Atte Räisänen

Oppimisalustan personalisointi, tekoälyn soveltaminen oppimisen hallintajärjestelmään

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

20.5.2025

Tiivistelmä

Tekijä:	Atte Räisänen
Otsikko:	Oppimisalustan personalisointi, tekoälyn soveltaminen oppimisen hallintajärjestelmään
Sivumäärä:	59 sivua + 1 liite
Aika:	20.5.2025
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Ohjelmistotuotanto
Ohjaajat:	Lehtori Amir Dirin

Viimeaikainen tekoälyn liittyvän innovaation kasvu on tehnyt siitä nopeasti lähes väistämättömän aiheen teknologian alalla. Tekoälylle löydetään uusia ja parannettuja käyttötapoja nykyään nopeammin kuin koskaan ennen, mikä on johtanut sen kyvykkyyksien valtavaan kehitykseen vain viimeisen muutaman vuoden aikana.

Tämä opinnäytetyö on tapaustutkimus, jonka tavoitteena on tutkia, kuinka tekoälyä voitaisiin hyödyntää opiskelijoiden oppimiskyvyn arvioimisessa ja yksilöllisten oppimispolkujen luomisessa osana oppimisen hallintajärjestelmää. Tämä toteutetaan kehittämällä virtuaalinen koeympäristö, joka mukautuu käyttäjän osaamistasoon analysoimalla hänen vastauksiaan ja säätämällä kysymysten vaikeustasoa kokeen aikana. Kokeen lopuksi järjestelmä suosittelee opiskelijalle sopivan moduulin, josta hän voi aloittaa kurssin suorittamisen.

Virtuaalisen koeympäristön kehittämisen lisäksi tässä työssä tarkastellaan sen luomiseen tarvittavia teknologioita, kuten koneoppimista, suuria kielimalleja ja muita tekoälyn osa-alueita. Lisäksi työssä käsitellään tekoälyn hyödyntämisen luotettavuutta opintojen personalisoinnissa ja analysoidaan sen mahdollisia hyötyjä, kuten opiskelijoiden yksilöllisten tarpeiden huomioimista suuremmassa ryhmässä sekä sen toteuttamiseen liittyviä haasteita.

Johtopäätöksenä todetaan, että tekoälyn käyttö oppimisen personalisoinnissa voi tuottaa johdonmukaisia ja hyviä tuloksia, mutta tilanteen huolellinen arviointi ja kattava suunnittelu ennen kehitystä on välttämätöntä.

Avainsanat: AI, Tekoäly, Koneoppiminen, Oppimisen hallinta, Oppimisen hallintajärjestelmät

---

Abstract

Author:	Atte Räisänen
Title:	Personalization of Learning Platform, Applying AI in Learning Management System
Number of Pages:	59 pages + 1 appendices
Date:	20 May 2025
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communications Technology
Professional Major:	Software Development
Supervisors:	Amir Dirin, Senior Lecturer

The recent surge in AI-related innovation has made it an almost unavoidable topic in the field of technology. New and improved use cases for AI are being discovered at a faster rate than ever before, which has led to astronomical improvements in the capabilities of artificial intelligence in just the past few years.

This thesis presents a case study with the goal of exploring how AI could be used to assess the learning capabilities of students and create personalized learning paths inside of a Learning Management System. This was achieved through the planning, developing and testing of a virtual exam environment that adapts to the user's level of proficiency by analyzing their responses and tuning the difficulty of the questions as the exam is being taken, concluding with the system deciding on a recommended module of the course for the user to begin their studies.

In addition to the development of the virtual exam environment, this thesis examines the underlying technologies necessary to create it, including machine learning, large language models and other different branches of artificial intelligence. Furthermore, it addresses the feasibility of utilizing AI in personalized learning, analyzing the potential benefits, such as being able to individually cater to each student's needs in a larger group, as well as the challenges that are involved.

In conclusion, it was found that the usage of AI in personalized learning can produce good consistent results, but one must consider the situation carefully and create comprehensive plans before implementation.

Keywords: AI, Artificial Intelligence, Machine Learning, Learning management, Personalized learning

Sisällys

Lyhenteet

LMS: *Learning Management System*. Oppimisen hallintajärjestelmä. Ohjelmisto, jolla voidaan seurata ja raportoida opiskelijan edistystä sekä oppisisällön käyttöä kurssin aikana

HTML: *Hypertext Markup Language*. Merkintäkieli, jota käytetään muun muassa nettisivujen rakenteiden luomisessa.

JSON: *JavaScript Object Notation.* Yleinen ja helposti luettava avoimen standardin tiedostomuoto, joka käyttää avain-arvo-pareja tiedon tallentamisessa.

DOM: *Document Object Model.* Dokumenttioliomalli. Kuvaa rakenteellista dokumenttia puuna, jonka eri olioiden arvoja voi helposti hakea sekä muokata.

NLP: *Natural Language Processing.* Tekoäly, joka osaa prosessoida ihmisen luomaa tietoa ja kommunikoida ihmisen kanssa

HTTP: *Hypertext Transfer Protocol.* Protokolla, jonka avulla voidaan jakaa tietoa palvelimen ja asiakkaan välillä.

1 Johdanto

Teknologian jatkuva kehitys ja verkko-opetuksen kasvava maailmanlaajuinen käyttöönotto on lähivuosina tehnyt verkko-oppimisesta yhä suositumpaa. Verkkokursseille ilmoittautuneiden opiskelijoiden määrä näyttää olevan vuosi vuodelta kasvussa, varsinkin pandemian jälkeen. [1.] Verkko-opinnot tarjoavat nykyään yhä laajemmalle määrälle ihmisiä mahdollisuuden kouluttautua helpommin kuin koskaan ennen ja parantaa taitojaan.

Toimivan ja käytännöllisen verkkokurssin- tai tutkinnon tekemiseen vaaditaan paljon muutakin kuin pelkät oppimismateriaalit ja tehtävät. Esimerkiksi opiskelijoiden osaamistasot voivat vaihdella paljon yksilöittäin, jonka vuoksi on yleistä luoda kokeneille opiskelijoille yksilöllisiä oppimispolkuja. Organisaatioilta ja oppilaitoksilta löytyy usein monia eri työkaluja kurssien ja opiskelijoiden tietojen hallintaan ja seurantaan. Tätä työkalujen kokonaisuutta kutsutaan oppimisen hallintajärjestelmäksi (LMS). [2.]

Tämän työn tarkoituksena on luoda prototyyppi osasta oppimisen hallintajärjestelmää, joka suosittelisi opiskelijalle parasta mahdollista aloituspistettä kurssille hänen osaamisensa perusteella. Tämä toteutettaisiin luomalla lyhyen kokeen, jonka opiskelijan tulisi suorittaa ennen kurssin aloittamista. Koe tulisi sisältämään noin 15 monivalintakysymystä liittyen eri kurssin osuuksiin. Järjestelmä valitsisi opiskelijalle koetulosten perusteella ja tekoälyä hyödyntäen optimaalisen moduulin, josta hänen kannattaisi aloittaa kurssin suorittaminen. Opiskelija voisi myös halutessaan kerrata aiemmat moduulit, mutta ne eivät olisi pakollisia hyväksyttyä kurssisuoritusta varten.

2 Tutkimuskysymysten määritteleminen ja esitutkimus

Tämän työn tarkoituksena on prototyypin luomisen kautta tutkia tekoälyn soveltuvuutta oppimisen hallintajärjestelmään ja saada vastaukset ainakin seuraaviin kysymyksiin:

- Mitä mahdollisia hyötyjä tekoälyn käyttö tarjoaa verkko-oppimisen personalisoinnissa?
- Mitä mahdollisia haasteita tekoälyn hyödyntämisessä tällaisessa tilanteessa saattaa tulla vastaan?
- Voiko tekoälyä luotettavasti hyödyntää opiskelijan osaamistason määrittelyssä ja oppimismoduulin ehdottamisessa?

Useat oppimisalustat hyödyntävät nykyään tekoälyä tavalla tai toisella räätälöidäkseen opiskelijalle henkilökohtaisia oppikokemuksia tarpeen mukaan. Esimerkiksi oppimisalusta Khan Academy on kehittänyt oman tekoälyopastajan nimeltään Khanmigo, joka toimii opastajana kursseilla. Se on NLP (Natural Language Processing) -malli, joka on kehitetty auttamaan opiskelijoita kysymyksien kanssa antamatta suoraa oikeata vastausta. Se on myös opetettu käyttämällä Khan Academyn kurssien oppimismateriaaleja. [3.]

Tämän lisäksi eri koneoppimisalgoritmien käyttämistä opiskelijan osaamistason määrittelemisessä on myös tutkittu. Tuloksien perusteella koneoppimismallit osasivat yleensä ennustaa opiskelijan osaamistasoa, mutta eroavaisuuksia alkoi näkymään enemmän huonommin suoriutuvien opiskelijoiden joukossa. [4.]

3 Teknologioihin tutustuminen

3.1 Oppimisen hallintajärjestelmien historiaa

Uusimmat LMS-alustat tarjoavat monenlaisia tapoja hallita, seurata ja dokumentoida kursseja, opiskelijan edistystä sekä oppimateriaalin käyttöä. LMS ei kuitenkaan ole uusi keksintö.

Ensimmäisen LMS:n, nimeltään *Stand Alone LMS*, kehittivät Phil Bookman ja Rich Silton vuonna 1983. LMS kehitettiin Silton-Bookman Systems -yrityksen henkilöstöhallinnon avustukseksi. Sen pohjalta kehitettiin sovellus nimeltään Registrar, joka hyödynsi Stand Alone LMS:n ominaisuuksia ja antoi henkilöstöhallinnon ja opastajien ilmoittaa työntekijöitä kursseille heidän puolestaan sekä pitää kirjaa läsnäoloista ja seurata koulutuksen edistymistä. Järjestelmä vaati toimiakseen koulutetun ylläpitäjän, jonka työ oli manuaalisesti kirjata järjestelmän sisälle kaikki opiskelijan tiedot, koska verkkoteknologia ei ollut vielä tarpeeksi kehittynyt. [5.]

Ensimmäiset 80-luvulla kehitetyt LMS:t olivat suunniteltu yrityksille avustamaan työntekijöiden sisäisessä opastuksessa. Ne eivät olleet vielä yleisiä, koska niiden ylläpito ja kehitys oli työlästä, eikä niiden hyödyntäminen kouluissa tai julkisessa opetuksessa ollut myöskään vielä järkevää. Kaikki kuitenkin muuttui 90-luvun alkupuolella, kun tietoliikenne- ja verkkoteknologian kehitys mahdollisti sekä opastajien, että opiskelijoiden pääsyn järjestelmään internetin kautta. Tämä johti LMS-alustojen nopeaan kehitykseen ja niiden käyttöön oppilaitoksissa ja kouluissa. Iso-Britannian Open University otti käyttöön SoftArc:in *FirstClass* LMS:n [6] ja Norjan NKI kehitti ja julkaisi oman LMS-alustansa nimeltä *EKKO*. [7] Järjestelmien uusiin ominaisuuksiin kuului muun muassa opastajien mahdollisuus luoda sisältöä opiskelijoille sekä opiskelijoiden mahdollisuus rekisteröidä itsensä järjestelmään ja sen jälkeen osaksi valitsemaansa kurssia. Tästä alkoi modernin LMS:n kehitys.

3.2 Nykyisten LMS-alustojen tutkiminen

Nykypäivänä markkinoilta löytyy satoja LMS-alustoja eri tarpeisiin. Järjestelmät ovat useimmiten joko yrityksille tai oppilaitoksille suunnattuja, mutta jotkut soveltuvat molemmille.

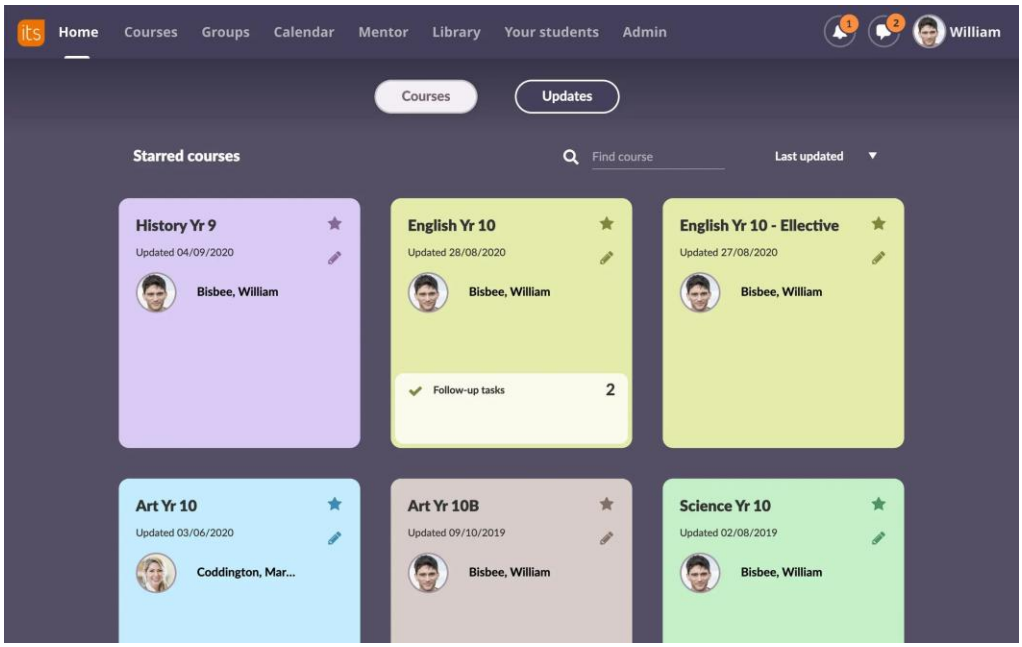
Moodle on Martin Dougiamasin vuonna 1999 kehittämä ilmainen avoimen lähdekoodin LMS. Se on suunnattu monenlaisille eri käyttäjäryhmille ja on muokattavissa tarpeen mukaan. Tällä hetkellä yli 140 000 rekisteröityä nettisivua

käyttää Moodlea. Se tukee yli 100 kieltä ja sitä on käytetty 235 eri maassa. Sen rekisteröityjen käyttäjien määrä on yli 430 miljoonaa, joka tekee siitä yhden tämän hetken suosituimmista LMS-alustoista. Suuri syy sen pitkäkestoiseen suosioon on sen GPLv3-lisenssi, joka sallii sen lataamisen, kopioimisen, muokkaamisen ja jakelun. Se tukee myös yhteisön tekemien laajennusten ja lisäosien käyttöä. [8.]

Moodlella voi luoda joko julkisia tai salattuja kursseja. Julkisiin kursseihin voi liittyä kuka tahansa, kun taas salatun kurssin liittymiseen tarvitaan opettajan luoma salainen ”avain”. Kurssit voivat sisältää oppimateriaalia, oppimistehtäviä, tenttejä sekä verkkokeskusteluja. Opiskelijoilla on mahdollisuus antaa alustan kautta opettajalle suoraa palautetta kurssista, ja opettaja voi seurata jokaisen opiskelijan edistystä kurssilla sekä tarjota tarvittaessa ohjausta. [9.]

Moodlea ylläpidetään ja kehitetään Moodle-hankkeella. Hanketta tukee taloudellisesti tällä hetkellä yli 100 kumppania eri puolilta maailmaa. Suomessa Moodlesta on tullut korkeakouluissa eniten käytetty LMS. Vuonna 2018 tehdyn raportin mukaan 85 % suomalaisista korkeakouluista käytti joko Moodlea tai Moodleroomsia verkko-oppimisympäristönä. [10.]

Toinen suosioon noussut LMS on oppilaitoksille suunnattu norjalaisen yrityksen *Itslearning AS* vuonna 1999 kehittämä järjestelmä itslearning. Toisin kuin Moodle, se ei ole ilmainen, eikä se tue ylimääräisiä lisäosia tai laajennuksia. Sen sijaan itslearning on luotu nopeasti asennettavaksi ja helposti käytettäväksi alustaksi, joka tarjoaa opettajille kaikki opetukseen tarpeelliset työkalut, ja antaa heidän keskittyä opettamiseen. Itslearningin käyttöliittymä on suunniteltu pitäen myös opiskelijat mielessä. Omia käynnissä olevia kursseja kuvataan eri värisillä ”korteilla”, jotka sisältävät myös ilmoituksia uusista tehtävistä. Kuva 1 esittää itslearningin käyttöliittymää. [11.]



