Tekstistä kuvaksi -generaattori palvelun ohjelmoiminen

Case DALL-E-2

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutus
Kevät, 2023

Niko Rantala

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutus		Tiivistelmä
Tekijä	Niko Rantala	Vuosi 2023
Työn nimi	Tekstistä kuvaksi -generaattori palvelu ohjelmoiminen	
Ohjaaja	Tommi Lahti	

TIIVISTELMÄ

Opinnäytetyön aiheena oli tekstistä kuvaksi -generaattorin ohjelmoiminen React-kielellä. Aiheeksi valittiin tekstistä kuvaksi -generaattorin ohjelmoiminen, koska tekoälyt ovat juuri nyt nousussa. Työn tavoitteena oli oppia React-kieltä ja miten siinä voidaan hyödyntää tekoälyä.

Opinnäytetyön teoriaosuus koostuu kolmesta osiosta. Ohjelmiston idea -luvussa suunnitellaan, mitä ominaisuuksia generaattori tulee sisältämään. Tämän jälkeen käydään läpi, miten se tullaan toteuttamaan. Idealuvussa käydään myös jo olemassa olevia kuvagenerointi palveluita, joiden pohjalta lähdettiin rakentamaan generaattoria. Työkalut-luvussa käydään läpi tässä työssä käytetyt työkalut. Työkaluihin kuuluivat ohjelmointikielet, DALL-E-2 -tekoäly ja VSCode -koodieditori. Kehitysympäristön asennus -luvussa käydään askel askeleelta läpi kaikki työhön tarvittavat asennukset.

Opinnäytetyö on toiminnallinen, vaikka siinä tehdäänkin kaksi tutkimuksellista kyselyä. Tutkimuskyselyillä haluttiin selvittää, kuinka moni kokee tekemäni generaattorin kuvan yhteensopivaksi annetun generointitekstin kanssa. Aineistoa analysoitiin tutkimalla vastauksia ja pohtimalla mitkä asiat ovat aiheuttaneet poikkeamat.

Työn johtopäätöksenä voidaan todeta, että tekoälyn liittäminen React-ohjelmistoon on tehty nykypäivänä helpoksi ja vaivattomaksi. Testaamisen aikana havaittiin, että tekoäly osaa generoida monipuolisia ja tunnistettavia kuvia. Kyselyiden pohjalta voidaan kuitenkin todeta generoiduissa kuvissa olevan kuitenkin aina tulkinnanvaraa.

Avainsanat	AI, Dal-E-2, React, Digitaide
Sivut	34 sivua ja liitteitä 5 sivua

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology Abstract

Author Niko Rantala Year 2023

Subject Text to image generator service programming

Supervisor Tommi Lahti

ABSTRACT

The subject of the thesis was programming a text-to-image generator in React. A text-to-image generator subject was chosen because artificial intelligences are right now a trending subject. Object of the work was to learn more about React and how to use an artificial intelligence in that language.

The theory part of the thesis includes three sections. The idea of program chapter contains preparations what features the program will include and how the project will be done. The idea of program also include research about what kind of text-to-image generator services are. This chapter tells about three different generator services. The tools chapter tells about the tools what was used in project. These tools were programming languages, DALL-E-2 AI and VSCode code editor. The development environment installations chapter contains all the installation instructions needed in this project.

This thesis is practical even though it contains two theoretical inquiries. The Surveys tried to get answers to the question “How many people find the image of the generator I made to be compatible with the text?”. The data was analyzed by researching surveys answers and by thinking what make survey deviations.

The Conclusion of this work is that including an artificial intelligence into React-software has been made easy and effortless. During the testing phase, it was discovered that artificial intelligence can generate diverse and recognizable pictures. But data from surveys tell that every generated picture have room for interpretation.

Keywords AI, Dal-E-2, React, Digitaide

Pages 34 pages and appendices 5 pages

Sanasto

AI (tekoäly)	Lyhenne englannin kielen sanasta artificial intelligence, joka tarkoittaa tekoälyä.
Algoritmi	Ohje, miten ohjelmistokieli toteutetaan.
Alustariippumaton	Ohjelmointikieli, joka ei ole sidoksissa mihinkään tiettyyn käyttöjärjestelmään.
Avoin lähdekoodi	Tapa, jolla kehitetään tai jaetaan tietokoneohjelmistoja.
Backend kieli	Koodi, jota ajetaan sivuston palvelimella. Esim. kirjautuminen ja tietokantojen käsittely.
Dal-E-2	OpenAI yrityksen luoma tekoäly, joka osaa muovata tekstistä kuvia.
Digitaide	Kaikki taide, josta tulee digitaalinen lopputulema.
Discord	Keskustelualusta, jolla lähetetään kuvia, videoita, tekstiä ja ääntä.
Funktio	Ohjelmoinnissa osa, joka suorittaa tietyn toiminnon. Sitä voidaan kutsua pääohjelmistoissa ja aliohjelmissa.
Github	Sivusto, joka antaa tallennustilaa ja työkaluja versionhallintaan.
Kirjasto (tietotekniikka)	Kokoelmia, luokkia ja aliohjelmia, joita käytetään tietokoneohjelmien kehityksessä.
Käyttöjärjestelmä	Tietokoneen ohjelmisto, joka mahdollistaa ohjelmien toiminnan.
Käyttöliittymä	Ohjelmiston osa, joka näkyy käyttäjälle.
Laajennus	Työkaluja, joilla voidaan parantaa ohjelmiston käyttökokemusta.
Ohjelmistokehys	Vanhan ohjelmiston kehitys, kun halutaan sen tekevän jotain uutta.
Ohjelmointikieli	Algoritmin toteuttamiseen tarkoitettu formaali kieli.
Palvelin	Palvelinohjelmisto, jota suoritetaan tietoliikenteen yhteydessä.
Parametri	Tietotekniikassa ohjelmiston funktiolle käynnistyksen yhteydessä tietoja välittävä tieto.
Poletti (tietotekniikka)	Metafora automaateissa käytetylle kolikkomaiselle maksuvälineelle.
Projektin hallinta	Toiminnan suunnittelu ja organisointi.
React	Ohjelmointikieli, jota käytetään ohjelmiston luonnissa.

Resoluutio	Kuvassa olevien pikseleiden määrä pituusyksiköllä.
Scrum	Ketterä tapa hallita projekteja.
Tekstieditori	Tietokoneohjelma, jolla voi kirjoittaa ja muokata tekstiä.
Versiohallinta	Tekniikka, jolla pidetään kirjaa tiedostoihin tehdyistä muutoksista ja säilötään vanhat versiot.

Sisällys

Kuvat, ohjelmakoodit ja taulukot

Liitteet

1 Johdanto

Opinnäytetyössä ohjelmoidaan mahdollisimman monipuolinen tekstistä kuvaksi-generaattori, joka luo digitaidetta OpenAI-palvelua hyödyntäen. Generaattorille annetaan haluttu tekstinpätkä ja sen jälkeen valitaan kategoria, joka tyylittelee kuvan halutun mukaisesti.

Generaattori alkaa työstämään kuvassa yleensä kahta erilaista mallia. Ensimmäinen niistä on generoiva malli, joka lisää kuvaan uusia kerroksia, joissa näkyy koko ajan enemmän haluttuja piirteitä. Toinen malli tarkistaa kuvaa ja raportoi ensimmäiselle mallille, onko se luonut tunnistettavasti pyydetyn asian. Toisen mallin palautteen avulla ensimmäinen malli yrittää parantaa edelliskerralla tekemäänsä jälkeä. Mallit toistavat näitä työvaiheita, kunnes haluttu tulos on saavutettu.

Tekstistä kuvaksi-generaattori on loistava työkalu kenelle tahansa, joka haluaa saada inspiraatiota tai uusia näkemyksiä esimerkiksi työprojekteihin. Siksi näenkin, että ammatilliset yritykset että taiteen harrastajatkin voisivat hyödyntää tätä uusissa hahmoissa ja ympäristöissä.

Pohdin opinnäytetyössäni seuraavia asioita:

- Miten tekstistä kuvaksi-generaattori toimii?
- Miten monipuolisesti tekoäly piirtää?
- Kuinka moni kokee tekemäni generaattorin kuvan yhteensopivaksi tekstin kanssa?

2 Ohjelmiston idea

2.1 Ohjelmiston ominaisuudet

Ohjelmisto luo kuvan vain annetusta tekstistä. Sille ei siis voi syöttää kuvia, jonka päälle halutaan piirtää uudestaan. Sillä ohjelmisto ei hyödynnä DALL-E-2:n kuvan muokkaus ominaisuuksia. Generaattorilla pystyy luomaan vain yhden kuvan kerrallaan. Lopputulos tukee vain 256×256 resoluutiota.

Ohjelmisto mahdollistaa myös kuvien generoimisen halutulla tyylillä. Vaihtoehtoja on kaiken kaikkiaan viisi. Eli kun hakukenttään kirjoitetaan esimerkiksi sana "Cat" ja valitaan tyyliksi "Black&White". Lopputulokseksi rakentuu mustavalkoinen kuva, jossa ilmenee jollakin tapaa kissa. Generaattori ymmärtää myös lauseita, eli sillä voidaan hakea mitä vaan maan ja taivaan väliltä. Kun rajoja ei ole, voidaan hakea esimerkiksi "An astronaut riding a horse", lopputulokseksi saat siis astronautin, joka ratsastaa hevosella. Jos tehdään uudestaan täysin samanlainen haku, generaattori tekee kuitenkin aina erilaisen kuvan.

Kuva 1 Esimerkki kuva

Kuvan esimerkeissä käytettiin DALL-E-2 tekstistä kuvaksi- generaattoria ja tyyliksi on valittu "Pencil drawing", eli lyijykynällä piirretty.

2.2 Sovelluksen toteutus

Ohjelmointikielenä on React. Se toimii generaattorin runkona, johon kuuluu ohjelmiston ulkoasu ja sen Async-funktio. Tämä funktio antaa ohjelmistolle lupauksen jostakin tulevasta, jota jäädään odottamaan Await-funktiolla. DALL-E-2 vastaanottaa Async-pyynnön, jonka jälkeen se lähettää valmiin kuvan takaisin ohjelmistolle näytettäväksi sille ennalta määrätyssä kohdassa.

Versionhallintaan käytetään Github-palvelua, josta generaattoria tullaan jakelemaan toistaiseksi. Github sivu mistä projektin löytää on https://github.com/nicozez1. Ohjelmistoon on saattanut tulla muutoksia opinnäytetyön jälkeen.

Projektinhallinnassa käytetään Scrum-mallia. Scrum-mallia hyödyntämällä ohjelmisto saadaan ajallaan tarvittavan hyvään malliin. Opinnäytetyön kirjoittamista myös hallinnoidaan scrum-mallilla, jotta se menee ohjelmiston kanssa yhtä matkaa.

