksia. WCAG:n kriteeri, 1.4.4 *Resize Text*, ohjaa käyttämään skaalautuvaa tekstiä verkkopalveluissa, jotta tekstin voi suurentaa kaksinkertaiseksi ilman, että sisältö tai toiminnallisuus kärsii siitä. (27.) WCAG ei kuitenkaan ohjeista minimiä tekstin koolle, mutta värien kontrastiin liittyvässä kriteerissä mainitaan, että 18.5px koon lihavoitu teksti lasketaan isoksi, joka tarvitsee vain 3:1 värikontrastin (23).

Puuttelliset tai olemattomat otsikkorakenteet johtavat myös saavutettavuushaasteisiin verkkopalvelussa (28). Otsikoiden pitäisi olla visuaalisesti ja ohjelmallisesti tunnistettavissa otsikoiksi, eikä esimerkiksi tekstiä saisi suunnitella näyttämään otsikolta, jos sen yhteydessä ei käytetä HTML-merkkauskielen otsikkotasoja (29). Ilman ohjelmallisesti tunnistettavia otsikoita, avustavatteknologiat eivät pysty tunnistamaan otsikoita ja sisällön hierarkiaa. WCAG:n kriteeri 1.3.1 *Info and Relationships* vaatii, että informaatio ja rakenteet on pystyttävä tunnistamaan ohjelmallisesti. Otsikoiden kannalta tämä tarkoittaa sitä, että verkkopalveluissa pitäisi käyttää oikein h1-h6-otsikkorakenteita. (28.) Saavutettavuushaasteita voi myös syntyä, jos otsikkotasoja jätetään välistä, sillä avustavaa teknologiaa käyttävät käyttäjät usein käyttävät navigointia otsikoiden kautta, eli jos otsikkorakenne ei ole hierarkisesti oikein, käyttäjät eivät välttämättä ymmärrä sisällön ja informaation suhteita toisiinsa (29).

Vaikka design systemien tyylikirjastossa asetetaan jo systemin perusta, kuten värimaailma ja typografia, ja näiden saavutettavuuden varmistaminen on ensimmäisiä askeleita saavutettavan design systemin luomisessa, saavutettavuusongelmat ovat muutakin kuin riittämätön kontrasti värien välillä tai liian pieni kirjasintyyppi. Iso osa saavutettavuudesta liittyy rakenteisiin ja toiminnallisuuksiin. (21.)

Puutteellinen saavutettavuus näppäimistöllä tuottaa haasteita monille käyttäjille, jotka käyttävät näppäimistöä navigoidakseen ja käyttääkseen verkkopalveluita. Käyttäjät, jotka navigoivat näppäimistöllä, ovat esimerkiksi käyttäjät motorisilla rajoitteilla ja näkörajoitteiset, jotka saattavat hyödyntää esimerkiksi ruudunlukijaa. (22.) Näppäimistöllä navigoimiseen liittyy monia haasteita, jotka on hyvä ottaa huomioon design systemeissä, kuten olematon tai riittämätön visuaalinen kohdistin havainnoimaan käyttäjälle missä osoitin sijaitsee. Kaikki interaktiiviset elementit eivät ole kohdistettavissa tai että käyttäjälle ei ole tarjolla mekanismia, jolla hän voi ohittaa verkkopalvelussa toistuvia elementtejä, kuten ylänavigaatiota. WCAG:n näppäimistöön liittyviä kriteereitä on useampia. A-tason 2.1.1 *Keyboard* kriteeri vaatii, että näppäimistöllä on pystyttävä suorittamaan kaikki verkkopalvelun toiminnot. (30.) AA-tason 2.4.7 *Focus Visible* kriteerin tarkoituksena on varmistaa, että kaikilla elementeillä, jotka voivat vastaanottaa kohdistuksen, on visuaalinen kohdistinindikaattori (31). Lisäksi A-tason 2.4.1 *Bypass Blocks* -kriteerissä ohjeistetaan, että näppäimistökäyttäjille on pystyttävä tarjoamaan mekanismi, jolla he voivat ohittaa toistuvat elementit (32). Yle.fi-sivu tarjoaa näppäimistöllä navigoiville käyttäjille 2.4.1 *Bypass Blocks* -kriteeriin ominaisuuden, jolla käyttäjä voi ohittaa kaikilla sivuilla toistuvan ylänavigaation päästäkseen pääsisältöön nopeammin (ks. kuva 7) (33).



Kuva 7. Kuvakaappaus yle.fi-sivulta, jossa näppäimistöllä navigoivalle käyttäjälle tarjotaan painike, jolla käyttäjä voi hypätä ylänavigaation ohi sivun pääsisältöön (33).

Lomakkeisiin ja niiden syötteisiin liittyy myös paljon saavutettavuusongelmia, jotka vaikuttavat design systemien suunnitteluun ja kehittämiseen. Tyypillisimpiä puutteita lomakkeissa ja syötteissä ovat mm. syötteiden <label>-tagien puute sekä puutteellinen tuki syötteiden täyttämiseen. (22.) Syötteiden virheen tunnistamisessa on myös hyvä huomioida väreihin liittyvä kriteeri, 1.4.1 *Use of Color*, jossa määritetään, että pelkällä värillä ei saa viestiä informaatiota, vaan siihen pitää yhdistää aina kuva tai teksti. Kaikki käyttäjät eivät pysty havaitsemaan värejä, jonka takia informaatiota ei pidä viestiä pelkällä värillä. (34.) Syötteiden virheen tunnistukseen liittyvät kriteerit 3.3.1 *Error Identification* ja 3.3.3 *Error Suggestion* myös velvoittavat, että lomakkeiden täytössä syntyvät virheet havaitaa ja käyttäjälle on kerrottava miksi virhemuodostui sekä ohjata käyttäjä korjaamaan virhe (35).

4.3 Miten design systemit edesauttavat saavutettavuutta

Implementoimalla saavuttavuutta suoraan design systemiin on monia hyötyjä. Yksi hyöty on tehokkuuden lisääminen. Saavutettavalla design systemillä voidaan parantaa tehokkuutta, koska saavutettavalla design systemillä varmistetaan, että saavutettavuus on jo osa kokemusta alusta alkaen, jolloin

työryhmien tarvitsee käyttää vähemmän aikaa saavuttavuusongelmien testaamiseen, korjaamiseen ja laadunvarmistuksessa. (36.)

Saavutettavalla design systemillä voidaan taata myös johdonmukaiset käyttöliittymäkomponentit ja kokemukset käyttäjille, joilla on rajoitteita (37). Johdonmukainen kokemus on helpompi oppia ja johdonmukainen käyttöliittymä tukee kaikkia käyttäjiä suorittamaan haluamansa toiminnot verkkopalvelussa. Etenkin avustavaa teknologiaa käyttävä käyttäjä voi suorittaa haluamansa toiminnot helpommin johdonmukaisessa verkkopalvelussa. (36.)

Googlen kehittämän design systemin, Material Designin, saavutettavuusarvoihin kuuluu kaikille suunnittelun periaate, joka ottaa huomioon yksilöiden erilaiset tarpeet tarjoamalla ratkaisuja yksilöllisiin adaptaatioihin (40). Etenkin tämä lähestymistapa design systemien saavutettavuuteen parantaa käytettävyyttä kaikille käyttäjille ja antaa työryhmille mahdollisuuden ryhtyä toimiin esteettömän käyttökokemuksen tarjoamiseksi eri alustoilla (37). Universaalit kokemukset vastaavat harvoin kaikkien tarpeita, jonka vuoksi mukautettavat ominaisuudet luovat useammalle käyttäjälle mahdollisuuden käyttää palvelua sujuvasti (40).

Yksikään design systemi ei kuitenkaan voi taata verkkopalvelun täyttä saavutettavuutta. Saavutettavan verkkopalvelun kehityksessä on iso vaikutus sillä, miten design systemiä sovelletaan. Vaikka yksittäiset käyttöliittymäkomponentit ja -elementit onkin suunniteltu ja kehitetty saavutettavasti, design systemeitä käytetään kokonaisten sivujen ja verkkopalveluiden kehittämiseen. Onkin tärkeää, että design systemiä soveltavat työryhmät osaavat hyödyntää design systemiä saavutettavasti. (41.)

4.4 Parhaita käytänteitä saavutettavalle design systemille

Dokumentaatiolla on tärkeä rooli design systemeissä, sillä se edistää työryhmien yhteistyötä, koska kaikki voivat saada tarvittavat tiedot selkeästi ja johdonmukaisesti yhdestä paikasta. Saavutettavuuden dokumentointi osaksi

design systemiä varmistaa, että design systemiä käyttävät työryhmät saavat riittävän tiedon siitä, miten toteuttaa saavutettavia palveluita. Lisäksi esteettömyyden tavoitteista, joihin design systemi pyrkii, on hyvä tarjota selkeä kuvaus, jotta työryhmät ymmärtävät myös yhteiset tavoitteet. Tavoitteiden taustalla voi olla esimerkiksi vastaaminen WCAG-ohjeistukseen ja inklusiivisen kokemuksen tarjoaminen. (42.)

Lisäksi saavutettavuuskäytänteet on hyvä dokumentoida osaksi design systemin tyyli-, komponentti ja patternikirjastoja, tarjoamalla esimerkkejä ja ohjeita, miten impelmentoida design systemin osia saavutettavalla tavalla (42). Tyylikirjaston yhteyteen voi esimerkiksi dokumentoida saavutettavat väriyhdistelmät ja niiden kontrastiarvot (37). Design systemin saavutettavuuskäytänteet voivat esimerkiksi sisältää ohjeita semanttisen HTML-merkistön ja sen natiivien komponenttien käyttöön design systemin käyttöliittymäkomponenteissa, jotka varmistavat, että komponentit luetaan ja ne toimivat oikein ruudunlukijoille (43). Lisäksi monimutkaisille käyttöliittymäkomponenteille on tarpeen esittää WAI-ARIA (Web Accessibility Initiative - Accessible Rich Internet Applications) käyttöön liittyviä ohjeita ja esimerkkejä, jotta voidaan varmistua, että elementit, joista saattaa puuttua sisäänrakennettu semanttisuus, voidaan tarjota se ja parantaa saavutettavuutta sitä kautta (44).

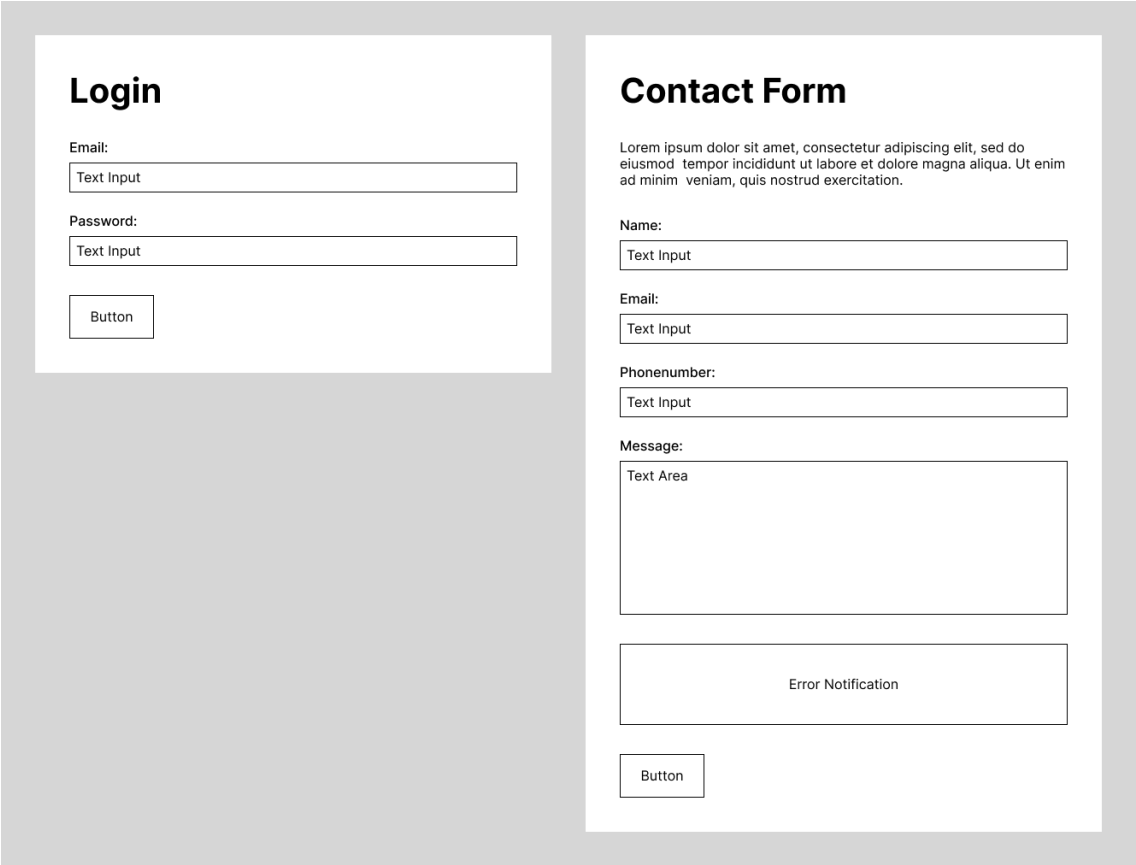
Saavutettavuuden kehittäminen ei pääty saavutettavasti toteutettuun design systemiin, vaan saavutettavuus vaatii jatkuvaa testausta ja iterointia. Design systemin saavutettavuutta voidaan auditoida automatisoiduilla työkaluilla, sekä manuaalisesti tarkastamalla, että se vastaa vaadittavia kriteereitä. Lisäksi käyttäjillä, joilla on erilaisia rajoitteita, testaaminen on ensisijaisen tärkeää, jotta saadaan oikeaa dataa design systemin saavutettavuudesta. Lisäksi on olennaista päivittää design systemiä vastaavaan muuttuvia saavutettavuuskriteereitä. (37.)

5 Saavutettavan design systemin kehitys lomakkeita varten

5.1 Tavoitteet ja resurssit

Design systemin suunnittelun alussa asetettiin selkeät tavoitteet sen toteutukselle. Koska tavoitteena oli soveltaa saavutettavuuden periaatteita design systemin kehityksessä, määritettiin design systemin saavutettavuuden tavoitteeksi WCAG 2.2 -ohjeistuksen AA-taso.

Tavoitteena oli suunnitella ja kehittää pieni design systemi, joka sisältäisi tyylikirjaston, muutaman komponentin komponenttikirjaston ja pari patternia. Luomalla yksinkertaiset rautalankamallit patterneista määritettiin, mitä komponentteja pitäisi suunnitella ja toteuttaa design systemiin. Suunniteltiin rautalankamallit yksinkertaisesta sisäänkirjautumislomakkeesta sekä yhteydenottolomakkeesta (ks. kuva 8), joissa yhdistyivät hyvin käyttöliittymäkomponenttien kulmakivet: painikkeet ja tekstisyötteet. Lisäksi suunniteltiin ilmoitustyyppinen käyttöliittymäkomponentti, joka ei esiinny natiivina HTML-merkistön elementtinä.



Kuva 8. Figmassa suunnitellut yksinkertaiset rautalankamallit design systemin patterneista, jotka toimivat pohjana tarvittavien elementtien määrittämisessä.