Kuva 1. Kuvakaappaus itslearning-oppimisalustan etusivulta [11]

Tämän lisäksi itslearning tarjoaa paljon muita opettajille ja opiskelijoille hyödyllisiä ominaisuuksia, kuten [11]

- viestintätyökalun oppilaiden ja opettajien välille
- oppimissuunnitelmien- ja tavoitteiden lisäämisen
- digitaalisen kirjaston opetusmateriaaleille
- kalenterin kursseille, suunnitelmille ja tapahtumille
- *ePortfolio* työkalun sähköisen portfolion luomiseen.

Vuonna 2019 itslearning myytiin suomalaisen Sanoma Group Oy:n haltuun. Itslearning toimii edelleen erillisenä yrityksenä, mutta Sanoma Groupin tuoma taloudellinen tuki vahvistaa sen asemaa ja jatkokehitysmahdollisuuksia.

3.3 Tekoäly ja koneoppiminen

Monet maailman suurimmista yrityksistä hyödyntävät tekoälyä tuotteiden sekä palveluiden kehityksessä ja käyttökokemuksen parantamisessa. Sitä käytetään esimerkiksi [12]

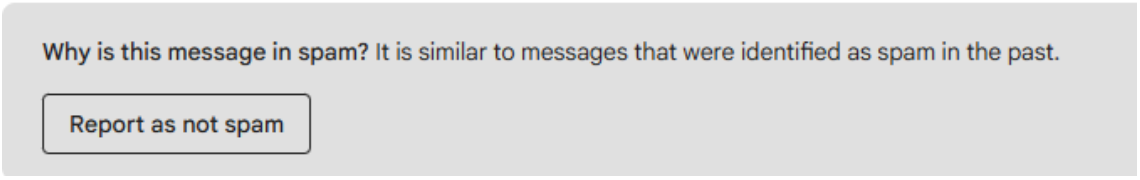
- asiakaspalvelussa usein kysyttyjen kysymysten vastaamisessa tai käyttäjän ohjaamisessa oikealle sivulle
- suosittelualgoritmeissa
- terveydenhuollossa avustamassa diagnoosin kanssa
- data-analyysissä suuren datamäärän tiivistämisessä ymmärrettävään muotoon.

Tässä opinnäytetyössä aikomuksena on käyttää tekoälyä apuna lyhyen kokeen toteuttamisessa, jonka perusteella opiskelijan osaamistasoa voisi ennustaa. Lyhyesti sanottuna tekoälyllä tarkoitetaan älykkääseen päättelyyn kykenevää tietokonetta tai ohjelmaa, mutta se on käsitteenä erittäin laaja. Siihen liittyy monta osa-aluetta, kuten koneoppiminen, neuroverkot, luonnollisen kielen käsittely, robotiikka ja syväoppiminen.

Koneoppimisella (Machine Learning) tarkoitetaan sellaista tekoälyä, joka osaa hyödyntää sille syötettyä dataa päätöksenteossa. Koneoppimismalli kehitetään usein jonkin tietyn ongelman ratkaisemiseksi. Sille syötetään paljon relevanttia dataa, josta se ”oppii” tekemään haluttuja päätöksiä. Mallia opetetaan tarkasti poimitulla datalla, kunnes se kykenee tekemään haluttuja päätöksiä. Tämän jälkeen mallin tarkkuutta arvioidaan käyttämällä uutta tarkistamatonta dataa. Arvioinnin tulosten perusteella malli joko otetaan käyttöön tai sen kehittämistä jatketaan.

Koneoppimista voi hyödyntää monessa eri tilanteessa, esimerkiksi roskapostin tunnistamisessa osana sähköpostipalvelua. Tässä tilanteessa koneoppimismallille annettaisiin syötteenä suuri lista sähköpostiviestejä, jotka merkittäisiin joko roskapostiksi tai ei. Syötettyjen viestien perusteella malli oppisi tunnistamaan roskapostiviestien piirteitä, esimerkiksi usein käytettyjä sanoja tai lauseita, ja lopulta sitä voitaisiin käyttää uusien tulevien sähköpostien suodattamisessa. Realistisessa tilanteessa tällainen koneoppimismalli vaatisi jatkuvaa opettamista, koska roskapostin lähettäjät tulisivat löytämään erilaisia keinoja suodattimen ohittamiseen. Esimerkiksi Googlen kehittämä järjestelmä roskapostin tunnistamiseen käyttää Gmail-palvelun käyttäjien raportteja sen parantamiseen. Jos käyttäjä merkitsee viestin roskapostiksi, sen sisältö analysoidaan ja viesti

tallennetaan järjestelmään. Jos jokin viesti on vahingossa luokiteltu roskapostiksi, voi käyttäjä merkitä sen virheellisesti luokitelluksi. Tässäkin tilanteessa käyttäjän raporttia käytetään koneoppimismallin parantamiseen. Kuva 2 näyttää, miten virheellisesti merkityn roskapostin voi ilmoittaa. [13.]



Kuva 2. Kuvakaappaus Gmail-palvelusta. Nappi virheellisesti luokitellun roskapostin raportoimiseen

Syväoppimisella (Deep Learning) taas puolestaan tarkoitetaan tietokoneen opettamista käsittelemään dataa ja tekemään johtopäätöksiä samalla tavalla kuin ihminen tekisi. Syväoppimismallit kykenevät ymmärtämään lauseita, kuvia ja puhetta. Ne voivat esimerkiksi kuvailla sanoin kuvan sisältöä, vastata ääneen kysyttyyn kysymykseen tai jopa luoda monen sivun pituisesta kirjoitelmasta lyhyen tiivistelmän. [14.]

Syväoppimismallit ovat laajoja neuroverkkoja, jotka koostuvat usein miljoonista simuloiduista ”neuroneista”, jotka yhdessä käsittelevät syötteen ja samalla oppivat siitä. Neuronit ovat yksittäisiä solmuja, jotka käsittelevät dataa käyttäen matemaattisia laskutoimituksia, ja syöttävät tuloksensa seuraavalle neuronille. Lopputuloksena saadaan vastaus alkuperäiseen kysymykseen. [14.]

Suuret kielimallit ovat suuria syväoppimismalleja, joita opetettu valtavalla määrällä dataa. Ne kykenevät opettamaan itseään ja ymmärtämään sekä sisäistämään tekstissä kirjoitettua tietoa. Ne voivat käyttää lähteenään miljardeja eri nettisivuja, kirjoja ja kirjoitelmia. [15.]

3.4 JavaScript/TypeScript

JavaScript on Netscapen 1995 vuonna luoma ohjelmointikieli, josta on tullut yksi maailman suosituimmista ohjelmointikielistä ja web-kehityksen oletuskieli. Web-kehityksessä JavaScript mahdollistaa dynaamisen toiminnallisuuden lisäämisen verkkosivulle tai web-sovellukseen. Sitä voidaan käyttää myös backend-ohjelmoinnissa, esimerkiksi palvelimien luomisessa Node.js-ajoympäristöä hyödyntäen.

JavaScriptin suosioon liittyy monia syitä; sillä on kaikista ohjelmointikielistä suurin pakettiarkisto, ja sitä voi nykyään käyttää melkein missä vain ympäristössä, jopa työpöytäsovellusten kehityksessä ja mobiilikehityksessä. Se on myös moneen muuhun ohjelmointikieleen verrattuna helppo oppia aloittelijalle. Koodia ei tarvitse erikseen kääntää, ja sen syntaksi on helposti luettavaa. Tämän lisäksi JavaScript on heikosti tyypitetty kieli, joka tarkoittaa, että muuttujien tietotyyppejä ei tarvitse merkitä erikseen, ja erityyppisten muuttujien muunnokset toisiinsa tehdään implisiittisesti eli päättelykykyä hyödyntäen.

Heikosti tyypitetyn kielen vastakohta on vahvasti tyypitetty kieli. Nämä kielet vaativat muuttujien tietotyyppien määrittelemisen ja erityyppisten muuttujien muunnokset joudutaan määritellä itse koodissa, esimerkiksi cast-operaattoria käyttäen. Vahvasti tyypitettyihin kieliin kuuluvat esimerkiksi Java, C# ja C++. Esimerkkikoodi 1 esittää JavaScript-ohjelmointikielen käyttöä.

```
function laskeKeskiarvo(numerot) {
     if(numerot.length === 0) return 0;
     let summa = 0;
     for(let num of numerot) {
          summa += num;
     }

     return summa / numerot.length;
}

const numerot = [15, 35, 80, 14, 56];
console.log("Keskiarvo:", laskeKeskiarvo(numerot));
```

Esimerkkikoodi 1. Lyhyt JavaScriptillä kirjoitettu ohjelma, joka ottaa syötteenä taulukon ja laskee sen sisältävien numeroiden keskiarvon.

Yllä esitetty Ohjelma toimii JavaScript-koodina, mutta sen käyttö tulee johtamaan virheelliseen palautusarvoon, jos funktiolle annetaan syötteenä taulukko, joka sisältää muun tyyppisiä muuttujia kuin numeroita. Esimerkkikoodi 2 näyttää saman keskiarvon laskuun tehdyn ohjelman kirjoitettuna Javalla. JavaScriptistä eroten kaikkien muuttujien tietotyypit sekä funktioiden palauttamien tietojen tyypit on määritelty koodissa. Java-koodissa tämä on pakollista. Javan kääntäjä ei suostu kääntämään koodia ja luomaan ohjelmaa, jos muuttujien tietotyypeissä on virheitä. Lisäksi Javassa koodin täytyy kuulua johonkin luokkaan, ja tässä tapauksessa ”main”-metodin luominen oli tarvittava syötteen saamiseksi.

```
public class KeskiarvoLaskin {
     public static double laskeKeskiarvo(int[] numerot) {
          if(numerot.length == 0) return 0;

          int summa = 0;
          for(int num : numerot) {
               summa += num;
          }

          return (double) summa / numerot.length;
     }

     public static void main(String[] args) {
          int[] numerot = {10, 20, 30, 40, 50};
          System.out.println("Keskiarvo: " + laskeKeskiarvo(numerot));
     }
}
```

Esimerkkikoodi 2. Keskiarvon laskeminen Javalla

Aloittelijan näkökulmasta heikosti tyypitetyt kielet voivat vaikuttaa paljon helppokäyttöisemmiltä, ja vahvasti tyypitettyyn kieleen tottuminen voi tuntua vaikealta, jos sellaista ei ole aiemmin käyttänyt. Vahvasti tyypitetyt kielet ovat kuitenkin suosittuja ja yleisempiä valintoja suuriin projekteihin ja koodipohjiin juuri siksi, että ne pakottavat ohjelmoijan miettimään koodin rakennetta ja määrittelemään selvästi funktioiden parametrit ja palautetyypit. Pienemmässä ohjelmassa, kuten koodiesimerkissä 2, tämä ei ole usein tarpeellista, mutta kymmenien tai satojen tuhansien koodirivien projekteissa funktioiden ja luokkien tarkka määrittely on erittäin tärkeää luettavuuden ja bugien estämisen kannalta. Juuri tämän takia JavaScriptin pohjalta on luotu vahvasti tyypitetty ohjelmointikieli nimeltään TypeScript.

TypeScript on tarkoituksella lähes identtinen JavaScriptiin. Se on vahvasti tyypitetty ja usein paremmin integroitu kehitysympäristöihin ja koodieditoreihin. TypeScript-koodi joudutaan aina ajaessa kääntämään JavaScriptiksi. Esimerkkikoodi 3 näyttää, miten aiempi keskiarvon laskeva ohjelma toteutetaan TypeScriptillä.

```
function laskeKeskiarvo(numerot: number[]): number {
     if(numerot.length === 0) return 0;

     let summa: number = 0;
     for(let num of numerot) {
          summa += num;
     }

     return summa / numerot.length;
}


const numerot: number[] = [10, 20, 30, 40, 50];
console.log("Keskiarvo:", laskeKeskiarvo(numerot));
```

Esimerkkikoodi 3. TypeScript-ohjelma, joka ottaa parametrina taulukon numeroita ja laskee sekä palauttaa niiden keskiarvon. Lähes identtinen aiemmin näytettyyn JavaScript-koodiin.

Tässä työssä tullaan hyödyntämään TypeScriptiä backend-ohjelmoinnissa.

3.5 Node.js

Node.js on JavaScriptille suunniteltu ajoympäristö, joka on suunniteltu JavaScript-koodin ajamiseen. Ilman sitä JavaScript-koodi jouduttaisiin yhdistämään osaksi nettisivua, ja sen ajaminen onnistuisi ainoastaan selaimella. Node.js mahdollistaa JavaScriptin ja TypeScriptin ajamisen komentoriviltä, jonka seurauksena sitä voidaan käyttää monessa eri tilanteessa, esimerkiksi palvelimen luomisessa.

3.6 React

React on vuonna 2013 luotu ilmainen avoimena lähdekoodina julkaistu käyttöliittymien tekemiseen tarkoitettu JavaScript-kirjasto. Se on helppokäyttöinen, nopea ja joustava, ja sitä voi käyttää myös mobiilisovellusten kehittämiseen.

Käyttöliittymän rakentaminen Reactilla tapahtuu komponenteilla. Komponentit toimivat kuin JavaScript-funktiot. Ne voivat olla niin isoja tai pieniä kuin halutaan ja voivat myös pitää sisällään muita komponentteja. Ne voivat myös ottaa syötteenä tietoa käyttäen ”propseja”, jotka ovat verrattavissa JavaScriptin funktioiden parametreihin. Esimerkkikoodissa 4 luodaan React-komponentti ”Kayttaja”, joka ottaa syötteenä käyttäjän nimen ja sähköpostin, ja palauttaa ne HTML-muodossa. Toinen React-komponentti ”KayttajaLista” ottaa parametrina listan käyttäjiä ”kayttajat”, luo jokaiselle käyttäjälle oman ”Kayttaja” komponentin, ja palauttaa komponentit.

```
function Kayttaja(props) {
  return (
    <div className="user-card">
      <h2>{props.nimi}</h2>
      <p>Email: {props.email}</p>
    </div>
  );
}

function KayttajaLista(props) {
  return (
    <div className="user-list">
      {props.kayttajat.map((kayttaja, index) => (
        <Kayttaja key={index} nimi={kayttaja.nimi}
          email={kayttaja.email}
        />
      ))}
    </div>
  );
}
```

Esimerkkikoodi 4. Toisia komponentteja sisältävien React-komponenttien luominen

Esimerkkikoodi 5 esittää, miten yllä olevien React-komponenttien pohjalta voidaan luoda vielä kolmas komponentti, joka hyödyntää molempia komponentteja lopullisen käyttäjälistan luomiseen.

```
function App() {
  const kayttajat = [
    { nimi: 'Alice', email: 'alice@example.com'},
    { nimi: 'Bob', email: 'bob@example.com'},
    { nimi: 'Charlie', email: 'charlie@example.com'}
  ];

  return (
    <div className="app">
      <h1>Käyttäjähakemisto</h1>
      <KayttajaLista kayttajat={kayttajat} />
    </div>
  );
}
```

Esimerkkikoodi 5. React-komponentti, joka määrittelee listan käyttäjistä ja niiden tiedoista, ja palauttaa ”KayttajaLista”-komponentin, jolle on lähetetty käyttäjätiedot.

Kaikki kolme komponenttia yhdessä renderöivät sivulle käyttäjähakemiston.

Kuva 3 on kuvakaappaus komponenttien syötteestä.

Kuva 3. Yllä olevien esimerkkikoodien syöte selaimessa (vasen) ja HTML-muodossa (oikea)

Toinen tärkeä Reactin ominaisuus on sen tarjoamat ”koukut” (hooks). Koukut antavat ohjelmoijalle pääsyn eri Reactin ominaisuuksiin suoraan komponenttien

sisältä. React tarjoaa oletuksena monia erilaisia koukkuja, kuten esimerkiksi State-, Context-, Ref- ja Effect-koukut.