Päiväkirjaan on ohjelmoinnin ohella kirjattu, että mihin kaikkiin ongelmiin projektissa törmättiin ja miten ne ratkaistiin.

2.3 Kuvageneraattorit

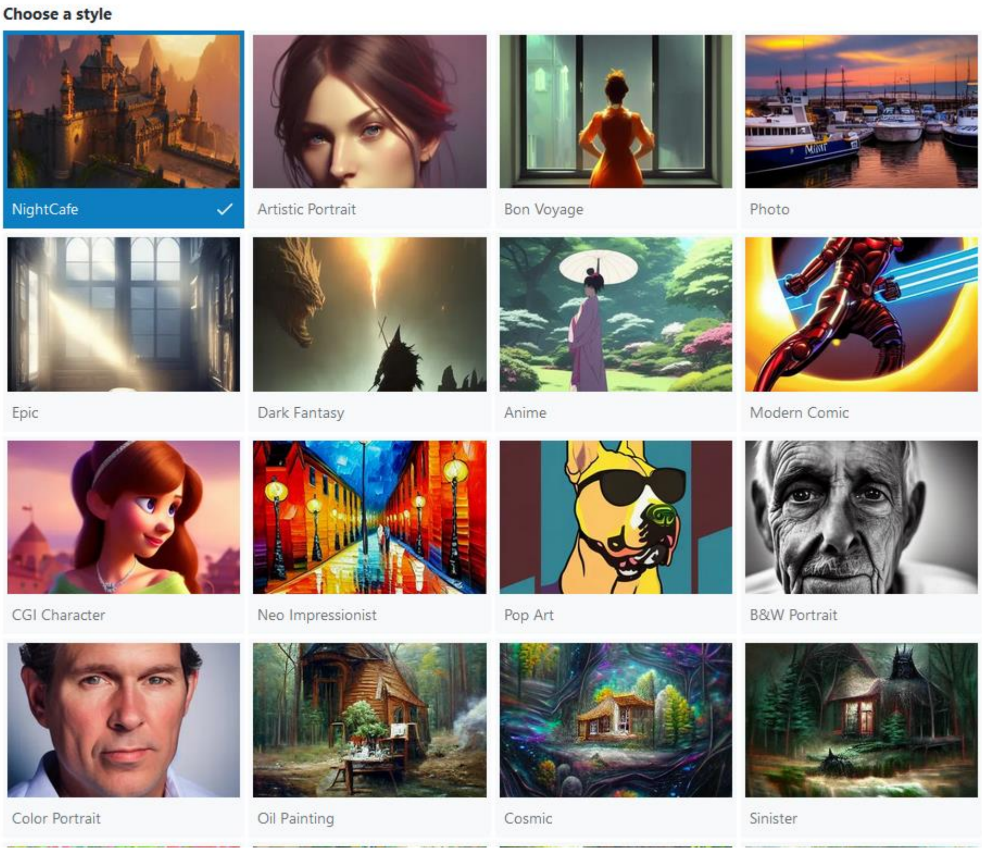
Kuvageneraattoreita on tehty jo vuosien ajan, vaikka se saattaa olla monelle uusi käsite. Nämä generaattorit nimensä mukaisesti siis generoivat kuvia tekoälykirjastoja hyödyntäen. Lopputulokseksi saadaan aina kuva, mutta syöte voi olla eri. Osa kuvageneraattoreista toimii tekstisyötteellä. On myös kuvageneraattoreita, jotka piirtävät oma kuvan päälle. Sivustoilla, jotka pyörittävät kuvageneraattoreita, on omat toimintaperiaatteensa. Yleensä sivustot ovat maksullisia, mutta usein tarjoavat ilmaisia poletteja eli virtuaalista valuuttaa.

2.3.1 Nightcafé

Nightcafé-sivustolla on mahdollista valita viidestä eri generaattorista, joilla halutaan luoda kuvia. Niihin kuuluvat tekstistä kuvaksi generaattorit Stable, DALL-E-2, Coherent ja Artistic. Sivulta löytyy myös Tranfer, joka piirtää tekstin sijasta käyttäjän itse lataaman kuvan päälle.

Sivusto antaa päivittäin rekisteröityneille käyttäjille viisi ilmaista polettia. Yhden kuvan luominen maksaa yhdestä poletista ylöspäin, riippuen halutusta resoluutiosta. (Nightcafe, n.d.-c.)

Kuva 2 Kuvassa on osa Nightcafe sivuston DALL-E-2 generaattorin 29. tyylivaihtoehdosta.



2.3.2 Midjourney

Midjourney on generaattori, joka pyörii ilmaisessa Discord-sovelluksessa. Tämä tekee siitä erilaisen kilpailijoihin verrattuna. Sillä Discord on ohjelmisto, jossa ihmiset pystyvät soittamaan puheluita ja kirjoittamaan toisilleen verkon välityksellä. Midjourneyn toiminta perustuu täysin tekstisyötteeseen, jolla määritettään tekoälylle raamit kuvan tekemiselle. (Midjourney, n.d.-c.)

Midjourneyta pääsee kokeilemaan ilmaiseksi muutaman kerran. Sen voi myös ostaa kuukaudeksi 12 eurolla, jolloin saa 200 minuuttia kuvien prosessointiaikaa. (Midjourney, n.d.-c.)

2.3.3 DreamStation

DreamStation perustuu Midjourneyn tavoin tekstisyötteeseen. Sivusto tarjoaa myös säädettäviä asetuksia. Yksi näistä on suosittu Step-asetus, jolla päätetään kuinka monta vaihetta generaattori käyttää kuvan tekemiseen. (DreamStudio, n.d.-c.)

DreamStation antaa ilmaiseksi 100 polettia uusille rekisteröityjille. Nämä riittävät noin 500 kuvaan, joiden resoluutio on 512x512. (DreamStudio, n.d.-c.)

Kuva 3 Nämä kuvat on luotu hakusanalla “A dinosaur dances ballet wearing a dress”.

3 Työkalut

3.1 Ohjelmointikielet

Jokaisen verkkosivun, puhelimen ja teollisuusrobotin käyttöjärjestelmä on luotu ohjelmointikielellä, jopa pakettiautomaatit toimivat samalla periaatteella. Listaa voidaan jatkaa loputtomiin. Jotta voidaan ohjelmoida kuvageneraattori, tarvitaan siis myös ohjelmointikieliä. Tässä projektissa käytettiin JavaScript-, React- ja Node.js-kieliä.

3.1.1 JavaScript

JavaScript on Netscapen kehittämä ja se julkaistiin jo vuonna 1995 Mocha-nimisenä. Tämän jälkeen nimi vaihdettiin LiveScriptiksi. Lopulta LiveScript liittoutui Sun Microsystemsin kanssa yhteen ja nimeksi tuli JavaScript. Tätä oli helpompi markkinoida, sillä Sun Microsystems on kehittänyt Java-kielen. (Pluralsight, n.d.-c.)

Tähän ohjelmointikieleen on tehty paljon ohjelmistokehyksiä, joista yksi esimerkki on myös tässä projektissa käytetty React. Eli eri ohjelmointikielet ovat käyttäneet JavaScriptiä runkona. Koska lähes kaikki selaimet tukevat JavaScriptia, se on yleisin käytetty ohjelmointikieli web-sivujen ohjelmoinnissa. (Pluralsight, n.d.-c.)

3.1.2 React

React on Meta Platforms inc. (entinen Facebook) kehittämä avoimen lähdekoodin JavaScript-kirjasto. Se julkaistiin ensimmäisen kerran vuonna 2013. Vuonna 2017 React julkaistiin uudestaan, MIT-lisenssillä ja ilman Facebookin patentteja. MIT-lisenssi mahdollisti sen, että lähdekoodia saa tarkastella, käyttää ja muokata kuka vain ja mihin tarkoitukseen tahansa, ilman että rikkoo tekijänoikeuslakia. Edellytyksenä on, että lisenssin teksti on sisällytetty lähdekoodiin. React on tarkoitettu lähinnä käyttöliittymien ja hallintapaneeleiden kehittämiseen, jotka ovat niitä ohjelmiston osia, jotka näkyvät käyttäjälle. (Meta Platforms, n.d.-c.)

3.1.3 Node.js

Node.js:n kehitti Linux Foundation ja se julkaistiin vuonna 2009. Ohjelmoinnissa on käytetty C-, C++- ja JavaScript-ohjelmointikielillä. Se on avoimen lähdekoodin sovellusohjelma, joka ei ole sidoksissa mihinkään tiettyyn alustaan. Node.js siis ajaa JavaScript-koodia ja se mahdollistaa koodin suorittamisen suoraan palvelimella. (netguru, n.d.-c.)

3.2 Kuvatekoäly DALL-E-2

DALL-E-2 on julkaistu vuonna 2022. Se on OpeanAI:n kehittämä tekoälyohjelma, joka luo kuvia annetuista teksteistä. DALL-E-2 käyttää GPT-3 tekoälyä, joka muuntaa syötetyn tekstin tekoälylle ymmärrettäväksi. Siinä on 3.5 miljardia parametria, jotka on ohjelmoitu oppimaan, miten yhdistetään tekstit oikeiden kuvien kanssa. (OpenAI, n.d.-c.)

DALL-E-2 käyttää kuvien generoimisessa kahta erilaista mallia. Ensimmäinen niistä on generoiva malli, joka lisää kuvaan uusia kerroksia, joissa näkyy koko ajan enemmän haluttuja piirteitä. Toinen malli tarkistaa kuvaa ja raportoi ensimmäiselle mallille, onko se luonut tunnistettavasti pyydetyn asian. Toinen malli antaa palautetta ensimmäiselle mallille, joka taas parantaa edelliskerralla tehtyä jälkeä. Mallit toistavat näitä työvaiheita, kunnes haluttu lopputulos on saavutettu. (Ryan O’Connor, 19.4.2022.)

DALL-E-2:sta pääsee käyttämään sen omilla sivuilla, johon tarvitsee luoda tunnukset. Generaattori luo aina yhdestä tekstistä neljä kuvaa. Yksi neljän kuvan generointi maksaa yhden krediitin, joita sivusto tarjoaa 50 kappaletta ilmaiseksi. Krediittejä saa lisää kuukausittain 15 kappaletta, mutta niitä voi halutessaan ostaa 15 dollarilla 115 kappaletta. Vaihtoehtoinen tapa on kirjautua OpenAI:n konsoliin, jossa voidaan luoda API Key. API Key on avain, jolla palvelu voi tunnistaa käyttäjän tai projektin. Tätä hyödyntämällä voidaan ohjelmoida ohjelmisto, joka pyytää DALL-E-2 generaattoria luomaan kuvan. API Key generointi tarjoaa kolme eri hinnoittelua, jotka riippuvat kuinka suuren kuvan tekee. OpenAI tarjoaa uusille käyttäjille 18 $ kokeilu rahaa, jonka voi käyttää mihinkä vaan heidän API Key

palveluun. Tässä projektissa yksi kuva, jonka resoluutio on 256X256 maksaa 0.016 $. (OpenAI, n.d.-c.)