Figmaa käytettiin design systemin suunnittelutyökaluna luomaan johdonmukainen ja skaalautuva kokonaisuus. Figmassa suunniteltiin design systemin visuaaliset tyylit ja design tokenit sekä tarvittavat käyttöliittymäkomponentit (38). Figmassa toteutettujen suunnitelmien pohjalta kehitettiin yksinkertainen React-kirjasto Storybook-työkalun avulla. Storybook on käyttöliittymäkomponenttien teknisen toteutuksen kehittämiseen ja dokumentointiin tarkoitettu työkalu. Sillä voi kehittää komponentteja omassa ympäristössä ja testata niiden eri toimintoja ja tiloja. Storybook tarjoaa dokumentaatioalustan design systemin teknisen toteutuksen versiolle. (39.)

Figman ja Storybookin lisäksi pilottiprojektille ei ollut tarpeelliseksi ottaa käyttöön kolmatta työkalua design systemin erilliseksi dokumentaatioalustaksi. Laajemmassa projektissa, jossa tehdään yhteistyötä eri työryhmien välillä,

dokumentaation rooli olisi korostunut tärkeämmäksi. Kolmannessa alustassa olisi voinut yhdistää suunnittelutyökalun ja teknisen toteutuksen design systemit, jotta niiden vertaaminen toisiinsa olisi ollut helpompaa.

5.2 Tyylikirjasto

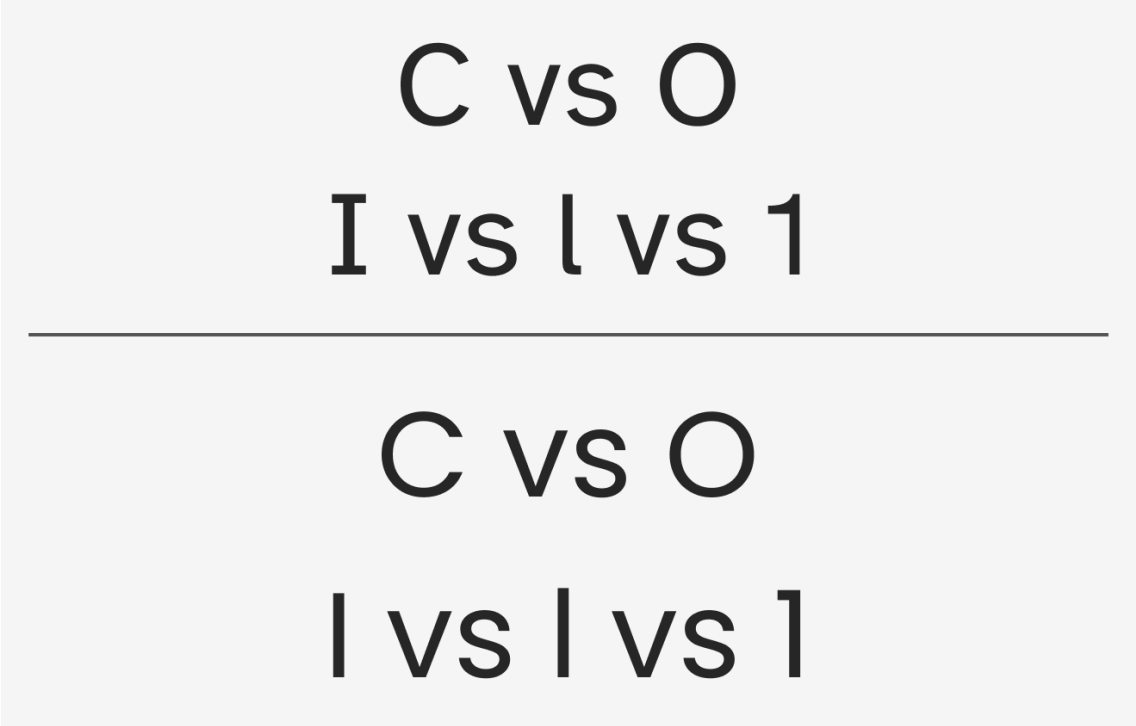
Design systemin tekeminen aloitettiin tyylikirjastosta. Design systemin tyylikirjastoon kuului typografia, värit, ikonit sekä mittasuhteet ja asettelu. Tyylikirjaston tyylit jaettiin primitiivisiin ja semanttisiin design tokeneihin. Primitiiviset design tokenit olivat perusarvoja, kuten tekstikoot, värit ja mittasuhteet. Ne toimivat perustana muille tyylikirjaston ratkaisuille. Semanttiset design tokenit puolestaan viittasivat primitiivisiin arvoihin, ja ne nimettiin käyttötarkoituksen mukaisesti. Esimerkiksi semanttinen värin design token nimeltä “—semantic-color-background-primary” kertoi design tokenin käyttötarkoituksesta enemmän kuin primitiivisen värin design token nimeltä “—primitive-color-violet-500”.

Tyylikirjaston toteutuksessa hyödynnettiin WCAG 2.2 -ohjeistuksen eri kriteereitä, jotka liittyivät esimerkiksi väreihin ja tekstiin. Tyylikirjastoa suunnitellessa pyrittiin ottamaan huomioon muita saavutettavuuteen ja käytettävyyteen vaikuttavia tekijöitä, joita ei mainita WCAG-ohjeistuksessa.

5.2.1 Saavutettava typografia

Saavutettavalla typografialla on suuri vaikutus siihen, että verkkopalvelut ja niiden sisältö ovat mahdollisimman havaittavia ja luettavia käyttäjälle. Saavutettavaan typografiaan vaikuttavat monet seikat, kuten kirjasintyyppien määrä, kirjasintyypin eri leikkaukset, välistys ja tekstin koko. Voidaan kuitenkin todeta, että kirjaisintyypin yksinkertaisuus on ratkaiseva ominaisuus luettavuuden kannalta. Tuttu tai helposti jäsenneltävä kirjasintyyppi on käyttäjälle helpompi hahmottaa, kun taas tuntematon ja monimutkainen kirjasintyyppi vaatii enemmän aikaa ja orientoitumista, mikä tekee merkkien ja sanojen jäsentämisestä kognitiivisesti väsyttävää ja haastavaa. (45.)

Design systemille sopivaa kirjasintyyppiä valittaessa huomioitiin useita ominaisuuksia, jotka vaikuttavat kirjasintyypin luettavuuteen. Monimutkaisen kirjasintyypin valitsemista vältettiin, koska yksinkertaiset muodot ja rakenteet tekstissä on nopeammin havaittavissa ja analysoitavissa (45). Jotta vältettäisiin valitsemasta kirjasintyyppiä, jonka merkit ovat samankaltaisia, vertailtiin yksittäisiä kirjasintyyppejä. Samankaltaiset merkit voivat lisätä epäselvyyttä tekstissä, mikä vaikuttaa lukunopeuteen ja tekstin ymmärtämiseen (45). Kuvassa 9 on esitetty kahden eri kirjasintyypin merkkejä, jotka muistuttavat toisiaan. Suurempi tyhjätila auttaa erottamaan kirjaimet toisistaan ja samanmuotoisten merkkien kuten ison “l”-kirjaimen, pienen “l”-kirjaimen ja numeron “1” erottaminen helpottavat tekstin luettavuutta (45).

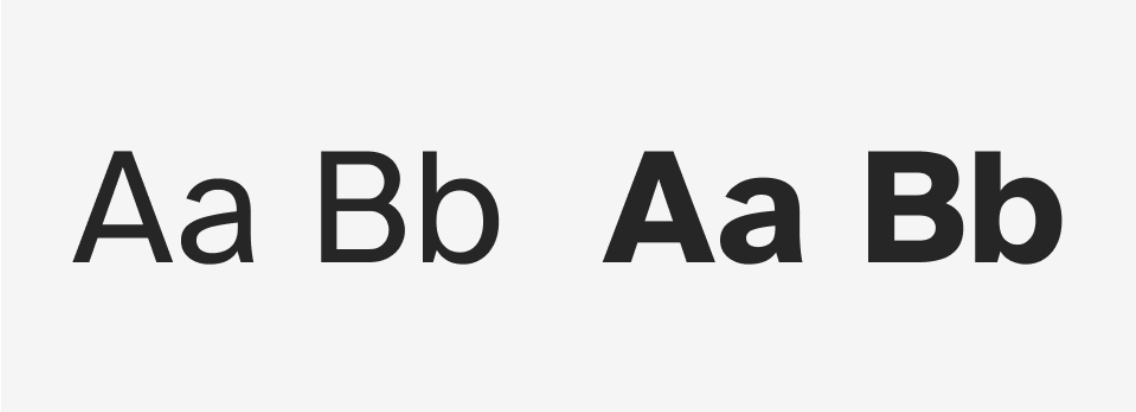


Kuva 9. Ylempänä Atkinson Hyperlegible -kirjasintyypin “C”- ja “O”-kirjaimet vertailussa sekä iso “l”-kirjain, pieni “l”-kirjain ja numero “1” vertailussa. Alempana Poppins-kirjasintyypin samat merkit vertailussa keskenään.

Atkinson Hyperlegible valikoitiin design systemin kirjasintyypiksi. Atkinson Hyperlegible on suunniteltu erityisesti heikkonäköisille, ja se parantaa luettavuutta selkeiden ja erottuvien kirjainten ja numeroiden avulla. Atkinson

Hyperlegible on päätteetön kirjasintyyppi, mutta se sisältää harkittuja päätteitä, jotka parantavat yksittäisten kirjainmuotojen erottuvuutta. Esimerkiksi kuvassa 9 näkyy, että iso “l”-kirjain on päätteellinen, mikä parantaa sen erottuvuutta samankaltaisista merkeistä. (46.)

Atkinson Hyperlegible -kirjasintyypissä on vain kaksi eri leikkausta, normaali ja lihavoitu (ks. kuva 10), jotka riittivät design systemin tarpeisiin. Otsikoille ja korostettaville teksteille määritettiin lihavoitu leikkaus, kun taas normaalileikkaus soveltui leipätekstiin. Normaalileikkauksen paksuus oli sopiva leipätekstiin, sillä se ei ollut liian ohut tai paksu, mikä voisi haitata tekstin havaittavuutta ja luettavuutta (45).



Kuva 10. Atkinson Hyperlegible -kirjasintyypissä on kaksi eri leikkausta: normaali ja lihavoitu.

Tekstin kokojen määrittämisessä käytettiin kokoyksikköä *rem*, joka perustuu juurielementin kirjasinkokoon. Monissa selaimissa juurielementin oletuskoko on 16px, mikä tarkoittaa, että yksi *rem*-yksikkö vastaa 16px-kokoa. (47.) WCAG 2.2 -kriteeri 1.4.4 *Resize Text* (AA-taso) vaatii, että tekstiä on pystyttävä suurentamaan kaksinkertaiseksi ilman, että sisältö tai toiminnot katoavat. Tämä kriteeri huomioi käyttäjien tavat suurentaa tekstiä eri tavoilla, kuten sivuston suurentamisen ja selaimen asetusten kautta muuttamalla juurielementin kirjasinkokoa. Tekijän vastuulla on luoda sisältöä, joka ei estä selaimen tekstin koon muuttamista. (27.) Käyttämällä *rem*-yksikköä varmistettiin, että teksti skaalautuu käyttäjän selaimen asetusten mukaisesti. Työpöytäkäyttöön

määritettiin tekstin minimikooksi 1rem (16px), koska WCAG 2.2 kriteerissä 1.4.2 *Contrast (Minimum)* (AA-taso) isoksi tekstiksi lasketaan 18.5px-kokoinen lihavoitu teksti (23). Normaalin leipätekstin kooksi asetettiin 1.125rem (18px) varmistamaan riittävä tekstin koko heikkonäköisille ilman, että heidän tarvitsee tukeutua muuttamaan selaimen asetuksia. Lisäksi tekstikokoja pienennettiin mobiilikäyttöliittymiä varten, jotta isot tekstikoot etenkin otsikkotyyleissä eivät veisi tilaa pienemmältä näytöltä.

Johdonmukaisen ja selkeän visuaalisen hierarkian varmistamiseksi eri tekstikokojen ja tasojen, kuten otsikoiden ja leipätekstin, välillä, hyödynnettiin 1.2 *Minor thirds* -skaalaussuhdetta (eng. *typescale*). Skaalaussuhteet luovat selkeitä rakenteita, jotka parantavat verkkopalveluiden havaittavuutta ja ymmärrettävyyttä luomalla loogisia suhteita elementtien välille. 1.2 *Minor thirds* on keskitason skaalaussuhde, joka sopii monipuolisesti erilaisiin näyttökokoihin. (48.) Kuvassa 11 on esitetty design systemin Figma-projektista kirjasinkokojen primitiivisten arvojen dokumentointi. Primitiiviset arvot toimivat tekstielementtien, kuten otsikkotasojen ja leipätekstin, semanttisten design tokenien määrittämisessä (kuva 12).

Kuva 11. Design systemin primitiiviset kirjasinkoot dokumentoituna Figmassa.

Kuva 12. Design systemin ensimmäisen otsikkotason määrittävät tyylit, jotka viittaavat primitiivisiin design tokeneihin.

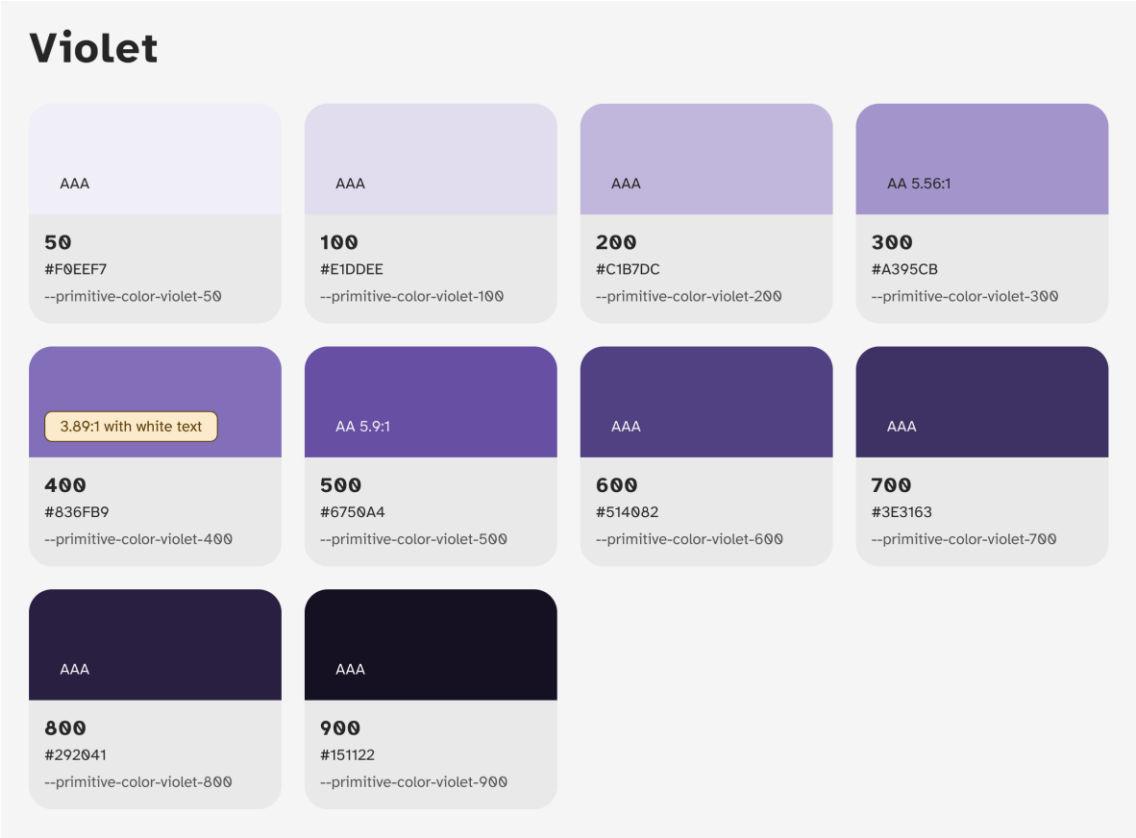
Tekstin havaittavuuteen ja luettavuuteen vaikuttaa myös tekstin välistys. Liian tiukka välistys voi heikentää merkkien havaittavuutta, ja kirjaimet voivat näyttää toisiltaan, jos ne ovat liian lähellä toisiaan. Toisaalta liian leveä välistys voi hankaloittaa sanojen jäsentelyä. (45.) WCAG 2.2 kriteeri 1.4.12 *Text Spacing* (AA-taso) vaatii lisäksi, että sisältöä tai toimintoja ei saa menettää, vaikka käyttäjä muuttaisi välistyksiä (49).