State-koukut mahdollistavat tiedon, kuten käyttäjän syötteen, muistamisen. Esimerkkikoodi 6 esittää state-muuttujan käyttämistä osana yksinkertaista laskinta. Esimerkissä luodaan state-muuttuja ”lasku”, jota voidaan kasvattaa tai vähentää painamalla nappia. Muistettava muuttuja luodaan useState funktiolla, jolle annetaan parametrinä muuttujan oletusarvo. State-koukulle täytyy myös antaa ”setter”-funktio, tässä tapauksessa setLasku, jota kutsutaan, kun muuttujan arvoa halutaan muuttaa.

```
function Laskin() {
  const [lasku, setLasku] = useState(0);

  const kasvata = () => setLasku(lasku + 1);
  const pienenna = () => setLasku(lasku - 1);

  return (
    <div>
      <h1>Laskin: {lasku}</h1>
      <button onClick={kasvata}>Kasvata</button>
      <button onClick={pienenna}>Pienennä</button>
    </div>
  );
}
```

Esimerkkikoodi 6. Yksinkertainen laskuri käyttäen Reactin state-koukkua

Context-koukut antavat komponenttien hakea tietoa ilman, että sitä olisi annettu propsien kautta. Niitä voi käyttää esimerkiksi sivun taustavärin muuttamiseen napilla, tai muissa tapauksissa, joissa samaa vaihtuvaa muuttujaa käytetään monessa eri paikassa (esimerkkikoodi 7).

```
const ThemeContext = createContext();
function ThemeProvider({ children }) {
  const [teema, setTeema] = useState('light');

  const vaihdaTeemaa = () => {
    setTeema((aiempiTeema) => (aiempiTeema === 'light' ? 'dark' :
     'light'));
  };

  return (
    <ThemeContext.Provider value={{ teema, vaihdaTeema }}>
      {children}
    </ThemeContext.Provider>
  );
}
function TeemanVaihtaja() {
  const { vaihdaTeema } = useContext(ThemeContext);

  return <button onClick={vaihdaTeema}>Vaihda teemaa</button>;
}
```

Esimerkkikoodi 7. Sivun taustavärin muuttaminen Context-koukulla.

Ref-koukut pitävät sisällään tietoa kuten State-koukut, mutta ne eroavat niistä siten, että niiden sisällön päivittäminen ei renderöi uudelleen koko komponenttia. Niitä käytetään usein DOM (Document Object Model) -elementtien, kuten tekstilaatikoiden hallintaan. Ne eivät myöskään tarvitse erillistä ”setter”-funktiota niiden sisältävän tiedon muokkaamiseen. Esimerkkikoodi 8 näyttää, miten ref-koukkua voidaan käyttää kursorin keskittämisessä.

```
function InputFocus() {
  const inputRef = useRef(null);
  const focusInput = () => {
    inputRef.current.focus(); //Keskitetään kursori elementtiin
  };

  return (
    <div>
      <input
        ref={inputRef} // Kytketään ref input-elementtiin
        type="text"
        placeholder="Click button to focus me"
      />
      <button onClick={focusInput}>Focus the input</button>
    </div>
  );
}
```

Esimerkkikoodi 8. Luodaan Ref-koukkua käyttäen tekstilaatikko ja nappi, joka keskittää kursorin siihen.

Effect-koukut ovat pääosin suunniteltu antamaan komponenteille mahdollisuuden kommunikoida ulkopuolisten järjestelmien, kuten esimerkiksi palvelimien kanssa. Effect-koukku voidaan määritellä kutsumaan sen sisältämän koodin kolmella eri tavalla:

- Ilman parametreja, jolloin se ajaa sisältämänsä koodin aina, kun komponentti, jonka sisälle se on määritelty, renderöidään uudelleen.
- Tyhjä taulukko parametrina, jolloin se ajaa sisältämänsä koodin yhden kerran silloin, kun komponentti, jonka sisälle se on määritelty, renderöidään ensimmäisen kerran.
- Parametrina taulukko, joka sisältää muuttujan tai muuttujia, jolloin se ajaa sisältämänsä koodin renderöinnin yhteydessä ainoastaan, jos jotain muuttujista on muokattu.

Esimerkkikoodi 9 näyttää konkreettisesti, miten näitä kolmea eri tapaa käytetään.

```
useEffect(() => {
  console.log("Tämä effect-koukku kutsutaan jokaisella
renderöinnillä");
});

useEffect(() => {
  console.log("Tämä effect-koukku kutsutaan vain kerran");
}, []);

useEffect(() => {
  console.log(`"Tämä effect-koukku kutsutaan aina countin
muuttuessa");
}, [count]);
```

Esimerkkikoodi 9. Esimerkki kolmesta eri tavasta kutsua Effect-koukkua

3.7 Python

Python on vuonna 1991 kehitetty dynaamisesti tyypitetty ohjelmointikieli, joka on suunniteltu olemaan helposti luettava. Siitä on tullut yksi koneoppimisen suosituimmista ohjelmointikielistä sen laajan kirjastovalikoiman sekä helpon syntaksin vuoksi.

Koneoppimisalgoritmit ovat usein erittäin monimutkaisia, jonka vuoksi koodin luettavuus ja helposti ymmärrettävä syntaksi ovat tärkeitä ohjelmointikielen

ominaisuuksia kehittäjälle. Python antaa kehittäjien keskittyä algoritmien rakentamiseen ja suunnittelemiseen koodin debuggauksen sijasta. Esimerkkikoodi 10 havainnollistaa pythonin yksinkertaista syntaksia. [16.]

```
def laskeKeskiarvo(num):
    return sum(num) / len(num) if num else 0

print(laskeKeskiarvo([1, 2, 3, 4, 5]))
```

Esimerkkikoodi 10. Python-funktio, joka ottaa parametrina listan numeroita ja palauttaa niiden keskiarvon

Yksi suosituimmista koneoppimiseen suunnatuista python-kirjastoista on scikit-learn. Se tarjoaa paljon helposti käytettäviä työkaluja sekä valmiita algoritmeja, joita voi käyttää data-analyysissä. Esimerkkikoodi 11 esittää tilannetta, jossa kirjastoa käytetään ennustamisessa.

```
from sklearn.datasets import load_iris
from sklearn.neighbors import KNeighborsClassifier

iris = load_iris()

model = KNeighborsClassifier()
model.fit(iris.data, iris.target)

print(model.predict([iris.data[0]]))
```

Esimerkkikoodi 11. Esimerkki luokittelusta käyttäen scikit-learn kirjastoa.

Esimerkissä ladataan aluksi iris-tietoaineisto, joka sisältää kerättyä dataa erilaisista kukista. Kukkia on kolmea eri lajiketta, ja haluamme käyttää tietoaineiston dataa lajikkeen ennustamiseen. Käytämme tähän KNeighborsClassifier-mallia. Opetamme mallin käyttäen aiemmin ladattua tietoaineistoa, ja tulostamme näytölle ennustuksen tuloksen. Esimerkkikoodi 12 esittää, miltä saman analyysin tekeminen näyttäisi ilman scikit-learn kirjaston työkaluja.

```
import numpy as np
from collections import Counter
from sklearn.datasets import load_iris

iris = load_iris()
X = iris.data
y = iris.target


def predict(x, X_train, y_train, k=3):
    distances = [np.sqrt(np.sum((x - sample) ** 2))
                 for sample in X_train]
    inds = np.argsort(distances)[:k]
    labs = [y_train[i] for i in inds]
    return Counter(labs).most_common(1)[0][0]
print(predict(X[13], X, y))
```

Esimerkkikoodi 12. Esimerkki luokittelusta ilman scikit-learn kirjastoa

Koska emme voi suoraan ladata mallia kirjastosta, joudumme toteuttamaan sen kokonaan itse. Laskemme syötetyn kukkanäytteen euklidisen etäisyyden jokaiseen tietoaineiston muuhun näytteeseen, jonka jälkeen valitsemme sen kolme lähintä naapuria ja valitsemme niistä yleisimmän lajikkeen.

3.8 HTTP

Tarvitsemme myös tavan, jolla pystymme yhdistämään käyttöliittymä järjestelmäämme. Tähän tulemme käyttämään HTTP (Hypertext Transfer Protocol) -pyyntöjä. HTTP-pyynnöt mahdollistavat datan lähettämisen sekä hakemisen palvelimelta luomalla päätepisteitä, jotka vastaanottavat ja käsittelevät tulevia pyyntöjä. HTTP-pyyntöjä löytyy moneen eri käyttötarkoitukseen. Taulukko 1 esittää yleisimmät HTTP-pyynnöt ja selittää lyhyesti niiden eri käyttötarkoitukset

Taulukko 1. HTTP tarjoaa monia erilaisia metodeja eri tarkoituksiin. Tässä taulukossa selitetään yleisimmät metodit ja niiden käyttötarkoitukset.

Metodi	**Käyttötarkoitus**
GET	Tiedon hakeminen palvelimelta
POST	Tiedon lähettäminen palvelimelle
PUT	Olemassa olevan tiedon korvaaminen uudella arvolla

Metodi	Käyttötarkoitus
PATCH	Olemassa olevan tiedon päivittäminen
DELETE	Tiedon poistaminen
HEAD	Sama kuin GET-metodi, mutta palvelin ei palauta takaisin mitään dataa
OPTIONS	Palauttaa kaikki palvelimella käytössä olevat metodit

3.9 MongoDB

Relaatiotietokannalla tarkoitetaan tietokantajärjestelmää, joka hyödyntää tauluja sekä niiden välisiä suhteita tiedon tallentamiseen. Jokaisella taululla on oltava tasan yksi pääavain, joka voi koostua yhdestä tai useammasta sarakkeesta. Pääavaimen arvon on erottava jokaisen taulukon rivin välillä, koska sen tarkoitus on erottaa taulun rivit toisistaan. Tämän lisäksi taulujen välille voi luoda yhteyksiä käyttäen viiteavaimia. Viiteavain on taulukon sarake, joka sisältää viitteen toisen taulukon riviin käyttäen sen pääavainta arvona. Viiteavaimen sisältävää taulukkoa kutsutaan usein lapseksi, ja sen viittaamaa taulukkoa äidiksi. Nämä termit viittaavat siihen, että äidillä voi olla useita lapsia, mutta lapsella ei voi olla useampaa äitiä. Taulukko 2 esittää tietokantataulua, joka sisältää asiakkaiden yhteystietoja.

Taulukko 2. Yksinkertainen esimerkki relaatiotietokannan taulusta, joka sisältää asiakastietoja

AsiakasID (Pääavain)	Nimi	Sähköposti
1	Liisa	liisa@esimerkki.com
2	Mikko	mikko@esimerkki.com

Taulukolle 2 voisi antaa nimeksi esimerkiksi ”Asiakkaat”, ja siihen voitaisiin liittää muita lapsitauluja käyttämällä viiteavainta. Taulun pohjalta voitaisiin luoda uusi taulu, joka viittaisi tiettyyn asiakkaaseen. Taulukko 3 on esimerkki tällaisesta taulusta.

Taulukko 3. Esimerkki ylemmän "Asiakkaat"-taulun pohjalta luodusta taulusta. Taulun viiteavain viittaa tiettyyn asiakkaaseen

TilausID (Pääavain)	**AsiakasID (Viiteavain)**	**TilausPvm**
101	1	2025-03-28
102	2	2025-03-29
103	2	2025-03-30

Kahden yllä olevan taulun avulla on luotu yksinkertainen suhde asiakkaiden ja tilausten välillä. Jokaisella tilauksella on oltava tasan yksi asiakas, ja yhdellä asiakkaalla voi olla monta tilausta.

Relaatiotietokantojen taulut luodaan usein käyttäen SQL-kyselykieltä. Taulukkoa luodessa täytyy olla tarkka, sillä jokaisen sarakkeen tietotyyppi on määriteltävä erikseen. Pääavaimet ja viiteavaimet määritellään myös taulukkoa luodessa. Esimerkkikoodi 13 esittää, miten aiemmin esitetyt taulut voitaisiin luoda SQL-kieltä käyttäen.

```
CREATE TABLE Asiakkaat (
    AsiakasID INT PRIMARY KEY,
    Nimi VARCHAR(50),
    Sähköposti VARCHAR(100)
);

CREATE TABLE Tilaukset (
    TilausID INT PRIMARY KEY,
    AsiakasID INT,
    TilausPvm DATE,
    FOREIGN KEY (AsiakasID) REFERENCES Asiakkaat(AsiakasID)
);
```

Esimerkkikoodi 13. Aiemmin esiteltyjen taulujen luominen SQL-kielellä.

Relaatiokannat ovat helppokäyttöisiä ja niiden luominen on yksinkertaista. Ne ovat nopeita, ja ovat oiva valinta, jos tallennettava tieto on helppoa jakaa suhteisiin ja sen muoto ei muutu.

Relaatiokantojen lisäksi on olemassa dokumenttitietokantoja, kuten MongoDB. Tämänkaltaisia tietokantajärjestelmiä kutsutaan NoSQL (Not only SQL) -tietokannoiksi koska ne eivät käytä perinteistä relaatiomallia eikä SQL-kyselykieltä hakuihin, muutoksiin ja lisäyksiin.

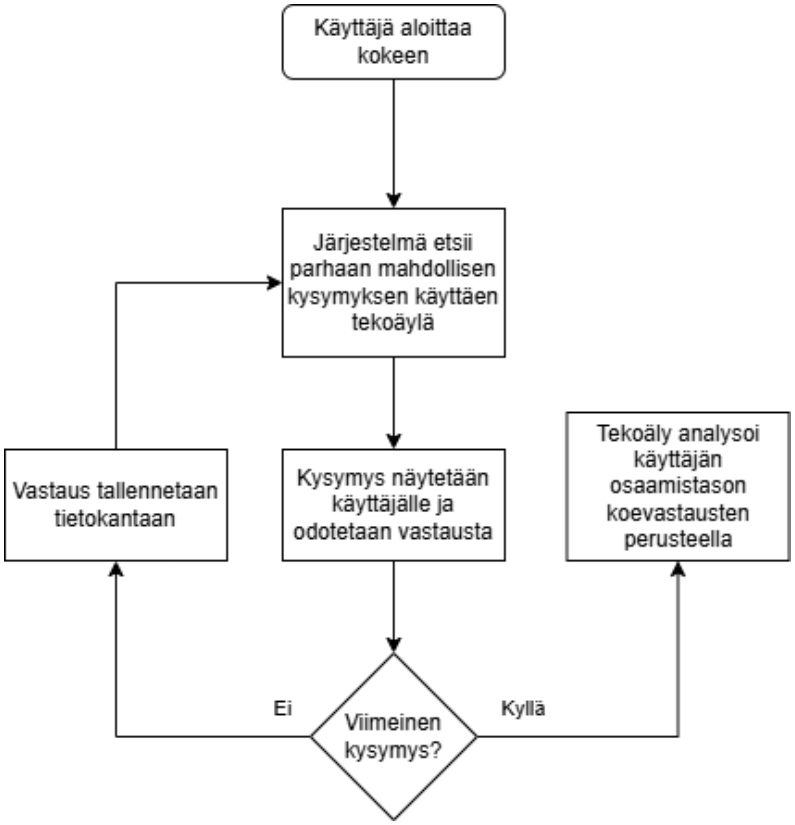
MongoDB tallentaa tiedot kantaan JSON (JavaScript Object Notation) -muodossa. Tämä tekee tietojen tallentamisesta ja sovelluskoodissa haettujen tietojen muuntamisesta objektiksi paljon helpompaa. JSON-dokumenttien kentät voivat myös erota toisistaan, ja ne voivat pitää sisällään merkkijonoja, tauluja ja muita erityyppisiä muuttujia ilman, että muuttujien tietotyyppejä olisi määritelty erikseen. MongoDB:ssä kaikki dokumentit ovat osana kokoelmaa. Kokoelmat eivät oletuksena estä erityyppisten dokumenttien tallentamista. Esimerkkikoodi 14 näyttää, miten aiemmin esitetyt asiakkaita ja tilauksia kuvaavat taulut voitaisiin tallentaa MongoDB-kantaan. Kaikki tiedot voi olla yhden dokumentin sisällä, koska MongoDB ei käytä relaatiomallia.

```
{
  "_id": 1,
  "nimi": "Liisa",
  "sahkoposti": "liisa@esimerkki.com",
  "tilaukset": [
    {
      "tilausId ": 101,
      "tilausPvm ": "2025-03-01"
    }
  ]
}
```

Esimerkkikoodi 14. Asiakkaiden ja tilausten tietojen tallentaminen MongoDB-tietokantaan

4 Suunnittelu

Tässä luvussa suunnitellaan, miten järjestelmän tulisi toimia ja miten eri teknologioita tullaan hyödyntämään siinä. Kuva 4 esittää yleisesti järjestelmän toimintaa.



Kuva 4. Kaavio, joka kuvastaa järjestelmän toimintaa

4.1 Koekysymykset

Koetta varten tarvitaan koekysymyksiä. Kysymykset ovat kaikki monivalintakysymyksiä, joten jokaiselle kysymykselle tarvitaan yksi oikea vastaus ja kolme

väärää vastausta. Kuva 5 on kuvakaappaus yhdestä kysymyksestä JSON-muodossa.