3.3 Editori VSCode

VSCode on vuonna 2015 Microsoftin julkaisema avoimen lähdekoodin tekstieditori. Se on luotu helpottamaan ohjelmointia. Se tukee monia lähdekooditiedostoja, kuten .HTML, .JS, .CSS, PHP, .JAVA ja .PY. Sen lisäksi siinä on tuki virheenkorjauksille. Nämä virheenkorjaustyökalut näyttävät virheet koodissa ja ehdottavat korjauksia siihen. Tämä työkalu mahdollistaa myös koodin rikkoutumispisteiden asettamisen. (Microsoft, n.d.-c.)

VSCode sisältää IntelliSense-ominaisuuden, joka taas täydentää koodin älykkäästi. Kun kirjoitetaan esimerkiksi HTML kielessä ”<p>”, IntelliSense tunnistaa tämän ja tarjoaa automaattisen täydennyksen loppuun ”</p>”. Ominaisuus tarjoaa myös koodiviitteitä. Eli kun koodiin kirjoitetaan pelkkä ”<”. IntelliSense tarjoaa heti vaihtoehtoja, miten sulkea tagi. (Microsoft, n.d.-c.)

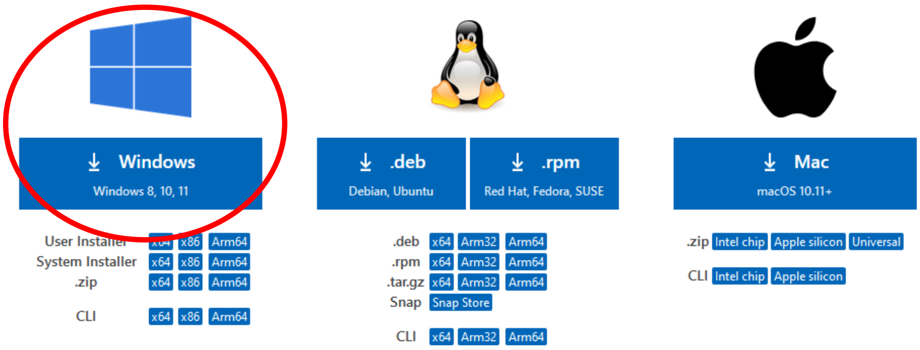
4 Kehitysympäristön asennus

Tässä luvussa käydään läpi tarvittavien ohjelmistojen asennus alusta loppuun. Sillä kun halutaan ohjelmoida, täytyy olla oikea kehitysympäristö halutun lopputuloksen aikaansaamiseksi. Asennusohjeet on tehty Windows-käyttöjärjestelmällä.

4.1 VSCoden asennus

Ensimmäisenä asennetaan VSCode, jolla kirjoitetaan ohjelman koodi. Asennustiedoston löytää seuraavasta linkistä: https://code.visualstudio.com/Download. Linkistä aukeaa sivu, josta valitaan Windows asennustiedosto.

Kuva 4 VSCode asennustiedoston lataussivu.



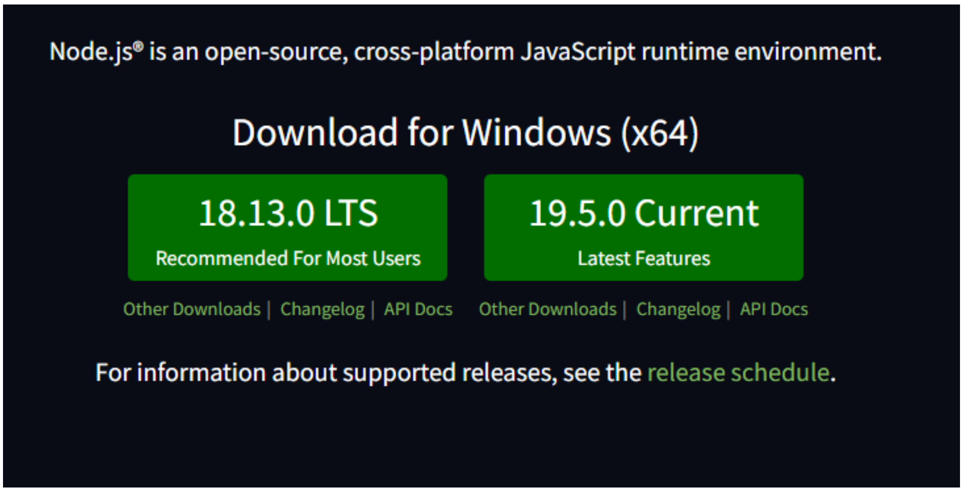
Asennus aloitetaan painamalla hiiren oikeaa näppäintä asennustiedoston päällä, jonka jälkeen painetaan ”Suorita järjestelmänvalvojana” painiketta. Seuraavaksi näytölle ilmestyy ”Sallitaanko tämän sovelluksen tehdä muutoksia tähän laitteeseen?” ikkuna, johon valitaan Kyllä-painike. Tämän jälkeen valitaan haluttu kieli ja valitaan OK-painike. Kielivalinnan jälkeen tulee lisenssin hyväksyntä ruutu. Hyväksytään käyttöehdot ja edetään Next-painikkeesta. Seuraavaksi tulee ”vaihtoehtoisia asennuksia” ruutu. Tähän ei tehdä muutoksia, joten valitaan Next-painike. Viimeiseksi tulee asennusruutu, joka kertoo sovelluksen olevan valmis asennettavaksi. Valitaan Install-painike, josta asennus alkaa. Kun

asennus on valmis, tulee ”Completin Visual Studio Code Setup Wizard” ikkuna. Valitaan Finish-painike, jonka jälkeen ohjelma aukeaa ja asennus on valmis.

4.2 Node.js asennus

VSCode asennuksen jälkeen asennetaan Node.js. Sillä luodaan ja suoritetaan React-ohjelmistoja. Aloitetaan hakemalla Node.js asennustiedosto seuraavasta osoitteesta: https://nodejs.org/en/. Linkistä aukeavalta sivulta valitaan vasemmanpuoleinen versio. Tämä on suositeltu versio suurimmalle osalle käyttäjistä.

Kuva 5 Valitaan vasemmanpuoleinen versio.



Kun lataus on valmis. Tupla klikataan juuri ladattua tiedostoa, jolla asennus alkaa. Tupla klikkauksen jälkeen ruudulle aukeaa tervetuloa ruutu. Painetaan Next-painiketta. Seuraavaksi näytölle tulee lisenssin hyväksyntä ruutu. Hyväksytään käyttöehdot, jonka jälkeen painetaan Next-painiketta. Lisenssin hyväksynnän jälkeen valitaan mihin Node.js asentuu. Tähän ikkunaan ei tehdä muutoksia, joten jatketaan painamalla Next-painiketta. Seuraavassa ikkunassa voi muokata asennusta. Tähänkään ikkunaan ei tehdä muutoksia, jolloin edetään painamalla Next-painiketta. Asennuksen muokkaus ikkunan jälkeen tulee lisäosa asennuksia kyselevä ikkuna. Koska projektissa ei tarvita näitä lisäosia, ei tehdä muutoksia tähän ruutuun ja edetään painamalla Next-painiketta. Lisäosa asennus ruudun jälkeen tulee ”Valmis asennettavaksi” ruutu. Tässä ruudussa painetaan Install-painiketta, josta asennus alkaa. Asennuksen alettua ruudun alareunaan ilmestyy vilkkuva kilpi. Kun kilpeä klikkaa, aukeaa ”Sallitaanko tämän sovelluksen tehdä muutoksia tähän laitteeseen?”

ikkuna. Juuri auenneeseen ikunnaan painetaan Kyllä-painiketta, josta asennus alkaa. Asennuksen valmistuttua tulee ”Completed the Node.js Setup Wizard” ikkuna. Tämän voi sulkea Finish-painikkeesta.

4.3 React-ohjelmiston luominen

Kun Node.js ja VSCode on asennettu, aloitetaan ohjelmiston luonti. Ensimmäisenä täytyy luoda kansio, jonne ohjelma tehdään. Kansion nimessä ei saa olla isoja kirjaimia, koska se rikkoo React-ohjelmiston nimi sääntöjä. Seuraavaksi avataan VSCode, jolla kirjoitetaan ohjelman koodi. Kun tekstieditori on auki, avataan juuri luotu kansio. Vasemmasta yläkulmasta löytyy File-painike, jota painamalla ilmestyy valikko. Tästä valikosta valitaan Open Folder -painike.

Kansion avaamisen jälkeen avataan terminaali, yläreunasta löytyvästä Terminal-painikkeesta. Tämän jälkeen aukeaa valikko, josta valitaan New Terminal -painike. Terminaalin saa myös auki näppäin yhdistelmällä ”Ctrl+Shift+Ö”. Terminaali aukeaa VScoden alareunaan. Siinä lukee ”PS”, jonka jälkeen tulee nykyisen kansion polku.

Kun terminaali on auki, voidaan kirjoittaa ”npx create-react-app ./” komentokehote siihen. Tämä komentokehote luo React ohjelmiston pohjan tähän kansioon. Kehotteen ”./” kohdan tilalle voidaan antaa nimi, esimerkiksi ”dall_e_generator”. Nimeämisessä tulee ottaa huomioon, ettei React tue ohjelmiston nimessä isoja kirjaimia. Nimen antamisen jälkeen Node.js luo kansion, jonne se asentaa ohjelmistopohjan. Tätä hyödyntämällä voidaan luoda helposti useita React-ohjelmistoja yhden kansion alle.

Tämän jälkeen painetaan Enter-näppäintä ja terminaalin valkoisen palkin pitäisi mennä toiselle riville. Palkin rivi saattaa olla vähän aikaa tyhjä. Hetken päästä siihen ilmestyy ”Creating a new React app in” teksti. Kun terminaaliin tulee teksti ”Happy hacking!” on ohjelmiston pohja asentunut. Ohjelmiston pohjan pystyy testaamaan myös kirjoittamalla ”npm start” komentokehotteen terminaaliin. Komennolla pitäisi aueta uusi ikkuna, jossa pyörii React-kielen logo.