5.2.2 Väripaletin valitseminen ja värien saavutettava käyttö

Määrittämällä design systemiin värijärjestelmän varmistetaan, että käytössä on hallittu värimaailma, joka on johdonmukainen, saavutettava ja jota käytetään oikein. Päävärien lisäksi design systemiin voi harkita lisävärejä, jotka viestivät toimintojen merkityksestä. Esimerkiksi virheiden käsittelyssä käyttäjälle voidaan viestiä tietyllä värillä viestin sisällöstä. Yleisesti ottaen punainen väri tarkoittaa virhettä, sininen tietoa ja vihreä onnistumista tai vahvistusta. (50.)

Design systemin värien suunnittelua lähestyttiin melko suppealla värimaailmalla. Määriteltiin pääväri, neutraali vaalea ja tumma väri, sininen viestimään tiedosta, vihreä viestimään onnistumisesta, oranssi viestimään varoituksesta ja punainen viestimään virheestä. Primitiiviset väripaletit luotiin ensimmäisenä, ja ne sisälsivät harmaan, violetin, sinisen, vihreän, oranssin ja punaisen eri sävyt vaaleasta tummaan. Väripalettien eri sävyjen kontrastit testattiin WebAIMin kontrastin tarkastustyökalulla joko vaalealla tai tummalla

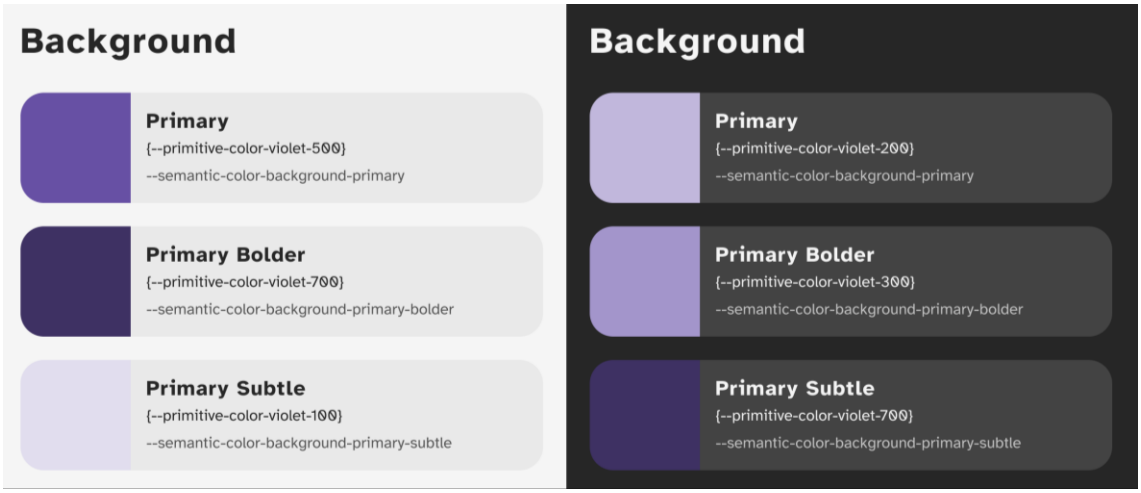
tekstillä varmistamaan, mitkä värit täyttivät vähintään WCAG 2.2 -kriteerin 1.4.3 *Contrast (Minimum)* (AA-taso) vaatimuksen, jonka mukaan tekstin ja sen taustavärin kontrastin on oltava vähintään 4.5:1 (23). Primitiiviset väripaletit dokumentoitiin Figmassa ja jokaiselle väriarvolle lisättiin tieto siitä, minkä tason saavutettavuuden se saavutti vaalean tai tumma tekstin taustavärinä (kuva 13).



Kuva 13. Violetin väripaletti, joka sisältää sävyt vaaleasta tummaan.

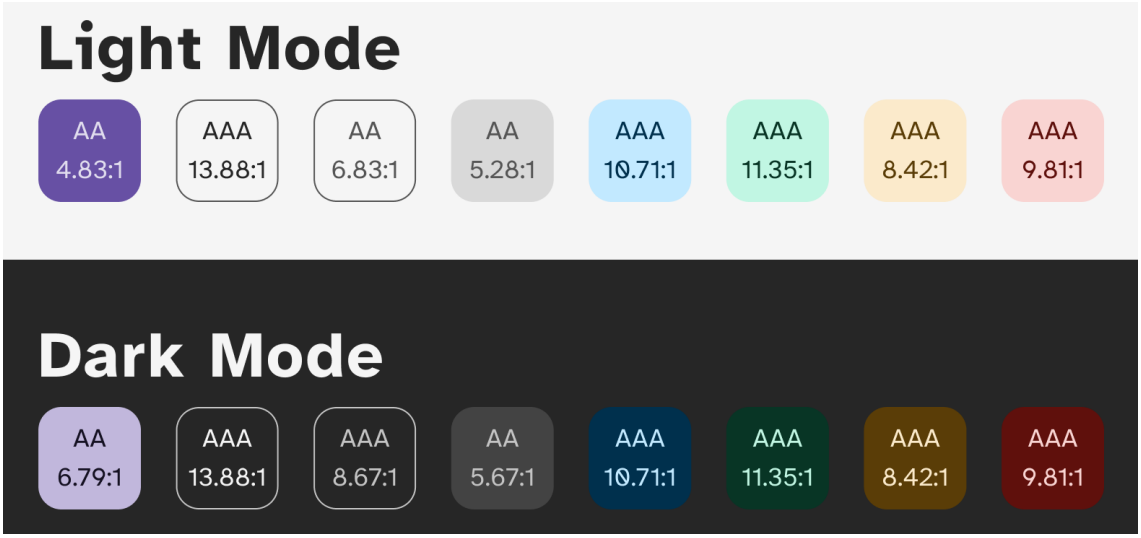
Primitiivisistä väripaleteista johdettiin semanttisesti nimetyt värit, jotka viittasivat primitiivisiin väreihin. Koska käytössä oli primitiivisten värien eri sävyjä, pystyttiin luomaan tarvittavista väreistä perusarvo sekä vaaleampi ja tummempi arvo, mikä mahdollisti värien monipuolisen käytön. Semanttisten värien käyttäytymiseen suunniteltiin, jos käyttäjä haluaisi käyttää tummaa käyttöliittymää vaalean sijasta. Kuvassa 14 on esitetty primäärivärin kolme eri arvoa: perusarvo, tummempi sekä vaaleampi arvo. Samaa lähestymistapaa käytettiin muidenkin semanttisten värien määrittelyssä. Lisäksi sininen, vihreä,

oranssi ja punainen väri nimettiin selkeämmin kuvastamaan niiden käyttötarkoituksia: *info*, *success*, *warning* ja *danger*.



Kuva 14. Semanttisen primäärivärin eri vaihtoehdot taustaväriksi.

Semanttisten värien määrittämisen yhteydessä testattiin erilaisten väriyhdistelmien kontrasti. Figmaan dokumentoitiin esimerkit saavutettavista väriyhdistelmistä, varmistamaan väriyhdistelmien saavutettava ja oikeanlainen käyttö (ks. kuva 15).



Kuva 15. Figma-projektiin luodut esimerkit saavutettavista väriyhdistelmistä vaalealla ja tummalla käyttöliittymällä.

Vaikka design systemin käyttöön suunniteltiin lisävärit viestimään erilaisista merkityksistä, pidettiin mielessä, ettei pelkällä värillä saa WCAG 2.2 kriteerin 1.4.1 *Use of Color* (A-taso) mukaan välittää tietoa. Kriteerin mukaan merkityksen välittämiseen kuuluu käyttää muitakin tapoja, kuten muotoa ja tekstiä. (34.)

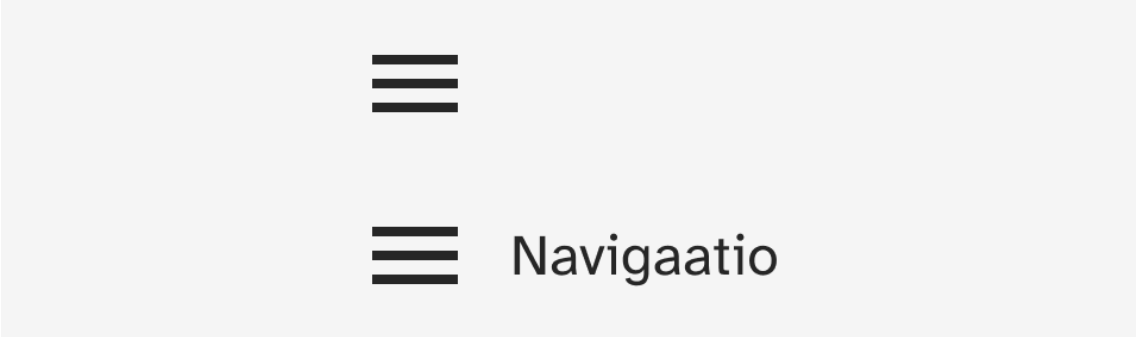
5.2.3 Ikonien saavutettavuus

Verkkopalvelun ja -sisällön ikoneita valittaessa täytyy kiinnittää huomiota niiden ymmärrettävyyteen ja käyttäjälähtöisyyteen. Huonot ikonit voivat luoda esteitä, jotka johtavat turhautumiseen ja väärinymmärryksiin ja pahimmassa tapauksessa rajoittavat käyttäjiä suorittamasta haluamiaan toimintoja. Ikoneiden tarkoitus on tarjota nopea ja intuitiivinen tapa kertoa informaatiosta ilman tekstiä. (51.)

Design systemin ikonikirjastoksi valittiin Material Symbols sen selkeiden ja helposti tunnistettavien ikoneiden sekä kattavan tuen takia Figmaan ja webkehitykseen. Saavutettavan ikonikirjaston valinnassa otettiin huomioon ikonien selkeys, jotta ne olisivat mahdollisimman helppo havannoida riippumatta käyttäjän visuaalisesta hahmotuskyvystä, ja että niiden merkitys olisi universaalisesti tunnistettavissa (51). Ikonien käytössä varmistettiin, että ne olisivat tarpeeksi isoja, jotta niiden mahdollinen kosketuspinta vastaisi WCAG 2.2 2.5.8 *Target Size (Minimum)* AA-tason kriteeriä. Kyseisessä kriteerissä määritetään, että interaktiivisten elementtien kosketuspinta-alan on vastattava vähintään 24x24px-kokoa. (52.) Tästä johtuen ikonin pienimmäksi kooksi määritettiin 1.5x1.5rem (24x24px) varmistamaan, että ikonin koko olisi riittävä, jos sitä käytettäisiin interaktiivisena käyttöliittymäelementtinä.

Tututkin muodot saattavat aiheuttaa käyttäjille sekaannuksia, eikä kaikki muodot ole yksiselitteisiä kaikille. Kaikki käyttäjät eivät välttämättä yhdistä hampurilaisikonin merkitystä navigaatioon, jolloin ikonin yhdistäminen tekstin kanssa parantaa palvelun ymmärrettävyyttä (ks. kuva 16). Yhdistämällä teksti ikonin kanssa, varmistetaan, että kaikki ymmärtävät tarkoituksen, parannetaan

ikonin näkyvyyttä, joka helpottaa etenkin käyttäjiä, joilla on näkörajoitteita, ja suurennetaan interaktiivisten ikonielementtien kosketuspintaa helpottaen käyttäjiä, joilla on motorisia rajoitteita. (51.)



Kuva 16. Hampurilaisikonin merkitys ei välttämättä avaudu kaikille käyttäjille, jolloin sen kanssa kannattaa yhdistää teksti, joka kertoo elementin merkityksen.

SVG-elementin käyttäminen on parempi vaihtoehto webkehityksessä kuin ikonikirjasintyypin käyttäminen, koska ruudunlukijat ja muut avustavat teknologiat tunnistavat SVG-elementin paremmin. SVG-elementille voidaan antaa HTML role -attribuutti, jossa voidaan kertoa, että elementti on kuva. Lisäksi SVG-elementeille voidaan antaa kuvaava otsikko. (51.) Esimerkkikoodi 1 on esitetty HTML SVG -elementti, jolle on annettu role-attribuutti ja <title>-tagi.

```
<svg
     xmlns="http://www.w3.org/2000/svg"
     viewBox="0 -960 960 960"
     role="img"
>
     <title>Check Circle Icon</title>
     <path d="m424-296 282-282-56-56-226 226-114-114-56 56 170 170Zm56
     216q-83 0-156-31.5T197-197q-54-54-85.5-127T80-480q0-83 31.5-
     156T197-763q54-54 127-85.5T480-880q83 0 156 31.5T763-763q54 54
     85.5 127T880-480q0 83-31.5 156T763-197q-54 54-127 85.5T480-80Zm0-
     80q134 0 227-93t93-227q0-134-93-227t-227-93q-134 0-227 93t-93
     227q0 134 93 227t227 93Zm0-320Z"/>
</svg>
```

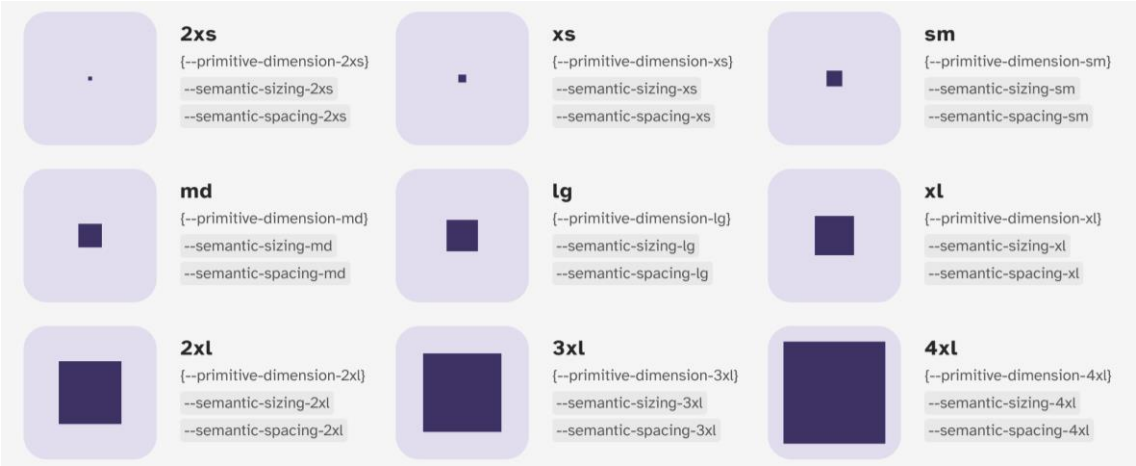
Esimerkkikoodi 1. Material Symbols -ikonikirjaston ikonia käytetty SVG HTML -elementtinä, jolle on annettu role-attribuutti kertomaan, mikä elementti on sekä <title>-tagi.