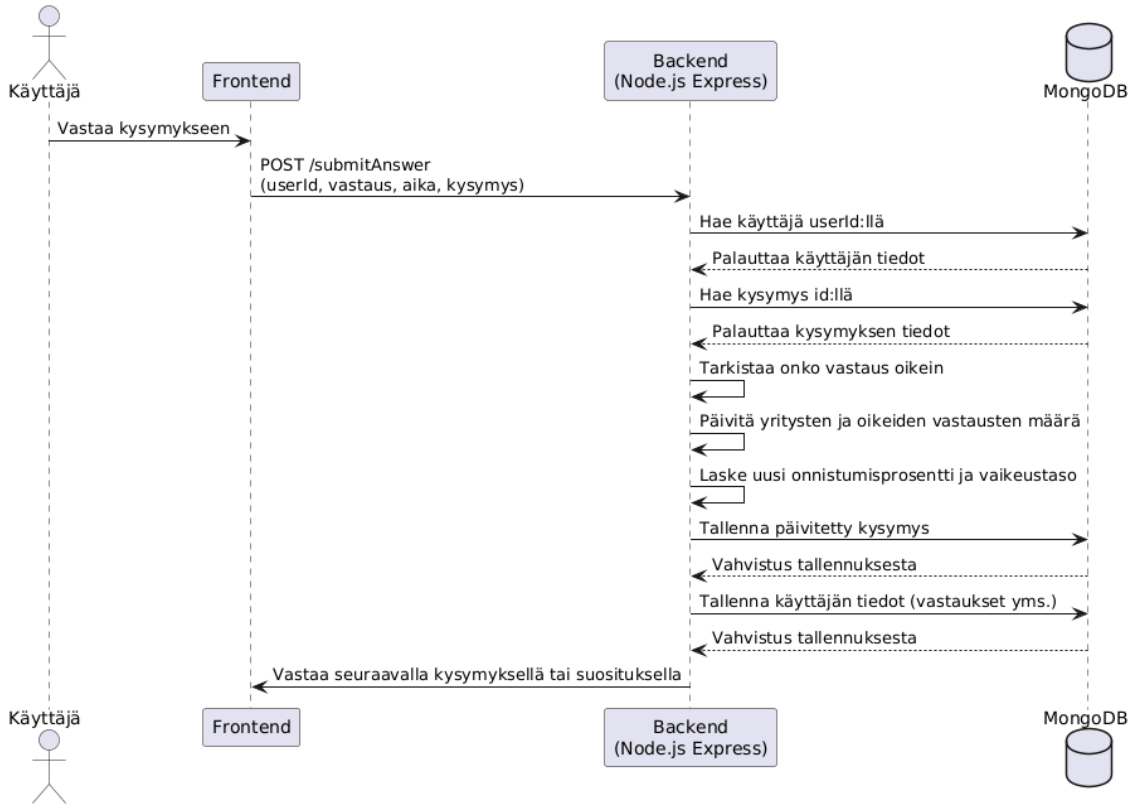
```
{
  category: 1,
  questionNo: 1,
  difficulty: 1,
  question: "Which of the following best defines software engineering?",
  options: [
    "Writing code for software applications",
    "Applying engineering principles to software development",
    "Designing only the UI of software",
    "Developing hardware components",
  ],
  correctIndex: 1,
},
```

Kuva 5. Yksittäinen kysymysobjekti JSON-muodossa

Kysymykset tallennetaan tietokantaan. Jokaisesta kysymyksestä tallennetaan seuraavat attribuutit:

- kategoria
- vaikeustaso
- kysymys
- vastausvaihtoehdot taulukkona
- oikean vastauksen indeksi
- kuinka monta kertaa kysymys on näytetty
- kuinka moni on vastannut kysymykseen oikein.

Kysymyksen vaikeustasoa voidaan säätää dynaamisesti katsomalla, kuinka monta prosenttia kysymykseen vastanneista on vastannut siihen oikein. Vaikeustasoa tullaan käyttämään kriteerinä uuden kysymyksen valitsemisessa. Jos käyttäjä on pärjännyt hyvin, voidaan häneltä kysyä vaikeampia kysymyksiä. Heikommin pärjänneeltä käyttäjältä voidaan puolestaan kysyä helpompia kysymyksiä. Kuva 6 esittää sekvenssikaaviolla, miten prosessi toimii.



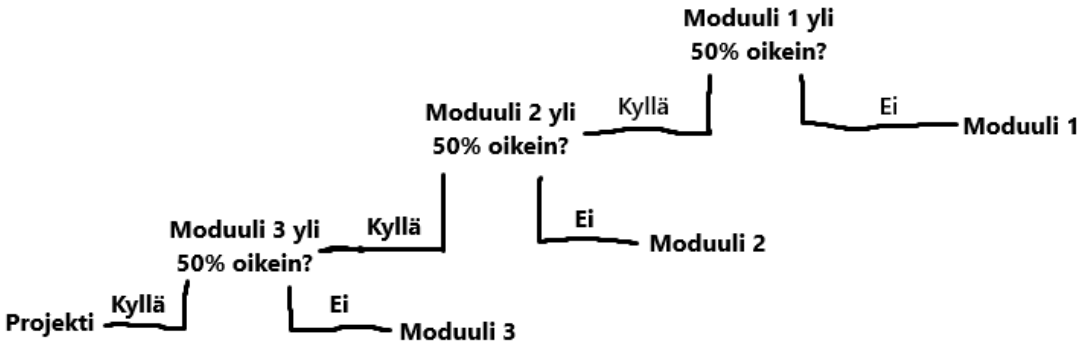
Kuva 6. Sekvenssikaavio kysymyksen vaikeustason päivittämisestä

Vaikeustaso tulee olemaan desimaaliluku, jonka arvo on väliltä 1–3.

4.2 Moduulin ennustaminen

Lopuksi järjestelmän on tarkoitus valita käyttäjän vastausten perusteella hänelle paras mahdollinen aloituskohta kurssille. Tähän tullaan hyödyntämään pythonin koneoppimiskirjastoja ja *Random Forest* -koneoppimismallia. [17] Malli opetetaan generoimalla dataa eri tasoisista käyttäjistä, koska oikeaa dataa ei vielä löydy. Sille syötetään käyttäjän oikein vastattujen kysymysten määrä jokaisesta kategoriasta sekä keskimääräinen vastausaika jokaisesta kategoriasta. Malli palauttaa näiden tietojen perusteella suosituksen aloitusmoduulista.

Random Forest soveltuu mainiosti juuri tähän tilanteeseen, koska se on kehitetty vastustamaan satunnaisia eroavaisuuksia datassa. Toisin sanottuna se osaa silti tarkasti ennustaa opiskelijan osaamistasoa, vaikka opiskelija olisi vahingossa vastannut muutamaan kysymykseen arvaamalla oikein. Malli käyttää logiikkana päätöspuita, joiden perusteella päädytään johonkin tiettyyn lopputulokseen. Kuva 5 näyttää, miltä yksi päätöspuista voisi mahdollisesti näyttää.



Kuva 7. Esimerkki mahdollisesta päätöspuusta moduulin ennustamiseen

Random Forest -malli generoi sille annettujen muuttujien pohjalta kymmeniä tai jopa satoja päätöspuita, joita käytetään lopullisen päätöksen tekemisessä. Tämä tekee siitä hyvän valinnan moduulin ennustamiseen, koska päätöspuita riittää moneen erilaiseen tilanteeseen. Vastaan saattaisi tulla esimerkiksi tilanne, jossa käyttäjä on saanut erinomaiset tulokset ensimmäisestä ja kolmannesta moduulista, mutta huonomman tuloksen toisesta moduulista. Tällaisessa tilanteessa ei ole yhtä selvää vastausta, mutta Random Forest voisi helposti generoida siihen säännön, esimerkiksi ”jos moduuli 2 on alle 50 % oikein, suosittele moduulia 2 ellei kaikki muut moduulit ole yli 90 % oikein, jolloin suosittele projektia”.

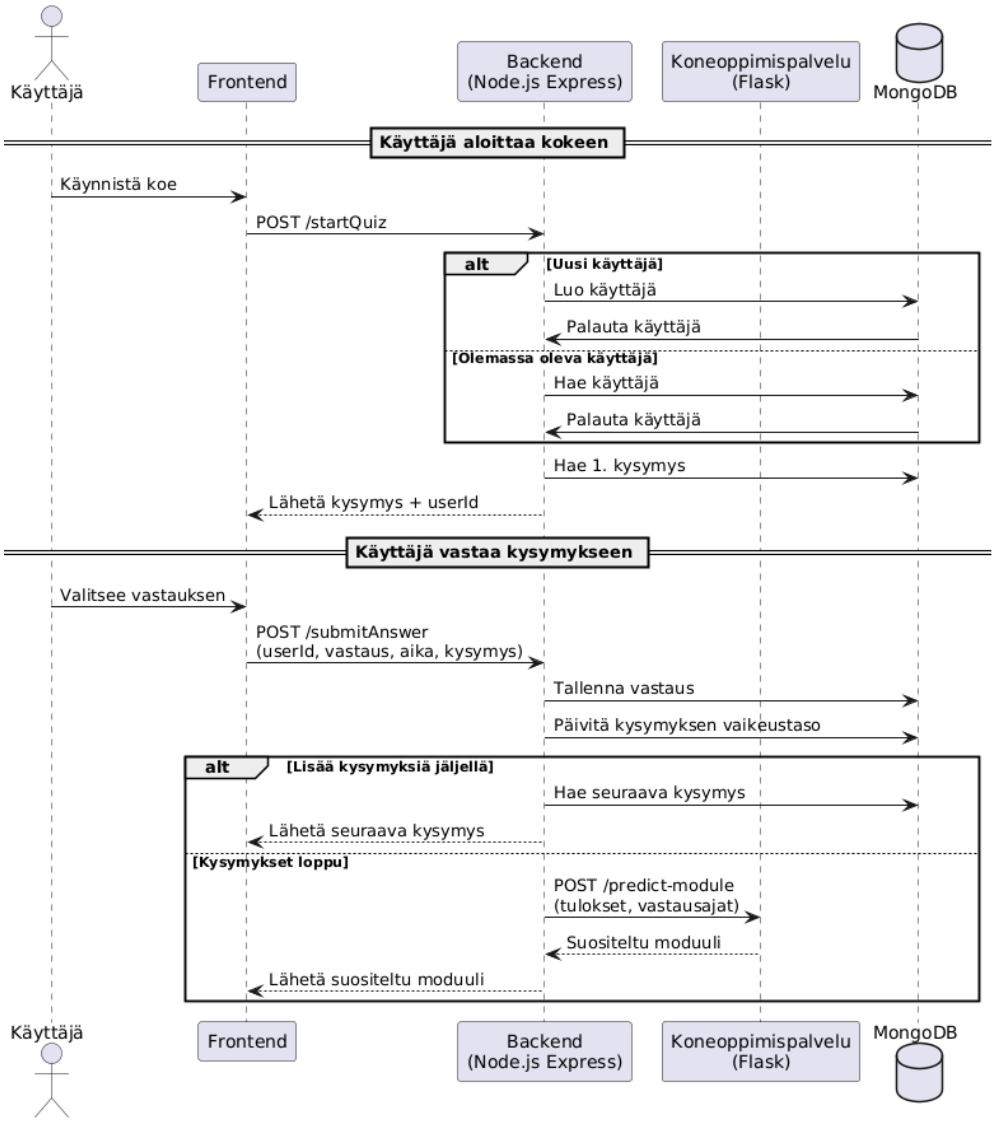
4.3 Backend-suunnittelu

Tarvitsemme myös tavan, jolla pystymme yhdistämään käyttöliittymä järjestelmäämme. Tähän tarkoitukseen tulemme käyttämään Node.js:n *Express*-kirjastoa. Expressillä voimme käsitellä HTTP-pyyntöjä luomalla päätepisteitä, joihin voidaan yhdistää käyttöliittymän puolelta.

Tässä projektissa tulemme käyttämään ainoastaan POST-metodia uuden kokeen aloittamiseen sekä kysymyksen vastaustietojen lähettämiseen. GET-metodille ei ole tarvetta, koska voimme hakea tarvittavat tiedot MongoDB:stä, ja palauttaa ne käyttöliittymälle POST-metodin palautusarvona.

4.4 Käyttöliittymän suunnittelu

Lopuksi tarvitsemme vielä käyttöliittymän, joka näyttää käyttäjälle kysymyksen sekä vastausvaihtoehdot. Käyttöliittymä luo ensin yhteyden palvelimemme päätepisteeseen, joka aloittaa kokeen. Palvelin hakee ensimmäisen kysymyksen ja luo samalla käyttäjän profiilin, ja palauttaa nämä tiedot käyttöliittymälle. Vastauksen saatua käyttöliittymä esittää kysymyksen käyttäjälle tekstinä, joka luo napin jokaiselle neljästä eri vastausvaihtoehdosta. Vastausnappia painettaessa käyttöliittymä lähettää käyttäjän tiedot, vastausajan, kysymyksen tiedot sekä käyttäjän vastauksen palvelimelle. Palvelin ottaa vastaan tiedot ja palauttaa käyttöliittymälle parhaan mahdollisen kysymyksen, jos kysymyksiä on vielä jäljellä. Jos kysymyksiä ei ole enää jäljellä, palvelin palauttaa moduulin, josta käyttäjä voi aloittaa kurssin. Käyttöliittymä tarkastelee palvelimen palauttamat tiedot ja katsoo, onko palvelin palauttanut seuraavaa kysymystä. Jos se ei ole, käyttäjälle voidaan esittää aloitusmoduuli. Jos seuraava kysymys löytyy, se esitetään käyttäjälle. Tämä jatkuu, kunnes kysymykset ovat loppuneet. Kuva 6 esittää sekvenssikaavion avulla järjestelmän toimintaa.



Kuva 8. Sekvenssikaavio järjestelmästä

Sekvenssikaavion avulla havainnollistamme järjestelmän toimintaa konkreettisesti.

5 Toteutus

5.1 Tietokannan pystyttäminen ja koekysymysten tallennus

Aloitamme toteutuksen pystyttämällä MongoDB tietokannan kysymyksiä varten. MongoDB Atlas on pilvitietokantapalvelu, jonka kautta voimme pystyttää

ilmaisen tietokantaklusterin pieniä projekteja varten. Luomme uuden klusterin projektia varten Atlas-palvelun kautta. Palvelu antaa meille linkin, jota voimme käyttää tietokantaan yhdistämisessä.

Seuraavaksi on aika viedä kaikki koekysymykset tietokantaan. Kysymykset on annettu valmiiksi tätä projektia varten, joten meidän ei tarvitse miettiä niiden sisältöä. Tarvitsemme tavan, jolla voimme luoda yhteyden tietokantaan. Tähän tarkoitukseen tulemme käyttämään Node.js:n *Mongoose*-kirjastoa. Mongoosen avulla voimme määritellä kaavion meidän yksittäistä koekysymystämme kuvaistavalle MongoDB-dokumentille. Luomme ensin kaavion, ja sen pohjalta objektin kysymykselle. Esimerkkikoodi 15 esittää, miten kaavio ja objekti luodaan.

```
const QuestionSchema = new mongoose.Schema({
  category: Number,
  questionNo: Number,
  difficulty: Number,
  question: String,
  options: [String],
  correctIndex: Number,
});
const Question = mongoose.model("Question", QuestionSchema);
```

Esimerkkikoodi 15. Kaavion ja objektin luominen koekysymyksiä varten

Tämän jälkeen tarvitsemme kaikkien kysymysten tiedot. Tätä varten voimme luoda JSON-muotoisen taulukko-objektin, joka sisältää jokaisen kysymyksen. Esimerkkikoodi 16 kuvaa yhtä kysymystä taulukossa.

```
{
    category: 1,
    questionNo: 1,
    difficulty: 1,
    question: "Which of the following best defines software
engineering?",
    options: [
      "Writing code for software applications",
      "Applying engineering principles to software development",
      "Designing only the UI of software",
      "Developing hardware components",
    ],
    correctIndex: 1,
}
```

Esimerkkikoodi 16. Koekysymystä kuvaava objekti JSON-muodossa

Kysymysten luomisen jälkeen voimme luoda yhteyden tietokantaan, käydä läpi kaikki kysymykset yksitellen ja lisätä niiden tiedot kantaan. Esimerkkikoodi 17 näyttää, miten yhteys luodaan ja miten kysymykset lisätään. Tässä tapauksessa kysymykset sisältävä JSON-taulukko on määritelty muuttujaksi ”questions”.

```
mongoose.connect("<connectionString>", {
  useNewUrlParser: true,
  useUnifiedTopology: true,
});
try {
    await Question.deleteMany({});
    await Question.insertMany(questions);
    mongoose.connection.close();
  } catch (error) {
    console.error("Error adding questions:", error);
    mongoose.connection.close();
  }
```

Esimerkkikoodi 17. Tietokantayhteyden luominen ja koekysymysten lisääminen

Voimme ajaa koodin Node.js-ajoympäristössä, mutta ennen ajamista meidän täytyy ladata mongoose. Esimerkkikoodi 18 näyttää, miten voimme ladata Mongoosen ja ajaa koodin käyttäen komentoriviä, kun koodi on tallennettu tiedostoon ”addquestions.js”.

```
npm i mongoose
node addquestions.js
```

Esimerkkikoodi 18. Pakettien asentaminen ja koodin ajaminen Node.js-ajoympäristössä

Kaikki koekysymykset löytyvät liitteestä 1.

