OULUN AMMATTIKORKEAKOULU

MOBIILIPOHJAINEN OPPIMISALUSTA

Timon Poutiainen, Joona Sivonen, Lassi Riekkola
Opinnäytetyö (AMK)
Kevät 2025
Tieto- ja Viestintätekniikka
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma, ohjelmistokehitys

Tekijä(t): Timon Poutiainen, Joona Sivonen, Lassi Riekkola
Opinnäytetyön otsikko: Mobiilipohjainen oppimisalusta
Työn ohjaaja(t): Meija Lohiniva
Työn valmistumislukukausi ja -vuosi: Kevät 2025
Sivumäärä: 61

Opinnäytetyön tarkoituksena oli suunnitella ja kehittää pelillistetty oppimisalusta, joka perustuu React Native-, Expo- ja Firebase -teknologioihin. Sovellus on suunniteltu erityisesti ala-asteikäisille lapsille ja sen tavoitteena on, että sitä käytetään esimerkiksi kouluopintojen tukena.

Projektin aikana tutkittiin miten luoda sovellus, joka olisi helppokäyttöinen ja miellyttävä. Selvitimme erilaisia tapoja pelillistää sovellus ja miten siitä saadaan tehtyä mielenkiintoinen nimenomaan lasten näkökulmasta.

Lopputuloksena tuotettiin sovelluskokonaisuus, joka toimii suunnitellusti. Sovelluksessa on kolme pääasiallista oppimisaihetta: matematiikka, maantieto ja englannin kieli. Jokaisessa pelissä on viisi tasoa, joita pelata. Tulevaisuudessa sovellukseen voisi lisätä lisää oppimisaiheita ja tasoja. Lisäksi tekoälyn hyödyntäminen voisi mahdollistaa yksilöllisesti mukautuvat oppimispolut. Myös opettajille suunnatut työkalut, kuten edistymisen seuranta ja räätälöidyt tehtävät voisivat laajentaa sovelluksen käyttömahdollisuuksia kouluympäristössä.

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Information Technology, Software Development

Author(s): Timon Poutiainen, Joona Sivonen, Lassi Riekkola
Title of thesis: Mobiilipohjainen oppimisalusta
Supervisor(s): Meija Lohiniva
Term and year when the thesis was submitted: Spring 2025
Number of pages: 61

The aim of this thesis was to design and develop a gamified learning platform based on React Native and Expo technologies. The app is specifically designed for elementary school children and aims to provide an engaging and motivating learning environment to support school education. The app includes three main subjects: mathematics, geography, and English.

The gamification of the application is designed to support learning and boost motivation. Players earn rewards and achievements that reinforce learned skills and encourage continued play. Progress is clearly visible, and players can compare their performance with others.

The result of the thesis was an application which works the same way it was planned to. In the future it is possible to add more subjects and levels to enhance the application.

SISÄLLYS

SANASTO

Figma	https://www.figma.com/ . Internetissä saatavilla oleva kaavio työkalu.
Duolingo	Duolingo on suosittu kielen opetus sovellus.
JavaScript	Pääasiassa verkkoympäristössä käytettävä ohjelmointi kieli.
Bézier-käyrä	Bézier-käyrä on matemaattinen käyrä, jota voi käyttää grafiikkasuunnittelussa luomaan sulavia muotoja.
SVG	SVG eli Scalable Vector Graphics on XML-merkintä-kieleen perustuva tapa esittää vektorikuvia.
Firebase	https://firebase.google.com/ . Googlen kehittämä tieto-kanta palvelu.
AsyncStorage	React Native ohjelmoinitkielen ominaisuus, joka antaa tallentaa pienikokoista tietoa lokaalisti.

1 JOHDANTO

Opinnäytetyön tavoitteena on kehittää pelillistetty oppimissovellus, joka on suunnattu alakouluikäisille lapsille. Oppimisalusta sisältää kolme peliä, jotka kattavat matematiikan perusteet, englannin kielen opetuksen sekä maantiedon. Maantietopelissä käyttäjälle esitetään jonkin valtion lippu, ja hänen tehtävänään on tunnistaa, minkä maan lipusta on kyse. Pelit on suunniteltu edistämään lasten oppimista tarjoamalla interaktiivisia ja motivoivia tehtäviä.

Oppimisalustan kehittämisessä on hyödynnetty pedagogisia ja teknologisia lähestymistapoja, jotka tukevat lapsen kognitiivista kehitystä. Oppimispsykologiset periaatteet, kuten palautteen merkitys ja vaiheittainen oppiminen, on otettu huomioon. Sovelluksen tavoitteena on tarjota positiivisia oppimiskokemuksia ja kannustaa lapsia aktiiviseen osallistumiseen.

Pelillistä oppimista on käytetty menestyksekkäästi monilla koulutustasoilla, ja se on erityisen hyödyllinen alakouluikäisten lasten kohdalla, jotka hyötyvät visuaalisesta ja kokemuksellisesta oppimisesta. Tämä näkemys tukee sovelluksen rakennetta, sillä se kannustaa lapsia oppimaan itsenäisesti ja hauskalla tavalla. (27)

Teknologisesti oppimisalusta on kehitetty ottaen huomioon saavutettavuus sekä responsiivisuus. Sovellus mukautuu erilaisiin näytön kokoihin, jotta sitä voi käyttää eri mobiililaitteilla. Lisäksi sovelluksen käyttöliittymän teossa on otettu huomioon hyvät suunnitteluperiaatteet, kuten selkeys ja helppokäyttöisyys.

Saavutettavuusnäkökohdat, kuten värien kontrastit ja fonttien selkeys on otettu huomioon sovelluksen kehityksessä. Oppimisalustan pedagoginen tausta perustuu konstruktivistiseen oppimiskäsitykseen, jossa oppiminen nähdään aktiivisena prosessina, jossa lapset rakentavat tietoa aiempien kokemustensa pohjalta.

Tässä opinnäytetyössä tarkastellaan oppimisalustan kehitysprosessia teknologisten valintojen, pedagogisten periaatteiden ja käyttäjäkokemuksen osalta. Opinnäytetyön tavoitteena on tarjota kokonaisvaltainen kuva oppimisalustan suunnittelusta ja toteutuksesta sekä arvioida sen pedagogista vaikuttavuutta.

Tulevaisuudessa oppimisalustaa voidaan laajentaa lisäämällä monipuolisuutta eli uusia eri aihealueiden pelejä ja sisältöä, jotka tukevat laajempaa oppimiskokonaisuutta. Oppimisalustan kehittäminen on jatkuva prosessi, jossa käyttäjäpalautteen ja uusien tutkimustulosten avulla voidaan parantaa sovelluksen toimivuutta ja pedagogista tehokkuutta.

Oppimisalustassa käytetään Firebasea ja AsyncStoragea tietojen tallentamiseen. Koska sovellus on suunnattu nuorille lapsille, tietoturva ja yksityisyys ovat keskeisiä näkökohtia. Henkilökohtaisia ja arkaluonteisia tietoja ei tallenneta, vaan käytämme Firebase Authenticationia anonyymiin kirjautumiseen ja pelidatan tallennukseen. AsyncStoragea hyödynnetään vain väliaikaisiin asetuksiin, kuten tumma/vaaleateeman valintaan, mutta ei arkaluontoisten tietojen säilyttämiseen.

2 KÄYTETYT TEKNOLOGIAT

2.1 Expo

Expo on avoimen lähdekoodin kehys, joka helpottaa JavaScriptillä ja React Native -teknologialla tehtyjen mobiilisovellusten kehittämistä. Expo tarjoaa kehittäjille laajan työkalupaketin, joka tekee sovellusten tekemisestä helpompaa ja mahdollistaa nopean prototyyppien luomisen ilman monimutkaisia asetuksia. Yksi Expon ominaisuuksista on "Expo Go" -sovellus. Sen avulla kehittäjät voivat esikatsella ja testata sovelluksia reaaliaikaisesti ilman, että niitä tarvitsee erikseen kääntää tai asentaa manuaalisesti. Kun sovellus käynnistetään Expo Go:ssa, se luo QR-koodin, jonka avulla kehittäjä voi ladata ja testata sovellustaan suoraan mobiililaitteellaan. Tämä eliminoi tarpeen käyttää USB-kaapeleita tai suorittaa monimutkaisia asennusprosesseja. (1)

2.2 React Native

Sovellus toteutettiin käyttämällä React Nativea, joka on JavaScript-pohjainen kehys. Se mahdollistaa natiivien mobiilisovellusten kehittämisen käyttämällä samoja periaatteita ja komponentteja kuin web-kehityksessä käytettävä React. React Native on suunniteltu erityisesti parantamaan mobiilisovellusten kehityksen nopeutta ja kustannustehokkuutta ja sillä onkin mahdollista kehittää sovelluksia niin Androidille kuin myös iOS:lle.

Valitsimme React Nativen, koska halusimme kehittää mobiilisovelluksen, joka on saatavilla sekä iOS- että Android-alustoilla yhdellä koodipohjalla. React Native tarjosi meille mahdollisuuden säästää kehitysaikaa ja resursseja, sillä voimme jakaa suurimman osan koodista molemmille alustoille. Tämä oli erityisen tärkeää projektin aikarajoitteiden ja budjetin vuoksi. Lisäksi React Native on rakennettu JavaScriptin päälle, mikä mahdollisti web-kehittäjien osallisuuden projektiin ilman, että heidän tarvitsee opetella uusia natiiveja ohjelmointikieliä. Kehitystiimimme arvosti myös React Nativelle tarjottuja työkaluja, kuten live reload ja hot

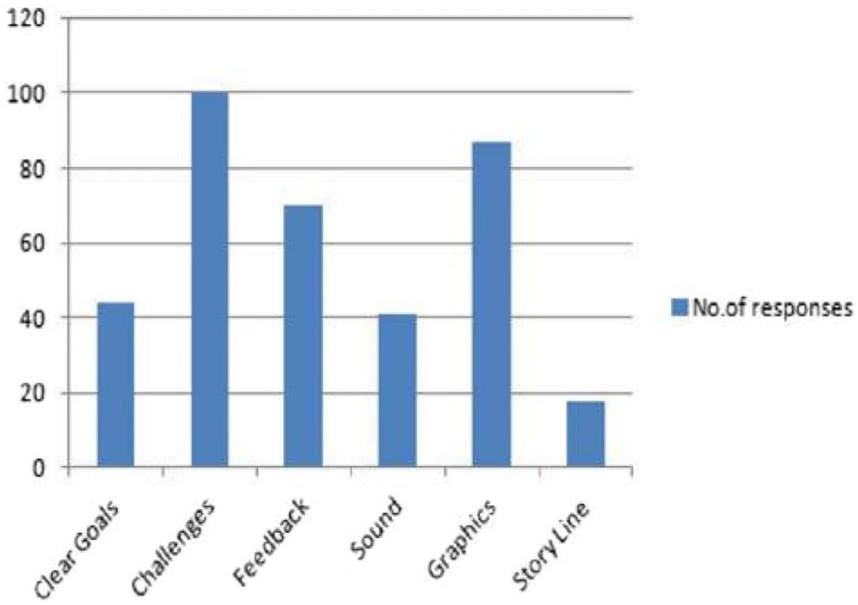
reload, jotka nopeuttivat kehitystyötä ja paransivat tuottavuutta. Sovelluksen kehityksen aikana huomasimme myös, että React Native tarjoaa hyvän suorituskyvyn ja laajennettavuuden, sillä voimme lisätä natiivikoodia tarvittaessa. (2)

2.3 Firebase

Käytämme Google Firebase –pilvipalvelua tallentamaan tietokantaan mm. pelien kysymykset, käyttäjän luomista varten sekä progressiivisten tietojen tallentamista varten kuten suoritettujen tietokilpailujen määrä. Kappaleessa esitellään lyhyesti Firebasen palvelut mitä tarvitsemme sovelluksessa: Firebase Cloud Storage on laajalti skaalautuva ja tietoturvallinen ratkaisu tiedostojen tallentamiseen ja sen voi integroida helposti muiden kuten Firebase Autentikaatio ja Firestore –palvelun kanssa yhteen ja näitä yhdistelemällä voidaan saavuttaa oikeinkin mukava sovelluskokonaisuus. (3) Firebase Autentikaatio palvelu yksinkertaa käyttäjän luomisen ja kirjautumisen, koska se tukee useita todennusmenetelmiä mm. Perinteinen sähköposti/salasana sekä kolmannen osapuolen todennukset ovat myös mahdollisia esim. Google tai Facebook. (4) Firestore on NoSQL tietokanta pilvipalvelussa, jonka avulla voidaan tallentaa ja synkronoida tietoja ns. Reaaliaikaisesti, tämä olisi hyödyllinen, koska Firestoren avulla käyttäjille voidaan näyttää reaaliajassa testituloksia. (5)

3 SUUNNITTELUN TEORIA

Projekti käynnistettiin suunnittelemalla, että miten lasten oppimista voidaan tukea pelillistetyllä mobiilisovelluksella. Tutkimuksessa ”Engaging children with educational content via Gamification” (15) valittiin 120 lasta iältään 9–10, joilta pyydettiin valitsemaan kolme mieluisinta ominaisuutta peleissä. Tämän tutkimuksen tulokset nähdään kuvassa 1.



KUVA 1. Peliominaisuus kysely (15)

Kyselyn lopputuloksena on, että seuraavat kolme ominaisuutta ovat lasten mieluisimpia peleissä: ”Challenges” eli Haasteet; että pelissä on eri tasoja, joita pitää läpäistä, ”Feedback” eli Palaute; palaute siitä kuinka monta pistettä on saanut ja ”Graphics”: hyvät visuaaliset näkymät.

3.1 Käyttöliittymä

Käyttöliittymän suunnittelussa mobiililaitteille rajallisen tilan vuoksi on tärkeää harkita tarkkaan, miten komponentit asetellaan näytölle, jotta käyttäjät eivät kuormitu. Esimerkiksi painikkeiden sijoittaminen peukaloiden ulottuville on oleellista käyttömukavuuden kannalta. Tämän lisäksi tärkeät painikkeet pitää sijoittaa näytössä mahdollisimman näkyville, jotta käyttäjä tietää mitä tehdä ja mistä painaa ilman että hänen pitää ajatella asiaa. (12)

Kuvassa 2 nähdään miten sovelluksessa ”peukaloalue” otetaan huomioon esimerkiksi ”Valitse oikea vastaus” -näkymässä. Käyttäjälle esitetyt vastaukset ovat kaikki ”peukaloalueella”, käyttökokemuksen mukavuuden vuoksi.

KUVA 2. "Valitse oikea vastaus" -näkymä testi versio

Lisäksi kysymys on esitetty selkeästi ja visuaalisesti erottuvasti näytön keskellä, mikä varmistaa, että käyttäjän huomio kiinnittyy heti tärkeimpään sisältöön. Painikkeiden tekstit ovat selkeitä ja yksiselitteisiä, mikä auttaa käyttäjää ymmärtämään, mitä toimintoa kukin painike suorittaa. Tämä selkeys ja yksinkertaisuus parantavat käyttökokemusta, sillä käyttäjä ei jää epävarmaksi siitä, mitä hänen tulisi tehdä seuraavaksi, ja pystyy tekemään valintansa nopeasti ja vaivattomasti. (12)

3.1.1 Värit

Tärkeä osa hyvän sovelluksen tekemisessä on oikeiden värien valinta. Hyvä väripaletti ei pelkästään paranna visuaalista mukavuutta, vaan se myös heijastaa sovelluksen tarkoitusta, kohderyhmää ja luo oikeanlaisen tunnelman käyttäjälle. Värit voivat vaikuttaa siihen, miten käyttäjät kokevat sovelluksen, ja niillä on suuri rooli siinä, kuinka helppokäyttöinen sovellus on (21).