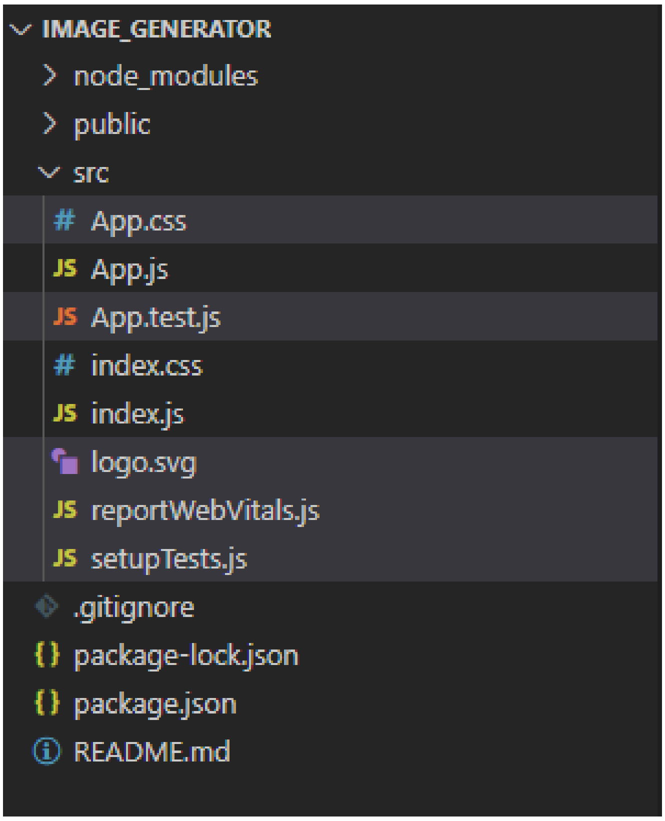
5 Kuvageneraattorin toteutusvaiheet

Tässä luvussa käydään läpi koko projektin ohjelmiston ohjelmoiminen. Ensimmäisenä putsataan React-pohjasta valmiit ohjelmakoodit, jonka jälkeen ohjelmoidaan kaikki mitä tarvitaan DALL-E-2:sen kuvapyynnön valmistelemiseen. Tämä sisältää käyttäjä syötteen, ulkoasun ja kuvatyyli nappuloiden ohjelmoimisen. Tämän jälkeen perehdytään, miten liitetään DALL-E-2:sen API Key ja ASYNC-pyyntö. Kun ohjelma on ohjelmoitu, testataan vielä toimivuus muutamalla erilaisella tekstillä.

5.1 React-pohjan valmistelu

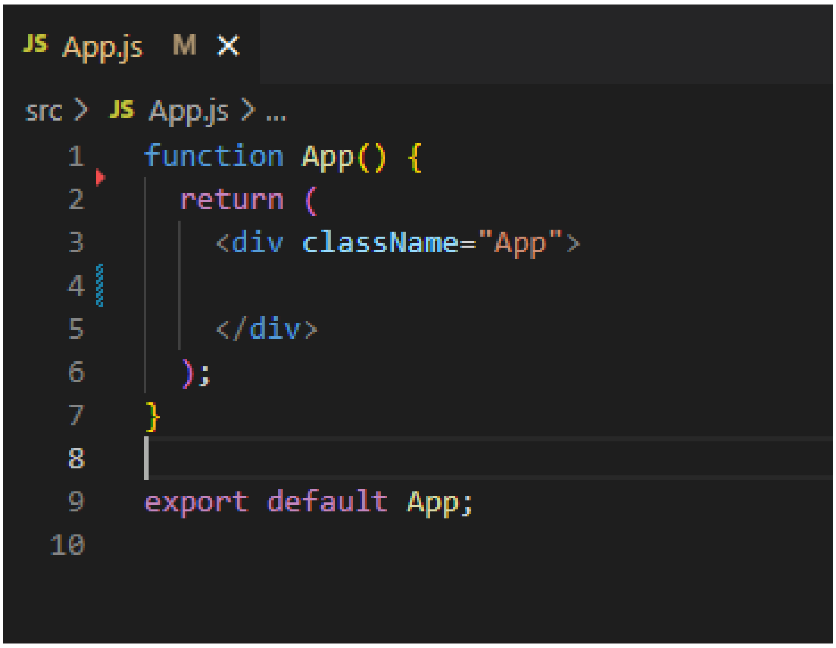
Tässä projektissa haluttiin tyhjä React-pohja, mihin ohjelmoida. Poistetaan React-pohjasta valmiit tyyli- ja testitiedostot, jotka ovat maalattuna alla olevassa kuvassa.

Kuva 6 React pohjasta poistettavat tiedostot.



Seuraavaksi avataan App.js ja index.js tiedostot. Niistä poistetaan osa koodista ja jäljelle jää Ohjelmakoodi 1:sen ja Ohjelmakoodi 2:sen rivit. App.js tiedostoon kirjoitetaan suurin osa tulevista koodeista, joihin kuuluu ulkoasu ja toiminnalliset koodit.

Ohjelmakoodi 1 App.js tiedostoon jääneet koodit.



Index.js tiedosto on node.js tiedosto, johon voisi kirjoittaa myös koko ohjelmiston. Tässä työssä index.js tiedosto kutsuu vain App.js tiedostoa. Ohjelmakoodi 2:sen rivillä kuusi ReactDOM-funktio luo polun index.html tiedoston div-elementtiin, jonka id on root. Tämä data tallennetaan const root -muuttujaan, jonka avulla piirretään näkyviin App.js tiedoston lopputulos. Lopputuloksen piirtäminen on ohjelmoitu Ohjelmakoodi 2:sen rivien seitsemän ja yksitoista välissä.

Ohjelmakoodi 2 index.js tiedostoon jääneet koodit.

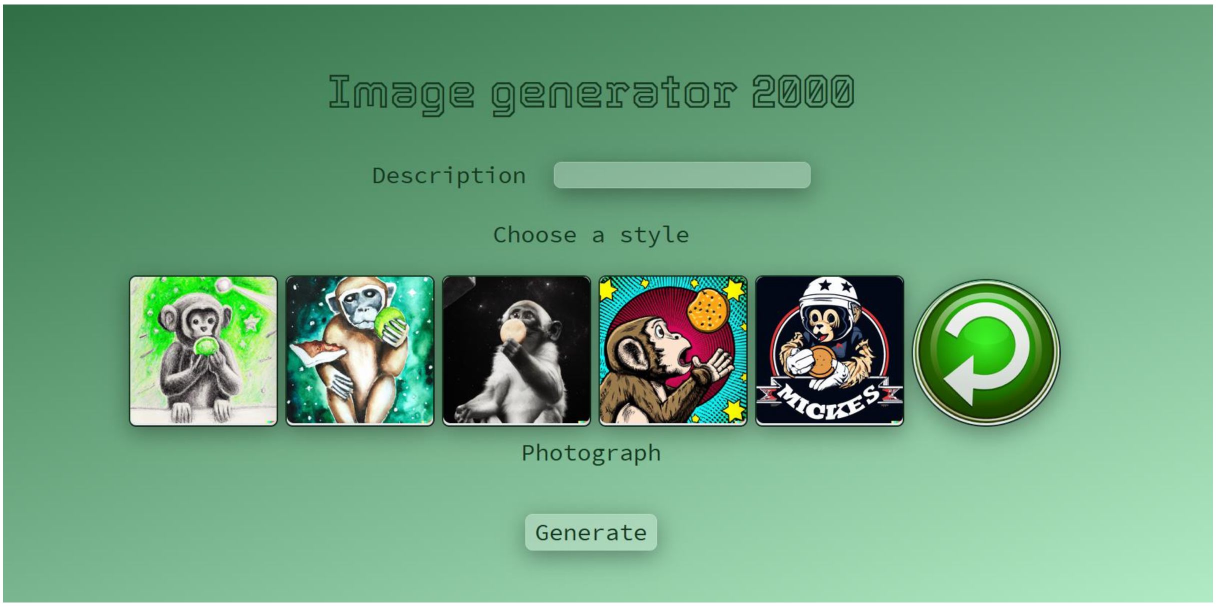
JS index.js M ×

src > JS index.js > ...

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

```

5.2 Käyttöliittymän ohjelmointi

Kuva 7 Ohjelmiston ulkoasu



Index.css eli tyylitiedostoon lisätään Liite 2 koodit, jotka antavat ohjelmistolle ulkoasun. Ulkoasu on yksinkertainen, mutta se sisältää kuitenkin nykypäivänä suositut pyöristetyt reunat ja hoover-toiminnot nappuloissa. Taustaväriksi valittiin vihreän eri sävyjä liukuvärinä tummasta vaaleaan. Generointinappula on vaaleanvihreä, joka muuttuu tummanvihreäksi hiiren osuessa siihen. Tekstinsyöttö kentän ja nappuloiden ympärille on lisätty hieman

varjostusta, jotta sivusto saisi enemmän ulottuvuuksia. Kaikkien nappuloiden reunukset on vaihdettu tummanvihreäksi, jotta ne sopisivat paremmin kokonaiskuvaan. Kirjasintyyli pidettiin yksinkertaisena ja helppolukuisena, joten sivulle valittiin googlen fonteista Source Code Pro ja Tourney. Tourney toimii otsikon fonttina ja muissa teksteissä on käytetty Source Code Pro -fonttia. Kuvatyylin valintanappuloihin tulee kuvat, jotka esittävät mahdollisen lopputuleman. Yksi nappuloista on reset nappula, jolla voidaan nollata kuvatyyli. Kuvassa 7 on lopputulos, joka tulee ohjetta noudattamalla.

Ohjelmakoodi 3 Nappuloiden ohjelmointi

```
75        <div className="image-input-pair">
76          <button className="button">
77            <img src={logo3} className="image" alt="Pencil drawing style" onClick={pencilDrawing}/>
78          </button>
79          <button className="button">
80            <img src={logo} className="image" alt="Watercolor style" onClick={watercolor}/>
81          </button>
82          <button className="button">
83            <img src={logo2} className="image" alt="Photograph style" onClick={photograph}/>
84          </button>
85          <button className="button">
86            <img src={logo4} className="image" alt="Pop Art style" onClick={popArt} />
87          </button>
88          <button className="button">
89            <img src={logo5} className="image" alt="NHL team emblem design style" onClick={NHL} />
90          </button>
91          <button className="buttonReset" >
92            <img src={logo6} className="imageReset" alt="Image style reset" onClick={noStyle} />
93          </button>
94        </div>
95        <p>{imageStyleText}</p>
96        <button className="main-button" onClick={generateImage}>
97          Generate
98        </button>
```

Index.css tiedoston muokkaamisen jälkeen vaihdetaan Ohjelmakoodi 1 rivin kolme div-elementin aloitus ja rivin viisi div-elementin lopetus main-elementiksi. Tämä kertoo ohjelmalle, että tässä osassa koodia on ohjelmiston tärkeät asiat. Tämän jälkeen luodaan ohjelmiston generointinappula Ohjelmakoodi 3 mukaisesti. Generointi nappula luodaan rivien 96 ja 98 välissä. Generointinappula kutsuu sitä klikatessa generateImage funktiota, joka käydään myöhemmin läpi. Seuraavaksi ohjelmoidaan tyylinvalinta nappulat, joita tulee yhteensä kuusi kappaletta. Luodaan Ohjelmakoodi 3 rivin 75 mukainen div-elementin aloitus ja sille lopetus. Tämä koodi rivittää nappulat vierekkäin ja pitää ne mahdollisimman keskitettynä. Näiden div-elementtien väliin ohjelmoidaan Ohjelmakoodi 3 rivien 75 ja 98 väliset nappula koodit. Nappulan sisällä on img-elementti, joka tekee nappulasta kuvan

näköisen. Img-elementti etsii tästä ohjelmasta logo-muuttujan, joka selitetään seuraavassa kappaleessa paremmin. Logo-muuttuja määrittää img-elementille sen kuvan, joka näytetään image-tyylittelyllä. Kun kuvaa klikataan se aktivoi onClick-tapahtuman, joka kutsuu sen hakasulkeisiin lisättyä funktiota. Esimerkiksi Ohjelmakoodi 3 rivillä 89 kutsutaan NHL funktiota, joka käydään läpi myöhemmin. Ohjelmakoodi 3 rivillä 92 ohjelmoidaan img-elementti, jolla on imageReset ulkoasu. Tämä nappula antaa käyttäjälle mahdollisuuden nollata tyylivalinta. Nappuloiden ohjelmoinnin jälkeen lisätään vielä p-elementti Ohjelmakoodi 3 rivin 95 mukaisesti. Tämä elementti hakee imageStyleText-muuttujan, joka päivittyy ruudulla aina tyylinappulaa painettaessa.

Ohjelmakoodi 4 Nappuloiden kuvien haku.