5.2 Backend-kehitys

Kun kysymykset ovat tallennettu tietokantaan, voimme aloittaa palvelimen tekemisen. Aloitamme alustamalla uuden Node-projektin asentamalla TypeScriptin käyttöön tarvittavat kirjastot, ja tekemällä tarvittavat muutokset konfigurointitiedostoihin. Esimerkkikoodi 19 näyttää komentoriviä käyttäen, miten alustamme projektin ja asennamme TypeScriptiin tarvittavan paketin komentorivillä

```
npm init -y
npm I –save-dev typescript
```

Esimerkkikoodi 19. Node.js-projektin alustaminen ja TypeScript-kirjaston asentaminen

Seuraavaksi tarvitsemme konfiguraatiotiedoston TypeScriptille. Luomme projektikansion sisälle uuden ”tsconfig.json”-tiedoston, ja määrittelemme sinne TypeScriptin asetukset JSON-muodossa. Esimerkkikoodi 20 näyttää tsconfig.json-tiedoston sisällön

```
{
    "compilerOptions": {
      "target": "ES6",
      "module": "CommonJS",
      "outDir": "./dist",
      "rootDir": "./src",
      "esModuleInterop": true,
      "strict": true
    }
}
```

Esimerkkikoodi 20. TypeScriptin asetusten määritteleminen

Joudumme myös muuttamaan Node.js:n generoiman ”package.json”-tiedoston sisältöä. Tiedosto määrittelee tietoja projektista JSON-muodossa. Lisäämme tiedoston ”scripts”-objektiin uuden merkinnän ”dev”. Objekti määrittelee komentoja, joita voidaan ajaa tietyllä sanalla. Tämä tekee projektin käynnistämisestä helpompaa (esimerkkikoodi 21).

```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "dev": "nodemon src/index.ts"
}
```

Esimerkkikoodi 21. Node.js:n konfigurointitiedoston mukauttaminen

Olemme nyt valmiita ajamaan TypeScript-koodia. Luomme projektin sisälle kansion ”src”, jonka sisälle luomme tiedoston ”index.ts”. Tämä tiedosto tulee olemaan oletustiedosto, joka ajetaan, kun projekti käynnistetään.

Aloitamme kehityksen alustamalla uuden express-sovelluksen ja luomalla yhteyden tietokantaan mongoose-kirjastoa käyttäen. Käytämme myös dotenv-

kirjastoa arkaluonteisien muuttujien, kuten tietokannan yhdistyslinkin, lukemiseen erillisestä tiedostosta (esimerkkikoodi 22).

```
dotenv.config();

const app = express();
const PORT = process.env.PORT || 5000;
const QUIZ_QUESTIONS_AMOUNT=15;
app.use(bodyParser.json());
app.use(cors());

const mongoUri = process.env.MONGO_URI;

if (!mongoUri) {
  throw new Error("Missing MONGO_URI in environment variables.");
}

mongoose.connect(mongoUri)
  .then(() => console.log("Connected to MongoDB"))
  .catch((err) => console.error("MongoDB connection error:", err));
```

Esimerkkikoodi 22. Express-sovelluksen alustaminen ja MongoDB-yhteyden luominen

Seuraavaksi luomme käyttäjille ja kysymyksille mongoose-kehykset, mallit sekä tietotyypit kehysten pohjalta. Tarvitsemme malleja, kun haluamme hakea tietoa tietokannasta, ja tietotyypit ovat tarpeellisia funktioiden palautusarvojen määrittelemiseen (esimerkkikoodi 23).

```
const UserSchema = new mongoose.Schema({
    name: String,
    answers: [{questionId: String, answerIndex: Number,
                correct: Boolean, timeTaken: Number}],
    scores: {category1: Number, category2: Number, category3: Number},
    questionsAsked: {category1: Number, category2: Number, category3:
Number},
})

const User = mongoose.model("User", UserSchema);

type UserType = HydratedDocument<InferSchemaType<typeof UserSchema>>;

const QuestionSchema = new mongoose.Schema({
    category: Number,
    questionNo: Number,
    difficulty: Number,
    question: String,
    totalAttempts: Number,
    correctAttempts: Number,
    options: [String],
    correctIndex: Number,
})

const Question = mongoose.model("Question", QuestionSchema);
type QuestionType = HydratedDocument<InferSchemaType<typeof
QuestionSchema>>;
```

Esimerkkikoodi 23. Mongoose-kehysten, mallien ja tietotyyppien luominen käyttäjille sekä kysymyksille

Tarvitsemme funktion, joka hakee käyttäjälle uuden kysymyksen aiempien vastausten perusteella. Funktio tarvitsee listan saatavilla olevista kysymyksistä. Jokaisesta moduulista tullaan kysymään tasan viisi kysymystä, joten voimme jättää pois moduulit, joista on jo kysytty viisi kysymystä. Haluamme myös jättää pois kysymykset, jotka on jo kysytty aiemmin käyttäjältä. Tämän jälkeen haluamme valita parhaan mahdollisen kysymyksen saatavilla olevien kysymysten joukosta.

Jokaisesta kysymyksestä on kirjattuna sen vaikeustaso desimaalilukuna asteikolla 1–3. Voimme käyttää sitä seuraavan kysymyksen etsimisessä laskemalla käyttäjälle optimaalisen vaikeustason hänen aiempien vastauksiensa perusteella (esimerkkikoodi 24).

```
let totalAnswered = user.answers.length || 1;
let correctAnswers = user.answers.filter((a) => a.correct).length;
let correctnessRate = correctAnswers / totalAnswered;

let desiredDifficulty = 1.0 + correctnessRate * 2.0;
```

Esimerkkikoodi 24. Seuraavan koekysymyksen optimaalisen vaikeustason laskeminen

Käytämme optimaalista vaikeustasoa seuraavan parhaan kysymyksen etsimisessä. Käymme yksitellen läpi jokaisen saatavilla olevan kysymyksen, ja laskemme sen onnistumisasteen, jonka saamme ottamalla kysymyksen kaikkien vastausten määrän ja jakamalla sen oikeiden vastausten määrällä. Tämän jälkeen laskemme kysymykselle ”explorationFactor”-attribuutin jota käytämme varmistaaksemme, että vähän kysyttyjä kysymyksiä ei jätetä pois. Attribuutin arvo laskee kysymyksen vastausten määrän kasvaessa keskiarvoon verrattuna. Lisäämme sen arvon onnistumisasteeseen. Tämä muodostaa kysymyksen lasketun UCB-arvon. Säädämme vielä UCB-arvoa sen perusteella, miten paljon se eroaa aiemmin lasketusta optimaalisesta vaikeustasosta. Jos kysymyksen UCB-arvo on suurin tähän mennessä, se merkitään valituksi kysymykseksi. Tämä toistetaan jokaiselle saatavilla olevalle kysymykselle, kunnes kaikki kysymykset on käyty läpi (esimerkkikoodi 25).

```
availableQuestions.forEach((q) => {
  let successRate = (q.correctAttempts ?? 0) / (q.totalAttempts || 1);
  let explorationFactor = Math.sqrt(
    Math.log(totalAttempts + 1) / (q.totalAttempts || 1)
  );
  let ucbValue = successRate + explorationFactor;

  let difficulty = q.difficulty ?? 1.0;
  let difficultyPenalty = Math.abs(difficulty - desiredDifficulty);
  let finalScore = ucbValue - difficultyPenalty * 0.2;

  if (finalScore > maxScore) {
    maxScore = finalScore;
    selectedQuestion = q;
  }
});
```

Esimerkkikoodi 25. Parhaan mahdollisen kysymyksen valitseminen

Kysymysten vaikeustasot ovat tällä hetkellä valmiiksi määriteltyjä, eivätkä ne muutu. Haluamme täten myös luoda funktion, jolla voitaisiin dynaamisesti päivittää kysymyksen vaikeustasoa, kun uusia vastauksia otetaan vastaan. Tämä voidaan toteuttaa laskemalla kysymyksen onnistumisasteen. Jos onnistumisaste on yli 80 %, voidaan kysymyksen vaikeustasoa kasvattaa hiljalleen lähemmäksi arvoa 1 (helppo). Jos onnistumisaste on alle 50 %, voidaan sitä siirtää lähemmäksi arvoa 3 (vaikea). Emme kuitenkaan halua, että vaikeustaso muuttuu liian nopeasti. Tämän takia käytämme eksponentiaalista tasoitusmallia vaikeustason säätämisessä (esimerkkikoodi 26).

```
let totalAttempts = (question.totalAttempts ?? 0) + 1;
let correctAttempts = question.correctAttempts ?? 0;
if (wasCorrect) correctAttempts++;
let correctnessRatio = correctAttempts / totalAttempts;
let difficulty = question.difficulty || 1.0;
const learningRate = 0.1;

if (correctnessRatio > 0.8) {
  difficulty = Math.max(
    1.0,
    difficulty - learningRate * (difficulty - 1.0)
  );
} else if (correctnessRatio < 0.5) {
  difficulty = Math.min(
    3.0,
    difficulty + learningRate * (3.0 - difficulty)
  );
}
```

Esimerkkikoodi 26. Kysymyksen vaikeustason säätäminen käyttäen eksponentiaalista tasoitusta

Kysymysten esittämisen lisäksi haluamme lopuksi ennustaa käyttäjälle moduulin, josta hänen olisi hyvä aloittaa kurssi. Moduulin ennustamiseen aiomme käyttää koneoppimista, joten se toteutetaan pythonin avulla.

Koneoppimista varten tarvitaan koneoppimismalli ja mallia varten tarvitaan dataa. Koska projekti ei ole vielä ollut varsinaisessa käytössä, joudumme generoimaan datan itse. Tulevaisuudessa malli tulisi opettaa uudelleen, kun varsinaista käyttäjädataa on riittävästi tietokannassa.

Generointia varten määrittelemme ensin parametrit, jotka määrittelevät sen laajuuden. Tässä tapauksessa haluamme generoida 1000 käyttäjän koetulokset.

Kysymyksiä tulee olemaan kolmesta eri moduulista, ja jokaisesta moduulista tullaan kysymään viisi kysymystä. Jokaiselle käyttäjälle arvotaan osaamistaso väliltä 0.2–0.95. Jokaista kysymystä kohden arvotaan sen tulos (oikein/väärin). Tuloksen arpomiseen käytetään binomijakaumaa, jossa osaamistaso toimii todennäköisyytenä. Tämän lisäksi jokaiselle kysymykselle arvotaan vastausaika väliltä 1–35 sekuntia. Mitä korkeampi käyttäjän arvottu osaamistaso on, sitä todennäköisemmin vastausajat ovat lyhyempiä (esimerkkikoodi 27).

```
NUM_USERS = 1000
QUESTIONS_PER_CATEGORY = 5
CATEGORIES = [1, 2, 3]
TOTAL_QUESTIONS = QUESTIONS_PER_CATEGORY * len(CATEGORIES)
data = []
for _ in range(NUM_USERS):
    scores = {}
    questions_asked = {}
    for cat in CATEGORIES:
        questions_asked[f'category{cat}'] = QUESTIONS_PER_CATEGORY
        skill = np.random.uniform(0.2, 0.95)
        correct = np.random.binomial(QUESTIONS_PER_CATEGORY, skill)
        scores[f'category{cat}'] = correct
        time_taken = np.random.normal(loc=10 - skill * 5, scale=1.5,
                                      size=QUESTIONS_PER_CATEGORY)
        avg_time = max(1.0, min(30.0, np.mean(time_taken)))
        scores[f'category{cat}_time'] = avg_time
```

Esimerkkikoodi 27. Käyttäjän koetulosten generointi pythonilla

Koetulosten generoinnin jälkeen laskemme oikeiden vastausten keskiarvon jokaiselle moduulille. Etsimme heikoimman moduulin keskiarvon perusteella, ja jos sen keskiarvo on alle 60 % oikein, valitsemme sen suositelluksi moduuliksi. Jos sen keskiarvo on 60 % oikein tai enemmän, valitsemme moduulin 4, joka tässä tapauksessa tarkoittaa, että käyttäjän kannattaisi keskittyä projektin tekemiseen, koska moduuleja on vain kolme. Lisäämme lopuksi käyttäjän tiedot taulukkoon ja toistamme prosessin jokaiselle käyttäjälle, kunnes taulukko sisältää kaikkien käyttäjien tiedot (esimerkkikoodi 28).

```
ratios = [scores[f'category{cat}'] / questions_asked[f'category{cat}']
          for cat in CATEGORIES]
weakest_category = np.argmin(ratios) + 1
min_ratio = ratios[weakest_category - 1]

if min_ratio < 0.6:
    recommended_module = weakest_category
else:
    recommended_module = 4
row = {
    **{f'score_cat{c}': scores[f'category{c}'] for c in CATEGORIES},
    **{f'avg_time_cat{c}': scores[f'category{c}_time'] for c in
CATEGORIES},
    **{f'asked_cat{c}': questions_asked[f'category{c}'] for c in
CATEGORIES},
    'recommended_module': recommended_module
}

 data.append(row)
```

Esimerkkikoodi 28. Moduulin valitseminen käyttäjän koetulosten perusteella ja datan tallennus taulukkoon

Kun data on generoitu, voimme muuntaa sen Pyhonin pandas-kirjaston DataFrame-muotoon ja syöttää sen koneoppimismallille opettamista varten. Merkitsemme datasta ominaisuudet, joita analyysiin halutaan käyttää, ja lopputuloksen, jota halutaan ennustaa. Tässä tapauksessa ominaisuudet ovat jokaisen kategorian tulokset sekä keskimääräiset vastausajat ja haluamme ennustaa suositeltua moduulia. Otamme opetusdatasta 20 % pois, ja määrittelemme sen testidataksi *train_test_split*-funktiota käyttäen. Testidatalla voimme testata, miten tarkasti malli osaa oikeasti ennustaa moduulia. Tulemme perehtymään testituloksiin tarkemmin seuraavassa kappaleessa. Lopuksi määrittelemme malliksi Random Forestin, ja lisäämme parametriksi n_estimators, jonka arvoksi asetamme 100. Tämä tarkoittaa, että malli tulee käyttämään 100 päätöspuuta ennustuksessa. Lopuksi tallennamme mallin tiedostoksi, jota voimme käyttää palvelimella (esimerkkikoodi 29).

```
df = generate_user_data(NUM_USERS)

X = df[['score_cat1', 'score_cat2', 'score_cat3',
        'avg_time_cat1', 'avg_time_cat2', 'avg_time_cat3']]
y = df['recommended_module']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=26)

model = RandomForestClassifier(n_estimators=100, random_state=26)
model.fit(X_train, y_train)

prediction = model.predict(X_test)
acc = accuracy_score(y_test, prediction)
print(f"Accuracy: {acc:.2f}")

joblib.dump(model, 'recommended_module_model.pkl')
```

Esimerkkikoodi 29. Random Forest -koneoppimisalgoritmin opettaminen generoidun datan avulla.