Jotkut ikonit ovat pelkästään dekoratiivisia, jolloin ne on syytä piilottaa ruudunlukijoilta aria-hidden-attribuutilla. Ikonikirjasintyyppikirjastoa käyttäessä on syytä aina piilottaa ikoni ruudunlukijoilta, jotta vältytään hämmennykseltä, eikä ruudunlukija lue käyttäjälle irrelevanttia tietoa. (51.)

5.2.4 Tila, koot ja asettelu

Tilan järjestelmällinen käyttö on oleellinen osa käyttöliittymiä. Erilaiset tilajärjestelmät, ruudukot ja asettelut tarjoavat sääntöjä, jotka antavat käyttöliittymille yhtenäisen rytmin. Selkeää asettelua on helpompi havainnoida ja ymmärtää. (53.)

Design systemin tilan ja kokojen mittojen määrittelyssä käytettiin 8 pisteen asteikkoa, joka tekee käyttöliittymän rytmistä ennustettavampaa ja helposti havaittavampaa, kun mittasuhteet ovat suhteellisia toisiinsa. 8 pisteen asteikossa arvot ovat jaettavissa 8:lla. (53.) Asteikkoon lisättiin myös puolikas arvo, jos tarvittaisiin pienempää tilaa. Koska design systemissä oltiin päätetty käyttää suhteellista mittayksikköä, vastasi 8px-arvo design systemissä 0.5rem-arvoa. Mittayksiköissä hyödynnettiin primitiivisiä ja semanttisia design tokeneita. Primitiivisissä design tokeneissa määritettiin kattavasti erilaisia numeraalisia kokoja, joihin viitattiin tila- ja kokojärjestelmien semanttisissa design tokeneissa, mikä loi järjestelmän, jonka arvot olivat pikselikoossa: 4px, 8px, 16px, 24px, 32px, 40px, 64px, 80px ja 104px (ks. kuva 17).



Kuva 17. Design systemin tila- ja kokojärjestelmässä käytettiin 8 pisteen asteikkoa luomaan selkeyttä käyttöliittymän rytmiin.

Design systemille suunniteltiin myös asettelujärjestelmä. Design systemi määritettiin käyttämään kolumniruudukkosysteemiä, jossa työpöydän käyttöliittymä käyttäisi 12 kolumnin ruudukkoa, tablettikoot 8 kolumnin ruudukkoa ja mobiilikäyttöliittymät 4 kolumnin ruudukkoa.

WCAG 2.2 -kriteerissä 1.4.10 *Reflow* (AA-taso) ohjeistetaan, että verkkopalveluiden sisältö on pystyttävä esittämään ilman tiedon tai toimintojen menetystä sekä ilman kaksisuuntaista vierittämistä pienillä näytöillä. Tämä tarkoittaa sitä, että käyttöliittymien on oltava responsiivia erikokoisille näytöille tai jos käyttäjä suurentaa selaimen ikkunaa jopa 400% asti. Hyödyntämällä CSS-tyylikielen (*Cascading Style Sheets*) grid, flexbox ja media queries voidaan toteuttaa helpommin responsiivisia käyttöliittymiä. (54.)

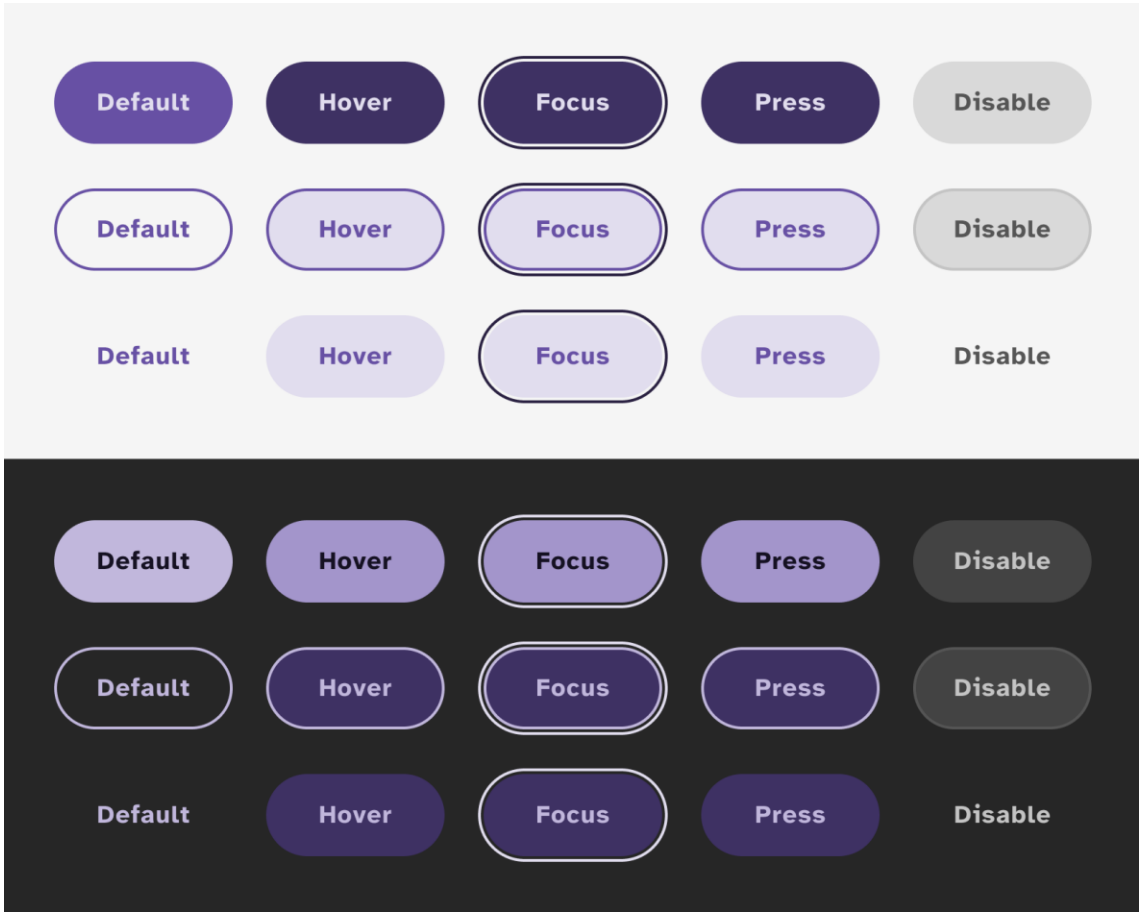
5.3 Komponenttikirjasto

Design systemin tyylikirjaston luomisen jälkeen siirryttiin suunnittelemaan ja toteuttamaan tarvittavia komponentteja. Aikaisemmin tehtyjen rautalankamallien pohjalta, oltiin määritetty etukäteen, että pilottiprojektin käyttöliittymäkomponentit olivat painike, tekstisyöte, pitkän tekstin syöte sekä statusviesti.

Tyylikirjasto oli luonut johdonmukaisen pohjan käyttöliittymäkomponenttien suunnitteluun. Valmiiksi määriteltyjä tyylejä käytettiin käyttöliittymäkomponenteissa, mikä nopeutti ja tehosti työskentelyä. Design systemin typografia ja väri olivat suunniteltu saavutettaviksi, eikä komponentteja suunnitellessa tarvinnut tarkistaa esimerkiksi väriyhdistelmien kontrasteja.

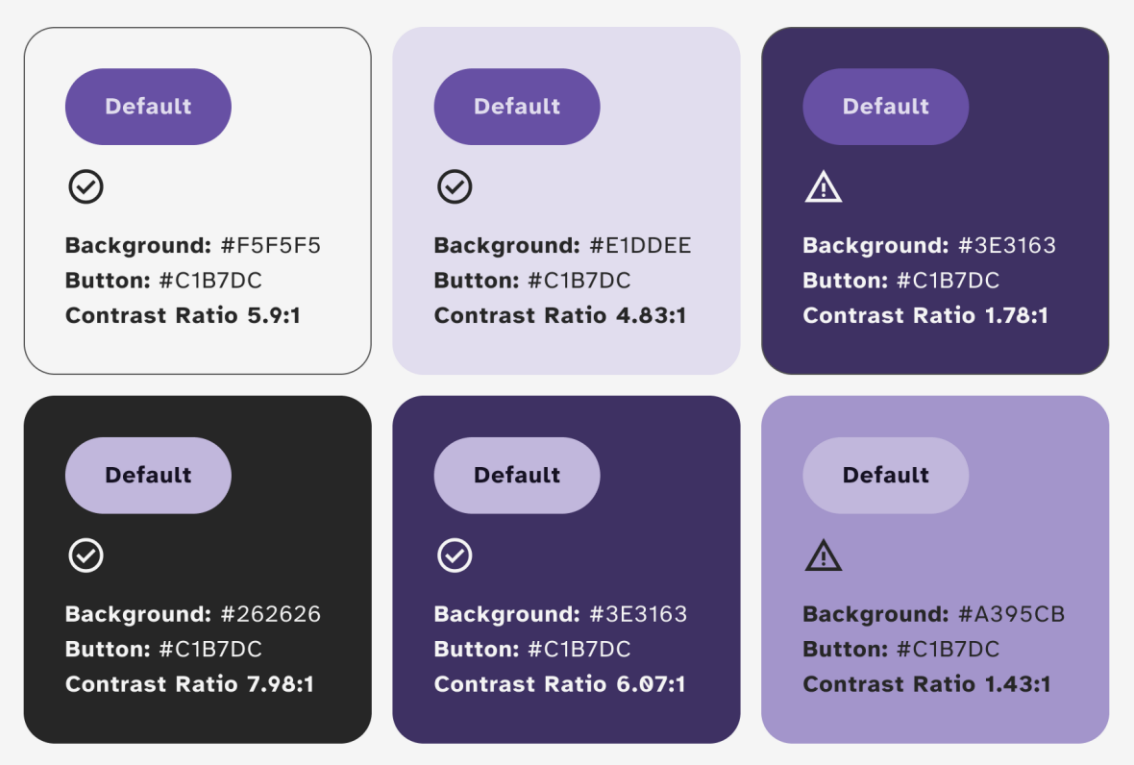
5.3.1 Painike

Painikkeet ovat käyttöliittymien peruselementtejä, joiden tarkoitus on ohjata käyttäjän toimintoja. Painikkeiden saavutettavuus on tärkeää kaikille käyttäjäryhmille. Painikkeesta toteutettiin kolme eri variaatiota: *primary*, *outline* ja *ghost,* jotka viestivät eri asteisista tärkeyksistä käyttäjän ohjaamisessa vuorovaikuttamaan painikkeen kanssa. *Primary*-painike edustaa tärkeintä CTA-toimintoa (*Call to Action*), *outline*-painike viestii toiseksi tärkeintä toimintoa ja *ghost*-painike ohjaa käyttäjää vähiten tärkeimpään toimintoon. Lisäksi toteutettiin erilaiset tyylit painikkeiden tiloille: *hover*, *focus*, *press* ja *disabled* (ks. kuva 18).



Kuva 18. Painikkeen eri variaatiot vaalealle ja tummalle käyttöliittymälle.

Painikkeen suunnittelun yhteydessä testattiin, miten saavutettavuus toteutui eriväristen taustojen kanssa, koska WCAG 2.2 -kriteeri 1.4.11 *Non-text Contrast* (AA-taso) vaatii, että vuorovaikutteisen käyttöliittymäkomponentin ja sen viereisten värien, kuten taustan, kontrastin on oltava vähintään 3:1 (lukuunottamatta epäaktiivista tilaa) (31). Testien kautta dokumentoitiin saavutettavan taustavärin käyttö painikkeen kanssa (ks. kuva 19).



Kuva 19. Painikkeen ja taustavärin kontrastin on oltava vähintään 3:1 täyttääkseen saavutettavuuskriteerin.

Koska painike oli ensimmäinen vuorovaikutteinen käyttöliittymäkomponentti, joka suunniteltiin ja kehitettiin, kiinnitettiin erityistä huomiota komponentin *focus* ja *disabled* tiloihin. Etenkin näppäimistöllä navigoiville käyttäjille visuaalinen kohdistinosoitin on oleellinen keino viestiä, missä käyttäjän osoitin sijaitsee käyttöliittymässä. Yleisesti modernien selainten oletuskohdistinosoittimet ovat riittäviä, ja monet käytetyimmistä selaimista, kuten Google Chrome ja Edge, ovat pyrkineet parantamaan oletuskohdistimien saavutettavuutta. Selainten oletuskohdistinosoittimien saavutettavuudesta käyttöliittymän omien värien kanssa ei kuitenkaan voida olla varmoja, ja oletuskohdistinosoittimiin ei kannata tukeutua, jos haluaa varmistaa saavutettavuuden täysimääräisen toteutumisen. (55.)

Design systemiin suunniteltiin ja toteutettiin oma visuaalinen kohdistinosoitin. WCAG 2.2 -kriteeri 2.4.7 *Focus Visible* (AA-taso) ei ota kantaa, miltä visuaalisen kohdistimen tulee näyttää, joten kohdistinosoittimen suunnittelussa

ja toteutuksessa käytettiin 2.4.13 *Focus Appearance* (AAA-taso) -kriteeriä (31). Kyseisessä kriteerissä määritetään, että kohdistimen on oltava 2px-paksuinen yhtenäinen viiva, jonka rajaama alue on vähintään yhtä iso kuin kohdistettavan komponentin, ja sen on oltava vähintään 3:1 kontrastissa kohdistettuun ja kohdistamattomaan tilaan nähden. (56.) Kohdistimessa hyödynnettiin WCAG 2.2 teknisiä ohjeita siitä, miten *Focus Visible* ja *Focus Appearance* -kriteerit voivat saavuttaa käyttämällä kaksiväristä kohdistinta, joka varmistaa, että kohdistin on näkyvä useammalla taustavärillä (57). Esimerkkikoodissa 2 on esitetty CSS-tiedoston tyylimäärittelyt :focus-visible CSS-pseudoluokalle. Luokalle on toteutettu kaksivärinen rajaus, joka koostuu sisemmästä ja ulommasta reunasta outline- ja box-shadow-tyylien avulla. Kaikki vanhat selaimet eivät tue :focus-visible-luokkaa, jonka vuoksi lisäsin ehdon, joka lisää :focus-luokkaan samat tyylit, jos :focus-visible ei ole tuettu. :focus ja :focus-visible -pseudoluokat toimivat hieman eri tavalla. :focus-pseudoluokka vastaa aina sillä hetkellä kohdistettua elementtiä riippumatta, millä tavalla se on saanut kohdistuksen. :focus-visible-pseudoluokka kohdistuu myös kohdistettuun elementtiin, mutta se näyttäytyy vain silloin, kun selain arvioi käyttäjän tarvitsevan visuaalisen indikaattorin kohdistuksesta, esimerkiksi näppäimistöllä navigoidessa. (58.)

```
*:focus-visible {
  /* Inner indicator */
  outline: var(--semantic-border-width-sm) solid
    var(--semantic-color-border-inverted);
  outline-offset: 0px;
  /* Outer indicator */
  box-shadow: 0px 0px 0px 4px var(--semantic-color-border-focus);
}

@supports not selector(:focus-visible) {
  *:focus {
    /* Inner indicator */
    outline: var(--semantic-border-width-sm) solid
      var(--semantic-color-border-inverted);
    outline-offset: 0px;
    /* Outer indicator */
    box-shadow: 0px 0px 0px 4px var(--semantic-color-border-focus);
  }
}
```

Esimerkkikoodi 2. CSS-tyylimäärittelyt design systemin :focus-visible pseudo-luokalle, joka lisää indikaattorin kun kohdistettavaan komponenttiin siirretään näppäimistökohdisin.