Tämä sovellus on suunnattu erityisesti lapsille, joten on tärkeää valita värit, jotka eivät ole liian voimakkaita tai häiritseviä, mutta kuitenkin kirkkaat ja selkeät, jotta ne kiinnittävät lasten huomion. Värit voivat toimia myös visuaalisena palautteena, joka auttaa lasta ymmärtämään, miten hän suoriutuu. Esimerkiksi jos lapsi vastaa oikein, nappi muuttuu vihreäksi, mikä symboloi oikeaa vastausta ja vahvistaa onnistumisen tunnetta. Jos taas vastaus on väärin, nappi muuttuu punaiseksi, jolloin lapsi saa selkeän visuaalisen vihjeen siitä, että jotain meni pieleen (10).

3.1.2 Äänimaailma lasten pelissä

Äänillä on merkittävä rooli digitaalisten oppimisympäristöjen luomisessa, erityisesti lapsille suunnatuissa peleissä. Taustamusiikki ja ääniefektit voivat tehostaa keskittymistä, lisätä pelin vetovoimaa ja tehdä oppimisesta miellyttävämpää. (16) On tärkeää, että äänimaailma tukee pelin oppimistavoitteita ja soveltuu käyttäjäryhmälleen. Lasten oppimispeleissä taustamusiikin tulisi olla miellyttävää, rauhoittavaa ja innostavaa. Tutkimukset osoittavat, että liian vauhdikas tai monimutkainen musiikki voi häiritä keskittymistä, kun taas pehmeä ja harmoninen tausta voi tukea oppimisprosessia (17). Musiikin tulee olla toistettavaa ilman, että se muuttuu ärsyttäväksi. Tämän vuoksi monet oppimispelit hyödyntävät lyhyitä, melodisesti yksinkertaisia sävelkulkuja, joissa on kevyitä vaihteluita (18).

Ääniefektit ovat tärkeä osa pelin vuorovaikutusta ja käyttäjäkokemusta. Ne voivat tarjota palautetta ja ohjata pelaajaa, mikä tekee oppimisesta sujuvampaa ja kannustavampaa (19). Hyvin suunnitellut ääniefektit voivat:

- **Vahvistaa onnistumisen kokemusta**, esimerkiksi korkeilla ja kirkkailla äänillä oikean vastauksen yhteydessä.
- **Auttaa navigoinnissa**, kuten selkeillä klik-äänillä nappeja painettaessa.
- **Vähentää turhautumista**, käyttämällä pehmeitä ja neutraaleja virheääniä, jotka eivät tunnu rangaistukselta.

Pienten lasten kohdalla liian voimakkaat tai äkilliset äänet voivat olla häiritseviä. Siksi suositeltavaa on käyttää pehmeitä, luonnollisia ääniä, kuten puhallinsoittimia tai luonnonääniä (20).

Yhteenvetona voidaan todeta, että oppimispelin äänimaailman tulee olla huolellisesti tasapainotettu yhdistelmä miellyttävää taustamusiikkia ja tarkoituksenmukaisia ääniefektejä. Hyvin suunniteltu ääniasettelu tukee keskittymistä, motivoi pelaajaa ja parantaa oppimiskokemusta ilman, että se kuormittaa tai häiritsee käyttäjää.

3.1.3 Suorituksen tunne

Luonnollisesti pelillistetyssä oppissovelluksessa on tärkeää ottaa huomioon, että kun saa esimerkiksi tason suoritettua se tuntuu hyvältä pelaajalle. Visuaalisesti tämän voi saada aikaan esimerkiksi jo edellä mainitulla tavalla, missä käyttäjä vastaa oikein niin nappi menee vihreäksi ja kun käyttäjä vastaa väärin nappi menee punaiseksi. Sovelluksessa on myös Tasonäkymä, jossa käyttäjä voi nähdä oman edistymisen visualisoituna siten, että suoritetut tasot ovat vihreitä, tämänhetkinen taso on sininen ja vielä lukitut tasot ovat harmaita.

Saavutuksien luominen peliin, on myös tapa tuottaa edistymisen tunnetta käyttäjälle. Saavutukset ovat tarkoin määrättyjä tavoitteita, joita käyttäjä voi yrittää saavuttaa pelissä. Ne visualisoivat käyttäjän edistymistä, tuntemusta ja omistautumista peliin. Saavutukset usein vaativat, että käyttäjä yrittää tehdä, jotain mitä pelissä suoraan ei ole vaadittu. (11.)

3.2 Saavutettavuus

Modernissa sovelluskehityksessä yksi tärkeistä asioista on saavutettavuuden implementointi, johon sisältyy mm. tumma ja vaalea tila (13), responsiivisuus sekä skaalautuvuus.

Tumma/Vaalea tila:

Tumma tila, eli tumma teema, on erittäin suosittu, sillä se vähentää merkittävästi silmien rasitusta hämärissä työskentely-ympäristöissä ja auttaa säästämään laitteiden akkua. Teemojen toteuttaminen React Nativella voidaan tehdä esimerkiksi käyttämällä Appearance-rajapintaa. Yksi tämän ominaisuuksista on auttaa sovellusta tunnistamaan laitteen valittu väriteema ja päivittämään ulkoasu dynaamisesti. Näin varmistetaan sovelluksen mukautuminen käyttäjän mieltymysten sekä järjestelmän asetusten mukaisesti. (13)

Responsiivisuus:

Responsiivisuudella tarkoitetaan sitä, että sovellus mukautuu eri näyttökokoihin ja suuntiin. React Nativella responsiivisuus voidaan implementoida käyttämällä ns. asettelukomponentteja, kuten flexboxia, sekä matemaattisia yksiköitä, kuten prosentteja, joiden avulla määritellään sopivat mitat. Rajapinta nimeltä Dimensions API voi auttaa komponenttien renderöimisessä eri laitteilla. Responsiivisuus on tärkeä osa sovelluskehitystä, sillä ilman sitä liian pienillä tai isoilla näytöillä näkymän ulkopuolelle voi jäädä käytettävyyden kannalta todella tärkeitä komponentteja. (22)

Skaalautuvuus:

Skaalautuvuudella viitataan sovellusten kasvavien palvelinpyyntöjen käsittelykykyyn ja ominaisuuksien monimutkaisuuteen. Implementoimalla järjestelmä, joka hallitsee tilanhallintaa sekä teemoja ja komponenttien renderöinnin optimointia, voidaan tehdä sovelluksesta skaalautuvampi. (23)

Erilaisten käyttäjien tarpeet saavutettavuudessa:

Saavutettavuus tarkoittaa, että sovellukset on suunniteltu palvelemaan monenlaisia käyttäjiä, mukaan lukien aistihaasteiset, neuroepätyypilliset ja värisokeat henkilöt. Esimerkiksi värisokeille suunnitellut teemat voivat auttaa erottelemaan käyttöliittymän elementtejä tehokkaammin, ja ruudunlukijayhteensopivuus on olennaista näkövammaisille käyttäjille. Neuroepätyypillisten käyttäjien kohdalla saavutettavuuden parantaminen voi tarkoittaa selkeitä navigointiratkaisuja sekä vältettäviä yllättäviä animoituja elementtejä. (24)

Testaaminen ja verifiointi:

Saavutettavuusominaisuuksia voi testata useilla työkaluilla, kuten Lighthousella tai React Native Accessibility Inspectorilla. Manuaalinen testaus erilaisilla apuvälineillä kuten ruudunlukijoilla (esim. VoiceOver ja TalkBack), auttaa varmistamaan, että sovellus toimii kaikille käyttäjille. Myös käyttäjätestaus erityisryhmiin kuuluvien henkilöiden kanssa voi tarjota arvokasta palautetta sovelluksen saavutettavuuden parantamiseksi.

Visuaalisten teemojen ja responsiivisuuden ohella saavutettavuus tarkoittaa siis sitä, että sovellus on suunniteltu kaikille käyttäjäryhmille. Näiden ominaisuuksien toteuttaminen ei pelkästään paranna käyttäjäkokemusta, vaan varmistaa, että sovellus on saavutettava ja joustavasti mukautuva erilaisiin laitteisiin ja tarpeisiin. (25)

3.3 Tietokantaratkaisu

AsyncStorage –moduuli ja Firebase –pilvipalvelu soveltuvat parhaiten tämän oppimisalustan tiedonhallintaan keskeisten vaatimusten perusteella, kuten käyttäjäasetusten ja progressiivisen datan tehokas verkkoyhteydetön tallentaminen, skaalautuvuus ja reaaliaikainen paikallisdatan synkronointi pilvipalvelu tietokantaan.

AsyncStoragen avulla voidaan tallentaa dataa paikallisesti käyttäjän omalle laitteelle. Tämä on tärkeää, jotta käyttäjäasetukset, kuten tumma/vaalea tila, säilyvät ilman internetyhteyttä. Koska kyseessä on ala-aste ikäisille lapsille suunnattu

oppimisalusta, käyttökokemuksen tulee olla saumaton eli käyttäjän on voitava jatkaa oppimista ilman ylimääräisiä uudelleenlatauksia ja kirjautumisia.

Jos data tallennettaisiin vain paikallisesti, eri laitetta käytettäessä käyttäjän dataa ei saisi siirrettyä toiseen laitteeseen. Firebase tarjoaa pilvipohjaisen tietokanta-ratkaisun, joka tukee reaaliaikaista tiedonkäsittelyä. Sen avulla voidaan tallentaa tietokilpailujen tulokset, käyttäjän edistyminen ja pistetilastot tietoturvallisesti, mikä mahdollistaa oppimisen jatkamisen eri laitteilla.

Tärkeimmät tallennusvaatimukset ovat:

- **Pysyvät käyttäjäasetukset** (esim. Tumman/Vaalean tilan valinta) tallennetaan paikallisesti.
- **Pilvipohjainen järjestelmä** synkronoi tietokilpailujen edistymisen, pisteet ja käyttäjän toiminnot Firebaseen.
- **Reaaliaikainen tiedonsiirto** mahdollistaa joustavan palautteen ja ajantasaiset pistetaulukot.
- **Laajennettavuus ja vahva tietosuoja** takaavat, että lasten käyttäjätiedot pysyvät suojattuina.
- **Firebasen ja AsyncStoragen yhdistäminen** tarjoaa tasapainon verkkoyhteydettömän käytön ja pilvipohjaisen toiminnallisuuden välillä, parantaen käyttökokemusta ja tukien sujuvaa oppimista.

Artikkelissa ”A guide to React Native’s AsyncStorage” (6), Käydään läpi AsyncStorage-moduulin toimintaa React Native -sovelluksissa. AsyncStorage on nimensä mukaan asynkroninen, salaamaton ja pysyvä avainarvo tallennusjärjestelmä, jolla mahdollistetaan tietojen säilyminen offline-tilassa. AsyncStorage ei salaa tietoja, joten sitä ei kannata käyttää arkaluontoisten tietojen tallentamiseen. Artikkelissa käydään läpi moduulin asentaminen ja sen keskeiset metodit, eli setItem(), getItem(), mergeItem(), removeItem() ja clear(). Näillä metodeilla sovelluksen kehittäjät voivat tallentaa, hakea, yhdistää ja poistaa tietoja tallennustilasta. Yhtenä esimerkkinä mainitaan teeman eli tumman/vaalean tilan tallentaminen AsyncStorageen, tämä parantaa käytettävyyttä sekä saavutettavuutta koska

näin käyttäjän asetukset ja data saadaan säilymään sovelluksen uudelleenkäynnistämisen yhteydessä. Juuri tämänlaista dataa mm. asetuksia, käyttäjän etunimi tai lempinimi, progressiivinen data (Kuinka monta tietokilpailua on suoritettu, Oikein/Väärin vastaukset yms.) voidaan tallentaa hyödyntäen tätä moduulia koska ne eivät ole mitenkään arkaluontoista (7). Kuvassa 3 nähdään koodiesimerkki, miten tietoa voidaan tallentaa paikallisesti.

```
let UID123_object = {
  name: 'Chris',
  age: 30,
  traits: {hair: 'brown', eyes: 'brown'},
};
// You only need to define what will be added or updated
let UID123_delta = {
  age: 31,
  traits: {eyes: 'blue', shoe_size: 10},
};

AsyncStorage.setItem(
  'UID123',
  JSON.stringify(UID123_object),
  () => {
    AsyncStorage.mergeItem(
      'UID123',
      JSON.stringify(UID123_delta),
      () => {
        AsyncStorage.getItem('UID123', (err, result) => {
          console.log(result);
        });
      },
    );
  },
);

// Console log result:
// => {'name':'Chris','age':31,'traits':
//    {'shoe_size':10,'hair':'brown','eyes':'blue'}}
```

KUVA 3. Datan tallennus paikallisesti ASyncStorageen

Kuvan 3 koodiesimerkki selkeyttää, kuinka kyseistä moduulia käytetään tietojen tallentamiseen, päivittämiseen sekä hakemiseen. Vaiheet ovat seuraavanlaiset:

1. Lähtöobjekti luonti: JavaScript-objekti nimeltä UID123_objext luodaan, ja se sisätää käyttäjän Chris perustiedot.
2. Delta-objektin luonti: Delta-objekti, UID123_delta, määritetään kuvaamaan niitä päivityksiä, jotka lisätään UID123_objectiin. Tässä tapauksessa iäksi muutetaan 30->31, silmien värin arvoksi asetetaan sininen, ja mukaan lisätään uusi ominaisuus kengänkoko: 10.

3. Lähtöobjektin tallennus: Metodia setItem käytetään tallentamaan UID123_object, joka on muunnettu merkkijonoksi, avaimen 'UID123' alle.
4. Tietojen yhdistäminen: Tallentamisen jälkeen metodi mergeItem yhdistää UID123_ delta-objektin jo tallennettuun UID123_objektiin.

Yhdistämisprosessi toimii näin:

Perusarvot kuten ikä päivitetään delta-objektin mukaisiksi.

Sisäkkäisissä olioissa, kuten piirteet, olemassa olevia ominaisuuksia päivitetään (silmienväri -> sininen), uusia ominaisuuksia lisätään (kengänkoko: 10), ja säilyvät ominaisuudet, kuten hiustenväri, pysyy samana.

- Päivitettyjen tietojen haku ja tulostus: Lopuksi metodi getItem hakee tallennetut ja päivitetyt tiedot, jonka jälkeen se muuntaa ne takaisin objektimuotoon.