Kun mallista on generoitu tiedosto, voimme pystyttää palvelimen ja luoda päätepisteen ennustuksen tekemistä varten Pythonin *flask*-kirjastolla. Päätepiste ottaa vastaan käyttäjän jokaisen moduulin koetulokset sekä keskimääräiset vastausajat JSON-muodossa ja palauttaa ennustetun moduulin palautuksena (esimerkkikoodi 30).

```
model = joblib.load('recommended_module_model.pkl')

@app.route('/predict-module', methods=['POST'])
def predict_module():
    data = request.get_json()

    try:
        scores = [
            data.get('score_cat1', 0),
            data.get('score_cat2', 0),
            data.get('score_cat3', 0),
            data.get('avg_time_cat1', 15),
            data.get('avg_time_cat2', 15),
            data.get('avg_time_cat3', 15),
        ]
        prediction = model.predict([scores])
        return jsonify({'recommended_module': int(prediction[0])})
    except Exception as e:
        return jsonify({'error': str(e)}), 500
```

```
const { userId } = req.body;
if (!userId) {
  const newUser = await createNewUser();
  const question = await Question.findOne({ category: 1, questionNo: 1
});
  if (!question) {
    return res.status(404).json({ message: "Question not found" });
  }

  return res.json({
    userId: newUser._id,
    question,
  });
}
```

Esimerkkikoodi 30. Päätepisteen luominen koneoppimismallia varten

Voimme nyt palata TypeScript-koodiimme. Käyttöliittymälle tullaan luomaan suora yhteys ainoastaan TypeScript-palvelimelle, joten tarvitsemme funktion, joka ottaa syötteenä käyttäjän ja hakee sille aloitusmoduulin yhdistämällä Flask-palvelimeen.

Aloitamme hakemalla kaikki tarvittavat tiedot käyttäjältä (esimerkkikoodi 31).

```
const categories = [1, 2, 3];
const scores: Record<string, number> = {};
const times: Record<string, number[]> = {};
for (const cat of categories) {
  scores[`score_cat${cat}`] =
    user.scores?.[`category${cat}` as keyof typeof user.scores] ?? 0;
  times[`cat${cat}`] = [];
}
for (const ans of user.answers) {
  const question = await Question.findById(ans.questionId);
  if (question) {
    const cat = question.category;
    if (typeof ans.timeTaken === "number") {
      times[`cat${cat}`].push(ans.timeTaken);
    }
  }
}
 const avgTimes: Record<string, number> = {};
 for (const cat of categories) {
  const timesArray = times[`cat${cat}`];
  avgTimes[`avg_time_cat${cat}`] =
    timesArray.length > 0
      ? timesArray.reduce((a, b) => a + b, 0) / timesArray.length
      : 15;
}
```

Esimerkkikoodi 31. Käyttäjän koetulosten ja vastausaikojen kerääminen taulukkoihin

Taulukot voidaan tämän jälkeen yhdistää osaksi yhtä muuttujaa ja lähettää Flask-palvelimelle käyttäen axios-kirjastoa, jonka jälkeen voimme palauttaa palvelimelta saamamme vastauksen (esimerkkikoodi 32).

```
const payload = {
  ...scores,
  ...avgTimes,
};

const response = await axios.post(
  "<SERVER_URL>/predict-module",
  payload
);
const recommendedModule = response.data.recommended_module;
return recommendedModule;
```

Esimerkkikoodi 32. Käyttäjän tietojen yhdistäminen ja lähettäminen flask-palvelimelle.

Lopulta voimme luoda varsinaiset päätepisteet, joihin käyttöliittymä tulee yhdistämään saadakseen tarvittavat tiedot. Tarvitsemme päätepisteen uuden kokeen aloittamiseen sekä päätepisteen uuden vastauksen kirjaamiseen.

Aloitamme luomalla päätepisteen, jonka kutsuminen aloittaa uuden kokeen. Päätepiste ottaa syötteenä mahdollisen olemassa olevan käyttäjän id:n. Jos käyttäjän id:tä ei ole syötetty, voidaan luoda uusi käyttäjä ja aloittaa koe kysymyksestä 1. Uuden käyttäjän id ja kysymysobjekti palautetaan käyttöliittymälle. Jos syötteenä on lähetetty käyttäjän id, ja kyseisellä id-arvolla löytyy tietokannasta käyttäjä, voidaan koetta jatkaa siitä pisteestä, mihin käyttäjä oli jäänyt. Esimerkkikoodi 33 esittää päätepisteen rakenteen.

```
app.post(
  "/startQuiz",
  async (req: Request, res: Response, next: NextFunction):
Promise<any> => {
    const { userId } = req.body;

    if (!userId) {
      const newUser = await createNewUser();
      const question = await Question.findOne({ category: 1,
questionNo: 1 });
      if (!question) {
        return res.status(404).json({ message: "Question not found"
});
      }

      return res.json({
        userId: newUser._id,
        question,
      });
    }

    try {
      const user = await User.findById(userId);
      if (!user) return res.status(404).json({ message: "User not
found" });

      if (user.answers.length >= QUIZ_QUESTIONS_AMOUNT) {
        let recommendedModule = await getRecommendedModule(user);
        return res.json({
          question: null,
          done: true,
          recommendedModule: recommendedModule,
        });
      }

      let nextQuestion = await getNextQuestion(user);

      return res.json({ question: nextQuestion });
    } catch (error: any) {
      console.error("Error in /startQuiz:", error);
      return res.status(500).json({ message: "Failed to start quiz"
});
    }
  }
);
```

Esimerkkikoodi 33. Express-kirjastoa käyttämällä luotu päätepiste, joka aloittaa uuden kokeen

Ottamalla mallia esimerkkikoodista 33 voimme luoda päätepisteen uuden vastauksen kirjaamista varten. Se ottaa syötteenä käyttäjän id:n, vastauksen indeksin, kysymyksen sekä vastausajan. Vastausindeksi on käyttäjän valitseman vastauksen indeksi asteikolla 0–3. Jokaisella kysymyksellä on tasan neljä vastausvaihtoehtoa, joten esimerkiksi vastausindeksin arvo 2 kuvaisi

vastausvaihtoehtolistan kolmatta arvoa. Käyttäjän vastausindeksiä voidaan verrata kysymyksen oikeaan vastausindeksiin, ja kasvattaa käyttäjän kyseisen moduulin oikeiden vastausten määrää, jos ne ovat samat. Tämän jälkeen kysymys voidaan tallentaa käyttäjän vastattujen kysymysten listalle, ja kysymysten vaikeustasoja voidaan päivittää kutsumalla aiemmin määriteltyä niiden päivittämiseen luotua funktiota. Lopuksi käyttäjän tiedot päivitetään tietokantaan, ja hänelle etsitään uusi kysymys hyödyntäen aiemmin luomaamme funktiota. Jos käyttäjä on vastannut kaikkiin kysymyksiin, hänelle etsitään suositeltu moduuli uuden kysymyksen sijasta. Esimerkkikoodi 34 näyttää, miten aiemmin tekemiämme funktioita hyödynnetään tässä tilanteessa.

```
await updateQuestionDifficulty(question._id.toString(), isCorrect);
await user.save();
if (user.answers.length >= QUIZ_QUESTIONS_AMOUNT) {
  let recommendedModule = await getRecommendedModule(user);
  return res.json({
    question: null,
    done: true,
    recommendedModule: recommendedModule,
  });
}

let nextQuestion = await getNextQuestion(user);
res.json({
  question: nextQuestion,
  done: user.answers.length >= QUIZ_QUESTIONS_AMOUNT,
});
```

Esimerkkikoodi 34. Uuden kysymyksen tai aloitusmoduulin hakeminen käyttäjälle

5.3 Käyttöliittymän kehitys

Lopuksi luomme käyttöliittymän. Tähän tulemme käyttämään Reactia sekä Tailwind-kirjastoa. Tailwind antaa meidän muotoilla sivua käyttämättä erillistä CSS-tiedostoa. Luomme käyttöliittymälle uuden projektin kirjoittamalla seuraavat komennot komentoriville (esimerkkikoodi 35).

```
npm create vite@latest quizbot-client -- --template react
cd quizbot-client
npm i
npm i tailwindcss @tailwindcss/vite
```

Esimerkkikoodi 35. React-projektin luominen ja tailwindcss-kirjaston asentaminen

Komennot luovat uuden React-projektin nimeltään ”quizbot-client” käyttäen Vite-työkalua, jonka jälkeen projektin sisälle asennetaan tailwindcss-kirjasto sekä kaikki muut projektin ajamiseen tarvittavat kirjastot. Projektin sisältä löytyy tiedosto vite.config.js, jonka sisältöä joudumme muuttamaan, jotta pystymme käyttämään tailwindiä. Esimerkkikoodi 36 kuvaa vite.config.js-tiedoston sisältöä muutosten jälkeen.

```
import { defineConfig } from 'vite'
import tailwindcss from '@tailwindcss/vite'
import react from '@vitejs/plugin-react'

// https://vite.dev/config/
export default defineConfig({
  plugins: [
    react(),
    tailwindcss(),
],
})
```

Esimerkkikoodi 36. Tailwind-kirjaston lisääminen vite.config.js-tiedostoon. Kirjastoon luodaan viite käyttämällä import-komentoa, ja se lisätään osaksi plugins-taulukkoa.

Nyt voimme aloittaa käyttöliittymän kehittämisen Reactilla. Aloitamme luomalla tarvittavat muuttujat käyttämällä useState-koukkua. Taulukko 4 esittelee luomamme muuttujat ja niiden käyttötarkoitukset.

Taulukko 4. Käyttöliittymän muuttujien nimet, käyttötarkoitukset ja oletusarvot

Muuttuja	**Käyttötarkoitus**	**Oletusarvo**
userId	Käyttäjän ID-arvon säilyttäminen. Tällä erotellaan käyttäjät toisistaan	null

Muuttuja	Käyttötarkoitus	Oletusarvo
question	Objekti, joka sisältää kysymyksen tiedot	null
loading	Totuusarvomuuttuja, joka kertoo, jos seuraavaa kysymystä ollaan lataamassa	true
done	Totuusarvomuuttuja, joka kertoo, onko koe valmis	false
recommendedModule	Sisältää arvon käyttäjälle valitusta moduulista	0
startTime	Sisältää tarkan ajan, jolloin viime kysymys on aloitettu	null

Kun sivu ladataan, meidän täytyy joko aloittaa uusi koe tai hakea mahdollisen olemassa olevan käyttäjän tiedot ja jatkaa koetta siitä, mihin käyttäjä on viimeksi jäänyt. Tätä varten voimme käyttää Reactin useEffect-koukkua, joka ajetaan kerran aina, kun sivu ladataan. Katsomme ensin, löytyykö selaimesta aiemman käyttäjän tallennettuja tietoja. Jos ei löydy, voimme yhdistää aiemmin luomaamme päätepisteeseen, joka luo automaattisesti uuden käyttäjän ja palauttaa meille ensimmäisen kysymyksen (esimerkkikoodi 37).

```
let storedUserId = localStorage.getItem("userId");
      if (!storedUserId || storedUserId.length !== 24) {
        try {
          const response = await
                              axios.post("<BACKEND_URL>/startQuiz", {});
          storedUserId = response.data.userId;
          localStorage.setItem("userId", storedUserId);
          setUserId(storedUserId);
          setQuestion(response.data.question);
        } catch (error) {
          console.error("Error creating user:", error);
        }
      }
```

Esimerkkikoodi 37. Uuden käyttäjän luominen ja ensimmäisen kysymyksen hakeminen

Jos käyttäjä on jo olemassa, voimme hakea sen tiedot samalla tavalla, liittäen käyttäjän id:n mukaan pyyntöön (esimerkkikoodi 38).

```
setUserId(storedUserId);
try {
  const response = await axios.post("http://localhost:5000/startQuiz",
{
    userId: storedUserId,
  });
  setQuestion(response.data.question);
  setDone(response.data.done);
  setRecommendedModule(response.data.recommendedModule);
} catch (error) {
  console.error("Error fetching question:", error);
}
```

Esimerkkikoodi 38. Olemassa olevan käyttäjän hakeminen päätepisteeltä

Käytämme myös useEffect-koukkua kysymyksen vastausajan mittaamiseen. Voimme luoda useEffect-koukun, joka kutsutaan aina, kun kysymyksen sisältävän useState-muuttujan tieto muuttuu. Tässä tilanteessa voimme asettaa startTime-muuttujan arvoksi tämänhetkisen ajan. Kun käyttäjä vastaa kysymykseen, voimme laskea sen hetkisen ajan ja tallennetun startTime-muuttujan ajan erotuksen, jolloin saamme kysymyksen vastausajan. Esimerkkikoodi 39 näyttää, miten useEffect-koukulla voidaan tallentaa vastausajan aloituskohta.

```
useEffect(() => {
  if (question) {
    setStartTime(Date.now());
  }
}, [question]);
```

Esimerkkikoodi 39. Reactin useEffect-koukun käyttäminen kysymyksen vastausajan laskemisen aloittamisessa.

Käyttöliittymän täytyy myös pystyä lähettämään käyttäjän vastauksia palvelimelle. Tätä varten voimme tehdä funktion, joka ottaa parametrina vastauksen indeksin ja lähettää tarvittavat tiedot POST-pyynnöllä palvelimelle. Vastauksen lähettämisen jälkeen odotetaan palvelimen vastausta ja asetetaan sen perusteella joko uusi kysymys tai näytetään käyttäjälle aloitusmoduuli. Funktio myös laskee sitä kutsuttaessa kysymyksen vastausajan. Esimerkkikoodi 40 näyttää valmiin funktion.

```
const submitAnswer = (answerIndex) => {
 const elapsedTime = Date.now() - startTime; // Elapsed time in ms
 setLoading(true);
 axios
   .post("http://localhost:5000/submitAnswer", {
     userId,
     answerIndex,
     currentQuestion: question,
     timeTaken: elapsedTime,
   })
   .then((response) => {
     setQuestion(response.data.question);
     setDone(response.data.done);
      setRecommendedModule(response.data.recommendedModule);
     setLoading(false);
   })
   .catch((error) => {
     console.error("Error submitting answer:", error);
   });
 };
```

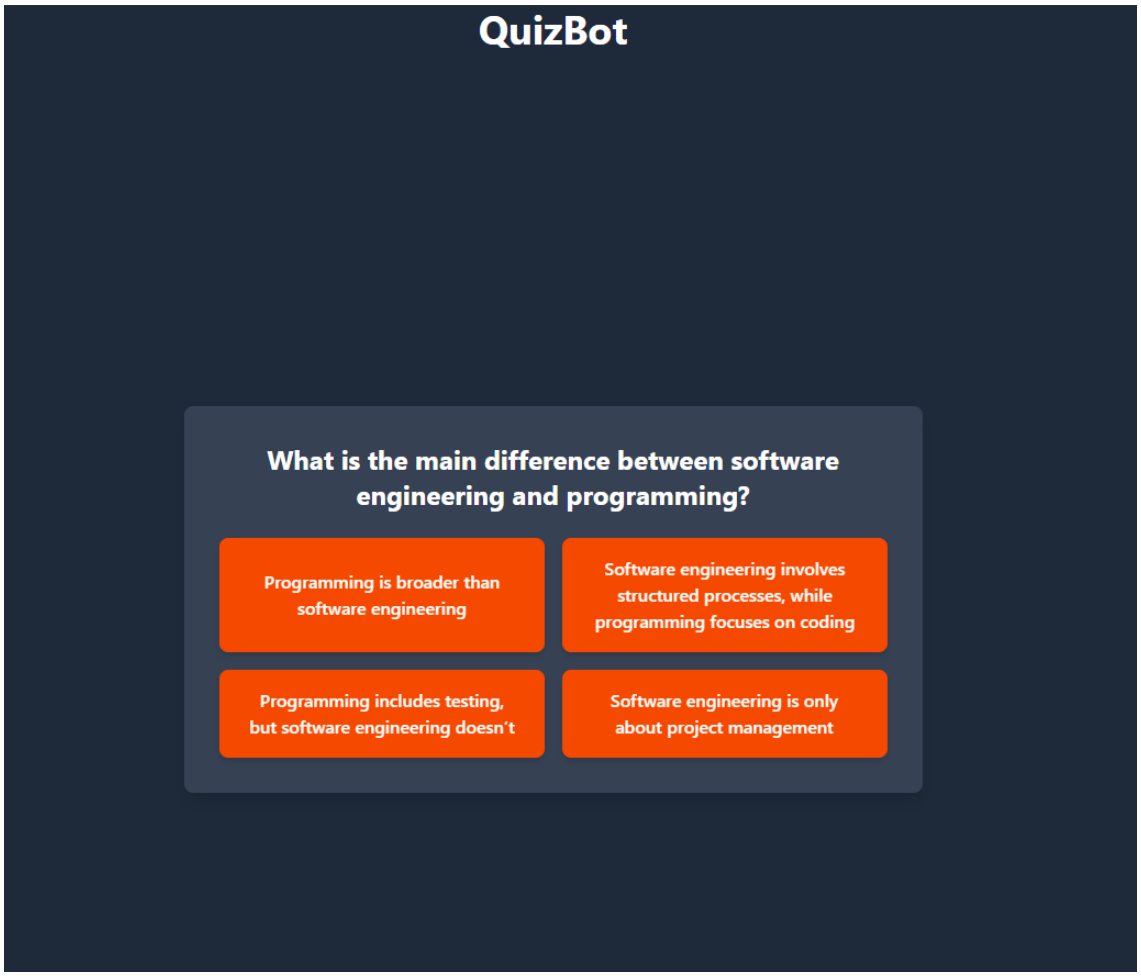
Esimerkkikoodi 40. Funktio, joka lähettää käyttäjän vastauksen palvelimelle ja samalla hakee uuden kysymyksen