```
const logo = require("./monkey.png");
const logo2 = require("./monkey2.png");
const logo3 = require("./monkey3.png");
const logo4 = require("./monkey4.png");
const logo5 = require("./monkey5.png");
const logo6 = require("./reset.png")

function App() {
```

Aiemmin ohjelmoidut nappulat sisälsivät img-elementin, joka hakee kuvan logo-muuttujasta. React-kielessä kuvaa ei voida suoraan hakea paikallisesta kansiosta img-elementtiin. Kuvat pitää ensimmäiseksi hakea Ohjelmakoodi 4 mukaisesti require-funktiolla muuttujaan, jonka jälkeen näitä muuttujia voidaan kutsua img-elementissä. Nämä muuttujat ohjelmoidaan juuri ennen App-funktion aloitusta, jotta nämä ladataan heti ohjelman käynnistyksessä.

Ohjelmakoodi 5 Kuvan generoimiseen tarvittavat muuttujat.

```
function App() {
  const [userPrompt, setUserPromnt] = useState("");
  let number = 1;
  let size = "256x256";
  const [imageUrl, setimageUrl] = useState("");
  const [imageStyle, setimageStyle] = useState("");
  const [imageStyleText, setimageStyleText] = useState("");

```

Kun kuvatyylien kuvat on määritetty, voidaan luoda kuvan generointiin tarvittavat muuttujat Ohjelmakoodi 5 mukaisesti. Muuttujat userPrompt, imageUrl, imageStyle ja imageStyleText ovat useState-muuttujia, jotka päivittyvät aina kun niiden kanssa määritettyä set-funktiota kutsutaan. Muuttujat number ja size ovat let tyyppisiä muuttujia, eli niille on annettava heti jokin arvo. Tätä arvoa ei voida muuttaa enää myöhemmin ohjelmassa.

Ohjelmakoodi 6 Kuvatyyli valinnan funktiot.

```
const pencilDrawing = () => {
  setimageStyle(" pencil drawing");
  setimageStyleText("Pencil drawing");
};
const watercolor = () => {
  setimageStyle(" watercolor painting");
  setimageStyleText("Watercolor painting");
};
const NHL = () => {
  setimageStyle(" NHL team emblem design");
  setimageStyleText("NHL team emblem design");
};
const photograph = () => {
  setimageStyle(" photograph");
  setimageStyleText("Photograph");
};
const popArt = () => {
  setimageStyle(" pop art");
  setimageStyleText("Pop art");
};
const noStyle = () => {
  setimageStyle("");
  setimageStyleText("");
};
```

Seuraavaksi voidaan ohjelmoida kuvatyylin valintaa varten funktiot Ohjelmakoodi 6 mukaisesti. Nämä funktiot asettavat imageStyle-muuttujalle arvon kutsumalla setimageStyle-funktiota. Funktio setImageStyleText asettaa muuttujalle imageStyleText tekstin, joka näytetään käyttäjälle tyylinvalinnassa. Näitä funktioita kutsutaan Ohjelmakoodi 3 kuvan img-elementeissä.

Ohjelmakoodi 7 Generoidun kuvan näyttämisen ja syötteen antamisen koodi.

```
64 {imageUrl && <img src={imageUrl} className="image" alt="ai image" />}
65 <InputBox label={"Description"} setAttribute={setUserPromnt} />
66 <div className="image-input-pair">
```

Kuvatyylin asettamisen jälkeen ohjelmoidaan generoidun kuvan näyttäminen Ohjelmakoodi 7 rivin 64 mukaisesti. Tämän rivin alussa tarkistetaan, onko imageUrl-muuttujassa tekstiä. Jos muuttujassa on tekstiä, se lähetetään img-elementille näytettäväksi. Ohjelmakoodi 7 rivillä 65 tuodaan tekstinsyöttö kenttä, joka on ohjelmoitu valmiiksi toisessa tiedostossa. Jotta voidaan käyttää toisen tiedoston funktiota, se täytyy tuoda haluttuun tiedostoon. Tämä tapahtuu kirjoittamalla App.js tiedoston ensimmäiselle riville ”import {InputBox} from ”./InputBox”;”, joka tuo tiedostoon InputBox-funktion. InputBox-funktio ottaa vastaan label ja setAttribute nimiset argumentit. Argumentti label asettaa tekstinsyöttö kentän viereen halutun tekstin ja setAttribute lähettää setUserPromnt-funktion.

Ohjelmakoodi 8 InputBox.jsx tiedoston koodit

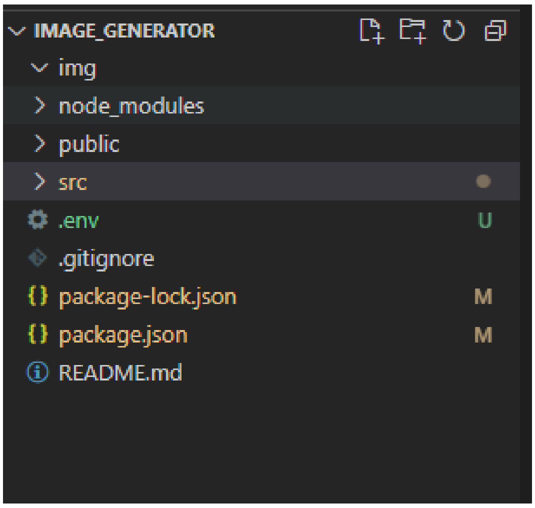
```
import React from "react";

export const InputBox = ({ label, setAttribute }) => {
  return (
    <div className="label-input-pair">
      <label className="label">{label}</label>
      <input
        className="main-input"
        onChange={(e) => setAttribute(e.target.value)}
      />
    </div>
  );
};

```

Seuraavaksi luodaan src-kansion alle InputBox.jsx tiedosto, johon ohjelmoidaan käyttäjänsyöte osio. Juuri luotuun tiedostoon kirjoitetaan Ohjelmakoodi 8 mukaisesti ohjelmakoodi. Tässä tiedostossa luodaan funktio InputBox, joka on laitettu näkyviin muille tiedostoille export-ilmoituksella. Funktiolle on asetettu label ja setAttribute parametrit, jotka se ottaa vastaan sitä kutsuttaessa. Funktion div-elementti sisältää label- ja input -elementin, jotka se rivittää samalle riville muotoilulla label-input-pair. Elementti label hakee funktiolle lähetetyn argumentin label, joka näytetään muotoilulla label. Tekstikenttä luodaan input-

elementillä, joka saa muotoilun main-input muotoilusta. Input-elementissä on onChange-tapahtuma, joka tarkkailee siinä tapahtuvia muutoksia. Kun Input-elementtiin kirjoitetaan, aktivoituu onChange-tapahtuma. Tämä lähettää uuden tekstin setAttribute-parametrillä userPrompt-muuttujaan. Kun käyttäjänsyötteen tekee tällä tavalla, voidaan syötteitä tehdä tarvittaessa useampi pienellä koodi määrällä. Tämä tarkoittaa, että tarvitsee vain tehdä uusi kutsu App.js tiedostossa. Uudelle kutsulle annetaan vain toiset argumentit, jolla luoda uusi käyttäjänsyöte.

5.3 DALL-E-2 liittäminen sovellukseen

Kuva 8 Luodaan .env tiedosto projekti kansioon.



Kun käyttöliittymä on ohjelmoitu, voidaan edetä DALL-E-2 API avaimen hakemiseen. Luodaan .env-tiedosto projektikansioon, joka on kuvassa kahdeksan image_generator. Tämän jälkeen avataan juuri luotu tiedosto ja kirjoitetaan siihen ”REACT_APP_API_KEY=” teksti. Env tiedostoon tallennetaan salaista dataa. Tähän dataan kuuluu salasanat ja tässä tapauksessa API key. Seuraavaksi mennään osoitteeseen https://openai.com/api/, johon tarvitsee kirjautua. Kun kirjautuminen on mennyt läpi ja ”Welcome to OpenAI” sivu on näkyvissä, painetaan oikeassa yläkulmassa olevaa tilin logoa. Logosta aukeaa valikko, josta valitaan ”View API keys” kohta. API keys sivulta valitaan Create new secret key -painike, joka avaa ponnahdusikkunan. Ponnahdusikkunassa näkyy generoitu API key, joka voidaan kopioida leikepöydälle vihreästä painikkeesta. Kun API key on kopioitu leikepöydälle, kopioidaan se env tiedoston ”REACT_APP_API_KEY=” kohdan perään. Kun API key on kirjoitettu env tiedostoon, voidaan se tuoda sieltä turvallisesti Ohjelmakoodi 9 mukaisesti.

Ohjelmakoodi 9 Tuodaan API key env tiedostosta App.js tiedoston muuttujaan.

```
const configuration = new Configuration({
  apiKey: process.env.REACT_APP_API_KEY,
});
```

Jotta Ohjelmakoodi 9 koodi toimisi, vaatii se openai kirjaston asennuksen. Kirjoitetaan terminaaliin ”npm install openai”, jolla kirjasto asentuu. Asennuksen jälkeen tehdään riville kaksi uusi import Ohjelmakoodi 10 mukaisesti. Kuvan rivillä kaksi tuodaan openai-kirjastosta Configuration- ja OpenAiAPI -oliot, joilla luodaan pyyntö DALL-E-2:lle. Ohjelmakoodi 9 kuvan muuttuja on Configuration-olio, joka sisältää API keyn. Olio on ohjelmoinnissa muuttuja, joka voi sisältää useamman muuttujan sisällään. Esimerkiksi voisi olla olio nimeltä kissa, jolla on muuttujat nimi, rotu ja ikä.

Ohjelmakoodi 10 Tuodaan openai kirjastosta Configuration ja OpenAiAPI oliot.