Suunnitelman toteutus:

Kuvasta 4 näkee, kuinka otimme käyttöön ASyncStorage –moduulin, jonka avulla tallennamme käyttäjän progressiivisen datan paikallisesti ja ohjelmoin StatsManager –komponentin, joka varastoi kolmea eri muuttujaa (Yhteenlaskettuna oikeat vastaukset, Yhteenlaskettuna väärät vastaukset, Suoritettujen tietokilpailujen summa):

```
import AsyncStorage from '@react-native-async-storage/async-storage';

const STORAGE_KEYS = {
  totalCorrectAnswers: 'TotalCorrectAnswers',
  totalWrongAnswers: 'TotalWrongAnswers',
  completedQuizzes: 'CompletedQuizzes',
};

const StatsManager = {
  getStats: async () => {
    const correct = await AsyncStorage.getItem(STORAGE_KEYS.totalCorrectAnswers);
    const wrong = await AsyncStorage.getItem(STORAGE_KEYS.totalWrongAnswers);
    const completed = await AsyncStorage.getItem(STORAGE_KEYS.completedQuizzes);

    return {
      totalCorrectAnswers: Number(correct) || 0,
      totalWrongAnswers: Number(wrong) || 0,
      completedQuizzes: Number(completed) || 0,
    };
  },

  updateStats: async (isCorrectAnswer: boolean, isQuizComplete: boolean) => {
    if (isCorrectAnswer) {
      const correct = (await AsyncStorage.getItem(STORAGE_KEYS.totalCorrectAnswers)) || '0';
      await AsyncStorage.setItem(STORAGE_KEYS.totalCorrectAnswers, (Number(correct) + 1).toString());
    } else {
      const wrong = (await AsyncStorage.getItem(STORAGE_KEYS.totalWrongAnswers)) || '0';
      await AsyncStorage.setItem(STORAGE_KEYS.totalWrongAnswers, (Number(wrong) + 1).toString());
    }

    if (isQuizComplete) {
      const completed = (await AsyncStorage.getItem(STORAGE_KEYS.completedQuizzes)) || '0';
      await AsyncStorage.setItem(STORAGE_KEYS.completedQuizzes, (Number(completed) + 1).toString());
    }
  },
};

export default StatsManager;
```

KUVA 4. ASyncStorage ja StatsManager

Tietokantasuunnitelman toteuttaminen ja lopullinen versio näkyy tarkemmin Komponentit –osiossa kohdassa Profiili

3.4 Kilpailullisuus

Digitaalisissa oppimisympäristöissä pelillisyyden elementit ovat nousseet merkittäväksi työkaluksi käyttäjien sitoutumisen ja motivaation lisäämisessä. Yksi keskeisimmistä pelillisyyden osa-alueista on pistenäyttö, joka toimii visuaalisena mittarina käyttäjien edistymiselle ja saavutuksille. Tämä teksti käsittelee pistenäytön tarkoitusta, sen psykologista vaikutusta lapsiin sekä sen roolia oppimisalustojen jatkuvassa käytössä.

Pistenäyttö on suunniteltu tarjoamaan käyttäjille, erityisesti lapsille, selkeän ja mitattavan tavan seurata omaa suorituskykyään ja edistymistään oppimisalustalla. Näyttämällä statistiikkaa, kuten oikeita vastauksia, vääriä vastauksia, suoritettuja tehtäviä ja tarkkuusprosentteja, pistenäyttö tarjoaa käyttäjille välittömän

palautteen suorituksestaan. Tämä palaute on ratkaisevan tärkeää saavutuksen tunteen luomisessa ja kannustaa käyttäjiä kehittämään itseään.

Pistenäyttö tuo myös mukanaan terveellisen kilpailun elementin esittelemällä parhaiten menestyneitä käyttäjiä. Kilpailullinen puoli motivoi käyttäjiä pyrkimään parempiin tuloksiin, mikä lisää heidän sitoutumistaan alustaan. Lisäksi pistenäyttö toimii edistymisen seuraajana, mahdollistaen käyttäjien visualisoida omaa kehitystään ajan myötä. Tämä on keskeistä pitkäaikaisen kiinnostuksen ja sitoutumisen ylläpitämiseksi. (26)

Psykologinen vaikutus lapsiin

Lapset ovat luontaisesti taipuvaisia aktiviteetteihin, jotka tarjoavat välittömiä palkintoja ja tunnustusta. Pistenäyttö hyödyntää tätä taipumusta tarjoamalla välittömän palautteen ja positiivista vahvistusta. Kun lapset näkevät nimensä tai saavutuksensa pistenäytöllä, he kokevat ylpeyttä ja saavutuksen tunnetta, mikä vahvistaa heidän haluaan jatkaa oppimista.

Lisäksi pistenäyttö hyödyntää tavoitteiden asettamisen psykologiaa. Asettamalla selkeät, mitattavat tavoitteet (esim. korkeampi tarkkuusprosentti tai useampien tehtävien suorittaminen) lapset pysyvät motivoituneina ja sitoutuneina. Pistenäyttö edistää myös sisäistä motivaatiota kannustamalla lapsia kehittämään taitojaan ja tietämystään henkilökohtaisen kasvun vuoksi, ei pelkästään ulkoisten palkintojen takia. (14)

Jatkuvan käytön edistäminen

Pistenäytöllä on keskeinen rooli lapsien kannustamisessa jatkamaan oppimisalustojen käyttöä. Tarjoamalla visuaalisen edistymisen kuvan pistenäyttö auttaa lapsia näkemään konkreettiset tulokset omista ponnisteluistaan. Tämä edistymisen tunne on voimakas motivaattori, sillä se vahvistaa ajatusta siitä, että heidän työnsä tuottaa tulosta.

Lisäksi se luo yhteisöllisyyden tunnetta mahdollistamalla lasten vertailla suorituskykyään vertaistensa kanssa. Tämä sosiaalinen puoli oppimisessa voi olla erittäin motivoiva, sillä lapset usein inspiroituvat ystäviensä ja luokkatoveriensa saavutuksista. Tarkkuusprosenttien lisääminen tuo mukanaan lisähaasteen,

kannustaen lapsia paitsi suorittamaan tehtäviä, myös tavoittelemaan mestaruutta ja tarkkuutta. (26)

3.5 Verkkosivun kehittäminen

Ohjelmistoprojektien menestyksekäs hallinta edellyttää selkeää ja saavutettavaa dokumentaatiota, joka helpottaa tiedon jakamista ja pitkäaikaista ylläpitoa. Vaikka GitHubin README -tiedostot tarjoavat perustason dokumentoinnin, ne eivät ole niin käyttäjäystävällisiä eivätkä tarjoa visuaalisia ja interaktiivisia ominaisuuksia.

Verkkosivun tarkoituksena on tarjota parempi vaihtoehto, jossa projektin tärkeät tiedot esitetään helposti lähestyttävässä ja selkeässä muodossa. Se mahdollistaa tietojen tarkastelun ilman tarvetta selata arkistoja tai kooditiedostoja.

Sivusto rakennetaan tukemaan seuraavia päämääriä:

- Helppolukuinen ja selkeä sisältö
- Skaalautuva muotoilu
- Vuorovaikutteiset ominaisuudet, kuten osiot, jotka laajenevat hover -ominaisuudella
- Sujuvampi navigointi verrattuna perinteisiin tekstipohjaisiin tiedostoihin
- Mahdollisuus hyödyntää kuvia, videoita ja latauslinkkejä

Sivusto toteutetaan HTML:n ja CSS:n avulla. Kehityksessä panostetaan yksinkertaisuuteen ja saavutettavuuteen.

Sivun eri osiot järjestellään siten, että käyttäjä löytää tarvitsemansa tiedot nopeasti. Jokainen osa sijoitetaan omalle alueelleen <section> -elementtejä käyttäen, mikä tekee sisällöstä loogisesti etenevän ja helpottaa navigointia.

CSS-tyylit suunnitellaan takaamaan miellyttävä käyttökokemus kaikilla päätelaitteilla. Pääpiirteitä ovat:

- Taustakuvan käyttö visuaalisen houkuttelevuuden lisäämiseksi ilman, että se heikentää luettavuutta.
- Dynaamiset osiot jotka avautuvat käyttäjän osoittaessa niitä, tehden sisällöstä joustavampaa ja vähemmän kuormittavaa.
- Responsiivinen muotoilu, joka varmistaa toimivuuden eri näyttökokoisilla laitteilla.
- Sivuston kehityksessä kiinnitetään erityistä huomiota saavutettavuuteen. Käyttökokemus optimoidaan värikontrastien, selkeiden fonttien ja helppokäyttöisten vuorovaikutuselementtien avulla

4 PEDAGOGISET NÄKÖKULMAT

Peruskoulun opetussuunnitelma korostaa oppimisympäristöjen turvallisuutta ja innostavuutta, huomioiden eri oppiaineiden tarpeet. Oppimista tapahtuu niin luokkahuoneessa kuin luonnossa tai vaikka kirjastoissa. Nykyään myös digitaaliset ratkaisut ovat keskeisessä roolissa, ja monipuoliset työtavat auttavat oppilaita kehittämään erilaisia taitoja. Teknologian merkitys kasvaa jatkuvasti, ja oppilaat osallistuvat aktiivisesti oppimisympäristöjen suunnitteluun ja valintoihin. Erilaiset digitaaliset ratkaisut ovat myös oiva apu oppilaan tuen saamiseen niin koulussa kuin sen ulkopuolellakin.

Tavoitteenamme oli suunnata oppimisalusta 7–10-vuotiaille lapsille, minkä vuoksi hankimme pedagogista tietoa siitä, millaiset kysymykset ja vaikeustasot sopivat heille parhaiten. Jos sovellus olisi liian vaikea tai helppo, sen käyttö menettäisi nopeasti kiinnostavuutensa, eivätkä lapset kokisi onnistumisen tai oppimisen iloa – mikä on juuri tämän sovelluksen perimmäinen tarkoitus.

Kognitiivisen, sosiaalisen ja emotionaalisen kehityksen tukeminen

Lapset innostuvat ja motivoituvat opiskelemaan helpommin, kun oppimiseen sisältyy esimerkiksi kilpailua. Monille lapsille on tärkeää olla parempi kuin muut ja näyttää se esimerkiksi pelien avulla. Peli tukee lapsen itsenäistä oppimista ja sitä lapsi voi pelata niin monesti kuin haluaa ja näin oppia lisää. Sovelluksen kannustava ympäristö kannustaa lasta tekemään tehtäviä niin kauan, että ne menevät oikein. Sisältöjen suunnittelussa painotetaan lasten kognitiivisen, sosiaalisen ja emotionaalisen kehityksen tukemista. Sovellus edistää lasten kognitiivista kehitystä tarjoamalla monipuolisia tehtäviä, jotka vahvistavat muistia, loogista ajattelua ja ongelmanratkaisutaitoja. Sosiaalista kehitystä tuetaan vertailemalla tuloksia ja kannustamalla pelaajia saavuttamaan omia ja yhteisiä tavoitteita, mikä luo positiivista vuorovaikutusta. Emotionaalinen kehitys huomioidaan palkitsemalla edistymisestä, mikä lisää motivaatiota ja itseluottamusta. Haasteiden asteittainen vaikeutuminen varmistaa, että oppiminen pysyy motivoivana ilman turhautumista. (8)

Pelilliset elementit

Sovelluksessa hyödynnetään pelillisiä elementtejä ja etenemiseen perustuvia rakenteita, jotka tekevät oppimisesta innostavaa ja merkityksellistä. Sovellus hyödyntää useita pelillistämisen keinoja, jotka tekevät oppimisesta motivoivaa ja interaktiivista:

- Tasojärjestelmä: Pelaaja etenee tasolta toiselle vaikeustason noustessa.
- Pisteytys ja palkinnot: Oikeista vastauksista ja onnistumisista saa pisteitä sekä virtuaalisia palkintoja, jotka kannustavat jatkamaan pelaamista.
- Saavutukset: Pelaajat voivat ansaita erilaisia saavutuksia, mikä lisää pitkäjänteistä motivaatiota.
- Ranking-lista: Mahdollisuus verrata omaa edistymistä muihin tuo kevyttä kilpailullisuutta.
- Visuaaliset ja äänitehosteet: Animaatiot ja äänipalautteet tekevät oppimisesta miellyttävää ja innostavaa.

Erilaiset oppimispolut

Sovellus tarjoaa erilaisia oppimispolkuja, joiden avulla käyttäjä voi edetä omassa tahdissaan ja kehittyä omien vahvuuksiensa mukaisesti:

- Tasolta toiselle eteneminen: Jokainen oppimiskategoria sisältää viisi vaikeustasoa, jotka monimutkaistuvat pelaajan edetessä.
- Vapaa harjoittelu: Pelaaja voi valita haluamansa aiheen ja harjoitella sitä omaan tahtiin ilman aikarajoja.

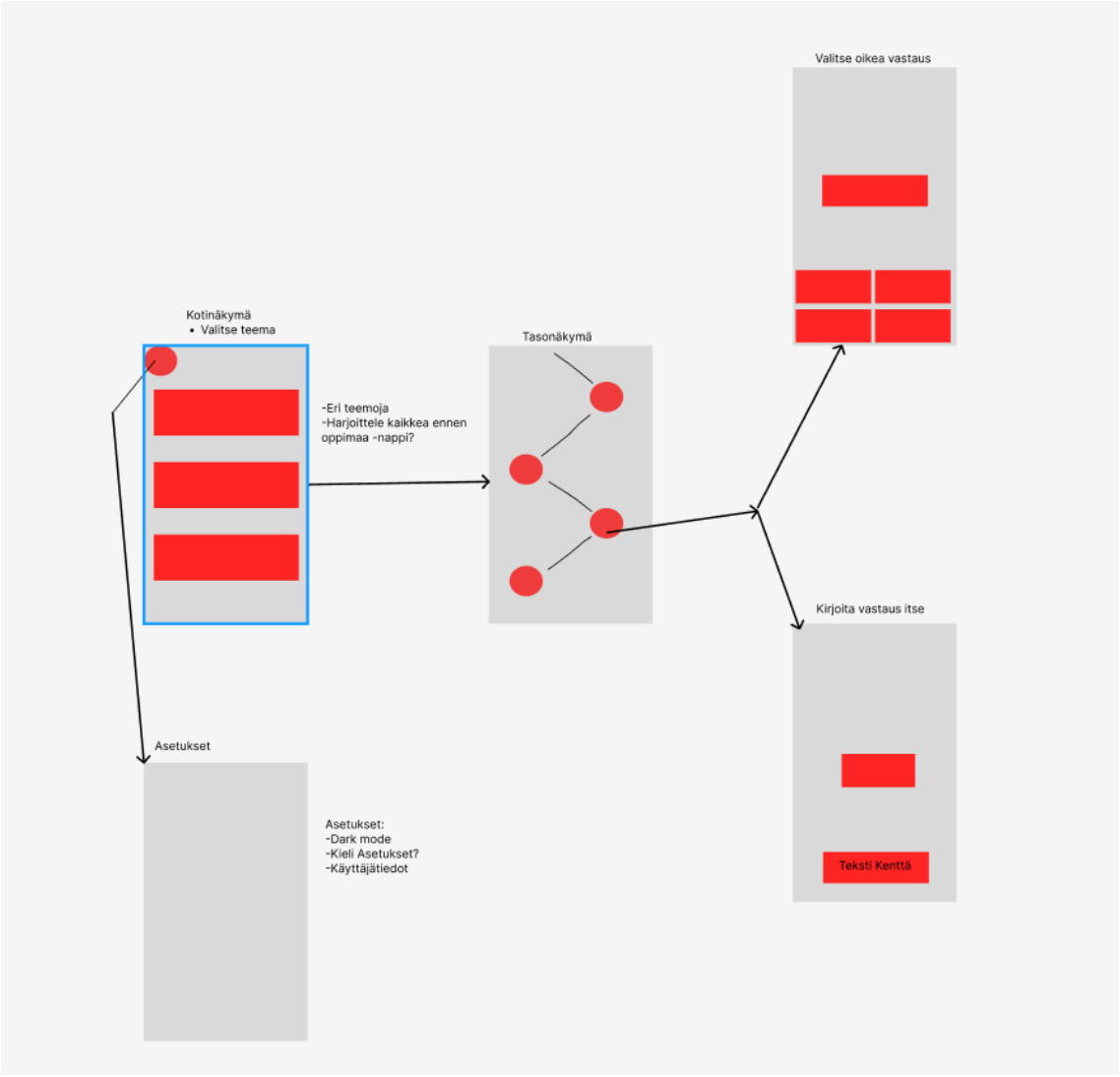
Käyttöliittymä sekä tehtävien vaativuus on mukautettu lapsen kehitystasoon, mikä auttaa luomaan palkitsevia oppimiskokemuksia ja vahvistamaan onnistumisen tunnetta. Lisäksi sovellus kannustaa lasta itsenäiseen ongelmanratkaisuun

ja luovuuteen tarjoamalla joustavia oppimispolkuja. Visuaalisesti miellyttävä ja selkeä käyttöliittymä tukee keskittymistä, ja tehtävät on suunniteltu tarjoamaan sopivasti haasteita, jotta lapsi pysyy motivoituneena. Oppimista tuetaan myös palautteella, joka rohkaisee ja ohjaa kohti seuraavia tavoitteita, samalla vahvistaen itsetuntoa ja oppimisen iloa. Kuvassa 5 nähdään kolme tukipilaria, joihin tällaisen sovelluksen olisi hyvä nojata.