Tarvitsemme vielä napit, jolla käyttäjä voi vastata kysymyksiin. Kun uusi kysymys on haettu palvelimelta, voimme generoida niiden elementit käyttäen palvelimelta haettuja tietoja. Jokaista palvelimelta haettua vastausvaihtoehtoa kohden luodaan uusi nappi, joka kutsuu vastauksen lähettämiseen tehtyä funktiota sitä painettaessa. Esimerkkikoodi 41 esittää, miten napit luodaan.

```
{question.options.map((option, index) => (
 <button
  key={index}
  onClick={() => submitAnswer(index)}
  className="py-4 px-6 bg-orange-600 hover:bg-orange-700 text-white
  font-semibold rounded-lg shadow-md transform transition-all
  duration-300 hover:scale-105"
 >
   {option}
 </button>
))}
```

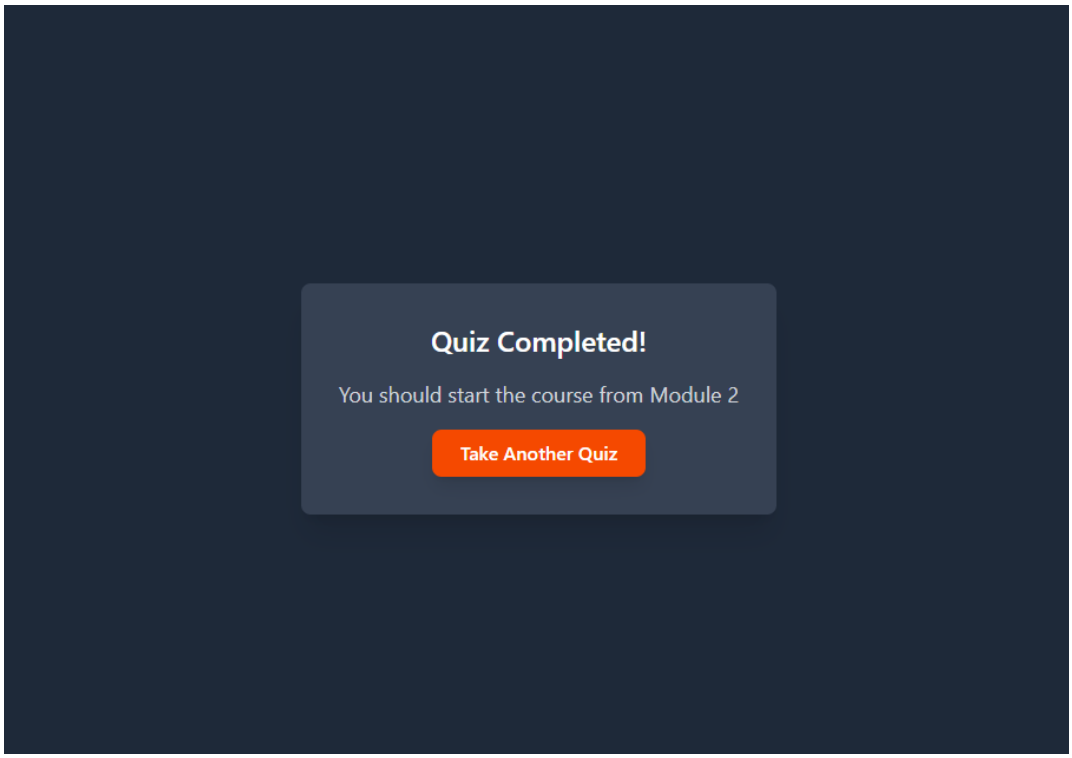
Esimerkkikoodi 41. Nappien luominen palvelimelta haettujen kysymysten perusteella

Nappien luomisen jälkeen käyttöliittymä on valmis. Kuva 7 on kuvakaappaus valmiista käyttöliittymästä



Kuva 9. Kuvakaappaus käyttöliittymästä, kun koe on kesken.

Kun koe on tehty, käyttöliittymä esittää käyttäjälle moduulin, josta hänen kannattaisi aloittaa kurssi. Kuva 8 näyttää, miten viesti näkyy käyttöliittymässä.



Kuva 10. Kuvakaappaus käyttöliittymästä, kun koe on valmis.

Kehitysprosessin aikana tutkittiin myös suuren kielimallin soveltuvuutta parhaan seuraavan kysymyksen ja aloitusmoduulin valitsemisessa. Kielimalliksi testiin valittiin OpenAI:n kehittämä GPT-4. Kysymyksen valintaa varten mallille syötettiin käyttäjän aiemmat vastaukset sekä lista kysymyksistä, ja mallia pyydettiin valitsemaan paras seuraava kysymys saatavilla olevien kysymysten joukosta. Moduulia valitessa mallille syötettiin käyttäjän kaikki vastaukset sekä selitykset jokaisen moduulin sisällöstä, ja sitä pyydettiin valitsemaan käyttäjälle sopivin moduuli.

Suuren kielimallin käyttämisen suurin etu tällaisessa data-analyysitilanteessa on sen kyky ymmärtää sille syötettyä tietoa johtopäätösten tekemisessä, mutta se ei kykene vertaamaan käyttäjän tuloksia muiden kokeen suorittaneiden käyttäjien kanssa. Tämän lisäksi jokainen mallille lähetetty pyyntö on maksullinen, joten sen liiallinen käyttäminen voi nopeasti tulla kalliiksi. Jokainen pyyntö täytyy myös prosessoida, joka vie keskimäärin noin 5–10 sekuntia.

Järjestelmässämme tämä tarkoitti, että jokaisen kysymyksen jälkeen käyttäjän täytyi odottaa pyynnön prosessointia, joka teki kokeen suorittamisesta paljon hitaampaa.

Yllä mainittujen syiden vuoksi suuren kielimallin käyttäminen järjestelmässä jätettiin välistä, mutta siitä voisi silti hyödyntää esimerkiksi palautteen antamisessa kokeen jälkeen. Mallille voitaisiin syöttää käyttäjän koevastaukset sekä eri moduulien tiedot, ja se voisi ehdottaa käyttäjälle resursseja hänen heikompien taitojensa parantamiseen.

6 Testaus

Järjestelmän toimivuuden takaamiseksi on erittäin tärkeää, että sitä on testattu kunnolla. Vaikka käyttäjän näkökulmasta sovellus saattaa näyttää toimivalta, voi sen sisäisestä logiikasta silti löytyä virheitä. Tämän vuoksi haluamme testata sovelluksen toimintaa mahdollisimman monesta eri kulmasta.

Testaukseen löytyy monia eri käytäntötapoja ja työkaluja. TypeScript palvelimen testaukseen aiomme hyödyntää Jest- ja Supertest-kirjastoja. Jestin avulla voimme käyttää luomiamme testiolioita (”mock”-oliota) järjestelmän eri osien testaamisessa. Esimerkkikoodi 42 näyttää, miten luomme mock-olion käyttäjälle sekä yhdelle kysymykselle ja testaamme sillä, että järjestelmä antaa takaisin oikean palautteen, kun käyttäjä lähettää oikein vastatun kysymyksen.

```
const mockUser = {
      _id: 'userId1',
      answers: [] as { questionId: string; correct: boolean;
                       timeTaken: number }[],
      scores: { category1: 0, category2: 0, category3: 0 },
      questionsAsked: { category1: 0, category2: 0, category3: 0 },
      save: jest.fn().mockResolvedValue(true)
    };

    const mockQuestion = {
      _id: 'question1',
      category: 1,
      correctIndex: 0,
      difficulty: 1
    };
it('should handle correct answer submission', async () => {
      const res = await request(app).post('/api/submitAnswer').send({
        userId: 'userId1',
        currentQuestion: { _id: 'question1', category: 1 },
        answerIndex: 0,
        timeTaken: 10
      });

      expect(res.status).toBe(200);
      expect(mockUser.answers.length).toBe(1);
      expect(mockUser.answers[0].correct).toBe(true);
      expect(mockUser.scores.category1).toBe(1);
      expect(res.body.question).toBeDefined();
});
```

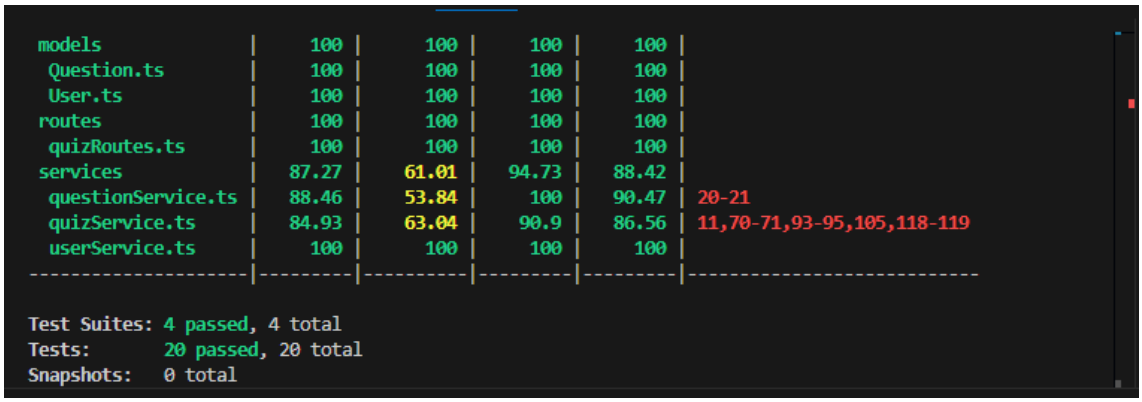
Esimerkkikoodi 42. Backendin testaaminen mock-olioilla

Palautusarvoja verrataan odotettuihin arvoihin expect-funktiolla. Jos palautusarvo eroaa oletetusta arvosta millään tavalla, testi epäonnistuu. Käytämme testeissä kolmea eri mock-kysymystä sekä mock-käyttäjää. Mock-käyttäjän sekä yhden mock-kysymyksen toteutus näkyy esimerkkikoodissa 42, ja esimerkkikoodi 43 näyttää kahden muun mock-kysymyksen rakennetta.

```
const mockQuestions = [
  {
    _id: new Types.ObjectId('680d64b19092b3cf04c15782'),
    category: 1,
    questionNo: 1,
    difficulty: 1.5,
    question: "Test question 1",
    totalAttempts: 10,
    correctAttempts: 5,
    options: ["A", "B", "C", "D"],
    correctIndex: 0
  },
  {
    _id: new Types.ObjectId('680d64b19092b3cf04c15783'),
    category: 2,
    questionNo: 1,
    difficulty: 2.0,
    question: "Test question 2",
    totalAttempts: 8,
    correctAttempts: 4,
    options: ["A", "B", "C", "D"],
    correctIndex: 1
  }
];
```

Esimerkkikoodi 43. Kahden mock-kysymyksen luominen backend-testejä varten

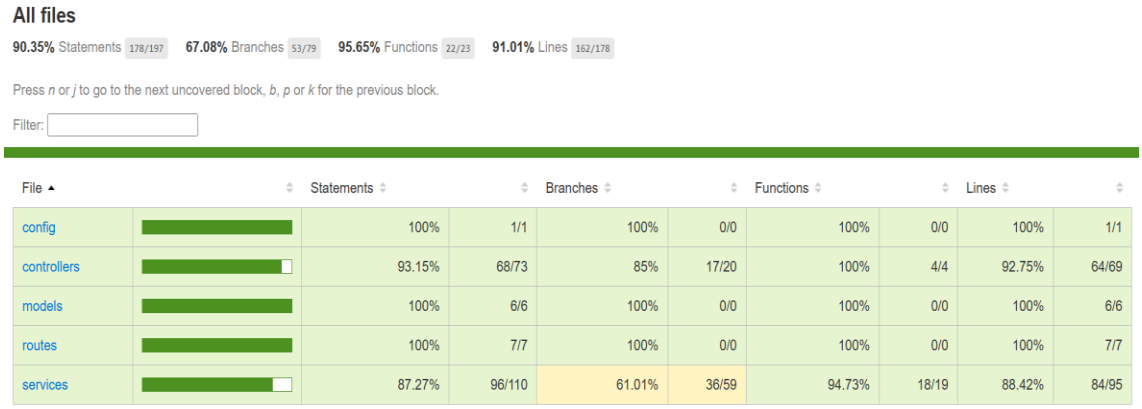
Teemme yhteensä 20 testiä erilaisista mahdollisista tilanteista, kunnes testit kattavat suurimman osan ohjelmasta. Testit voidaan ajaa komentoriviltä. Jos testien kanssa tulee virheitä, siitä ilmoitetaan ja testi epäonnistuu. Kuva 9 näyttää, miltä testitulokset näyttävät komentorivillä.



Kuva 11. Kuvakaappaus testituloksista komentorivillä

Voimme myös generoida testien pohjalta kattavuusraportin, joka voidaan avata selaimessa. Raportin voi generoida komentoriviltä kutsumalla jest-työkalua parametrilla ”coverage”. Työkalu luo projektin hakemistoon uuden kansion raportille, josta se löytyy HTML-muodossa.

Raportti näyttää jokaisen tiedoston testikattavuuden. Sillä voidaan myös katsoa yksittäisten funktioiden testikattavuutta. Kuvasta 10 nähdään testiemme kattavuus, joka on keskimäärin noin 90 %.



Kuva 12. Kuvakaappaus testikattavuusraportista

Koneoppimismallia on myös mahdollista testata. Kun generoimme mallin opettamiseen käytetyn datan, voimme ottaa osan datasta talteen ja käyttää sitä opettamisen sijasta mallin testaamiseen. Esimerkkikoodi 44 näyttää, miten

otamme generoidusta datasta 20 % talteen testiä varten, syötämme testidatan mallille, ja katsomme, miten tarkasti malli on osannut ennustaa käyttäjälle moduulin.

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

prediction = model.predict(X_test)
acc = accuracy_score(y_test, prediction)
print(f"Accuracy: {acc:.2f}")
```

Esimerkkikoodi 44. Koneoppimismallin testaaminen generoidulla datalla

Koodin ajaminen antaa meille testin tarkkuudeksi 99 %.

Seuraavaksi haluamme testata käyttöliittymän toimintaa ja varmistaa, että se toimii odotetulla tavalla myös virhetilanteissa. Käyttöliittymän testaamiseen käytämme Cypress-työkalua. Cypress on erityisesti frontend-testauksen automaatioon luotu työkalu. Backend-testauksen testit luodaan samalla tavalla kuin backend-testit, jotka käyttävät mock-olioita. Tällä kertaa käyttöliittymän lähettämät pyynnöt ”siepataan”, ja niiden sisällöt korvataan mock-objekteillamme. Esimerkkikoodi 45 näyttää, miten testaamme kysymyksen vastaamista sekä uuden kysymyksen näyttämistä. Sieppaamme sekä startQuiz- että submitAnswer-pyyntöjen palautusarvot ja korvaamme ne omilla mock-kysymyksillä. Voimme sitten varmistaa, että palautetut arvot vastaavat mock-kysymyksiämme.