```
1 import { InputBox } from "./InputBox";
2 import { Configuration, OpenAIApi } from "openai";
3 import { useState} from "react";
```

Kun API key on hoidettu olioon, voidaan ohjelmoida itse DALL-E-2 kuva generointi pyyntö prosessi. Ensimmäiseksi luodaan OpenAIApi-olio Ohjelmakoodi 11 mukaisesti. Olion nimi on openai ja se saa configureration-oliolta API keyn. Seuraavaksi ohjelmoidaan kuvan generoimisesta generateImage async -funktio. Funktion määrittelyssä kerrottaan ohjelmistolle async tekstillä, että jotain ollaan pyytämässä. Seuraavaksi luodaan imageParameters-olio, jolla on kuvan generointiin tarvittavat muuttujat. Muuttuja prompt hakee userPrompt- ja imageStyle -muuttujat, jotka se yhdistää pilkun avulla kokonaiseksi lauseeksi. Oliolla on myös muuttuja n, joka hakee muuttujan number. Muutuja n kertoo, kuinka monta kuvaa halutaan generoida. Number-muuttuja muunnetaan vielä numeroksi parseInt-funktiolla, jolla voidaan saada tulevaisuudessa ei numeraaliset pyynnöt estettyä. Size-muuttuja hakee muuttujan size. Size-muuttuja kertoo ohjelmistolle, kuinka suuri generoitava kuva on. Olion luonnin jälkeen ohjelmoidaan itse kuvan generointi pyyntö Ohjelmakoodi 11 rivin 31 mukaisesti. Rivillä 31 luodaan muuttuja, joka odottaa vastausta openai-olion luomalta kuvan generointi pyynnöltä. Openai-olio luo pyynnön funktiolla

createImage, johon annettaan imageParameters-olio argumentiksi. Seuraavaksi luodaan urlData-muuttuja, joka hakee response-muuttujasta kuvan url-linkin. Viimeisenä kuvan linkki haetaan funktiolla setimageUrl, joka laittaa datan imageUrl-muuttujaan.

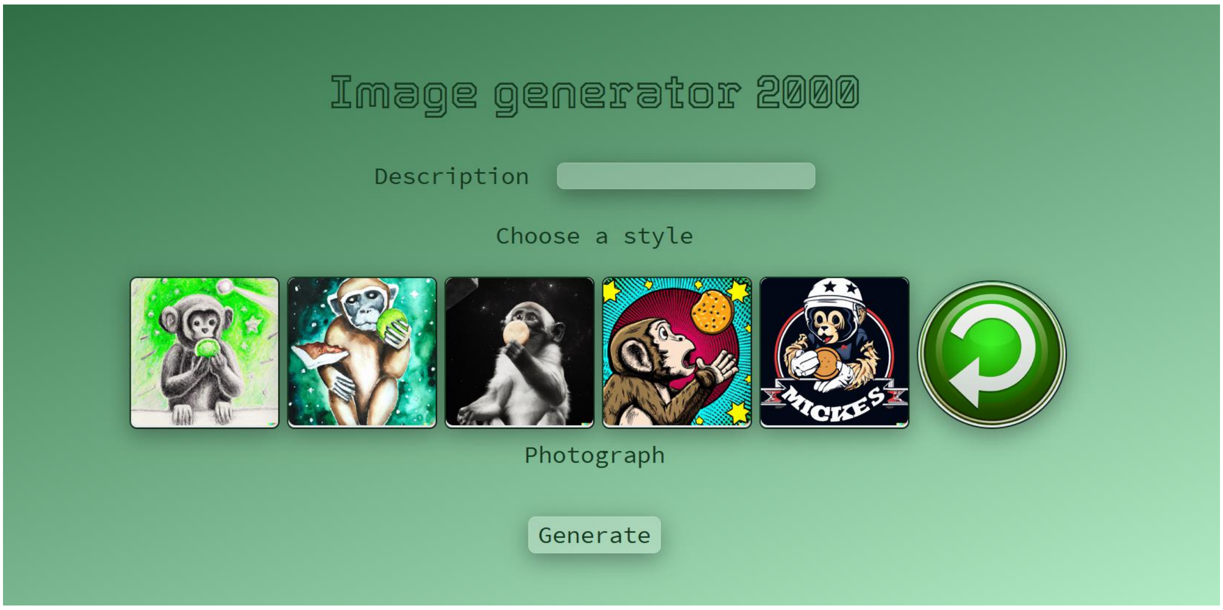
Ohjelmakoodi 11 DALL-E-2:sen kuva genrointi pyyntö.