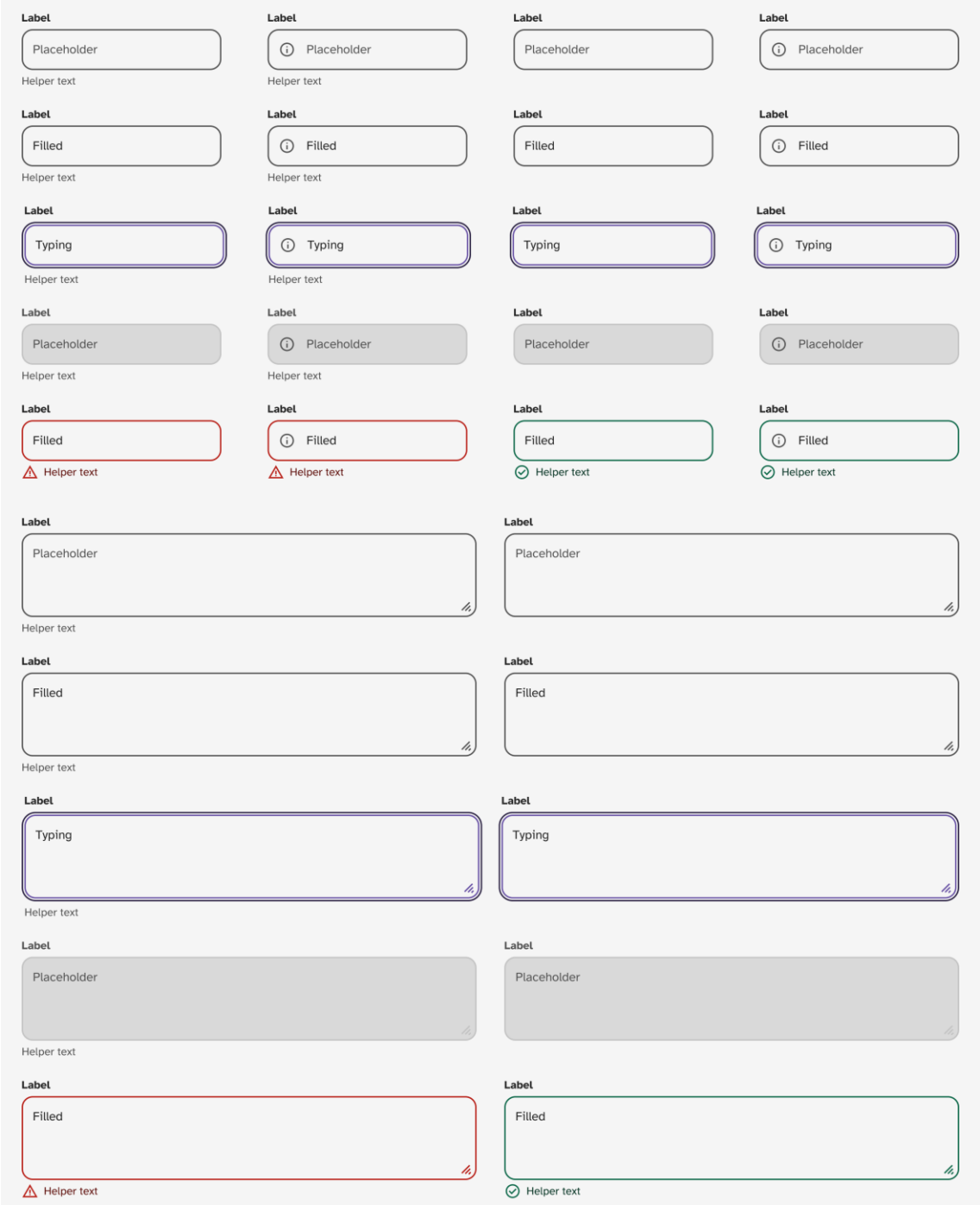
Käyttöliittymäkomponenttien *disabled*-tila on melko yleinen tapa viestiä siitä, ettei komponentilla voi suorittaa jotain toimintoa. WCAG 2.2 -kriteeri 1.4.11 *Non-text Contrast* (AA-taso) ei vaadi, että epäaktiivisten komponenttien kontrasti viereisiin väreihin olisi vähintään 3:1 (31). Painikkeessa varmistettiin kuitenkin, että painikkeen teksti vastaisi riittävää kontrastia. Epäaktiivisiin käyttöliittymäelementteihin liittyy monia haasteita. Esimerkiksi HTML-attribuutti disabled=true estää komponentin painamista sekä kohdistimen vastaanottamista, mikä puolestaan estää ruudunlukijoita havainnoimasta kyseistä komponenttia. Epäaktiivinen komponentti ei kerro miten tilanne korjataan, vaan sen yhteydessä olisi tärkeää viestiä käyttäjälle, mitä hänen kuuluu tehdä tilanteen korjaamiseksi. Vaihtamalla disabled-attribuutti aria-disabled-attribuuttiin voidaan varmistaa, että komponentti on myös ruudunlukijoille havaittavissa ja se voi vastaanottaa kohdistuksen. Aria-disabled-attribuutti ei kuitenkaan estä komponentin toimintaa, vaan on kehittäjän vastuulla varmistaa, ettei esimerkiksi epäaktiivinen painike lähetä lomakkeen tietoja eteenpäin. (59.)

Suunnittelijan ja kehittäjän näkökulmasta on myös huomioitava, että painikkeet ja linkit ovat eri asioita. Painikkeiden tarkoitus on suorittaa jokin toiminto käyttöliittymässä, ja linkit ohjaavat käyttäjän jonnekin muualle. Yleisesti nämä kaksi elementtiä näyttävät erilaisilta. Kaikki käyttäjät eivät kuitenkaan voi havaita visuaalisia eroavaisuuksia, ja he käyttävät muita tapoja elementtien havaitsemiseksi. Lisäksi suunnittelijat voivat joskus suunnitella yksittäisen linkin näyttämään painikkeelta tai toisinpäin. Tästä syystä on tärkeää kommunikoida, milloin kyseessä on HTML-painike-elementti <button> ja milloin linkkielementti <a>, sillä ne viestivät ohjelmallisesti eri asioita. (60.) Painikekomponentista kehitettiin mukautuva elementti, jolle on helppo lisätä arvo, jos sitä käytetään <a>-linkkielementtinä.

5.3.2 Tekstisyöteet

Design systemin seuraavina käyttöliittymäkomponentteina toteuttiin tekstisyötteet, <input>-elementin yksirivisenä tekstisyötteenä ja <textarea>-

elementin monirivisenä tekstisyötteenä. Tekstisyötteistä suunniteltiin ja kehitettiin useampi eri variaatio ja niiden eri tilat (ks. kuva 20). Painike-elementistä tuttuja oppeja hyödynnettiin esimerkiksi *disabled*- ja *focus*-tiloihin.



Kuva 20. Suunniteltu yksirivinen ja monirivinen tekstisyöte ja niiden eri variaatiot ja tilat.

Tekstisyötteitä suunnitellessa ja toteutettaessa kiinnitettiin tarkasti huomiota, miten elementeissä viestittäisiin syntyneistä virheistä. WCAG 2.2 3.3.1 *Error Identification* (A-taso) -kriteerissä ohjeistetaan, että käyttäjälle on ilmoitettava virheestä ja tarjottava kuvaus siitä. Kriteerin tarkoituksena on antaa käyttäjälle riittävä tieto siitä, että virhe on tapahtunut. Kriteerin onnistumiseen ei riitä, että esimerkiksi epäonnistuneen lomakkeen lähetyksen yhteydessä käyttäjälle näytetään lomake uudestaan ilman vihjettä, mistä lähetyksen epäonnistuminen johtui. (35.) Koska kriteerissä 1.4.1 *Use of Color* (A-taso) ohjeistetaan, ettei pelkällä värillä saa viestiä tietoa, tekstisyötteiden virheiden käsittelyyn yhdistettiin teksti, ikoni ja väri, jotta varmistettiin, että virhe olisi mahdollista havaita useammalla aistilla (34). Lisäksi jotta ruudunlukijat voisivat tunnistaa, että syötteessä on virhe, lisättiin syötteille aria-invalid-attribuutti, jotta avustavat teknologiat tunnistaisivat virheen ohjelmallisesti (61). Esimerkkikoodi 3 on esitetty yksirivisen tekstisyötteen React-komponentista osa, jossa esiintyy elementille annettavia erilaisia attribuutteja, kuten aria-invalid ja aria-disabled.

```
<input
     id={id}
     type={type}
     placeholder={placeholder}
     className={inputClasses}
     aria-describedby={errorId || successId || helperId || undefined}
     aria-disabled={disabled}
     aria-invalid={invalid}
     {...props}
/>
```

Esimerkkikoodi 3. Tekstisyötteen React-komponentti, jolle voi antaa aria-invalid ja aria-disabled-attribuutteja parantamaan komponentin saavutettavuutta avustavien teknologioiden kanssa.

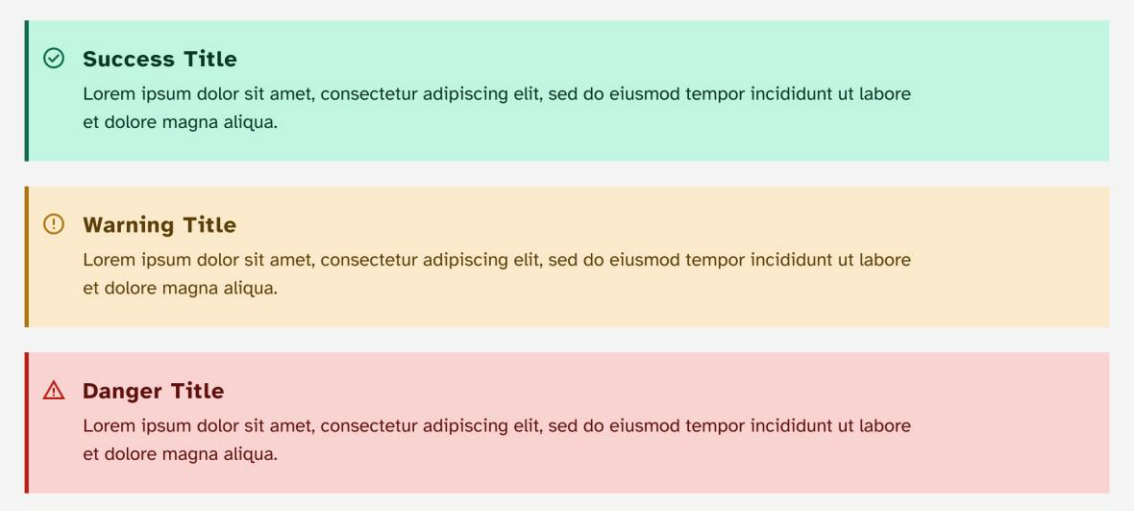
WCAG 2.2 3.3.2 *Labels or Instructions* (A-taso) -kriteerissä ohjeistetaan käyttämään nimilappuja (eng. *label*) tai ohjeita syötteille, jotta käyttäjät tietävät, millaista tietoa syötteeseen pitää antaa. Kriteerin onnistumiselle on annettu useampi tekniikka, joista kuvaavien nimikylttien tarjoaminen on pakollinen tekniikka. Lisäksi kriteerin onnistumisen kannalta myös toisen hyväksytyn tekniikan täytyy toteutua. (62.) Nimikylttien lisäämisen pakolliseksi tekemisen lisäksi määritettiin, että yksiriviselle syötteelle on pakko valita, minkä tyyppistä tietoa se vastaanottaa. Koska käyttöliittymäkomponentti oli vain tekstiä

vastaanottava syöte, määritettiin että React-komponentti voisi vastaanottaa yhden näistä informaatiotyypeistä, jotka kaikki käyttävät tekstimuotoista merkintää: *date*, *datetime-local*, *email*, *month*, *number*, *password*, *search*, *tel*, *text*, *time*, *url* ja *week*. Tämä tekniikka vastasi yhtä *Label of Instructions* -kriteeriin hyväksytyistä tekniikoista. (63.) Lisäksi molemmille tekstisyöteille oli mahdollista antaa kuvaus, jonka yhdistin ohjelmallisesti syötteeseen aria-describedby-attribuutin avulla (64).

Monirivisen tekstisyötteen ominaisuudet olivat lähes samanlaiset kuin yksirivisen teksisyötteen. Moniriviselle tekstisyötteelle ei kuitenkaan tarvinnut määritellä erikseen informaatiotyyppiä, koska <textarea>-elementti voi vastaanottaa vain tekstiä.

5.3.3 Statusviesti

Statusviestin käyttötärkoitus oli viestiä käyttäjälle erilaisista tilanteista, kuten lomakkeiden virheistä tai onnistuneesta lomakkeen lähetyksestä. Statusviestille suunniteltiin kolme eri variaatiota aikaisemmin suunniteltujen värien käyttötarkoitusten perusteella. Statusviestin erilaiset suunnitellut variaatiot olivat *success*, *warning* ja *danger*. Suunniteltiin, että käyttöliittymäkomponentin pitäisi sisältää otsikko antamaan riittävä tieto käyttöliittymäkomponentin sisällöstä, ja sisältö, joka kertoisi käyttäjälle tarvittavan informaation statuksesta. Suunnitteluvaiheessa huomioitiin samoja asioita kuin mitä aikaisemmissakin käyttöliittymäkomponenteissa, kuten ettei pelkällä värillä saisi viestiä informaatiota ja että tekstin ja taustavärin välillä tulisi olla riittävä kontrasti. Kuvassa 21 on esitetty statusviestin eri variaatiot.



Kuva 21. Statusviestikäyttöliittymäkomponentin kolme eri variaatiota: *success*, *warning* ja *danger*.