KUVA 5. Pedagogiikan tukipilarit (9)

5 SOVELLUKSEN TOTEUTUS

Ottamalla teoria osuuden tutkimuksen ”Engaging children with educational content via Gamification” (15) tulokset huomioon lähdettiin luomaan sovelluksen rakennetta. Käyttämällä Figma -prototypointityökalua, muotoiltiin sovelluksen ensimmäinen rakenne, joka nähdään kuvassa 6.

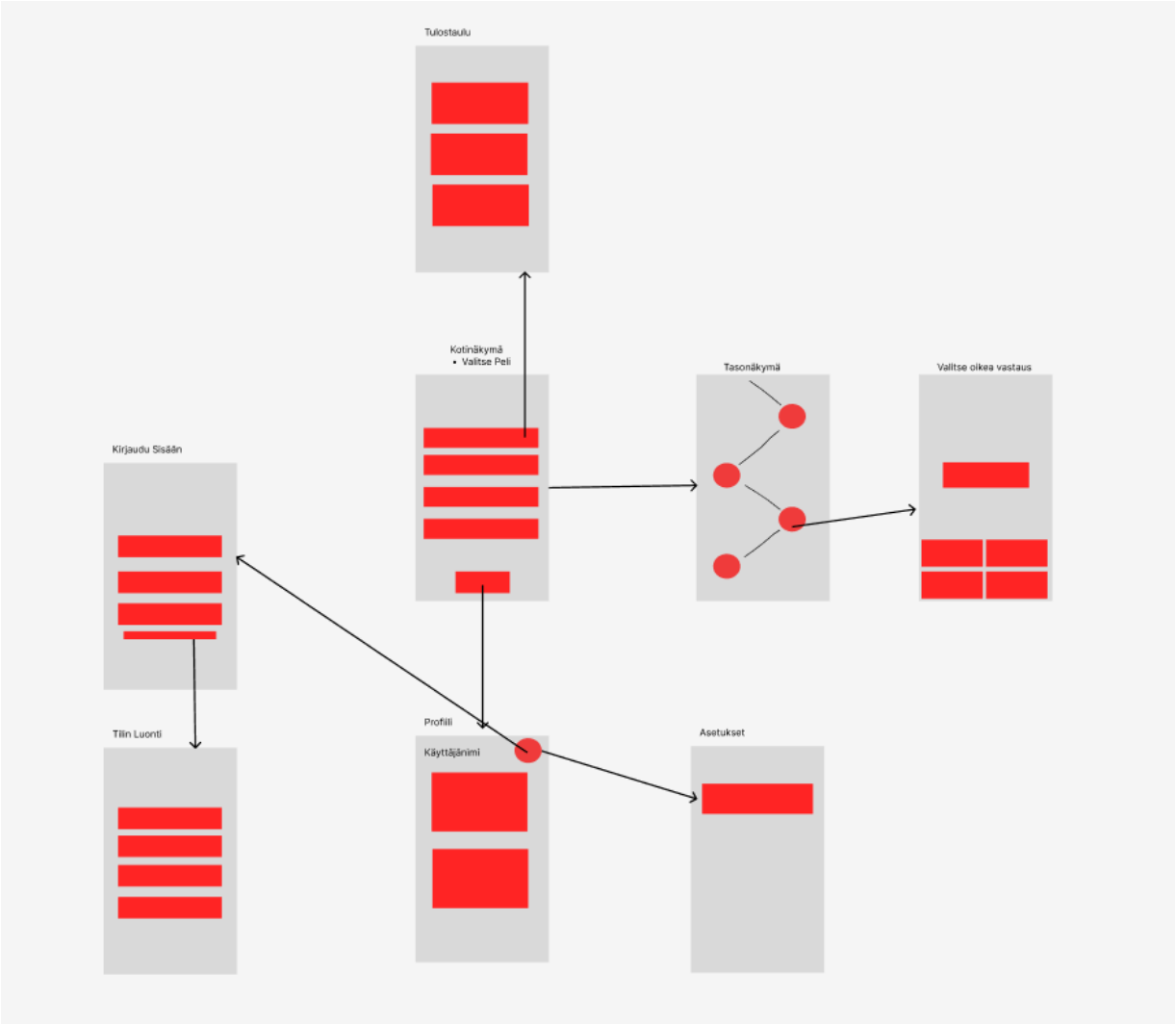


KUVA 6. Sovelluksen rakenne suunnitelma

Tämän lisäksi tutkittuaan samankaltaisia sovelluksia, kuten Duolingoa, saatiin seuraavat näkymät suunniteltua sovellukseen. Sovelluksen ensimmäisessä versiossa siis olisi suunnitelman mukaisesti kuusi eri näkymää: ”Tervetuloa”, ”Koti”, ”Asetukset” ja ”Taso” -näkymät. Tämän lisäksi ”Taso” -näkymästä, kun valitsee

tason, johdetaan käyttäjä satunnaisesti joko ”Valitse oikea vastaus” -näkymään tai sitten ”Kirjoita vastaus itse” -näkymään. ”Valitse oikea vastaus” -näkymässä käyttäjällä esitetään pulma ja hänen pitää valita oikea vastaus neljästä eri vaihtoehdosta. ”Kirjoita vastaus itse” -näkymässä käyttäjälle esitetään pulma ja hänen pitää itse pähkäillä mikä pulman ratkaisu on.

Kuvassa 6 nähty rakenne suunniteltiin ihan opinnäytetyön alussa ja samalla, kun teoriaa luettiin, niin sovelluksen rakenne muuttui myös sen mukaan. Kuvassa 7 nähdään sovelluksen lopullinen rakenne. ”Kirjoita vastaus itse” -näkymä poistettiin sovelluksesta, mutta neljä uutta näkymää lisättiin; ”Tulostaulu”- ”Profiili”-, ”Kirjaudu sisään”- ja ”Tilin luonti” -näkymät.



KUVA 7. Sovelluksen lopullinen rakenne

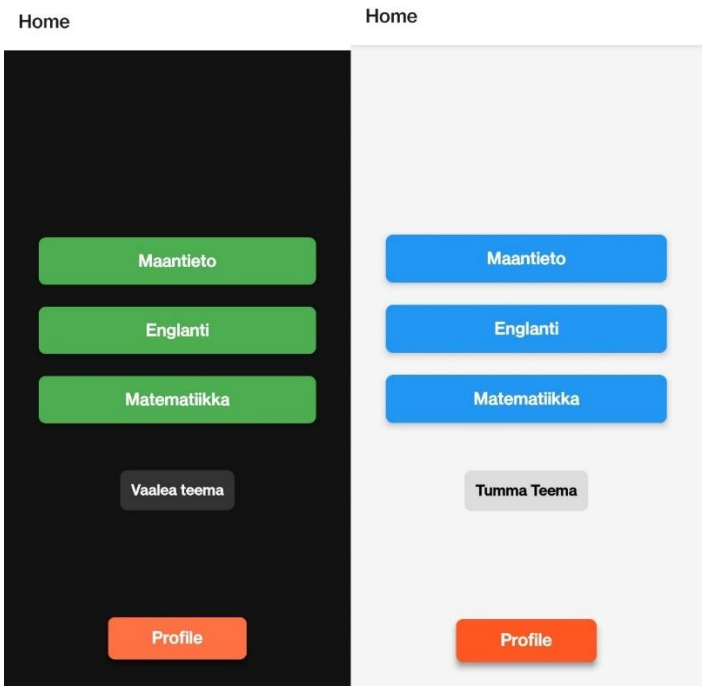
”Kirjoita vastaus itse” -näkymä poistettiin sovelluksesta siksi, koska pelien tasot eivät ole pitkiä eikä tasoja ole, kun vain viisi per peli. Jos jokaisessa pelissä olisi enemmän tasoja ja vanhoja kysymyksiä tulisi harjoitteena uusiksi käyttäjälle ”Kirjoita vastaus itse” -näkymä olisi hyvä oppimisen vuoksi, mutta kun sovelluksessa ei ole minkäänlaista mekaniikka, jossa suoritettuja kysymyksiä esiteltäisiin uusikis käyttäjälle eri tasoissa, tämä näkymä tuntui turhalta lisätä. ”Tulostaulu”- näkymä ”Profiili” -näkymä lisättiin, koska sovelluksessa on mekaniikka, joka laskee, kuinka monta vastausta käyttäjä on saanut oikein ja kuinka monta käyttäjä on saanut väärin, niin täältä käyttäjä voi nähdä tämän tilaston. ”Kirjaudu sisään”- ja ”Tilin luonti” -näkymät lisättiin, jos käyttäjä haluaa tallentaa tilitiedot Firebaseen, jotta hänen tilastotietonsa ovat saatavilla myös muilla laitteilla.

5.1 Käyttöliittymä -komponentti

Käyttöliittymän ensimmäinen versio nähdään kuvassa 8.

KUVA 8. Päänäkymä versio 1

Sovelluksen ensimmäinen versio oli tarkoitettu vain testaamiseen ja käyttöliittymän suunnittelun hahmottamiseen. Toisessa versiossa käyttöliittymää parannettiin yhtenäisemmällä tyylillä, kuten vakioimalla painikkeiden koot ja selventämällä tekstejä. Käyttöliittymän toinen versio on esitetty kuvassa 9.

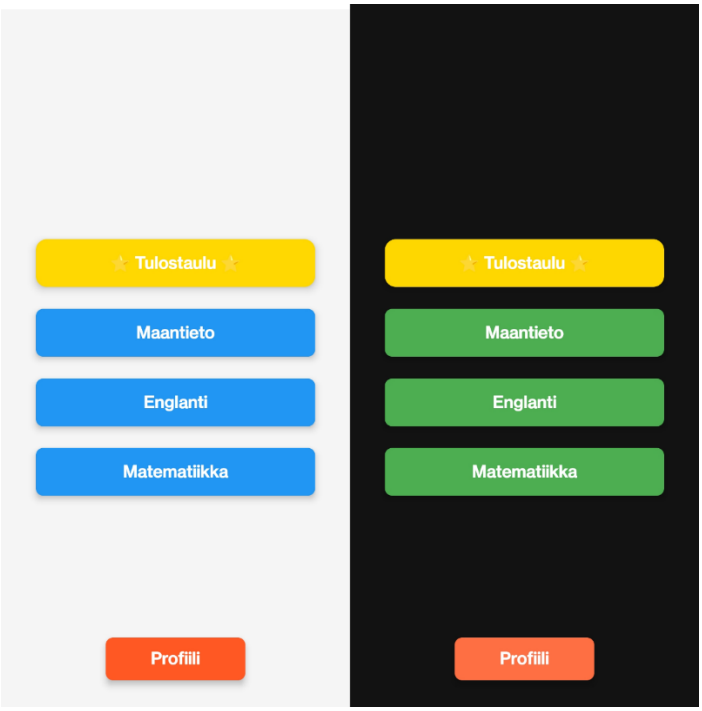


KUVA 9. Käyttöliittymän toinen versio

Kuten kuvassa näkyy, käyttöliittymän toisessa versiossa on implementoitu ”Dark mode”, jonka avulla käyttäjä voi halutessaan muuttaa sovelluksen ulkoista teemaa joko vaaleaksi tai tummaksi. Sen lisäksi toisessa versiossa sovelluksen ulkonäköä on hienostettu hieman tekemällä napeista samankokoiset ja lisäämällä värejä.

Yllä oleva ”Home”-palkki on tullut sivujen navigointiin muutosten myötä. Versiossa 1 navigointi toimi suoralla linkillä tiedostoon, mutta versiossa 2 siirryttiin käyttämään React Nativin ”StackNavigator”-ominaisuutta. ”StackNavigator” toimii siten, että uudet näkymät ”kerrostetaan” edellisen päälle.

Käyttöliittymän viimeisimmälle versiolle, joka näkyy kuvassa 10, tehtiin vain minimaalisia muutoksia. ”Dark mode” -näppäin siirrettiin ”Asetukset”-näkymään, ja ”Tulostaulut” -näppäin lisättiin, joka vie ”Tulostaulu” -näkymään.



KUVA 10. Käyttöliittymän lopullinen versio

5.2 Profiili -komponentti

Profiili on näkymä, joka pitää sisällään käyttäjän tallennetut progressiiviset tiedot. Kuvassa 11 nähdään oikeat vastaukset, väärät vastaukset, suoritetut tietokilpailut.

KUVA 11. Profiili –komponentti Versio 1

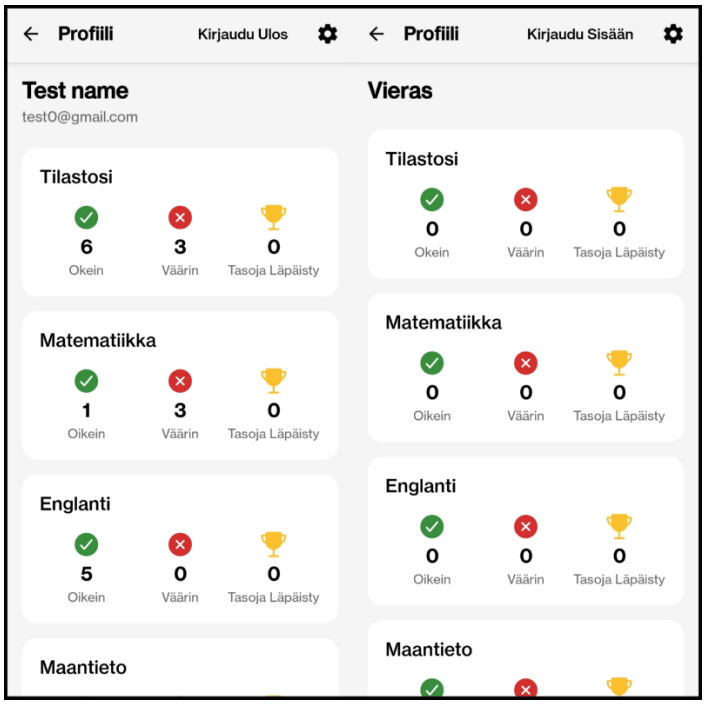
Profiilin tiedot tulevat näkyviin StatsManager –komponentin kautta ja kuvassa 12 näkyy sen ohjelmallinen toteutus:

```
import AsyncStorage from '@react-native-async-storage/async-storage';

const STORAGE_KEYS = {
  totalCorrectAnswers: 'TotalCorrectAnswers',
  totalWrongAnswers: 'TotalWrongAnswers',
  completedQuizzes: 'CompletedQuizzes',
};

const StatsManager = {
  getStats: async () => {
    const correct = await AsyncStorage.getItem(STORAGE_KEYS.totalCorrectAnswers);
    const wrong = await AsyncStorage.getItem(STORAGE_KEYS.totalWrongAnswers);
    const completed = await AsyncStorage.getItem(STORAGE_KEYS.completedQuizzes);

    return {
      totalCorrectAnswers: Number(correct) || 0,
      totalWrongAnswers: Number(wrong) || 0,
      completedQuizzes: Number(completed) || 0,
    };
  },

  updateStats: async (isCorrectAnswer: boolean, isQuizComplete: boolean) => {
    if (isCorrectAnswer) {
      const correct = (await AsyncStorage.getItem(STORAGE_KEYS.totalCorrectAnswers)) || '0';
      await AsyncStorage.setItem(STORAGE_KEYS.totalCorrectAnswers, (Number(correct) + 1).toString());
    } else {
      const wrong = (await AsyncStorage.getItem(STORAGE_KEYS.totalWrongAnswers)) || '0';
      await AsyncStorage.setItem(STORAGE_KEYS.totalWrongAnswers, (Number(wrong) + 1).toString());
    }

    if (isQuizComplete) {
      const completed = (await AsyncStorage.getItem(STORAGE_KEYS.completedQuizzes)) || '0';
      await AsyncStorage.setItem(STORAGE_KEYS.completedQuizzes, (Number(completed) + 1).toString());
    }
  },
};

export default StatsManager;
```

KUVA 12. ASyncStorage StatsManager

Kuten kuvasta näkyy, StatsManager on toteutettu hyödyntämällä ASyncStorage –moduulia, jonka teoreettista puolta käsiteltiin lähteissä (6 ja 7). Enemmän asiaa StatsManager –komponentista sen omassa kappaleessa.