```

const openai = new OpenAIApi(configuration);
const generateImage = async () => {
  const imageParameters = {
    prompt: userPrompt + "," + imageStyle,
    n: parseInt(number),
    size: size,
  };
  const response = await openai.createImage(imageParameters);
  const urlData = response.data.data[0].url;
  setimageUrl(urlData);
};

```

5.4 Testaus

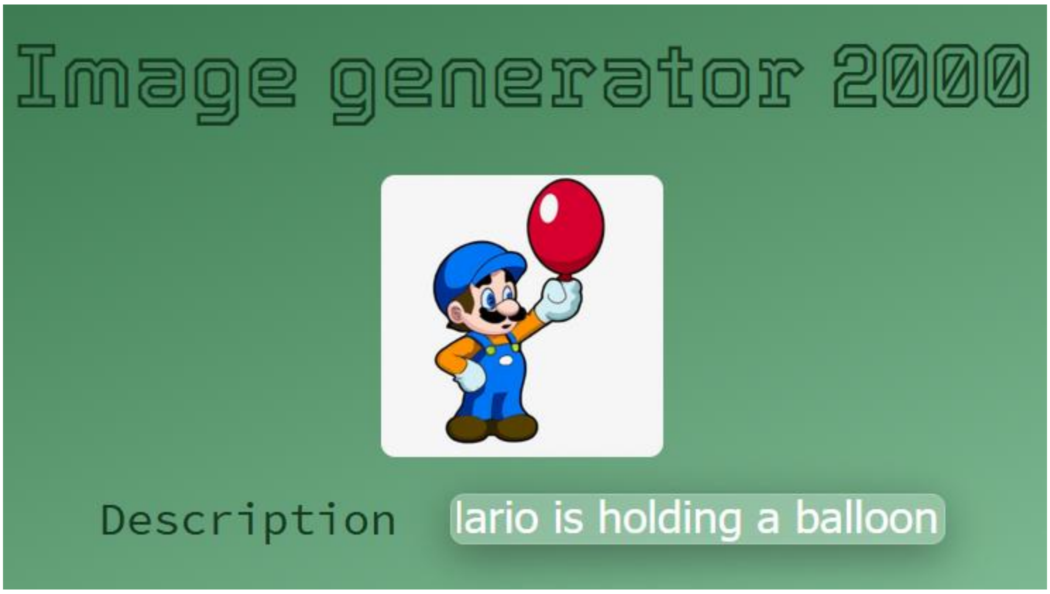
Kuva generaattorin valmistuttua, aloitetaan sen testaaminen. Ensimmäisenä testataan, että tyylinvalinta nappulaa painettaessa tulee aina oikea teksti. Kun valitaan valokuva tyyli, pitäisi Kuvan 8 mukaisesti ilmestyä Photograph teksti. Seuraavaksi testataan reset-nappulan toimivuus. Reset-nappulan tulisi nollata sen hetkinen tyylivalinta, eli tyylivalinta teksti katoaa ruudulta.

Kuva 9 Kuvassa valittu Photograph tyyli.



Kun jokaisen tyylinvalinta nappulan toimivuus on testattu, testataan itse kuvan generointi pyyntöä. Ensimmäisenä lähetetään pyyntö ilman mitään kuvatyyliä, jolloin lopputulos on sattumanvarainen. Tekstillä ”Mario is holding a balloon” saatiin kuvan 9 mukainen lopputulos. Kuvasta hahmotetaan helposti videopelihahmo Mario, joka pitelee ilmapalloa.

Kuva 10 Generointi pyyntö ilman tyylivalintaa



Seuraavaksi testataan kaikki tyylivalinnat samalla tekstillä, jotta voidaan verrata niiden lopputulemia. Kuvassa 10 on kaikki generointi pyynnöt vasemmalta oikealle samassa

järjestyksessä, kun ohjelman tyylivalinnat. Kuvasta voidaan huomata, että tyylivallinnat toimivat. Kaikissa kuvissa esiintyy taas Mario, joka pitelee ilmapalloa.

Kuva 11 Kaikkien tyylivalintojen generointi testit yhdessä kuvassa.

6 Kysely ja kyselyn tulokset

Image generator 2000 toimivuuden analysointia varten luotiin kaksi kyselyä. Tuloksien avulla pohditaan, kuinka moni kokee generaattorin kuvan yhteensopivaksi annetun tekstin kanssa. Ensimmäiseen kyselyyn vastasi kahdeksan ihmistä ja toiseen kyselyyn vastasi yhdeksän ihmistä.

Ennen kyselyä tutkittiin, onko tehty muita samankaltaisia kyselyitä. Aiheesta ei ole aikaisemmin tehty tutkimusta, joten tässä opinnäytetyössä ei voitu hyödyntää jo olemassa olevia tuloksia. Nämä kaksi kyselyä on tehty tätä opinnäytetyötä varten.

Kaikki kyselyn kuvat ovat ensimmäisiä tuloksia, jotka generaattori antoi. Tämä varmistaa, ettei kyselyn luojalla ole vaikutusvaltaa tuloksiin. Kyselyn kuvat on testattu googlen käänteisellä kuvahaulla, jotta varmistettiin ettei generaattori ole kopioinut kuvaa suoraan netistä.

6.1 Kyselyn tekeminen

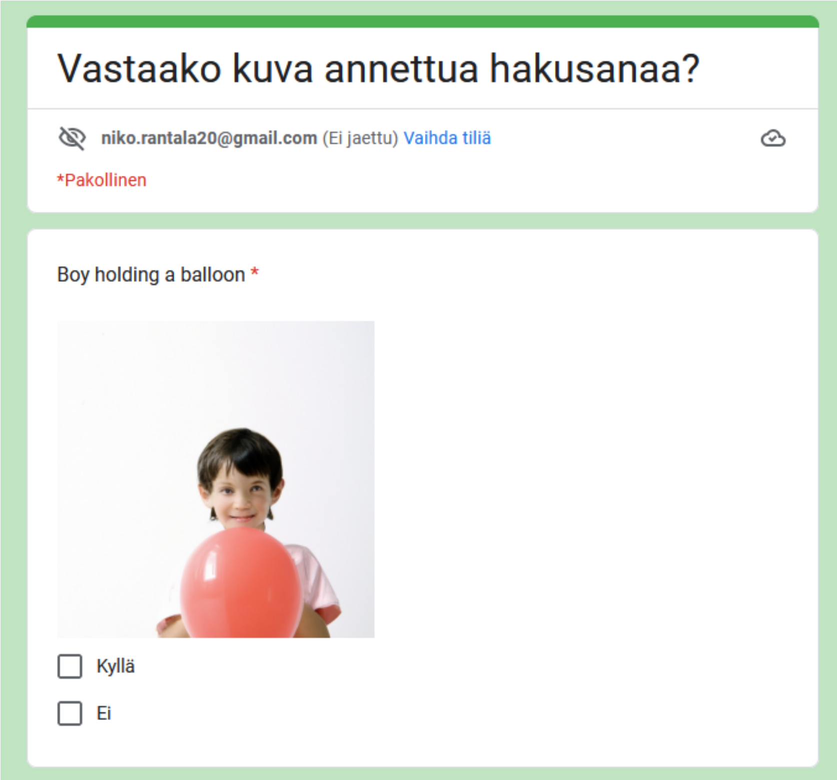
Kyselyä varten luotiin viisi erilaista kuvaa. Kuvien luomiseen käytettiin seuraavia tekstejä:

1. Boy holding a balloon.
2. Old woman ties a shoe.
3. Cat is driving a tractor.
4. Penguin between hamburger buns.
5. A robot donut.

Generointi tekstit tehtiin tarkoituksella hieman kieliopillisesti väärin ja ympäripyöreiksi, jotta nähdään osaako tekoäly generoida halutut kuvat virheistä huolimatta. Näille kuville ei myöskään annettu tyylivalintoja, ettei se vaikuttaisi lopputulokseen.

Kuvan 11 ensimmäisessä kyselyssä vastaajalta kysytään ”Vastaako kuva annettua hakusanaa?”. Kyselyyn vastaaja valitsee ruudun ”Kyllä” tai ”Ei”.

Kuva 12 Ensimmäinen kysely.



Kuvan 12 toisessa kyselyssä annetaan kuva, johon vastaajan pitää itse kirjoittaa teksti millä se olisi voitu generoida. Kyselyt annetaan eri ihmisryhmille, jotta saadaan mahdollisimman paljon näkemyksiä. Kun kaikki henkilöt ovat vastanneet, aloitetaan tulosten analysointi.

Kuva 13 Toinen kysely.

6.2 Kyselyn analysointi

6.2.1 Kuvien vastauksien analysointi

Kyselyjä pidettiin auki viikon ajan, jonka aikana ensimmäiseen kyselyyn vastattiin kahdeksan kertaa ja toiseen yhdeksän kertaa. Seuraavaksi käydään jokainen kuva läpi ja näiden vastauksien perusteella voidaan todeta, kuinka moni kokee generaattorin kuvan yhteensopivaksi annetun tekstin kanssa.

Kuva 14 Kyselyiden kuva yksi, jonka generointi tekstinä oli Boy holding a balloon.

Ensimmäisenä kuvana kyselyissä oli Kuvassa 13 esiintyvä poika, joka pitelee ilmapalloa. Ensimmäisessä kyselyssä seitsemän vastaajaa kahdeksasta koki kuvan vastaavaan annettua hakusanaa. Toisen kyselyn kaikissa vastauksissa ilmeni sana lapsi tai poika, jonka yhteyteen liitettiin ilmapallo ja pallo. Yhdeksästä vastauksesta kahdessa ei mainita pojan pitelevän palloa. Molempien kyselyjen vastauksien perusteella voidaan päätellä kuvan olevan tulkinnanvarainen, koska poika ei näkyvästi pitele ilmapallosta kiinni.

Kuva 15 Kyselyiden kuva kaksi, jonka generointi tekstinä oli Old woman ties a shoe.

Toisena kyselyjen kuvana oli Kuvassa 14 oleva iäkäs nainen, joka sitoo kengännauhaa. Ensimmäisessä kyselyssä yksi vastaaja kahdeksasta oli sitä mieltä, ettei kuva vastaa annettua hakusanaa. Toisessa kyselyssä yhdeksästä vastauksesta kolmessa ilmeni, että iäkäs ihminen

sitoo kengännauhaa. Kun taas kuusi vastaajaa vastasi, että kuvassa on nainen. Näihin vastauksiin voi olla syynä, ettei kuvassa olevan henkilön kasvoja paljasteta.

Kuva 16 Kyselyiden kuva kolme, jonka generointi tekstinä oli Cat is driving a tractor.

Kolmantena kuvana kyselyissä oli Kuvassa 15 oleva piirros kissasta, joka ajaa traktoria. Ensimmäisessä kyselyssä kaikki vastaajat vastasivat kuvan muistuttavan kissaa, joka ajaa traktoria. Toisen kyselyn kaikissa vastauksissa ilmeni kissa, joka on traktorissa. Tässäkin kuvassa on kuitenkin tulkinnanvaraa, koska yhdeksästä ihmisestä kahdeksan tulkitsi kissan ajavan traktoria ja yksi vastasi kissan vilkuttavan traktorista. Tämä johtuu todennäköisesti siitä, ettei kissan molemmat käpälät ole ratilla ja traktori ei selkeästi ole käynnissä.

Kuva 17 Kyselyiden kuva neljä, jonka generointi tekstinä oli Penguin between hamburger buns.

Neljäntenä kuvana oli Kuvassa 16 esiintyvä pingviini, joka on hampurilaisen välissä. Ensimmäisessä kyselyssä kaikki vastaajat olivat sitä mieltä, että kuva vastaa annettua hakusanaa. Toisen kyselyn kahdeksassa vastauksesta yhdeksästä ilmeni hampurilainen, jonka välissä on pingviini. Yksi oli vastannut, että kyseessä on ankka.

Kuva 18 Kyselyn kuva viisi, jonka generointi tekstinä oli A robot donut.

Viimeisenä kuvana kyselyissä oli Kuvassa 17 esiintyvä robotti donitsi. Kaikki ensimmäisen kyselyn vastaajat vastasivat, että kuva on robotti donitsin näköinen. Toisessa kyselyssä vastaukset sisälsivät donitsin ja robotin. Toisen kyselyn perusteella kuvaa voidaan tulkita

kahdella tapaa. Viisi vastaajaa vastasi kuvan muistuttavan robottia, joka näyttää donitsilta. Kun taas neljä vastaajaa näki donitsin, joka on koristeltu robotin näköiseksi.

6.2.2 Kyselyn yhteenveto

Ensimmäisen kyselyn tulokset nähdään Kuvassa 18. Tuloksista voidaan päätellä, miten valmiin generointi tekstin antaminen vaikutti vastaukseen. Sillä kun vastaaja näkee ”oikean” vastauksen, sitä ei luultavasti epäillä niin helposti. Suurin osa vastaajista valitsi kaikkiin kysymyksiin vastauksen ”Kyllä”. Ainoastaan kahteen ensimmäiseen kuvaan vastattiin kerran ”Ei”. Tämä luultavasti johtui siitä, että kuvat jättivät tiettyjä yksityiskohtia pois. Ensimmäisen kuvan poika ei pitänyt näkyvästi pallosta kiinni ja toisen kuvan kenkää ei voitu liittää naiseen.

Jälkikäteen ajateltuna ensimmäiseen kyselyyn olisi ollut hyvä lisätä ”Ei” vaihtoehtoon kysymys, että ”Miksi kuva ei vastaa annettua tekstiä?”. Jotta tuloksia olisi voinut tarkastella vielä yksityiskohtaisemmin.

Kuvan 18 ja 19 kyselyiden tulos taulukot tehtiin Google sheet palvelussa.

Kuva 19 Ensimmäisen kyselyn tulokset.

Generointi tekstit	Boy holding a balloon	Old women tie a shoe	A cat is driving a tractor	Penguin between hamburger buns	A robot donut
Vastaaja 1	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä
Vastaaja 2	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä
Vastaaja 3	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä
Vastaaja 4	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä
Vastaaja 5	Ei	Kyllä	Kyllä	Kyllä	Kyllä
Vastaaja 6	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä
Vastaaja 7	Kyllä	Kyllä	Kyllä	Kyllä	Kyllä
Vastaaja 8	Kyllä	Ei	Kyllä	Kyllä	Kyllä

Toisen kyselyn tulokset näkyvät kuvissa 19 ja 20. Ensimmäisen kuvan kahdessa vastauksessa kuvaa haettaisiin tekstillä ”Poika ja pallo”. Toisen kuvan kolmessa vastauksessa ei määritetty, että kyseessä olisi ollut iäkäs nainen. Kolmannessa kuvassa vain yksi vastasi, että kissa vilkuttaa traktorista ajamisen sijaan.

Kuva 20 Toisen kyselyn kolmen ensimmäisen kuvan tulokset.

Vastaajien kuvangenerointi tekstit			
	Kuva 1	Kuva 2	Kuva 3
Vastaaja 1	Boy and red balloon	Granny ties a shoe	A Cat waving inside a tractor
Vastaaja 2	Lapsi pitelee punaista ilmapalloa	Vanhus solmii kengännauhoja	Kissa ajaa värikkäällä traktorilla
Vastaaja 3	Poika joka pitelee punaista ilmapalloa	Vanha nainen solmimassa kengän nauhaansa	Piirretty kuva kissasta, joka ajaa traktoria
Vastaaja 4	Poika pitää palloa	Mummo solmii kenkiä	Kissa ajaa traktoria
Vastaaja 5	Lapsi pitelee punaista ilmapalloa	Vanhus sitoo ruskeita kengännauhoja	Kissa ajaa traktorilla
Vastaaja 6	Lapsi pitelee ilmapalloa	Mummo solmii kengännauhoja	Kissa ajaa traktoria piirrustus
Vastaaja 7	Nuori hymyilevä lapsi pitelee punaista ilmapalloa edessään. Valkoinen tausta	Vanha nainen kukkamekossa sitoo kengän nauhoja.	Valokuvarealistinen kissa ajaa piirretyssä maailmassa traktorilla.
Vastaaja 8	Poikalapsi pitelee punaista ilmapalloa	Mummo kumartuu sitomaan kengännauhoja	Piirustus kissasta ajamassa traktoria
Vastaaja 9	Pieni poika ja ilmapallo	Kengän sitominen	Kissa traktorissa maalaus

Kuvassa 21 nähdään kahden viimeisen kuvan vastaukset. Yksi vastaaja oli tulkinnut neljännen kuvan eläimen olevan pingviinin sijasta ankka. Viidennen kuvan viisi vastaajaa vastasi kuvassa olevan donitsi, joka on koristeltu robotin näköiseksi. Neljä vastasi kuvassa olevan robotti, joka näyttää donitsilta.

Kuva 21 Toisen kyselyn kahden viimeisen kuvan tulokset.

Vastaajien kuvangenerointi tekstit		
	Kuva 4	Kuva 5
Vastaaja 1	Penguin hamburger	Donut with robot decoration
Vastaaja 2	Pingviini hampurilaisen välissä	Donitsi-robotti
Vastaaja 3	Muovailuvaha pingviini hampurilaisen välissä	Suklaa donitsi joka on koristeltu robotin päällä
Vastaaja 4	Pingviini hampurilaissämpylän välissä	Robotti donitsi
Vastaaja 5	Duck sandwich	Robottidonitsi
Vastaaja 6	Pingviini hampurilaissämpylöiden välissä	Robottikoriste donitsissa
Vastaaja 7	Hampurilainen, mutta pihvi on pingviini	Suklaadonitsi strösselillä jossa on robotin pää päällä
Vastaaja 8	Pingviini hampuralaisen välissä	Donitsi, jolla on robotin pää
Vastaaja 9	Pingviini burgeri	Robotti donitsi

Ensimmäisen kyselyn tulosten perusteella seitsemän kahdeksasta vastaajasta kokee kuvan muistuttavan annettua generointi tekstiä. Toisessa kyselyssä keskimäärin seitsemän yhdeksästä vastaajasta osasi antaa kuvan perusteella oikean generointi tekstin.

7 Yhteenveto

Tutkimuskysymyksiin vastaaminen onnistui hyvin. Kysymykseen ”Miten tekstistä kuvaksi-generaattori toimii?” saatiin vastaus jo heti opinnäytetyön johdantoa kirjoittaessa. Itse ohjelmointi osuuden jälkeen saatiin vastaus kysymykseen ”Miten monipuolisesti tekstistä kuvaksi -generaattori piirtää?”. Sillä kun generaattoria testattiin useita kertoja eri teksteillä ja ilman tyylivalintoja, sain aina täysin eri tyylillä tehdyn kuvan. Generaattorille pystyy syöttämään niin paljon eri tyylejä, että tyylivalinta-nappulat piti karsia viiteen kappaleeseen. Kyselyistä saatiin dataa, jonka pohjalta pystyttiin vastaamaan kysymykseen ” Kuinka moni kokee tekemäni generaattorin kuvan yhteensopivaksi tekstin kanssa?”. Vastauksista voidaan päätellä, että jokainen generoitu kuva on tulkinnanvarainen. Tutkimuskysymyksiin saatiin vastaukset sujuvasti opinnäytetyötä kirjoittaessa ilman sen suurempia ongelmia.

Tavoitteenani tässä opinnäytetyössä oli oppia React-kieltä ja miten tekoälyn liittäminen ohjelmistoon tapahtuu käytännössä. Ennen ohjelmoinnin aloittamista jännitin, että onnistuuko DALL-E-2:sen liittäminen ohjelmistoon. Ohjelmoinnin alettua ilmeni, että jännitys olikin turhaa ja projekti eteni sutjakkaasti. Se miten tekstistä kuvaksi tekoälyt toimivat, oli suurin asia minkä opin opinnäytetyötä tehdessä. Opin myös hieman React-kielen perusteita ja miten API Key -palvelut toimivat. Työn alussa lähtökohdat olivat huonot asiatekstin kirjoittamisessa, mutta opin sitäkin koko ajan enemmän opinnäytetyön edetessä.

Projektia tullaan tulevaisuudessa jakamaan opinnäytetyön tekijän Github-tilillä. Projektia tullaan kehittämään vielä, jotta siitä saadaan entistä käyttäjäystävällisempi ja mobiiliyhteensopiva. Generaattoriin tullaan ohjelmoimaan hahmo, joka kertoo ohjeistuksen generaattorin käytöstä. Kuvatyyli-nappuloita tullaan ohjelmoimaan lisää ja ne tulevat sisältämään monipuolisempia tyylivalintoja. Mobiiliyhteensopivuus tullaan luomaan React-kielelle rakennetulla Bootstrap -kirjastolla.

Lähteet

Verke. (n.d). Miten teen tekoälytaidetta?. https://www.verke.org/vinkit/miten-teen-tekoalytaidetta/

OpenAI. (n.d). DALL-E-2. https://openai.com/dall-e-2/

OpenAI. (n.d). Image generation. https://beta.openai.com/docs/guides/images/language-specific-tips

George Web Dev. (n.d.) Generate crazy AI images (DALL·E 2) by creating this app - React JS. https://www.youtube.com/watch?v=bFwS5PSHRD8&t=48s

Ryan O'Connor. (19.4.2022). How DALL-E 2 Actually Works. AssemblyAI. https://www.assemblyai.com/blog/how-dall-e-2-actually-works/

Guy Parsons. (13.7.2022). The DALL·E 2 Prompt Book. dall-ery gall-ery. https://dallery.gallery/the-dalle-2-prompt-book/

Microsoft. (n.d). Download Visual Studio Code. https://code.visualstudio.com/Download

DreamStudio. (n.d). [Kuvanteko tekoäly]. [ohjelmisto]. https://beta.dreamstudio.ai/

Nightcafe. (n.d). [Kuvanteko tekoäly]. [ohjelmisto]. https://nightcafe.studio/

Midjourney. (n.d). [Kuvanteko tekoäly]. [ohjelmisto]. http://midjourney.com

Meta Platforms. (n.d). React. https://Reactjs.org

Pluralsight. (n.d). 25 years of JavaScript. https://Javascript.com/about

Lauri Järvenpää. (n.d). Webkehitys ja verkkopalveluiden kehitys. Lamia Oy. https://Itewiki.fi/opas/webkehitys/

netguru. (n.d). What Is Node.js? Complex Guide for 2022. https://Netguru.com/glossary/node-js

Microsoft. (n.d). Learn to code with Visual Studio Code. https://Code.visualstudio.com/learn

Liite 1: Aineistonhallintasuunnitelma

Kehitysprojekti:

Opinnäytetyö tulee sisältämään ohjelmiston luomisessa syntynyttä koodia, havainnollistavia kuvia ja kaavioita. Kyselyt tehdään google Forms-palvelulla ja niihin vastaaminen on täysin anonyymia. Kaikki valokuvat ovat joko itse otettuja tai tekoälyn luomia.

Kaikkea materiaalia, mikä on luotu kehitys projektin aikana, tullaan säilyttämään tekijän pöytätietokoneen C-asemalla opinnäytetyön tekoprosessin ajan. Kaikki materiaali varmuuskopioidaan OneDrive-palveluun viikoittain. Projektissa luotu ohjelmisto tullaan säilyttämään toistaiseksi tekijän Github-tilillä. Opinnäytetyön aineistoa tullaan säilyttämään hyväksymispäivästä ainakin seuraavan vuoden ajan, jotta opinnäytetyön tulokset voidaan tarvittaessa varmistaa.

Opinnäytetyössä ei tulla keräämään henkilötietoja tai muita luottamuksellisia tietoja, joten aineistoa ei tarvitse tuhota säilytysjakson päätyttyä.

Liite 2: index.css koodit