Statisviestin kaltaiselle komponentille ei ole olemassa natiivia HTML-elementtiä, joten selvitettiin, miten kyseisestä käyttöliittymäkomponentista saisi tehtyä saavutettavan. WCAG 2.2 4.1.3 *Status Messages* (AA-taso) -kriteerissä ohjeistetaan, että käyttäjille, jotka käyttävät avustavaa teknologiaa, on kerrottava tilamuutoksista, jotka eivät ota vastaan kohdistinta, jotta myös käyttäjät, jotka eivät näe niitä, saavat tarvittavan tiedon. Statusviestillä tarkoitetaan sisällön muutosta, joka ei ole kontekstin muutos ja joka antaa käyttäjälle tietoa toimenpiteen onnistumisesta tai tuloksista, ohjelman odotustilasta, prosessin edistymisestä tai virheiden olemassaolosta. Jos statusviesti kertoo toimenpiteen onnistumisesta, tuloksista tai sovelluksen tilasta, statusviestille on annettava role=status ja käyttäjälle on tarjottava palaute. Jos informaatiota lähetetään onnistuneesti eteenpäin, tai jos statusviesti sisältää ohjeistuksen tai varoituksen virheistä, sille pitää antaa role=alert ja statusviestin kanssa on yhdistettävä jokin toinen onnistumiskriteeri. Statusviestille valittiin tekniikka, jossa käyttäjälle pitäisi tarjota riittävät kuvaukset tunnistamaan kentät joissa virhe esiintyy. (65.) Esimerkkikoodissa 4 on esitetty elementin React-komponentti, jolle voidaan antaa role-attribuutti riippuen siitä, minkälaisesta statusviestistä on kyse.

```
<section
          className={statusMessageClasses}
          aria-label={ariaLabel}
          role={role}
          {...props}
>
      <div className='thesis-notification-icon-column'>
          <Icon
               color={variant}
               icon={iconVariant}
               size='default'
               ariaHidden />
      </div>
      <div className='thesis-notification-content'>
          <HeadingTag className='thesis-notification-title'>
               {heading}
          </HeadingTag>
          <div className='thesis-notification-caption'>
               {children}
          </div>
      </div>
</section>
```

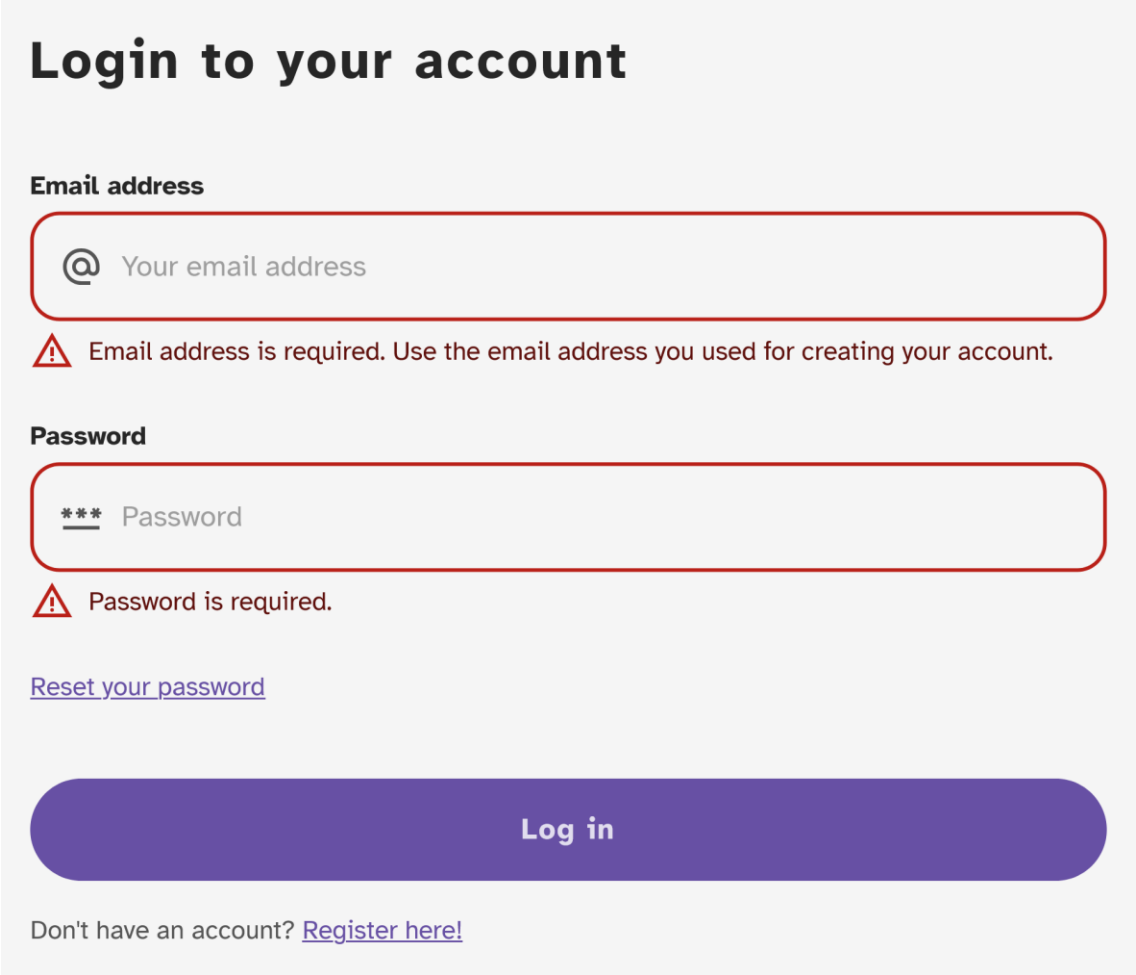
Esimerkkikoodi 4. Statusviestin React-komponentista koodia, jossa sille voidaan antaa role sekä aria-label parantamaan käyttöliittymänkomponentin toimivuutta avustavien teknologioiden kanssa.

Statusviesti on hyvin monimuotoinen käyttöliittymäkomponentti, jolla voi viestiä käyttäjälle esimerkiksi järjestelmän tilanteesta, onnistuneesta lomakkeen lähettämisestä tai virheistä lomakkeessa. Alunperin oli suunniteltu, että statusviestikomponentilla olisi ollut variaatio infoviestille. Todettiin kuitenkin infoviestin käyttötapauksen olevan erilainen verrattuna statusviestiin, jonka tarkoitus on viestiä käyttäjälle nykytilasta, kun taas infoviesti olisi toiminut lisäinformaation antamisena tai informaation korostamisena käyttöliittymässä.

5.4 Lomakepatternit

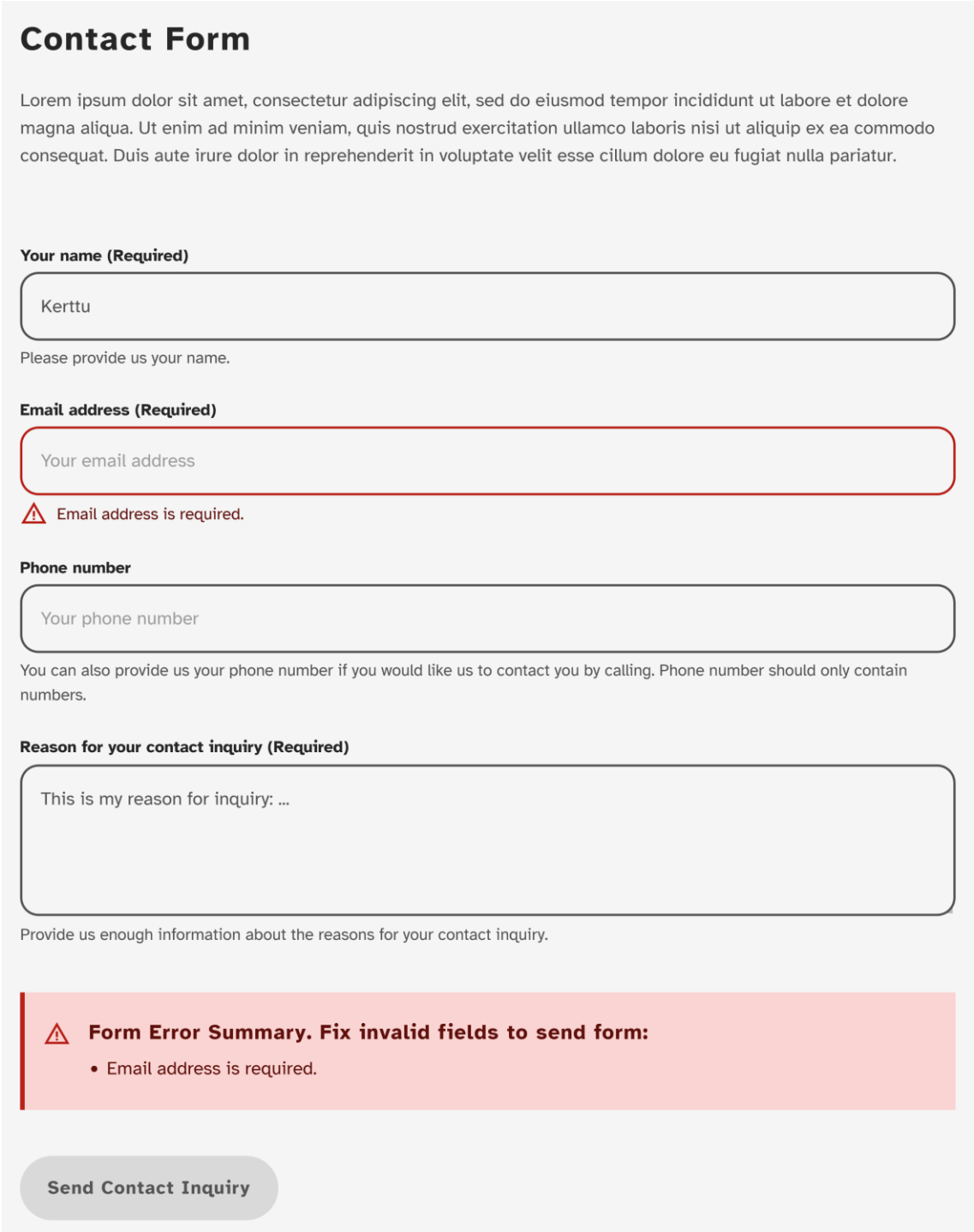
Tyylien ja komponenttien toteutuksen jälkeen niitä sovellettiin suunnitelluissa lomakepatterneissa. Aloitettiin kirjaudu sisään -lomakkeen patternista, joka tarvitsi kaksi yhden rivin tekstisyötettä sekä painikkeen ja erinäisiä tekstielementtejä. Kun tyylit ja komponentit oli suunniteltu ja toteutettu huolellisesti alusta alkaen saavutettavaksi, patternin toteuttaminen onnistui nopeasti. Lomakkeen virheiden käsittelyyn keskityttiin ja miten muodostuneista virheistä viestittäisiin käyttäjälle. Kirjaudu sisään -lomakkeessa virheistä

ilmoitettiin käyttäjälle, kun lomakkeen lähetys ei onnistunut. Kuvassa 22 on kuvakaappaus kirjaudu sisään -lomakepatternin kehitetystä versiosta, kun tekstisyötteissä on virheilmoitukset.



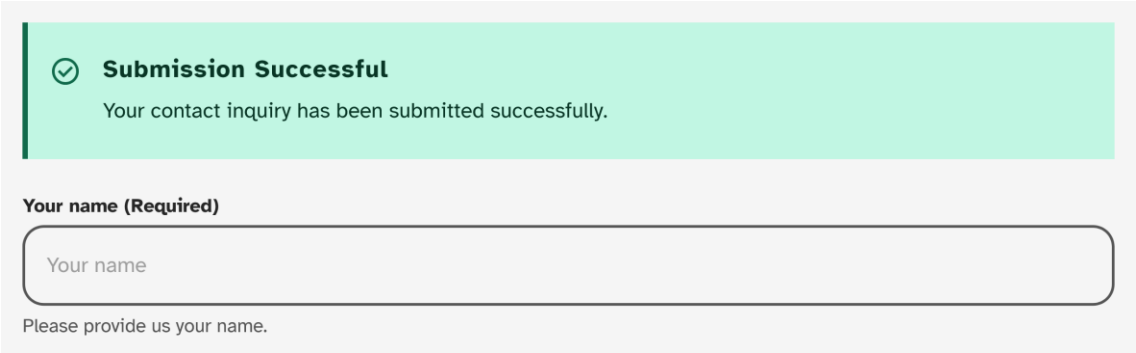
Kuva 22. Kuvakaappaus kirjaudu sisään -lomakkeen patternista.

Yhteydenottopyyntölomakepatternin kehittäminen oli myös nopeaa, kun käytössä oli valmiit käyttöliittymäkomponentit ja tyylit. Yhteydenottopyyntölomakepatternissa jokainen kenttä validoidaan, kun käyttäjän kohdistin siirtyy pois kentästä, jolloin virheestä viestitään käyttäjälle heti. Lisäksi virheet listataan käyttäjälle statusviestille lomakkeen lopussa (ks. kuva 23). Statusviestille on annettu role-attribuutti alert kertomaan, että lomakkeessa on muodostunut virhe.



Kuva 23. Yhteydenottolomakepatternin lopussa virheet kootaan listaan.

Lomakepatternissa käytettiin myös statusviestikäyttöliittymäkomponenttia kertomaan onnistuneesta lomakkeen lähettämisestä (ks. kuva 24). Statusviestille oli lisätty role-attribuuttiksi status kertomaan, että kyseinen komponentti kertoo onnistuneesta toiminnosta.

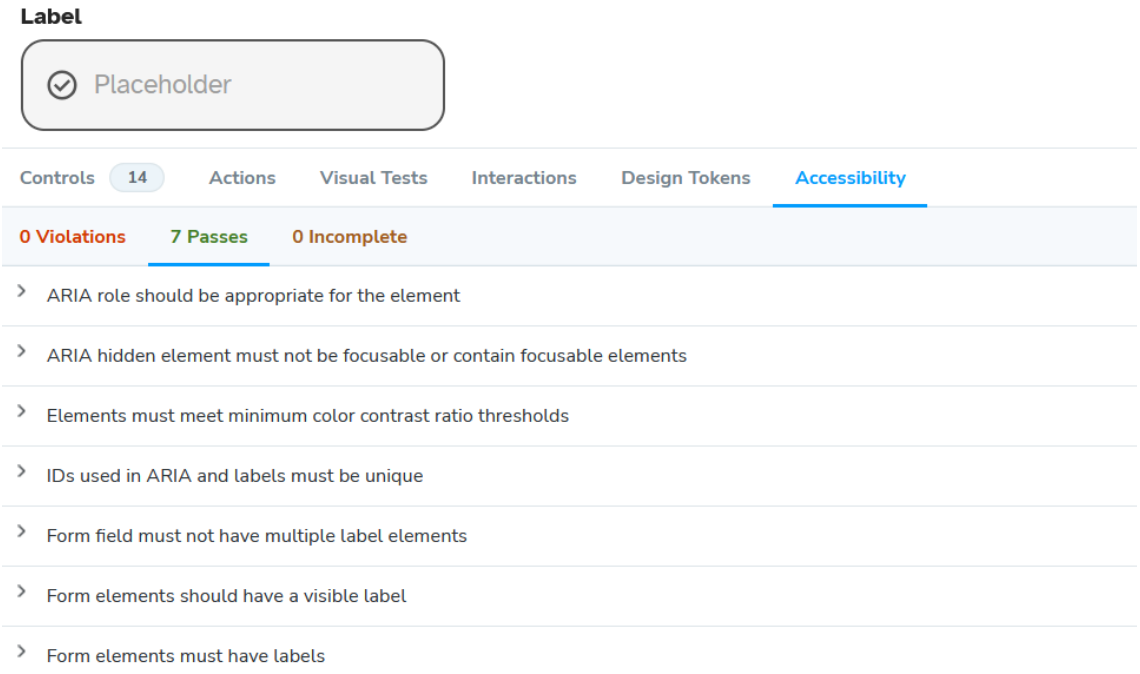


Kuva 24. Kun lomake on lähetetty onnistuneesti, käyttäjälle ilmoitetaan siitä statusviestillä.