”Profiili”-näkymän ensimmäinen versio oli pelkistetty versio näkymän toiminnallisuudesta. Käyttäjänluonnin lisäämisen myötä oli tarpeellista laajentaa näkymää ja sen ominaisuuksia. Kuvassa 13 nähdään profiilin viimeinen versio, jossa edellä mainittu tiedonhaku StatsManagerin kautta on laajennettu.



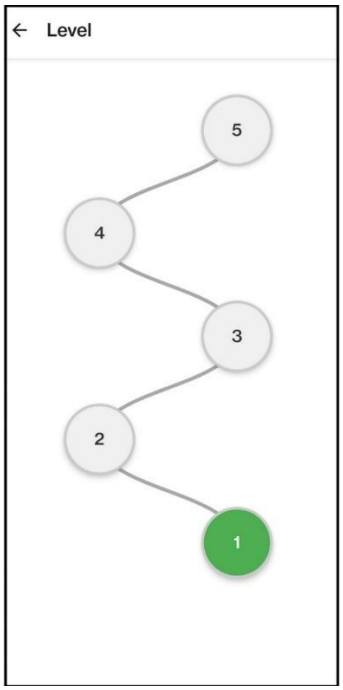
KUVA 13. Profiilin lopullinen näkymä

Kuten kuvassa näkyy, jos käyttäjä on kirjautunut sisään, hänen käyttäjänimensä ja sähköpostinsa näytetään ruudun vasemmassa yläkulmassa. Jos käyttäjä ei ole kirjautunut sisään, nämä tiedot korvataan 'Vieras'-nimikkeellä.

Ensimmäisessä versiossa käyttäjän tiedot tallennettiin vain paikallisesti, eikä pelien yksittäistä tilastotallennusta ollut lainkaan. Viimeisessä versiossa käyttäjän tiedot tallennetaan yhä lokaalisti, mutta jos käyttäjä on kirjautunut sisään, tiedot tallennetaan ja haetaan myös Firebasesta. Tilastot esitetään ruudulla 'kortteina', joista ylimmässä kortissa näkyvät kaikkien pelien yhteiset tilastot ja sen alla pelikohtaiset tilastot erikseen.

5.3 Tasonäkymä -komponentti

Tasonäkymässä käyttäjä näkee valitsemansa pelin kentät. Kuvassa 14 nähdään sovelluksen tasonäkymä.



KUVA 14. Tasonäkymä ensimmäinen versio

Tasonäkymän käyttöliittymä on toteutettu käyttäen hyväksi React nativen eri ominaisuuksia, kuten ”Animated” -kirjastoa, jota käytettiin tekemään napeista interaktiivisemmat ja miellyttävämmät käyttäjälle. ”Animated” -kirjaston käyttö voidaan nähdä kuvassa 15.

```
// on pressing the node
const handlePressIn = () => {
  Animated.spring(scaleAnim, {
    toValue: 0.8,
    friction: 1,
    useNativeDriver: false,
  }).start();
};

// on releasing the node
const handlePressOut = () => {
  Animated.spring(scaleAnim, {
    toValue: 1,
    friction: 3,
    useNativeDriver: false,
  }).start();
};
```

KUVA 15. ”Animated” -kirjaston käyttö

- ”toValue” - määrittää, kuinka pieneksi painike kutistuu, kun käyttäjä painaa sitä

- ”friction” - määrittää napin jäykkyyden tai kimmoisuuden, eli mitä alempi arvo on syötetty sitä kimmoisampi nappi on ja mitä isompi arvo niin sitä jäykempi nappi on.

Kuten kuvassa 15 nähdään, tasonäkymässä on painikkeiden lisäksi Bézier-käyrä. Suunnitelleessa sovellusta Tasonäkymän oli tarkoitus näyttää ”karttamaiselta”, eli eri kentät olisi yhdistetty polulla, jossa käyttäjä kulkee. Tämän saavuttamiseksi Bézier-käyrä oli sopiva ratkaisu. Bézier-käyrä on toteutettu käyttäen React nativen SVG-kansiota ja sen toiminta näkyy kuvissa 16 ja 17.

```
// get node positions
const getNodePosition = (index: number) => {
  const y = (LEVELS - index - 1) * SPACING + PADDING;
  const x = width / 2 + ((index % 2 === 0) ? OFFSET : -OFFSET);
  return { x, y };
};
```

KUVA 16. "getNodePositions" -funktio

- y: Napin pystysuuntainen sijainti määräytyy sen indeksin perusteella tasojen listassa. Napit on sijoitettu tasaisesti y-akselilla käyttäen SPACING-vakiota.
- x: Napin vaakasuuntainen sijainti vaihtuu ruudun vasemman ja oikean puolen välillä sen perusteella, onko napin indeksi parillinen vai pariton.

```
// calculate Bézier curve between points
const getControlPoints = (prevX: number, prevY: number, currX: number, currY: number) => {
  const midY = (prevY + currY) / 2;
  return {
    controlPoint1: { x: prevX, y: midY },
    controlPoint2: { x: currX, y: midY },
  };
};
```

KUVA 17. " getControlPoints" -funktio

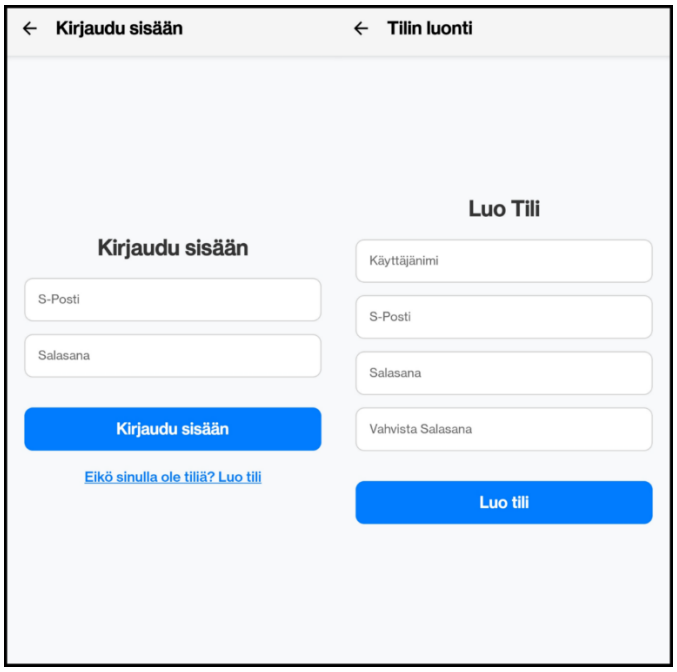
- midY: Lasketaan edellisen napin y-koordinaatin (prevY) ja nykyisen napin y-koordinaatin (currY) välinen keskipiste, mikä varmistaa, että käyrä on sujuva ja symmetrinen.
- controlPoint1 ja controlPoint2: Nämä ovat Bézier-käyrän pisteet, jotka sijoitetaan samalla y-koordinaatilla (midY) mutta edellisen ja nykyisen napin x-koordinaattien kohdalle.

Tasonäkymään on lisätty taustakuva sekä ohjelmallinen logiikka. Kuvassa 18 nähdään lopullinen näkymä, jossa ilmenee, että jos taso on suoritettu, pallon väri näkyy sinisenä, uusin avattu taso näkyy vihreänä ja lukitut tasot näkyvät valkoisena.

KUVA 18. Tasonäkymän lopullinen versio

5.4 Tilinluonti -komponentti

Sovelluksessa voi halutessaan luoda itselleen käyttäjän tallentaakseen pelaaja tiedot Firebaseen. Tilin luonnissa käyttäjää pyydetään antamaan käyttäjänimi, sähköposti ja salasana. ”Kirjaudu sisään” -näkymä ja ”Tilin luonti” -näkymä nähdään kuvassa 19.

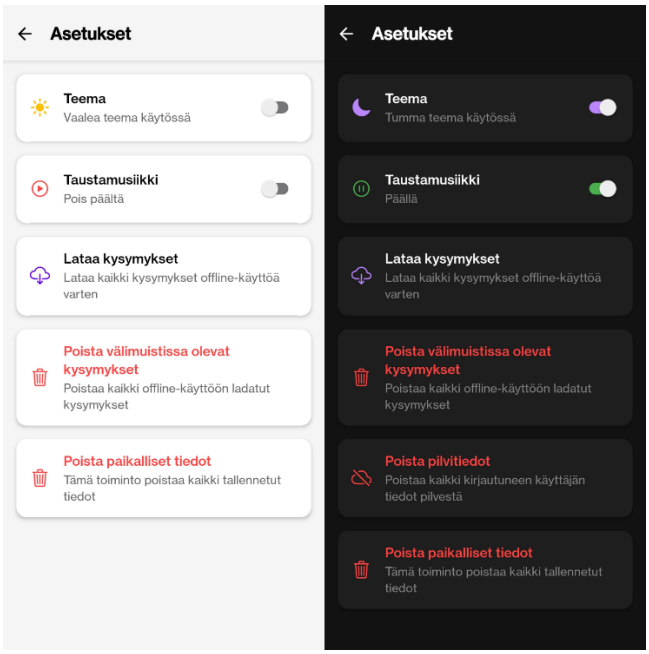


KUVA 19. "Kirjaudu Sisään"- ja "Luo Tili" -näkymät

5.5 Asetukset -komponentti

Asetukset-näkymä antaa käyttäjän muuttaa sovelluksen käyttökokemusta. Täältä käyttäjä voi muuttaa sovelluksen teemaa vaaleasta tummaan. Tämän lisäksi näkymästä voi vaihtaa, haluaako sovellukseen taustamusiikkia vai ei. Internet-yhteyttömää käyttöä varten käyttäjä voi halutessaan ladata englanti- ja maantietopelien kysymykset omaan laitteeseen.

Täältä käyttäjä voi myös poistaa omat tilastotietonsa halutessaan. Jos on kirjautunut sisään, näkymään ilmestyy uusi nappi, ”Poista pilvitiedot”. Tämä nappi antaa kirjautuneen käyttäjän poistaa tilastotietonsa Firebasesta. On kuitenkin huomioitava, että tämä poistaa vain tilastotiedot, ei käyttäjätiliä. Asetukset -näkymä nähdään kuvassa 20.



KUVA 20. ”Asetukset” -näkymä

5.6 Tulostaulu –komponentti

Tässä osiossa nähdään kuvista 21, 22 ja 23 miten tulostaulu on toteutettu ohjelmallisesti ja miltä se näyttää mobiililaitteella avattuna

```
const ScoreBoard = () => {
  const { isDarkMode, toggleTheme } = useTheme(); // Access theme context
  const styles = createStyles(isDarkMode); // Generate dynamic styles

  const [topPerformers, setTopPerformers] = useState({
    totalCorrectAnswers: { username: '', value: 0 },
    totalWrongAnswers: { username: '', value: 0 },
    completedQuizzes: { username: '', value: 0 },
    totalCorrectAnswersMath: { username: '', value: 0 },
    totalCorrectAnswersEng: { username: '', value: 0 },
    totalCorrectAnswersCountry: { username: '', value: 0 },
  });
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    const fetchAllUsersStats = async () => {
      try {
        const usersCollection = collection(db, 'users');
        const usersSnapshot = await getDocs(usersCollection);
        const usersData = usersSnapshot.docs.map((doc) => doc.data());

        // Calculate top performers
        const performers = {
          totalCorrectAnswers: { username: '', value: 0 },
          totalWrongAnswers: { username: '', value: 0 },
          completedQuizzes: { username: '', value: 0 },
          totalCorrectAnswersMath: { username: '', value: 0 },
          totalCorrectAnswersEng: { username: '', value: 0 },
          totalCorrectAnswersCountry: { username: '', value: 0 },
        };
```

KUVA 21. Tulostaulu -komponentin rakenne

Jokaisen rekisteröityneen käyttäjän tiedot haetaan Firebasen tietokannasta ja ne käydään läpi sekä verrataan forEach –funktiota käyttäen, jos jonkun käyttäjän datakentän arvo on suurempi kuin toisten niin asetetaan tämä kärkipäähän. Toteutus näkyy kuvasta 22.