```css
@import url('https://fonts.googleapis.com/css2?family=Tourney');

/* Global Styles */
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}

body {
  font-family: "Source Code Pro", sans-serif;
  font-size: 2rem;
  line-height: 1.6;
  color: #153821;
}
.header{
  font-size: 4rem;
  font-family: "Tourney";
}
.chooseStyle{
  margin-top: 30px;
  font-family: "Source Code Pro", sans-serif;
}

p {
  margin-bottom: 1.25rem;
}

.App {
  width: 100vw;
  height: 100vh;
  position: fixed;
  top: 0;
  right: 0;
  bottom: 0;
  left: 0;
  overflow-y: auto;
  margin: 0 auto;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  background-image: linear-gradient(160deg, #25643b 0%, #b9f3cd 100%);
}
.tutorialText{
  text-align: center;
```

```
  font-size: 1rem;
}

.main-text {
  width: 450px;
  text-align: justify;
}

.label {
  margin-right: 15px;
}

.label-input-pair {
  margin-top: 18px;
  display: flex;
  justify-content: space-between;
  align-content: space-between;
  align-items: center;
  width: 600px;
}
.image{
  width: 200px;
  border-radius: 10px;
}
.imageReset{
  width: 200px;
  border-radius: 50%;
}
.whiteBox{
  width: 200px;
  background-color: white;
  height: 200px;
  border-radius: 10px;
}
.button{
  border-radius: 10px;
  margin-top: 10px;
  margin-left: 10px;
  box-shadow: 0 8px 32px 0 rgba(0, 0, 0, 0.37);
  border-color: #153821;
}
.buttonReset{
  border-radius: 50%;
  margin-top: 10px;
  margin-left: 10px;
  box-shadow: 0 8px 32px 0 rgba(0, 0, 0, 0.37);
  border-color: #153821;

}
.buttonReset:hover{
```

```css
  border-radius: 50%;
  margin-left: 10px;
  border-color: #01d44b;
  box-shadow: 0 8px 100px 0 rgba(0, 0, 0, 0.37);

}
.button:hover{
  border-radius: 10px;
  margin-left: 10px;
  border-color: #01d44b;
  box-shadow: 0 8px 100px 0 rgba(0, 0, 0, 0.37);
}
.image-input-pairs {

  display: flex;
  justify-content: space-between;
  align-content: space-between;
  flex-direction: row;
  align-items: center;
}

.main-input {
  padding: 0 4px;
  height: 36px;
  color: #fff;
  font-size: 2rem;
  background: rgba(255, 255, 255, 0.25);
  box-shadow: 0 8px 32px 0 rgba(0, 0, 0, 0.37);
  backdrop-filter: blur(4px);
  border-radius: 10px;
  border: 1px solid rgba(255, 255, 255, 0.18);
  outline: none;
}


.main-button {
  font-family: "Source Code Pro";
  margin-top: 38px;
  font-size: 2rem;
  color: #153821;
  display: inline-flex;
  justify-content: center;
  align-items: center;
  width: 180px;
  box-sizing: border-box;
  border: none;
  border-radius: 1px;
  cursor: pointer;
  font-weight: 500px;
  position: relative;
```

```
  outline: none;
  gap: 8px;
  height: 50px;
  user-select: none;
  padding: 16px 25px;
  text-decoration: none;
  background: rgba(255, 255, 255, 0.25);
  box-shadow: 0 8px 32px 0 rgba(1, 2, 7, 0.37);
  backdrop-filter: blur(4px);
  border-radius: 10px;
  border: 1px solid rgba(255, 255, 255, 0.18);
}

.main-button:hover {
  background: rgba(133, 132, 132, 0.55);
  box-shadow: 0 8px 32px 0 rgba(0, 0, 0, 0.37);
  backdrop-filter: blur(4px);
  border-radius: 10px;
  border: 1px solid rgba(255, 255, 255, 0.18);
}
```