Huolehtimalla lomakkeiden virheiden käsittelystä ja tarjoamalla käyttäjälle riittävät ohjeet virheiden korjaamiseen ja oikean tiedon antamiseen, varmistettiin lomakkeiden saavutettavuus. Kun virheiden ilmoittamisessa käytettiin aria-attribuutteja, virheet ilmoitettiin myös avustaville teknologioille. Toimiva virheiden käsittely lisää myös lomakkeiden käytettävyyttä, kun käyttäjän ei tarvitse arvata, mikä lomakkeen lähettämisessä on mennyt pieleen tai missä muodossa haluttu data pitää antaa.

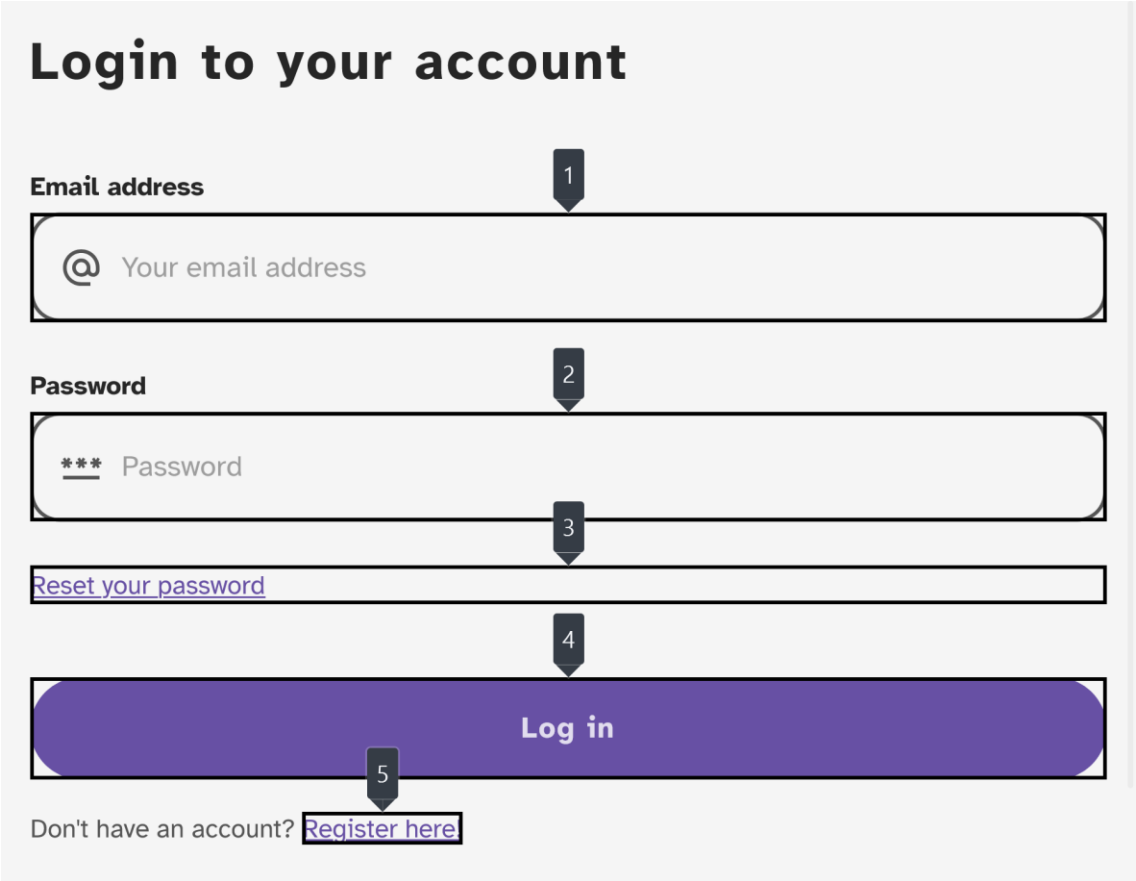
5.5 Saavutettavuuden auditointi

Design systemin suunnittelun ja kehityksen yhteydessä tehdyn saavutettavuustyön ja WCAG 2.2 -ohjeistuksen hyödyntämisen lisäksi design systemin kehitettyjen komponenttien ja patternien saavutettavuus auditoitiin. Auditoimisessa käytettiin Storybookissa saavutettavuuden automatisoituun testaukseen lisäosaa, joka ajaa komponenteille tietyt saavutettavuustestit ja antaa yhteenvedon toteutetuista testeistä (ks. kuva 25). Lisäosa auditoi renderöidyt elementit WCAG-kriteereihin ja muihin alan hyväksymiin parhaisiin käytänteisiin perustuvien heurustisten ominaisuuksian perusteella (66). Lisäosan automatisoidut testit osoittautuivat hyödylliseksi etenkin ensimmäisenä laadunvarmistajana havaitsemaan räikeimmät puutteet saavutettavuudessa.



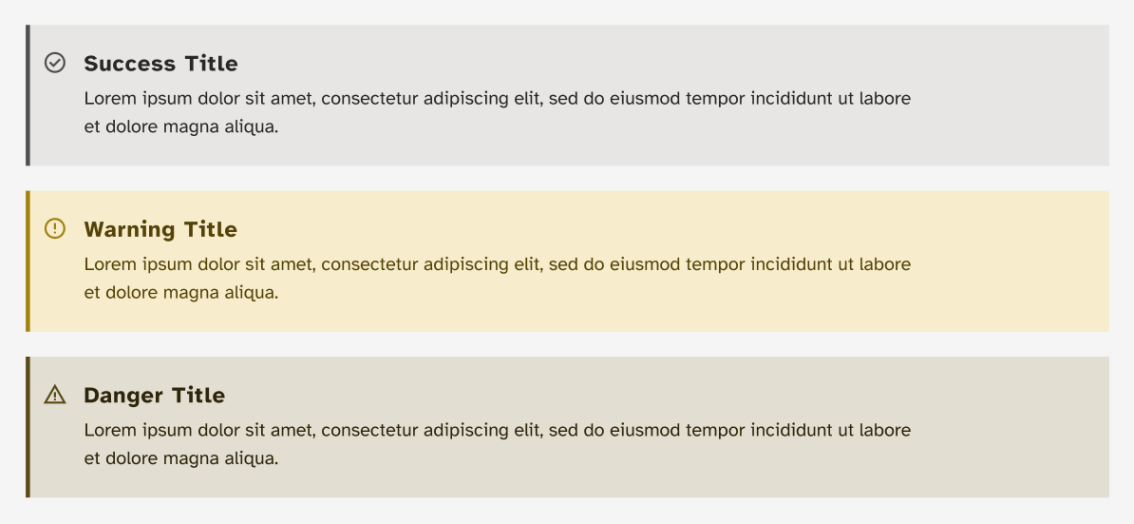
Kuva 25. Storybookin saavutettavuuslisäosan yhteenveto toteutetuista testeistä tesktisyötteelle (67).

Storybookin lisäosan lisäksi design systemin saavutettavuutta auditoitiin Firefox-selaimen saavutettavuustyökalulla. Firefox-selaimen työkalulla pystyttiin tarkistamaan näppäimistökäyttöön liittyviä puutteita ja tarkistamaan lomakepatternien näppäimistökohdistusjärjestyksen. Kuvassa 26 on kirjaudu sisään -lomakepatternin näppäimistökohdistusjärjestys esitettynä Firefox-selaimen saavutettavuustyökalulla.



Kuva 26. Firefox-selaimen saavutettavuustyökalulla voidaan esittää sivustojen näppäimistökohdistusjärjestys ja päätellä sen kautta toteutuuko sivustolla selkeä navigointi (68).

Firefox-selaimen saavutettavuustyökalulla oli mahdollista simuloida erilaisia näköön liittyviä rajoitteita kuten punavihersokeutta. Etenkin värien suunnittelun yhteydessä olisi ollut hyödyllistä simuloida erilaisia näkörajoitteita, jotta olisi voinut varmistaa, että värit ovat mahdollisimman havaittavia eri värisokeuksilla. Värisokeuksia simuloimalla korostui etenkin vaatimus siitä, miten pelkällä värillä informaation jakaminen käyttäjälle ei ole saavutettavaa, ja informaation esittäminen useammalla havaittavalla keinolla takaa sen saavutettavuuden riippumatta käyttäjän rajoitteista. Kuvassa 27 on statusviestikäyttöliittymäkomponentin eri variaatiot punavirhersokeutta simuloiden. Kuvasta voi havaita, kuinka statusviestien värit eivät kerro viestin merkityksestä.



Kuva 27. Statusviestikäyttöliittymäkomponentin eri variaatiot punavihersokeutta simuloiden (68).

Automatisoidun saavutettavuusauditoinnin lisäksi on tärkeää myös testata saavutettavuutta manuaalisesti. Manuaalisen testauksen tärkeys korostui, kun ruudunlukijalla auditoidessa, havaittiin puutteita statusviestin ääneenlukemisen yhteydessä. NVDA:n ilmaisella ruudunlukijaa käyttämällä huomattiin, ettei yhteydenottolomakepatternin onnistuneen lähetyksen jälkeen ruudunlukija lukenut statusviestiä. Selvitettiin, että ruudunlukija ei vastaanottanut dynaamisesti lisätyn komponentin status-roolia, joten lomakepatternille luotiin tyhjä -tagi, jolle annettiin valmiiksi status-rooli. Statusviesti lisättiin dynaamisesti jo olemassa olevaan -tagin sisälle, jolloin ruudunlukija tunnisti muutoksen kyseisen tagin sisällä ja luki ääneen statusviestikomponentin sisällön. Keskimäärin 57% esteettömyysongelmista on tunnistettavissa automatisoidulla testaamisella, jonka vuoksi manuaalinen testaaminen on hyödyllistä, vaikkakin automatisoidulla prosessilla voidaan tehostaa auditointia (69).

6 Yhteenveto

Insinöörityön keskeisimmät tulokset ja havainnot osoittivat, että saavutettavuuden ja design systemien yhdistäminen vaatii sekä suunnittelun

että teknisen toteutuksen huolellista huomiointia. Erityisesti havaittiin, että täysimääräisen saavutettavuuden toteutuminen edellyttää design systemin iterointia sekä yksittäisten elementtien että laajempien yksittäisistä elementeistä koostettujen kokonaisuuksien auditointia. Vaikka yksittäinen tyylikirjaston elementti tai komponenttikirjaston käyttöliittymäkomponentti olisi suunniteltu ja toteutettu saavutettavasti, niiden käytännön sovellukset voivat silti luoda saavutettavuuspuutteita. Tämä havaittiin erityisesti lomakepatternien saavutettavuusauditoinnissa ruudunlukijan kanssa.

Johtopäätöksenä voidaan todeta, että saavutettavuuden ottaminen design systemin suunnittelun ja kehityksen pääarvoksi alusta alkaen parantaa verkkopalveluiden tasa-arvoista käyttäjäkokemusta ja tehostaa laadunvarmistusta ja iterointia kehityksen loppuvaiheessa. Yksittäisten savutettavuuspuutteiden korjaaminen on helpompaa ja tehokkaampaa kuin laajempien puutteiden arviointi ja korjaaminen, kun ongelmat voivat liittyä design systemin perusperiaatteisiin ja vaikuttaa useampiin käyttöliittymäkomponentteihin ja patterneihin. Design systemien dokumentaatiolla on tässä oleellinen rooli, erityisesti saavutettavuuden dokumentoinnissa, jotta kaikilla design systemiä käyttävillä työryhmillä on yhtenäiset ohjeet ja tavoitteet saavutettavuuden toteutumiselle ja käytänteille. Dokumentaatio myös tehostaa työtä ja vähentää jatkuvaa saavutettavien toteutustapojen etsimisestä johtuvaa kuormaa, kun design systemiin on dokumentoitu hyväksytyt käytänteet esimerkiksi värien saavutettava käyttö sekä ohjeet ARIA-attribuuttien hyödyntämiseen.

Koska insinöörityössä tarkasteltiin saavutettavuutta erityisesti visuaalisesta ja teknisestä näkökulmasta, kognitiivisen saavutettavuuden laajempi tarkastelu rajattiin työn ulkopuolelle. Lisäksi toteutetun design systemin saavutettavuutta ei arvioitu oikeiden käyttäjien kanssa, mikä rajasi auditoinnin tuloksia käyttäjältä saadun tiedon osalta. Tästä syystä design systemin saavutettavuusarvioinnin tulokset painottuvat tekniseen arviointiin.

Jatkoa ajatellen saavutettavan design systemin kehittämisessä olisi olennaista ottaa mukaan design systemin, etenkin sen patternien, testaamiseen oikeita käyttäjiä. Kognitiivisen saavutettavuuden huomioiminen parantaisi myös design systemin saavutettavuutta erityisesti sisällöntuotannossa. Tehdyn design systemin dokumeentio jäi lopuksi hyvin niukaksi, joten sen kehittäminen sekä selkeämpien ohjeiden ja käytänteiden laatiminen olisi myös selkeä seuraava askel design systemin kehittämisessä.

Insinöörityö antoi arvokasta teoreettista ja käytännön tietoa saavutettavuuden ja design systemien yhteensovittamisessa. Aiheen merkkitävyys korostuu laajenevan esteettömyysdirektiivin myötä sekä nykyaikaisen verkkokehityksen kannalta, jossa käyttöliittymät koostuvat design systemeistä. Työ herätti ajatuksia siitä, millä tasolla saavutettavuuden on tarpeellista toteutua design systemillä toteutettujen palveluiden laadusta ja käyttäjäryhmistä riippuen – vai onko kenties tarpeellista toteuttaa kaikki palvelut saavutettavasti, jotta jokaisella on tasavertainen pääsy palveluihin riippumatta rajoitteista.

Lähteet

1 Verkkosisältöjen saavutettavuus. Verkkoaineisto. Saavutettavuuskirjasto Celia. <https://www.saavutettavasti.fi/verkkosisaltojen-saavutettavuus/>. Luettu 1.9.2024.

2 Yleistä saavutettavuudesta. Verkkoaineisto. Aluehallintovirasto. <https://www.saavutettavuusvaatimukset.fi/yleista-saavutettavuudesta/>. Luettu 1.9.2024.

3 Kenelle saavutettavuus on tärkeää. Verkkoaineisto. Aluehallintovirasto. <https://www.saavutettavuusvaatimukset.fi/yleista-saavutettavuudesta/kenelle-saavutettavuus-on-tarkeaa/>. Luettu 1.9.2024.

4 Kuka hyötyy saavutettavuudesta. 2023. Verkkoaineisto. Papunet. <https://papunet.net/saavutettavuus/miksi-saavutettava/kuka-hyotyy-saavutettavuudesta/>. 3.4.2023. Luettu 1.9.2024.