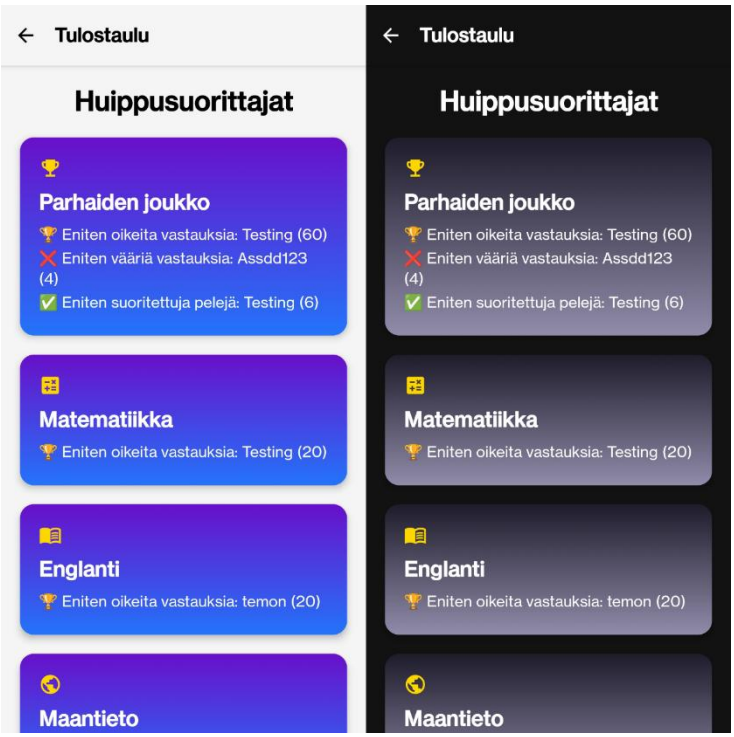
```
usersData.forEach((user) => {
  // Total Correct Answers
  if (user.totalCorrectAnswers > performers.totalCorrectAnswers.value) {
    performers.totalCorrectAnswers.username = user.username || 'Anonymous';
    performers.totalCorrectAnswers.value = user.totalCorrectAnswers;
  }

  // Total Wrong Answers
  if (user.totalWrongAnswers > performers.totalWrongAnswers.value) {
    performers.totalWrongAnswers.username = user.username || 'Anonymous';
    performers.totalWrongAnswers.value = user.totalWrongAnswers;
  }

  // Completed Quizzes
  if (user.completedQuizzes > performers.completedQuizzes.value) {
    performers.completedQuizzes.username = user.username || 'Anonymous';
    performers.completedQuizzes.value = user.completedQuizzes;
  }

  // Math Correct Answers
  if (user.totalCorrectAnswersMath > performers.totalCorrectAnswersMath.value) {
    performers.totalCorrectAnswersMath.username = user.username || 'Anonymous';
    performers.totalCorrectAnswersMath.value = user.totalCorrectAnswersMath;
  }

  // English Correct Answers
  if (user.totalCorrectAnswersEng > performers.totalCorrectAnswersEng.value) {
    performers.totalCorrectAnswersEng.username = user.username || 'Anonymous';
    performers.totalCorrectAnswersEng.value = user.totalCorrectAnswersEng;
  }

  // Country Correct Answers
  if (user.totalCorrectAnswersCountry > performers.totalCorrectAnswersCountry.value) {
    performers.totalCorrectAnswersCountry.username = user.username || 'Anonymous';
    performers.totalCorrectAnswersCountry.value = user.totalCorrectAnswersCountry;
  }
});

setTopPerformers(performers);
setLoading(false);
} catch (error) {
console.error('Error fetching all users stats:', error);
```

KUVA 22. Ohjelmallinen toteutus käyttäjien datan vertaamisesta

Kuvassa 23 nähdään datan visualisoinnin toteutus ohjelmallisesti ja kuvassa 24 nähdään lopullinen näkymä mobiililaitteessa avattuna:

```
return (
  <ScrollView contentContainerStyle={styles.container}>
    <Text style={styles.headerTitle}>Huippusuorittajat</Text>

    {/* Overall Performance Card */}
    <LinearGradient
      colors={isDarkMode ? ['#1F1C2C', '#928DAB'] : ['#6A11CB', '#2575FC']}
      style={styles.card}
    >
      <Icon name="emoji-events" size={24} color="#FFD700" style={styles.icon} />
      <Text style={styles.sectionTitle}>Parhaiden joukko</Text>
      <Text style={styles.text}>
        🏆 Eniten oikeita vastauksia: {topPerformers.totalCorrectAnswers.username} (
        {topPerformers.totalCorrectAnswers.value})
      </Text>
      <Text style={styles.text}>
        ❌ Eniten vääriä vastauksia: {topPerformers.totalWrongAnswers.username} (
        {topPerformers.totalWrongAnswers.value})
      </Text>
      <Text style={styles.text}>
        ✅ Eniten suoritettuja pelejä: {topPerformers.completedQuizzes.username} (
        {topPerformers.completedQuizzes.value})
      </Text>
    </LinearGradient>

    {/* Math Card */}
    <LinearGradient
      colors={isDarkMode ? ['#1F1C2C', '#928DAB'] : ['#6A11CB', '#2575FC']}
      style={styles.card}
    >
      <Icon name="calculate" size={24} color="#FFD700" style={styles.icon} />
      <Text style={styles.sectionTitle}>Matematiikka</Text>
      <Text style={styles.text}>
        🏆 Eniten oikeita vastauksia: {topPerformers.totalCorrectAnswersMath.username} (
        {topPerformers.totalCorrectAnswersMath.value})
      </Text>
    </LinearGradient>
```

KUVA 23. Pistetaulun datan visualisointi ohjelmallisesti



KUVA 24. Pistetaulun lopullinen näkymä + vaalea/tumma teema

5.7 StatsManager –komponentti

StatsManager-komponentti on keskeinen osa oppimissovellusta, joka hallinnoi käyttäjän suoritustilastoja ja asetuksia sekä paikallisesti että pilvipohjaisesti. Se hyödyntää AsyncStorage -kirjastoa tietojen tallentamiseen paikallisesti ja Firebase Firestore -tietokantaa pilvipohjaiseen tallennukseen.

Komponentin päätehtävät voidaan jakaa seuraaviin osa-alueisiin:

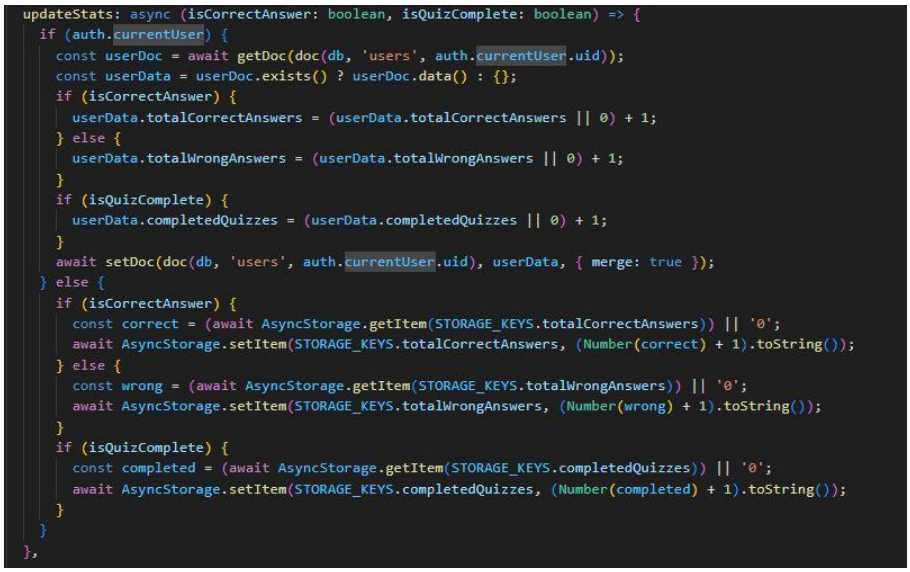
Tilastojen tallentaminen ja hakeminen: Kuvissa 25, 26 ja 27 nähdään ohjelmallisesti, miten StatsManager hakee ja tallentaa käyttäjän vastausten oikeellisuuden, suoritetut tietokilpailut ja edistymisen eri aihealueilla, eli matematiikassa, englannissa ja maantieteessä. Jos käyttäjä on kirjautunut sisään, tiedot tallennetaan Firestore tietokantaan. Jos käyttäjä ei ole kirjautunut, tiedot säilytetään paikallisesti AsyncStoragessa.

```
getStats: async () => {
  if (auth.currentUser) {
    const userDoc = await getDoc(doc(db, 'users', auth.currentUser.uid));
    if (userDoc.exists()) {
      const userData = userDoc.data();

      return {
        totalCorrectAnswers: userData.totalCorrectAnswers || 0,
        totalWrongAnswers: userData.totalWrongAnswers || 0,
        completedQuizzes: userData.completedQuizzes || 0,
        totalCorrectAnswersMath: userData.totalCorrectAnswersMath || 0,
        totalWrongAnswersMath: userData.totalWrongAnswersMath || 0,
        completedQuizzesMath: userData.completedQuizzesMath || 0,
        totalCorrectAnswersEng: userData.totalCorrectAnswersEng || 0,
        totalWrongAnswersEng: userData.totalWrongAnswersEng || 0,
        completedQuizzesEng: userData.completedQuizzesEng || 0,
        totalCorrectAnswersCountry: userData.totalCorrectAnswersCountry || 0,
        totalWrongAnswersCountry: userData.totalWrongAnswersCountry || 0,
        completedQuizzesCountry: userData.completedQuizzesCountry || 0,
        theme: userData.theme || 'light',
        unlockedLevelsMath: userData.unlockedLevelsMath || [1], // Ensure unlockedLevelsMath is included
        passedLevelsMath: userData.passedLevelsMath || [0], // Ensure passedLevelsMath is included
        unlockedLevelsEng: userData.unlockedLevelsEng || [1],
        passedLevelsEng: userData.passedLevelsEng || [0],
        unlockedLevelsCountry: userData.unlockedLevelsCountry || [1],
        passedLevelsCountry: userData.passedLevelsCountry || [],

      };
    }
  } else {
```

KUVA 25. Tietojen hakeminen Firestoresta.

```
} else {
  const correct = await AsyncStorage.getItem(STORAGE_KEYS.totalCorrectAnswers);
  const wrong = await AsyncStorage.getItem(STORAGE_KEYS.totalWrongAnswers);
  const completed = await AsyncStorage.getItem(STORAGE_KEYS.completedQuizzes);
  const correctMath = await AsyncStorage.getItem(STORAGE_KEYS.totalCorrectAnswersMath);
  const wrongMath = await AsyncStorage.getItem(STORAGE_KEYS.totalWrongAnswersMath);
  const completedMath = await AsyncStorage.getItem(STORAGE_KEYS.completedQuizzesMath);
  const correctEng = await AsyncStorage.getItem(STORAGE_KEYS.totalCorrectAnswersEng);
  const wrongEng = await AsyncStorage.getItem(STORAGE_KEYS.totalWrongAnswersEng);
  const completedEng = await AsyncStorage.getItem(STORAGE_KEYS.completedQuizzesEng);
  const correctCountry = await AsyncStorage.getItem(STORAGE_KEYS.totalCorrectAnswersCountry);
  const wrongCountry = await AsyncStorage.getItem(STORAGE_KEYS.totalWrongAnswersCountry);
  const completedCountry = await AsyncStorage.getItem(STORAGE_KEYS.completedQuizzesCountry);
  const theme = await AsyncStorage.getItem(STORAGE_KEYS.theme);
  const unlockedLevelsMath = await AsyncStorage.getItem(LEVELSCREEN_STORAGE_KEYS.unlockedLevelsMath);
  const passedLevelsMath = await AsyncStorage.getItem(LEVELSCREEN_STORAGE_KEYS.passedLevelsMath)
  const unlockedLevelsEng = await AsyncStorage.getItem(LEVELSCREEN_STORAGE_KEYS.unlockedLevelsEng);
  const passedLevelsEng = await AsyncStorage.getItem(LEVELSCREEN_STORAGE_KEYS.passedLevelsEng)
  const unlockedLevelsCountry = await AsyncStorage.getItem(LEVELSCREEN_STORAGE_KEYS.unlockedLevelsCountry)
  const passedLevelsCountry = await AsyncStorage.getItem(LEVELSCREEN_STORAGE_KEYS.passedLevelsCountry)
  return {
    totalCorrectAnswers: Number(correct) || 0,
    totalWrongAnswers: Number(wrong) || 0,
    completedQuizzes: Number(completed) || 0,
    totalCorrectAnswersMath: Number(correctMath) || 0,
    totalWrongAnswersMath: Number(wrongMath) || 0,
    completedQuizzesMath: Number(completedMath) || 0,
    totalCorrectAnswersEng: Number(correctEng) || 0,
    totalWrongAnswersEng: Number(wrongEng) || 0,
    completedQuizzesEng: Number(completedEng) || 0,
    totalCorrectAnswersCountry: Number(correctCountry) || 0,
    totalWrongAnswersCountry: Number(wrongCountry) || 0,
    completedQuizzesCountry: Number(completedCountry) || 0,
    theme: theme || 'light',
    // Ensure that unlocked and passed levels are included ->
    unlockedLevelsMath: unlockedLevelsMath ? JSON.parse(unlockedLevelsMath) : [1],
    passedLevelsMath: passedLevelsMath ? JSON.parse(passedLevelsMath) : [],
    unlockedLevelsEng: unlockedLevelsEng ? JSON.parse(unlockedLevelsEng) : [1],
    passedLevelsEng: passedLevelsEng ? JSON.parse(passedLevelsEng) : [],
    unlockedLevelsCountry: unlockedLevelsCountry ? JSON.parse(unlockedLevelsCountry) : [1],
    passedLevelsCountry: passedLevelsCountry ? JSON.parse(passedLevelsCountry) : [],
  };
}
```

KUVA 26. Tietojen hakeminen paikallisesta muistista.



```
updateStats: async (isCorrectAnswer: boolean, isQuizComplete: boolean) => {
  if (auth.currentUser) {
    const userDoc = await getDoc(doc(db, 'users', auth.currentUser.uid));
    const userData = userDoc.exists() ? userDoc.data() : {};
    if (isCorrectAnswer) {
      userData.totalCorrectAnswers = (userData.totalCorrectAnswers || 0) + 1;
    } else {
      userData.totalWrongAnswers = (userData.totalWrongAnswers || 0) + 1;
    }
    if (isQuizComplete) {
      userData.completedQuizzes = (userData.completedQuizzes || 0) + 1;
    }
    await setDoc(doc(db, 'users', auth.currentUser.uid), userData, { merge: true });
  } else {
    if (isCorrectAnswer) {
      const correct = (await AsyncStorage.getItem(STORAGE_KEYS.totalCorrectAnswers)) || '0';
      await AsyncStorage.setItem(STORAGE_KEYS.totalCorrectAnswers, (Number(correct) + 1).toString());
    } else {
      const wrong = (await AsyncStorage.getItem(STORAGE_KEYS.totalWrongAnswers)) || '0';
      await AsyncStorage.setItem(STORAGE_KEYS.totalWrongAnswers, (Number(wrong) + 1).toString());
    }
    if (isQuizComplete) {
      const completed = (await AsyncStorage.getItem(STORAGE_KEYS.completedQuizzes)) || '0';
      await AsyncStorage.setItem(STORAGE_KEYS.completedQuizzes, (Number(completed) + 1).toString());
    }
  }
},
```

KUVA 27. Tietojen tallentaminen/päivittäminen Firestoreen tai paikalliseen muistiin.

Kysymysten hakeminen: Kuvassa 28 nähdään miten StatsManager hakee käyttäjälle sopivia kysymyksiä Firestoresta.

```
const StatsManager = {

  getQuestionsForSubject: async (
    subject: 'english' | 'country',
    level: number,
    shouldShuffle: boolean
  ) => {
    const config = {
      english: {
        collection: 'englishQuestions',
        storageKey: STORAGE_KEYS.englishQuestions,
        docId: `level${level}`
      },
      country: {
        collection: 'countryLevels',
        storageKey: STORAGE_KEYS.countryQuestions,
        docId: `level_${level}`
      }
    };

    const { collection, storageKey, docId } = config[subject];
    const storageKeyWithLevel = `${storageKey}_level${level}`;

    try {
      const cachedQuestions = await AsyncStorage.getItem(storageKeyWithLevel);
      let questions = cachedQuestions ? JSON.parse(cachedQuestions) : [];

      if (!questions.length) {
        const docSnapshot = await getDoc(doc(db, collection, docId));
        if (docSnapshot.exists()) {
          questions = docSnapshot.data()?.questions || [];
          await AsyncStorage.setItem(storageKeyWithLevel, JSON.stringify(questions));
        }
      }

      if (shouldShuffle) {
        const shuffledQuestions = [...questions].sort(() => Math.random() - 0.5);
        return shuffledQuestions.map(question => ({
          ...question,
          options: [...question.options].sort(() => Math.random() - 0.5)
        }));
      }

      return questions;
```

KUVA 28. Kysymysten hakeminen Firestoresta.

Tasonhallinta: Kuvissa 29 ja 30 näkyy kuinka StatsManager seuraa käyttäjän etenemistä yksittäisissä aihealueissa hakemalla ja tallentamalla avatut ja suoritetut tasot. Tämä mahdollistaa käyttäjäkohtaisen oppimispolun säilyttämisen sekä paikallisesti että pilvessä.

```
// Get unlocked levels for a specific subject
getUnlockedLevels:  async (subject: 'Math' | 'Eng' | 'Country') => {
  if (auth.currentUser) {
    const userDoc = await getDoc(doc(db, 'users', auth.currentUser.uid));
    if (userDoc.exists()) {
      const userData = userDoc.data();
      const key = `unlockedLevels${subject}` as keyof typeof userData;
      return userData[key] || [1];
    }
  } else {
    const key = `unlockedLevels${subject}` as LevelScreen_StorageKey;
    const unlockedLevels = await AsyncStorage.getItem(LEVELSCREEN_STORAGE_KEYS[key]);
    return unlockedLevels ? JSON.parse(unlockedLevels) : [1];
  }
},

// Get passed levels for a specific subject
getPassedLevels: async (subject: 'Math' | 'Eng' | 'Country') => {
  if (auth.currentUser) {
    const userDoc = await getDoc(doc(db, 'users', auth.currentUser.uid));
    if (userDoc.exists()) {
      const userData = userDoc.data();
      const key = `passedLevels${subject}` as keyof typeof userData;
      return userData[key] || [];
    }
  } else {
    const key = `passedLevels${subject}` as LevelScreen_StorageKey;
    const passedLevels = await AsyncStorage.getItem(LEVELSCREEN_STORAGE_KEYS[key]);
    return passedLevels ? JSON.parse(passedLevels) : [];
  }
},
```

KUVA 29. Tasojen hakeminen paikallisesta muistista tai Firestoresta.

```
// Set unlocked levels for a specific subject
setUnlockedLevels: async (subject:  'Math' | 'Eng' | 'Country', level: number) => {

  const stats = await StatsManager.getStats();
  const key = `unlockedLevels${subject}` as keyof typeof stats;
  const unlockedLevels = (stats as Record<string, any>)[key] || [1];

  if (!unlockedLevels.includes(level)) {
    unlockedLevels.push(level);
    if (auth.currentUser) {
      await setDoc(doc(db, 'users', auth.currentUser.uid), { [key]: unlockedLevels }, { merge: true });
    } else {
      await AsyncStorage.setItem(STORAGE_KEYS[key], JSON.stringify(unlockedLevels));
    }
  }
},

// Set passed levels for a specific subject
setPassedLevels: async (subject: 'Math' | 'Eng' | 'Country', level: number) => {
  const stats = await StatsManager.getStats();
  const key = `passedLevels${subject}` as keyof typeof stats;
  const passedLevels = (stats as Record<string, any>)[key] || [];

  if (!passedLevels.includes(level)) {
    passedLevels.push(level);
    if (auth.currentUser) {
      await setDoc(doc(db, 'users', auth.currentUser.uid), { [key]: passedLevels }, { merge: true });
    } else {
      await AsyncStorage.setItem(STORAGE_KEYS[key], JSON.stringify(passedLevels));
    }
  }
},
```

KUVA 30. Tasojen datan päivittäminen paikalliseen muistiin tai Firestoreen.

Teeman hallinta: Kuvassa 31 nähdään että käyttäjän valitsema tumma tai vaalea teema tallennetaan ja haetaan, jotta sovelluksen ulkoasu pysyy yhtenäisenä eri käyttökertojen välillä. Tämä tieto tallennetaan joko AsyncStorageen tai Firestoreen riippuen siitä, onko käyttäjä kirjautuneena vai “vieraana”.

```
// Save the theme preference
setTheme: async (theme: 'light' | 'dark') => {
  if (auth.currentUser) {
    await setDoc(doc(db, 'users', auth.currentUser.uid), { theme }, { merge: true });
  } else {
    await AsyncStorage.setItem(STORAGE_KEYS.theme, theme);
  }
},

getTheme: async () => {
  if (auth.currentUser) {
    const userDoc = await getDoc(doc(db, 'users', auth.currentUser.uid));
    if (userDoc.exists()) {
      const userData = userDoc.data();
      return userData.theme || 'light';
    }
  } else {
    const theme = await AsyncStorage.getItem(STORAGE_KEYS.theme);
    return theme || 'light';
  }
},
```

KUVA 31. Teeman tilan hakeminen ja tallentaminen.

Yhteenvetona voidaan todeta, että StatsManagerin toiminnallisuus tukee oppimissovelluksen käyttäjäkokemusta tarjoamalla luotettavan tavan säilyttää ja hakea tietoja eri laitteilla ja käyttökerroilla.

5.8 Pelit

Sovelluksen pääominaisuutena on kolme peliä, joiden teemat ovat: matematiikka, maantieto ja englanti. Jokaisessa pelissä käyttäjälle esitetään kysymys ja neljä vastausvaihtoehtoa, joista yksi on oikea. Vastattuaan käyttäjä saa visuaalisen ja äänellisen palautteen siitä, menikö vastaus oikein vai väärin. Jos vastaus on oikein, nappi muuttuu vihreäksi ja soitetaan korkea sävel. Jos vastaus on väärin, nappi muuttuu punaiseksi ja soitetaan matala sävel.

Tämän lisäksi käyttäjän tasoedistystä seuraa sovellukseen implementoitu ”Progressbar” eli edistymispalkki, joka täyttyy sinisellä värillä sen mukaan, kuinka

moneen kysymykseen käyttäjä on vastannut. Edistymispalkin ensimmäinen implementointi sovellukseen voidaan nähdä kuvassa 35.

5.8.1 Matematiikkapeli

Tämä komponentti pitää sisällään suhteellisen helppoja yhteen, vähennys, jako ja kerto –laskuja, jotka soveltuvat noin 7-10-vuotiaille.

Ensimmäisessä versiossa kysymykset ovat vielä kovakoodattuja, tämä on tarkoitus muuttaa niin että matemaattisten kysymysten muuttujien arvot muodostetaan käyttämällä satunnaisnumero generaattoria, jotta kysymykset olisivat joka kerta eriarvoiset. Versio 1. nähdään kuvissa 32 ja 33.

KUVA 32. Oikein vastattu

KUVA 33. Väärin vastattu

Versio 2. Kuvista 34 ja 35 näkyy lisätty pistetarkkuus, edistymispalkki ja satunnaisnumero generaattori.

KUVA 34. Pistetarkkuus alle 50% punaisena, 50% tai yli vihreänä ja edistymisen palkki näkyy oikeanpuoleisessa kuvassa.

```
const generateRandomQuestion = (operation: string): QuizOption => {
  let A = Math.floor(Math.random() * 12) + 1;
  let B = Math.floor(Math.random() * 12) + 1;
  if (operation === "Vähennyslaskut" && A < B) [A, B] = [B, A];

  let question = "";
  let answer = 0;

  switch (operation) {
    case "Yhteenlaskut":
      question = `Kuinka paljon on ${A} + ${B}?`;
      answer = A + B;
      break;
    case "Vähennyslaskut":
      question = `Kuinka paljon on ${A} - ${B}?`;
      answer = A - B;
      break;
    case "Kertolaskut":
      question = `Kuinka paljon on ${A} × ${B}?`;
      answer = A * B;
      break;
    case "Jakolaskut":
    default:
      B = Math.floor(Math.random() * 5) + 1;
      A = B * (Math.floor(Math.random() * 10) + 1);
      question = `Kuinka paljon on ${A} ÷ ${B}?`;
      answer = A / B;
      break;
  }
```

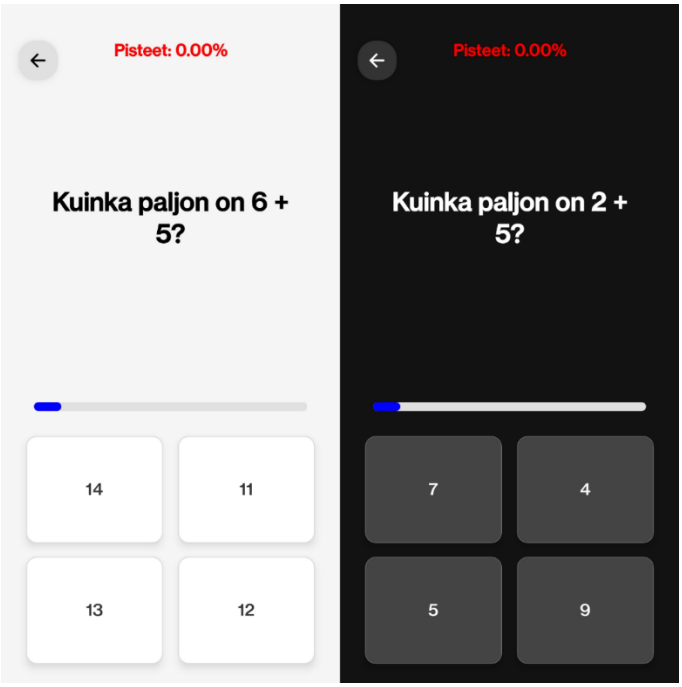
KUVA 35. "Satunnaisnumero generaattorin ohjelmallinen toteutus"

Kuvasta 36 nähdään, miten kysymykset generoidaan.

```
Generated question for Yhteenlaskut: Kuinka paljon on 17 + 10?, Answer: 27
Incorrect options generated: 29,30,31
Final options for Kuinka paljon on 17 + 10?: 27,30,29,31
Generated question for Yhteenlaskut: Kuinka paljon on 3 + 3?, Answer: 6
Incorrect options generated: 8,7,4
Final options for Kuinka paljon on 3 + 3?: 7,6,8,4
Generated question for Yhteenlaskut: Kuinka paljon on 17 + 1?, Answer: 18
Incorrect options generated: 17,15,16
Final options for Kuinka paljon on 17 + 1?: 17,18,15,16
Generated question for Vähennyslaskut: Kuinka paljon on 17 - 13?, Answer: 4
Incorrect options generated: 6,8,3
Final options for Kuinka paljon on 17 - 13?: 4,6,3,8
```

KUVA 36. "Kysymysten generoimisen periaate konsolista"

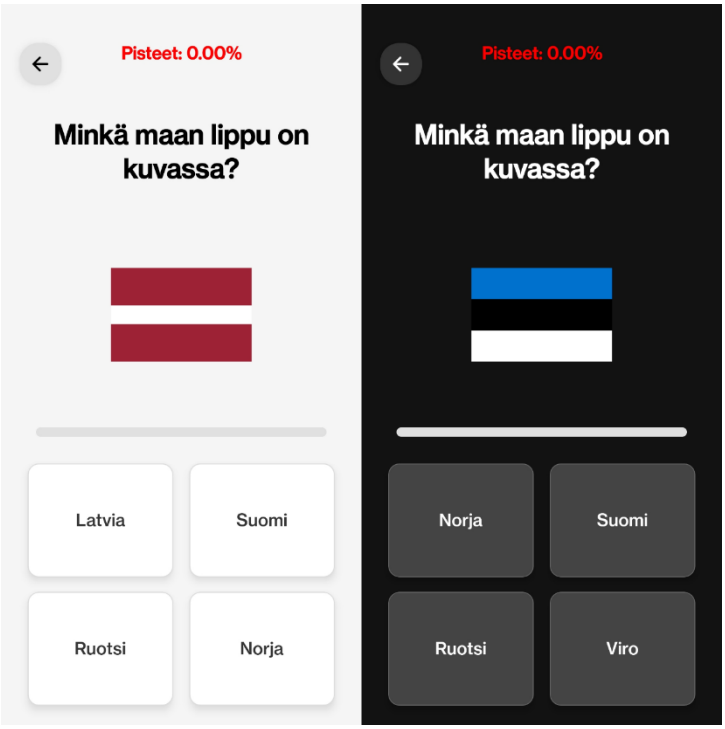
Kuvasta 37 nähdään lopullinen versio.



KUVA 37. Lopullinen näkymä vaalea/tumma teema asetuksella

5.8.2 Maantietopeli

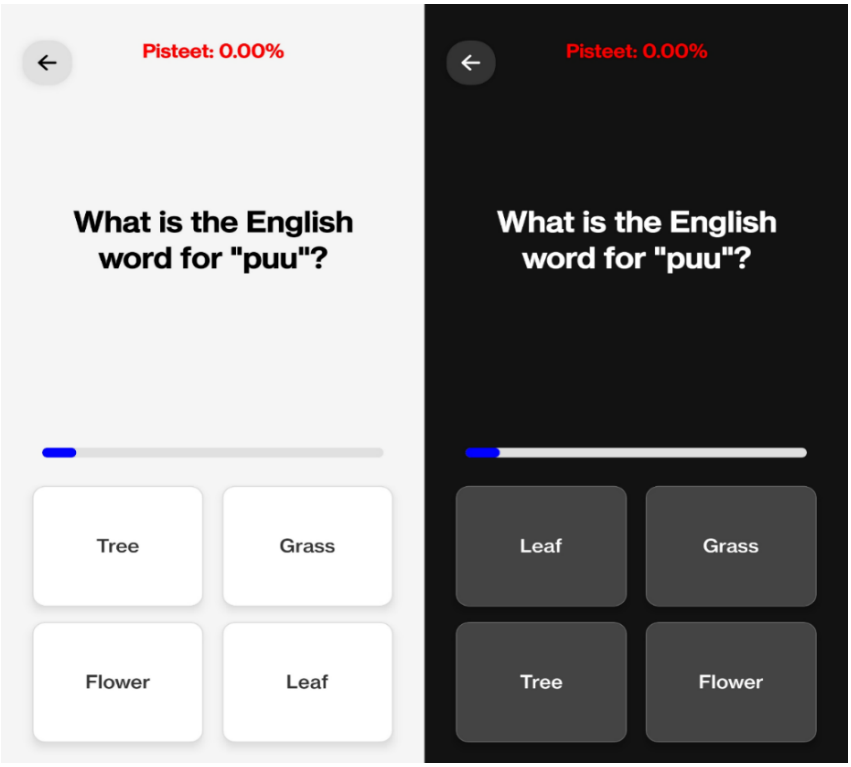
Maantietopeli on toteutettu samalla monivalintakaavalla, kuin muutkin pelit sovelluksessa. Peli pitää sisällään viisi eri tasoa, joissa jokaisessa on 10 tunnistettavaa lippua. Näihin viiteen tasoon sisältyy Suomen naapurimaat, Baltian maat ja lisäksi Puola ja Valko-Venäjä. Seuraavalla tasolla peli vaikeutuu ja mukaan tulee maita koko Euroopasta. Kolmannella tasolla tutustutaan Pohjois-Amerikan maihin ja neljännellä Etelä-Amerikan maihin. Pelin viimeisellä tasolla tunnistetaan Aasian maiden lippuja. Kuvassa 38 näkyy maantietopelin näkymä.