5 Digipalvelulain vaatimukset. Verkkoaineisto. Aluehallintovirasto. <https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/>. Luettu 1.9.2024.

6 Esteettömyysdirektiivi. Verkkoaineisto. Sosiaali- ja terveysministeriö. <https://stm.fi/esteettomyysdirektiivi>. Luettu 1.9.2024.

7 WCAG 2 Overview. 2005. Verkkoaineisto. W3C Web Accessibility Initiative (WAI). <https://www.w3.org/WAI/standards-guidelines/wcag/>. Päivitetty 7.3.2024. Luettu 5.9.2024.

8 Introduction to Understanding WCAG 2. 2023. Verkkoaineisto. W3C Web Accessibility Initiative (WAI). <https://www.w3.org/WAI/WCAG22/Understanding/intro>. Päivitetty 27.8.2024. Luettu 5.9.2024.

9 Tietoa WCAG-ohjeistuksesta. Verkkoaineisto. Aluehallintovirasto. <https://www.saavutettavuusvaatimukset.fi/digipalvelulain-vaatimukset/tietoa-wcag-kriteereista/>. Luettu 5.9.2024.

10 Web Content Accessibility Guidelines (WCAG) 2.2. 2023. Verkkoaineisto. W3C Web Accessibility Initiative (WAI). <https://www.w3.org/TR/WCAG22/>. Päivitetty 5.10.2023. Luettu 10.9.2024.

11 Understanding Guideline 4.1: Compatible. 2023. Verkkoaineisto. W3C Web Accessibility Initiative (WAI).

<https://www.w3.org/WAI/WCAG22/Understanding/compatible.html>. Päivitetty 2.4.2024. Luettu 10.9.2024.

12 What are Design Systems? 2021. Verkkoaineisto. Interaction Design Foundation – IxDF. <https://www.interaction-design.org/literature/topics/design-systems>. Luettu 10.9.2024.

13 Fessenden, Therese. 2021. Design Systems 101. Verkkoaineisto. Nielsen Norman Group. <https://www.nngroup.com/articles/design-systems-101/>. 11.4.2021. Luettu 10.9.2024.

14 Frost, Brad. 2013. Atomic Design. Verkkoaineisto. <https://bradfrost.com/blog/post/atomic-web-design/>. Luettu 14.9.2024.

15 Aalto design system. Verkkoaineisto. Aalto University. <https://brand.aalto.fi/ds>. Luettu 14.9.2024.

16 Suomi.fi Design System. Verkkoaineisto. Digi- ja väestötietovirasto. <https://designsystem.suomi.fi/>. Luettu 14.9.2024.

17 Helsinki Design System. Verkkoaineisto. Helsinki. <https://hds.hel.fi/>. Luettu 14.9.2024.

18 User Centered Design (UCD). 2016. Verkkoaineisto. Interaction Design Foundation – IxDF. <https://www.interaction-design.org/literature/topics/user-centered-design>. Luettu 19.9.2024.

19 Design for All. 2016. Verkkoaineisto. Interaction Design Foundation – IxDF. <https://www.interaction-design.org/literature/topics/design-for-all>. Luettu 19.9.2024.

20 Wikipedia, the free enclycopedia. Verkkoainesto. Wikipedia. <https://en.wikipedia.org/wiki/Main_Page>. Luettu 19.9.2024.

21 Cook, Anna. 2021. Auditing Design Systems for Accessibility. Verkkoaineisto. Deque Systems. <https://www.deque.com/blog/auditing-design-systems-for-accessibility/>. 4.5.2021. Luettu 19.9.2024.

22 Common Accessibility Pitfalls in Web Design (And How to Avoid Them). 2024. Verkkoaineisto. AllAccessible. <https://www.allaccessible.org/common-accessibility-pitfalls-in-web-design-and-how-to-avoid-them/>. 25.6.2024. Luettu 19.9.2024.

23 Understanding Success Criterion 1.4.3: Contrast (Minimum). 2023. Verkkoaineisto. W3C Web Accessibility Initiative (WAI).

<https://www.w3.org/WAI/WCAG22/Understanding/contrast-minimum>. Päivitetty 16.4.2024. Luettu 19.9.2024.

24 Understanding Success Criterion 1.4.11: Non-text Contrast. 2023. Verkkoaineisto. W3C Web Accessibility Initiative (WAI). <https://www.w3.org/WAI/WCAG22/Understanding/non-text-contrast.html>. Päivitetty 13.9.2024. Luettu 20.9.2024.

25 WebAIM: Color Contrast Checker. Verkkoaineisto. WebAIM. <https://webaim.org/resources/contrastchecker/>. Luettu 20.9.2024.

26 Introducing accessibility in typography. Verkkoaineisto. Google Fonts. <https://fonts.google.com/knowledge/readability_and_accessibility/introducing_accessibility_in_typography>. Luettu 20.9.2024.

27 Understanding Success Criterion 1.4.4: Resize. 2023. Verkkoaineisto. W3C Web Accessibility Initiative (WAI). <https://www.w3.org/WAI/WCAG22/Understanding/resize-text.html>. Päivitetty 07.8.2024. Luettu 20.9.2024.

28 Technique H42: Using h1-h6 to identify headings. 2023. Verkkoaineisto. W3C Web Accessibility Initiative (WAI). <https://www.w3.org/WAI/WCAG22/Techniques/html/H42>. Päivitetty 16.4.2024. Luettu 20.9.2024.

29 How to structure headings for web accessibility. 2017. Verkkoaineisto. Nomensa. <https://www.nomensa.com/blog/how-structure-headings-web-accessibility/>. 27.7.2017. Luettu 20.9.2024.

30 Understanding Success Criterion 2.1.1: Keyboard. 2023. Verkkoaineisto. W3C Web Accessibility Initiative (WAI). <https://www.w3.org/WAI/WCAG22/Understanding/keyboard.html>. Päivitetty 27.8.2024. Luettu 20.9.2024.

31 Understanding Success Criterion 2.4.7: Focus Visible. 2023. Verkkoaineisto. W3C Web Accessibility Initiative (WAI). <https://www.w3.org/WAI/WCAG22/Understanding/focus-visible.html>. Päivitetty 14.5.2024. Luettu 20.9.2024.

32 Understanding Success Criterion 2.4.1: Bypass Blocks. 2023. Verkkoaineisto. W3C Web Accessibility Initiative (WAI). <https://www.w3.org/WAI/WCAG22/Understanding/bypass-blocks.html>. Päivitetty 16.4.2024. Luettu 20.9.2024.

33 Yle.fi. Verkkoaineisto. Yleisradio. <https://yle.fi/>. Luettu 15.10.2024.

34 Understanding Success Criterion 1.4.1: Use of Color. 2023. Verkkoaineisto. W3C Web Accessibility Initiative (WAI). <https://www.w3.org/WAI/WCAG22/Understanding/use-of-color.html>. Päivitetty 10.7.2024. Luettu 20.9.2024.

35 Understanding Success Criterion 3.3.1: Error Identification. 2023. Verkkoaineisto. W3C Web Accessibility Initiative (WAI). <https://www.w3.org/WAI/WCAG22/Understanding/error-identification.html>. Päivitetty 14.5.2024. Luettu 20.9.2024.

36 Design Systems: The Key to Accelerating Accessible UX Design. 2023. Verkkoaineisto. Level Access. <https://www.levelaccess.com/blog/design-systems-the-key-to-accelerating-accessible-ux-design/>. 10.5.2023. Luettu 22.9.2024.

37 Wong, Brienne. 2023. Create an accessible design system. Verkkoaineisto. zeroheight. <https://zeroheight.com/help/article/create-an-accessible-design-system/>. Päivitetty 7.6.2023. Luettu 22.9.2024.

38 Figma: The Collaborative Interface Design Tool. Verkkoaineisto. Figma. <https://www.figma.com/>. Luettu 22.9.2024.

39 Get started with Storybook. Verkkoaineisto. Storybook. <https://storybook.js.org/docs>. Luettu 22.9.2024.

40 Accessibility overview. Verkkoaineisto. Material Design. <https://m3.material.io/foundations/overview/principles>. Luettu 22.9.2024.

41 Curtis, Nathan. 2018. “Accessible” Design Systems Don’t Guarantee Accessible Products. Verkkoaineisto. Medium. <https://medium.com/eightshapes-llc/accessible-design-systems-dont-guarantee-accessible-products-3478e3a462ba>. 29.11.2018. Luettu 22.9.2024.

42 Guavara, Gastón. 2023. Accessibility in Design Systems: The Role of Documentation. Verkkoaineisto. Dodonut. <https://dodonut.com/blog/accessibility-in-design-systems/>. 29.8.2023. Luettu 22.9.2024.

43 HTML: A good basis for accessibility. 2024. Verkkoaineisto. MDN. <https://developer.mozilla.org/en-US/docs/Learn/Accessibility/HTML>. Päivitetty 26.7.2024. Luettu 22.9.2024.

44 WAI-ARIA basics. 2024. Verkkoaineisto. MDN. <https://developer.mozilla.org/en-US/docs/Learn/Accessibility/WAI-ARIA_basics>. Päivitetty 4.10.2024. Luettu 5.10.2024.

45 WebAIM: Typefaces and Fonts. 2020. Verkkoaineisto. WebAIM. <https://webaim.org/techniques/fonts/>. Päivitetty 27.10.2020. Luettu 7.10.2024.

46 Zaraysky, Susanna. 2022. From Rebranding to Readability with Atkinson Hyperlegible. Verkkoaineisto. Material Design. <https://m3.material.io/blog/atkinson-hyperlegible-design>. 9.6.2022. Luettu 7.10.2024.

47 Christopher, Esther. 2024. CSS Units – When to Use rem, em, px, and More. Verkkoaineisto. freeCodeCamp. <https://www.freecodecamp.org/news/css-units-when-to-use-each-one/>. 25.1.2024. Luettu 7.10.2024.

48 What Is a Type Scale? 2023. Verkkoaineisto. Supercharge Design. <https://supercharge.design/blog/what-is-a-type-scale>. 5.9.2023. Luettu 7.10.2024.

49 Understanding Success Criterion 1.4.12: Text Spacing. 2023. Verkkoaineisto. W3C Web Accessibility Initiative (WAI). <https://www.w3.org/WAI/WCAG22/Understanding/text-spacing.html>. Päivitetty 14.5.2024. Luettu 7.10.2024.

50 Color. Verkkoaineisto. Figma. <https://www.designsystems.com/color-guides/>. Luettu 7.10.2024.

51 Lange, Andrée. 2023. Improving Icon Usability and Accessibility: 6 Valuable Tips. Verkkoaineisto. The A11Y Collective. <https://www.a11y-collective.com/blog/icon-usability-and-accessibility/>. 21.12.2023. Luettu 7.10.2024.

52 Understanding Success Criterion 2.5.8: Target Size (Minimum). 2023. Verkkoaineisto. W3C Web Accessibility Initiative (WAI). <https://www.w3.org/WAI/WCAG22/Understanding/target-size-minimum.html>. Päivitetty 14.5.2024. Luettu 7.10.2024.

53 Dahl, Elliot. Space, grids, and layouts. Verkkoaineisto. Figma. <https://www.designsystems.com/space-grids-and-layouts/>. Luettu 7.10.2024.

54 Understanding Success Criterion 1.4.10: Reflow. 2023. Verkkoaineisto. W3C Web Accessibility Initiative (WAI). <https://www.w3.org/WAI/WCAG22/Understanding/reflow.html>. Päivitetty 27.8.2024. Luettu 7.10.2024.

55 Roselli, Adrian. 2017. Avoid Default Browser Focus Styles. Verkkoaineisto. <https://adrianroselli.com/2017/02/avoid-default-browser-focus-styles.html>. Päivitetty 5.7.2020. Luettu 9.10.2024.

56 Understanding Success Criterion 2.4.13: Focus Appearance. 2023. Verkkoaineisto. W3C Web Accessibility Initiative (WAI). <https://www.w3.org/WAI/WCAG22/Understanding/focus-appearance>. Päivitetty 25.7.2024. Luettu 9.10.2024.

57 Technique C40: Creating a two-color focus indicator to ensure sufficient contrast with all components. 2023. Verkkoaineisto. W3C Web Accessibility Initiative (WAI). <https://www.w3.org/WAI/WCAG22/Techniques/css/C40>. Päivitetty 14.5.2024. Luettu 9.10.2024.

58 :focus-visible. Verkkoaineisto. 2024. MDN. <https://developer.mozilla.org/en-US/docs/Web/CSS/:focus-visible>. Päivitetty 23.7.2024. Luettu 9.10.2024.

59 Friedman, Vitaly. 2021. A Complete Guide To Accessible Front-End Components. Verkkoaineisto. Smashing Magazine. <https://www.smashingmagazine.com/2021/03/complete-guide-accessible-front-end-components/>. Päivitetty 25.5.2022. Luettu 9.10.2024.

60 Designer Tips: Improving Button Accessibility. 2023. Verkkoaineisto. Level Access. <https://www.levelaccess.com/blog/designer-tips-improving-button-accessibility/>. 7.9.2023. Luettu 9.10.2024.

61 Technique ARIA21: Using aria-invalid to Indicate An Error Field. 2023. Verkkoaineisto. W3C Web Accessibility Initiative (WAI). <https://www.w3.org/WAI/WCAG22/Techniques/aria/ARIA21>. Päivitetty 2.4.2024. Luettu 10.10.2024.

62 Understanding Success Criterion 3.3.2: Labels and Instructions. 2023. Verkkoaineisto. W3C Web Accessibility Initiative (WAI). <https://www.w3.org/WAI/WCAG22/Understanding/labels-or-instructions.html>. Päivitetty 10.7.2024. Luettu 10.10.2024.

63 Technique H44: Using label elements to associate text labels with form controls. 2023. Verkkoaineisto. W3C Web Accessibility Initiative (WAI). <https://www.w3.org/WAI/WCAG22/Techniques/html/H44>. Päivitetty 7.8.2024. Luettu 10.10.2024.

64 Technique ARIA1: Using the aria-describedby property to provide a descriptive label for user interface controls. 2023. Verkkoaineisto. W3C

Web Accessibility Initiative (WAI). <https://www.w3.org/WAI/WCAG22/Techniques/aria/ARIA1>. Päivitetty 2.4.2024. Luettu 10.10.2024.

65 Understanding Success Criterion 4.1.3: Status Messages. 2023. Verkkoaineisto. W3C Web Accessibility Initiative (WAI). <https://www.w3.org/WAI/WCAG22/Understanding/status-messages.html>. Päivitetty 27.8.2024. Luettu 10.10.2024.

66 Accessibility tests. Verkkoaineisto. Storybook. <https://storybook.js.org/docs/writing-tests/accessibility-testing>. Luettu 16.10.2024.

67 Storybook. Versio 8.3.6. 2024. Storybook.

68 Mozilla Firefox. Versio 131.0.3. 2024. Mozilla.

69 Deque Study Shows Its Automated Testing Identifies 57 Percent of Digital Accessibility Issues, Surpassing Accepted Industry Benchmarks. 2021. Verkkoaineisto. Deque Systems. <https://www.deque.com/blog/automated-testing-study-identifies-57-percent-of-digital-accessibility-issues/>. Päivitetty 10.3.2021. Luettu 16.10.2024.