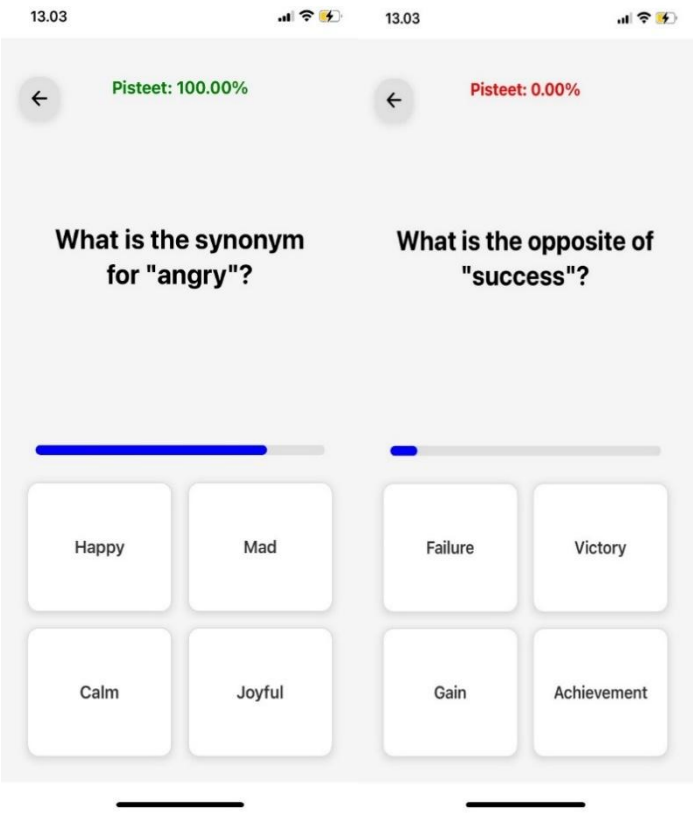
KUVA 38. Maantietopelin näkymä

5.8.3 Englantipeli

Tässä pelissä pelaajalle esitetään Englannin kielen kysymyksiä. Sen lisäksi, että kysymyksissä pitää osata englantia on seassa myös ripaus yleistietoa. Ensimmäisen tason kysymykset ovat lapsille tuttuja sanoja, joihin pitää löytää englannin kielen vastine. Myöhemmillä tasoilla pelaajan pitää osata esimerkiksi valita oikea aikamuoto tai löytää synonyymi tai vastakohta esitetylle sanalle, kuten kuvassa 39 ja 40 näkyy.



KUVA 39. Ensimmäisen tason tehtävä



KUVA 40. Kolmannen tason tehtävä

Englanninkielisten kysymysten käyttö pelissä tuo monia etuja, erityisesti lapsille, jotka opettelevat kieltä. Tässä syitä, miksi tämä lähestymistapa on tehokkaampi kuin suomenkielisten kysymysten käyttö:

1. Täydellinen kieliympäristö ja luonnollinen oppiminen

Kun peli esittää kysymykset englanniksi, pelaajan täytyy käyttää kieltä jatkuvasti. Tämä luo luonnollisen kieliympäristön, jossa englannin ymmärtäminen kehittyy tehokkaammin verrattuna siihen, että kysymykset olisivat suomeksi ja vastaukset englanniksi.

2. Sanaston ja lauserakenteiden oppiminen kontekstissa

Uuden kielen oppiminen on tehokkainta, kun sanat ja ilmaukset opitaan osana kokonaisia lauseita ja tilanteita. Englanninkieliset kysymykset auttavat pelaajaa ymmärtämään sanoja niiden käyttöyhteyksissä, mikä tekee oppimisesta luonte-vampaa ja pysyvämpää.

3. Suoraan englanniksi ajattelu

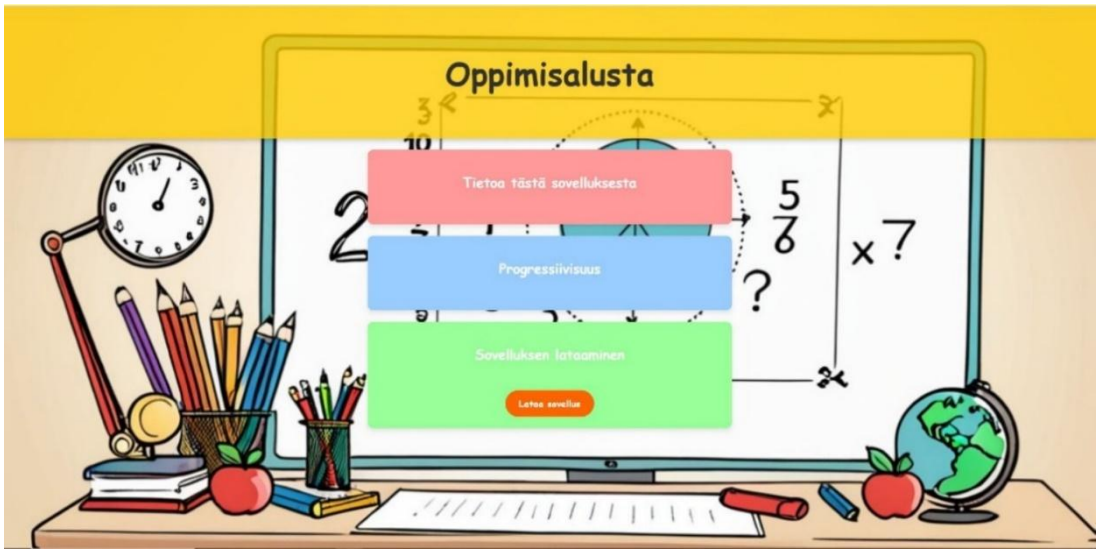
Jos kysymykset olisivat suomeksi, pelaaja voisi alkaa kääntää niitä mielessä, mikä hidastaa kielen oppimista. Kun kaikki on alusta asti englanniksi, oppiminen tapahtuu ilman ylimääräistä käännösprosessia, ja kielitaito kehittyy sujuvammin.

4. Valmius todellisiin tilanteisiin

Maailmalla englanti on keskeinen kieli, ja monissa tilanteissa (esim. pelit, koulutus, matkailu) ei ole tarjolla käännöksiä. Kun lapset tottuvat käsittelemään englantia heti ilman suomen tukea, he oppivat käyttämään kieltä varmemmin ja itsenäisemmin. (28)

5.9 Verkkosivu

Kuvista 41 ja 42 nähdään visuaalinen näkymä lopullisesta verkkosivusta.



KUVA 41. Verkkosivun ulkoasu

KUVA 42. Sivun osiot laajenevat, kun otsikkotekstiä painetaan

6 POHDINTA

Projektin toteuttaminen onnistui suunnitelmien mukaisesti. Emme alun perinkään aikoneet tehdä alustan sisältämistä tietokilpailuista/peleistä kovin laajoja tiukan aikarajan takia ja tarkoituksena olikin tehdä modulaarinen sovellus, jota voi jatkossa helposti kehittää ja laajentaa. Tarkoituksena oli tehdä yksinkertainen sovellus, jota esimerkiksi ala-asteikäiset lapset voivat käyttää kouluopintojen tukena.

Projektissa tuli tutuksi käyttäjän pelinsisäisten tietojen tallentaminen paikallisesti sekä pilvipalveluun ja varsinkin kun kohderyhmänä oli lapset, niin jouduimme tutkimaan ja oppimaan myös tietoturvaa sekä pedagogiikkaa. Helpoin ja fiksuin ratkaisu tietoturvan osalta oli se, että emme yksinkertaisesti tallenna mitään arkaluontoista dataa kuten oikeita nimiä ja osoitteita ja mikä sen parempi tapa varmistaa tietoturva kuin se, ettei tietoja ole olemassakaan.

Projektin aikana opittiin uusia React Nativen toimintatapoja ja miten sen kirjastoja voidaan käyttää, kuten ”Animated” -kirjasto, jota käytettiin Taso -näkymän nappien animoimiseen sekä “ASyncStorage” -kirjasto, jota käytettiin pelitietojen tallentamiseen paikallisesti.

Oppimisalustaa voisi laajentaa lisäämällä eri oppimisaiheita kuten esimerkiksi historia, biologia ja eri kieliä. Tasoja on työn valmistuttua 5 per aihe, näitäkin saisi lisättyä helposti ja nopeasti koska koodi on modulaarista. Sovellukseen voisi tulevaisuudessa myös lisätä erilaisia saavutuksia pelaajalle pelillistämisen tueksi. Data-analytiikkaa ja tekoälyä voisi myös lisätä koska niiden avulla olisi mahdollista kartoittaa pelaajan vahvuuksia sekä heikkouksia.

Sovelluksella on myös potentiaalia kaupalliseen käyttöön, erityisesti kouluympäristöissä. Oppilaitokset voisivat hyödyntää sitä opetuksen tukena, tarjoten oppilaille interaktiivisen ja pelillistetyn tavan oppia eri aihealueita. Sovellus voitaisiin lisensoida kouluille, jolloin ne saisivat käyttöönsä laajemmat raportointityökalut opettajille, kuten edistymisen seuranta ja mukautettavat oppimispolut. Lisäksi sovellus voisi sisältää koulujen tarpeisiin räätälöityjä sisältöjä, kuten

opetussuunnitelmaan sopivia tehtäviä. Tällainen kaupallinen malli mahdollistaisi jatkuvan kehityksen ja uusien oppimissisältöjen lisäämisen käyttäjäpalautteen perusteella.

Pelillistetty oppimissovellus oli uudenlainen projekti ryhmällemme, joten oli mieluisaa oppia, miten käyttää React Nativea ja luoda sen avulla sovellus, jota muut ihmiset voivat käyttää omaan oppimiseen. Jos jotain sovelluksessa voisi parantaa niin tasoista ja kysymyksistä varsinkin kielipeleissä voisi tehdä rakenteellisempia ja tarkempia. Vielä parempi tapa kehittää pelien kysymyksiä olisi, että kysymykset otettaisiin suoraan koulujen koulukirjoista.

LÄHTEET

1. Expo. 2025. Expo. Luettavissa: https://expo.dev/ . Luettu: 21.2.2025.

2. Meta Platforms, Inc. 2025. React Native. Luettavissa: https://reactnative.dev/. Luettu: 21.2.2025.

3. Google LLC. 2025. Cloud Storage for Firebase. Luettavissa: https://firebase.google.com/docs/storage. Luettu: 21.3.2025

4. Google LLC. 2025. Firebase Authentication. Luettavissa: https://firebase.google.com/docs/auth. Luettu: 21.3.2025

5. Google LLC. 2025. Cloud Firestore. Luettavissa: https://firebase.google.com/docs/firestore. Luettu: 21.3.2025

6. Innocent, C. 14.3.2022. A guide to React Native's AsyncStorage. LogRocket. Luettavissa: https://blog.logrocket.com/guide-react-natives-asyncstorage/ . Luettu 21.3.2025

7. Meta Platforms, Inc. 2025. AsyncStorage. Luettavissa: https://reactnative.dev/docs/asyncstorage. Luettu: 21.3.2025

8. Opetushallitus. 2025. Oppimisen ja koulunkäynnin tuki. Luettavissa: https://www.oph.fi/fi/koulutus-ja-tutkinnot/oppimisen-ja-koulunkaynnin-tuki. Luettu: 21.3.2025.

9. Markkanen, M. 17.2.2023. Varhaiskasvatus sovellusviidakossa – millainen on pedagogisesti laadukas digitaalinen ympäristö? LabFocus. Luettavissa: https://blogit.lab.fi/labfocus/varhaiskasvatus-sovellusviidakossa-millainen-on-pedagogisesti-laadukas-digitaalinen-ymparisto/. Luettu 21.2.2025.

10. Paige. 27.11.2017. Using colour to design for children. Prototypr. Luettavissa: https://blog.prototypr.io/week-8-using-colour-to-design-for-children-da5b98594a2a. Luettu: 23.2.2025.

11. Crudu, A. & MoldStud Research Team. 24.1.2024. The psychology of video game design: Understanding player motivations. MoldStud. Luettavissa:

https://moldstud.com/articles/p-the-psychology-of-video-game-design-understanding-player-motivations. Luettu: 23.2.2025.

12. Hannah, J. 21.12.2023. The ultimate guide to mobile app design: Follow these UI principles & best practices. UX Design Institute. Luettavissa: https://www.uxdesigninstitute.com/blog/ultimate-guide-to-mobile-app-design/. Luettu: 23.2.2025.

13. Aloka, H. 20.9.2023. How to implement dark/light themes in React Native? Medium. Luettavissa: https://medium.com/%40alokasamarathunge/how-to-implement-dark-light-themes-in-react-native-d9cee55596c6. Luettu: 21.3.2025

14. Locke, A, E., Latham, P, G. 6.2002. Building a Practically Useful Theory of Goal Setting and Task Motivation: A 35Year Odyssey. Luettavissa: https://www.researchgate.net/publication/254734316_Building_a_Practically_Useful_Theory_of_Goal_Setting_and_Task_Motivation_A_35Year_Odyssey. Luettu: 21.3.2025

15. Nand, K., Baghaei, N., Casey, J., Barmada, B., Mehdipour, F. & Liang, H.-N. 18.7.2019. Engaging children with educational content via Gamification. SpringerOpen. Luettavissa: https://slejournal.springeropen.com/articles/10.1186/s40561-019-0085-2. Luettu: 23.2.2025.

16. Jørgensen, K. (2011). "Game Sound: An Introduction to the History, Theory, and Practice of Video Game Music and Sound Design." MIT Press.

17. Hallam, S. (2010). "The Power of Music: Its Impact on the Intellectual, Social and Personal Development of Children and Young People." International Journal of Music Education.

18. Blythe, M. (2019). "Sound and Play: Interactive Audio in Digital Games." Cambridge University Press.

19. Collins, K. (2013). "Playing with Sound: A Theory of Interaction with Sound and Music in Video Games." MIT Press.

20. Meyer, B., & Gast, R. (2018). "Audio-based Feedback and Learning in Children's Educational Apps." Journal of Interactive Learning Research.

21. Flanders, C. 2.5.2024. “The Psychology of Color in Mobile App Design”. Medium. Luettavissa: https://medium.com/@MobileAppDesigner/the-psychology-of-color-in-mobile-app-design-558fa1f7b9f4. Luettu: 3.4.2025.

22. Abdulmuize, A. 2.2.2025. “Responsive Design in React Native: Building Apps for Multiple Screen Sizes”. Dev. Luettavissa: https://dev.to/aomuiz/responsive-design-in-react-native-building-apps-for-multiple-screen-sizes-1fnf. Luettu: 1.4.2025.

23. Admin, A. 17.12.2024 “State Management in React Native: A Comprehensive Guide for 2025”. React Native Insights.

Luettavissa: https://reactnativeinsights.com/state-management-in-react-native-a-comprehensive-guide-for-2025/ Luettu: 1.4.2025.

24. Eliezer, C. 12.5.2023 “React Native: Accessibility for Visually Impaired People”. Whitesmith. Luettavissa: https://www.whitesmith.co/blog/react-native-a11y/ Luettu: 2.4.2025.

25. Bhalodia, V. 8.4.2024 “Enhancing React Native App Accessibility: Guidelines & FAQs”. Dev. Luettavissa: https://dev.to/vikrant_bhalodia/enhancing-react-native-app-accessibility-guidelines-faqs-4njd Luettu: 2.4.2025.

26. Education Recoded. 28.2.2025 “How Gamification in Education Boosts Student Motivation and Engagement”.

Educationrecoded. Luettavissa: https://educationrecoded.org/how-gamification-in-education-boosts-student-motivation-and-engagement/ Luettu: 2.4.2025.

27. Mohammed, M. Fatemah, A. Hassan, L. 6.3.2024 “Effects of Gamification on Motivations of Elementary School Students: An Action Research Field Experiment”. SageJournals. Luettavissa: https://journals.sagepub.com/doi/full/10.1177/10468781241237389. Luettu: 4.4.2025.

28. Opetushallitus. 14.5.2019. ”Perusopetuksen opetussuunnitelman perusteiden 2014 muutokset ja täydennykset koskien A1-kielen opetusta vuosiluokilla 1–2”. Pdf-tiedosto. https://www.oph.fi/sites/default/files/documents/perusopetuksen_vuosiluokkien_1-2_a1-kielen_opetussuunnitelman_perusteet.pdf. Luettu: 6.4.2025.