```
it('should allow answering questions and proceed to next question', ()
=> {
  cy.intercept('POST', '/api/startQuiz', {
    statusCode: 200,
    body: {
      userId: '123456789012345678901234',
      question: {
        _id: '1',
        question: 'Sample question 1',
        options: ['Option 1', 'Option 2', 'Option 3', 'Option 4'],
        category: 1,
        questionNo: 1
      }
    }
  }).as('startQuiz');
  cy.intercept('POST', '/api/submitAnswer', {
    statusCode: 200,
    delay: 300,
    body: {
      question: {
        _id: '2',
        question: 'Sample question 2',
        options: ['Option A', 'Option B', 'Option C', 'Option D'],
        category: 2,
        questionNo: 2
      },
      done: false
    }
  }).as('submitAnswer');
    // Odotetaan kysymyksen lataamista
    cy.wait('@startQuiz');
    // Vastataan ensimmäiseen kysymykseen
    cy.get('[data-testid="quiz-option"]').first().click();
    // Varmistetaan, että lataus alkaa suoraan vastaamisen jälkeen
    cy.get('[data-testid="loading-spinner"]', { timeout: 1000
}).should('exist');
    // Odotetaan uutta kysymystä
    cy.wait('@submitAnswer');
    // Varmistetaan, että uusi kysymys on haettu
    cy.get('[data-testid="question-text"]').should('contain', 'Sample
question 2');
  });
```

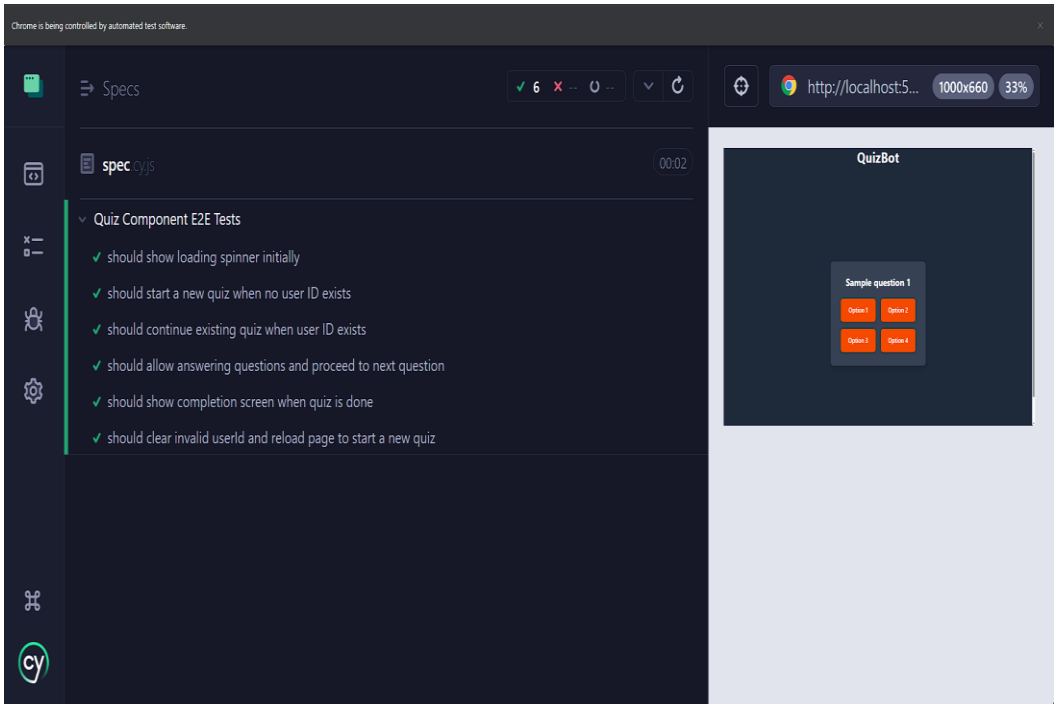
Esimerkkikoodi 45. Kysymyksen vastaamisen sekä uuden kysymyksen hakemisen testaus

Luomme samalla periaatteella loput testeistämme. Ne tulevat kattamaan seuraavat tilanteet:

- Lataus-ikonin näkyminen, kun sovellus käynnistetään.
- Uuden kokeen aloittaminen, kun aiempaa käyttäjää ei ole välimuistissa.
- Vanhan kokeen jatkaminen, kun aiempi käyttäjä löytyy välimuistista.
- Suositellun moduulin näyttäminen kokeen loputtua.

- Virheellisen käyttäjän poistaminen välimuistista ja uuden kokeen aloittaminen, jos välimuistiin tallennettua käyttäjää ei löydetä palvelimelta.

Voimme seurata testiemme edistystä Cypressin käyttöliittymästä. Cypress simuloi testejä varten nettiselaimen, jota ohjataan koodilla. Kuva 13 näyttää Cypress-käyttöliittymän sekä laatimamme testit, jotka ovat kaikki onnistuneet.



Kuva 13. Kuvakaappaus käyttöliittymän Cypress-testeistä

Voimme vielä lopuksi testata järjestelmää manuaalisesti itse tekemällä kokeen. Kuva 14 näyttää viiden eri koeyrityksen oikeiden vastausten lukumäärää jokaisesta moduulista, ja järjestelmän lopullista suositusta kurssin aloituspisteeksi.

Kuva 14. Viiden eri käyttäjän koetulokset tietokannassa

Tulokset ovat tallennettu ”scores”-taulukon sisälle, ja ”recommendedModule”-muuttujan arvo kuvaa lopullista suositeltua moduulia.

7 Tulokset

Lopputuloksena saatiin järjestelmä, joka esittää käyttäjälle halutun määrän kysymyksiä kurssin eri moduuleista, joiden haastavuutta säädetään käyttäjän arvioidun osaamistason perusteella. Kokeen loputtua käyttäjän vastaukset analysoidaan käyttäen koneoppimista, ja hänelle valitaan sopiva aloituspiste kurssille.

7.1 Tutkimuskysymyksiin vastaaminen

Lopputuloksen ja tehtyjen tutkimusten perusteella tekoälyn hyödyt oppimiskokemuksen personalisoinnissa tulevat selväksi. Tekoälyllä voidaan analysoida opiskelijan suorituksia reaaliajassa ja räätälöidä kurssia hänen osaamistasonsa perusteella, mikä johtaa parempaan oppimiskokemukseen ja mahdollisesti lopuksi parempiin oppimistuloksiin. Tekoäly voi lisäksi suositella opiskelijalle hänen heikompia osa-alueitansa tukevia resursseja. Tekoälyn hyödyllisyys tulee varsinkin näkyviin suurilla verkkokursseilla, joissa henkilökohtaisen ohjauksen tarjoaminen jokaiselle opiskelijalle ei olisi muuten käytännöllistä.

Vaikka tekoälyn hyödyntäminen tuo mukanaan paljon etuja, sen käyttöönottoon liittyy myös omat haasteensa. Yksi suurimmista haasteista on laadukkaan, luotettavan ja tarpeeksi kattavan datan löytäminen. Tekoäly oppii datan perusteella, joten jos data sisältää virheitä tai ei ole helposti ymmärrettävissä, on sen käyttäminen hyödytöntä. Toinen suuri haaste varsinkin koneoppimisessa on hyödyllisten datapisteiden löytäminen, ja niiden osoittamien johtopäätösten löytäminen. Esimerkiksi tässä projektissa luodussa koejärjestelmässä kerätään talteen käyttäjän vastausaika jokaiselle kysymykselle. Nopea vastausaika voi osoittaa, että käyttäjä ymmärsi kysymyksen ja tiesi vastauksen nopeasti, tai se voi tarkoittaa, että käyttäjä on arvannut vastauksen oikein ymmärtämättä kysymystä. Tällaisessa tilanteessa yksittäisen vastauksen perusteella on lähes mahdotonta tietää, onko vastaus valittu arvauksella vai tietämyksen perusteella. Usein tilanteen voi ratkaista vertaamalla vastausta käyttäjän aiempiin vastauksiin. Jos käyttäjä on vastannut kaikkiin muihin kysymyksiin oikein ja yhtä

nopeasti, voidaan olettaa vastauksen olevan aito. Voidaan todeta, että tekoälyä on mahdollista hyödyntää luotettavasti opiskelijan osaamistason arvioinnissa, mutta se vaatii järjestelmän huolellista suunnittelua sekä jatkuvaa seurantaa datamäärän kasvaessa. On erittäin tärkeää, että järjestelmää vahditaan tarkasti jonkin aikaa, kun se otetaan käyttöön ensimmäisen kerran ja varsinaista käyttäjien tuottamaa dataa aletaan käyttämään, koska siinä vaiheessa mahdolliset algoritmivirheet saattavat tulla esille.

7.2 Validointi

Tehtyjen testien ja tutkimusten perusteella voidaan todeta, että järjestelmämme kykenee ennustamaan opiskelijan osaamistasoa, ja käyttämämme koneoppimismalli on luotettava. Emme kuitenkaan voi vielä todeta sen olevan vielä käytännössä luotettava. Järjestelmän käytännöllistä validointia varten tarvitsemme dataa oikeilta käyttäjiltä, jolla järjestelmän toimivuuden voisi todistaa. Uutta dataa voitaisiin myös hyödyntää koneoppimismallin opettamisessa sekä mahdollisten uusien tärkeiden muuttujien havaitsemisessa.

8 Yhteenveto

Tässä työssä suunniteltiin ja kehitettiin prototyyppi personalisoidusta koejärjestelmästä, jonka voisi integroida osaksi oppimisen hallintajärjestelmää. Koejärjestelmä hyödyntää tekoälyä opiskelijoiden osaamistason arvioimiseen uusien kysymysten valintaa varten. Järjestelmä käyttää myös generoidun datan avulla opetettua koneoppimismallia, jolla se valitsee opiskelijalle sopivan aloituspisteen kurssille.

Työn aikana tutustuttiin myös projektin luomiseen tarvittaviin teknologioihin, tutkittiin tekoälyn soveltuvuutta oppimisen hallintajärjestelmään ja sen hyödyntämisen hyviä ja huonoja puolia. Koneoppimisen ja opiskelijan osaamistason arvioimisen lisäksi tutkittiin myös mahdollisia muita käyttötapoja tekoälylle osana oppimisen hallintajärjestelmää, esimerkiksi suurien kielimallien käyttämistä palautteen antamisessa sekä oppimisresurssien suosittelemisessa.

Lopulta järjestelmä testattiin, jolla varmistettiin sen toimivan oikein ja odotetulla tavalla. Todettiin, että tekoälyä, varsinkin koneoppimista, on mahdollista hyödyntää luotettavasti opiskelijan taidon arvioimisessa, kunhan sen suunnittelee ja toteuttaa harkitusti.

Seuraava tärkein askel jatkokehityksen kannalta on ehdottomasti järjestelmän integroiminen osaksi oppimisen hallintajärjestelmää, esimerkiksi Moodlea, jotta luotettavaa dataa pystyisi keräämään oikeilta käyttäjiltä. Uutta dataa voitaisiin hyödyntää koneoppimismallin opettamiseen. Tämän lisäksi olisi tärkeä tutkia yksittäisten opiskelijoiden kokemuksia järjestelmästä, jotta voitaisiin havaita mahdollisia uusia hyödyllisiä muuttujia, joilla opiskelijan osaamistasoa voisi ennustaa vielä tarkemmin.

Lähteet

1 Wood, Johnny. 2022. These 3 charts show the global growth in online learning. Verkkoaineisto. <https:/www.weforum.org/stories/2022/01/online-learning-courses-reskill-skills-gap>. Luettu 4.2.2025.

2 SAP. 2024. Mikä on oppimisen hallintajärjestelmä (LMS)?. Verkkoaineisto. <https://www.sap.com/finland/products/hcm/corporate-lms/what-is-lms.html>. Luettu 4.2.2025.

3 Khan Academy. 2025. Khanmingo for students. Verkkoaineisto. <https://www.khanacademy.org/college-careers-more/khanmigo-for-students>. Luettu 27.04.2025.

4 Aga Maulana, Ghazi Mauer idroes, Pati Kemala, Nur Balqis Maulydia, Leveraging Artificial Intelligence to Predict Student Performance: A Comparative Machine Learning Approach. 2023. Verkkoaineisto. <https://www.researchgate.net/publication/376684240_Leveraging_Artificial_Intelligence_to_Predict_Student_Performance_A_Comparative_Machine_Learning_Approach>. Luettu 27.04.2025.

5 Training Industry. 2025. The Evolution of the Learning Management System (LMS). <https://trainingindustry.com/wiki/learning-technologies/the-evolution-of-the-learning-management-system-lms/>. Luettu 4.2.2025.

6 Franconnect. 2025. What Was the First LMS Platform?. Verkkoaineisto. <https://www.franconnect.com/the-first-lms-platform/>. Luettu 11.2.2025.

7 Morten Paulsen & Torstein Rekkedal. The NKI Internet College: A review of 15 years delivery of 10,000 online courses. 2001. Verkkoaineisto. <https://www.irrodl.org/index.php/irrodl/article/view/17/354>. Luettu 13.02.2025.

8 Moodle. Statistics. 2025. Verkkoaineisto. <https://stats.moodle.org/>. Luettu 13.02.2025.

9 Moodle. Tietoa moodlesta. 2025. Verkkoaineisto. <https://docs.moodle.org/4x/fi/Tietoa_Moodlesta>. Luettu 13.02.2025.

10 Teemu Seesto. ECAR 2017 – FACULTY SURVEY SELVITYS SUOMALAISTEN KORKEAKOULUJEN OPETTAJIEN JA TUTKIJOIDEN NÄKEMYKSISTÄ INFORMAATIOTEKNOLOGIAN KÄYTÖSTÄ. 2018. Verkkoaineisto. <https://tt.eduuni.fi/sites/kity/publicAAPA-FUCIOdocs/ECAR/ECAR2017_FacultySurvey_Suomi.pdf>. Luettu 14.02.2025.

11 Itslearning. Verkkoaineisto. <https://fi.itslearning.com/solution/take-the-tour>. Luettu 14.02.2025.

12 European Parliament. What is artificial intelligence and how is it used? 2020. <https://www.europarl.europa.eu/topics/en/article/20200827STO85804/what-is-artificial-intelligence-and-how-is-it-used>. Luettu 16.04.2025.

13 Google. Understanding Gmail's spam filters. 2022. Verkkoaineisto. <https://workspace.google.com/blog/identity-and-security/an-overview-of-gmails-spam-filters>. Luettu 25.03.2025.

14 Amazon. What is Deep Learning? Verkkoaineisto. < https://aws.amazon.com/what-is/deep-learning/>. Luettu 25.03.2025.

15 Amazon. What is LLM (Large Language Model)? 2025. Saatavissa: <https://aws.amazon.com/what-is/large-language-model/> Luettu 25.03.2025.

16 Python Basics. Why Python for Machine Learning? < https://python-basics.org/why-python-for-machine-learning/>. Luettu 16.04.2025.

17 IBM. What is random forest? Verkkoaineisto. < https://www.ibm.com/think/topics/random-forest>. Luettu 27.04.2025.

Koekysymykset

Category 1: Theory of Software Engineering

1. Which of the following best defines software engineering?

☐ Writing code for software applications
☑ Applying engineering principles to software development
☐ Designing only the UI of software
☐ Developing hardware components

2. What is the main difference between software engineering and programming?

☐ Programming is broader than software engineering
☑ Software engineering involves structured processes, while programming focuses on coding
☐ Programming includes testing, but software engineering doesn’t
☐ Software engineering is only about project management

3. Which of the following is NOT a characteristic of good software?

☐ Maintainability
☐ Scalability
☑ High complexity
☐ Efficiency

4. Which SDLC phase involves feasibility studies?

☑ Requirement Analysis
☐ Design
☐ Implementation
☐ Maintenance

5. What is the primary purpose of software maintenance?

☐ To add new features only
☑ To remove bugs and improve performance
☐ To rewrite the entire system from scratch
☐ To make software completely error-free

6. Which of the following is an example of a non-functional requirement?

☑ The system should support 100 users simultaneously
☐ The user should be able to log in
☐ The system should allow password reset
☐ The user should receive email notifications

7. What does coupling refer to in software design?

☑ The degree of dependency between modules
☐ The number of functions in a module
☐ The size of the software
☐ The security of the software

8. Which type of testing ensures new changes do not break existing functionality?

☐ Unit testing
☐ Integration testing
☑ Regression testing
☐ System testing

9. What is the goal of software configuration management?

☐ To ensure code runs faster
☑ To manage changes in software artifacts
☐ To improve UI design
☐ To remove software bugs

10. What is a key disadvantage of the Waterfall model?

☐ It allows for changes at any stage

☑ It requires extensive documentation
☐ It follows an iterative approach
☐ It is ideal for rapidly changing requirements

Category 2: Software Development Methodologies

1. Which Agile framework uses time-boxed iterations called sprints?

☑ Scrum
☐ Kanban
☐ Waterfall
☐ Extreme Programming (XP)

2. What is the purpose of a sprint retrospective?

☐ Plan upcoming tasks
☑ Identify and improve team performance
☐ Present the product to stakeholders
☐ Estimate the budget

3. Which methodology is best suited for projects with clear, well-defined requirements?

☐ Scrum
☑ Waterfall
☐ Kanban
☐ Lean

4. What is the main purpose of a Kanban board?

☐ To document system architecture
☑ To track work in progress and visualize workflow
☐ To enforce strict deadlines
☐ To manage test cases

5. What is a defining feature of Lean Software Development?

☐ Heavy reliance on documentation
☐ Maximizing resource allocation

☑ Eliminating waste and optimizing efficiency
☐ Strict hierarchical team structures

Category 3: DevOps Methods & Applied Tools

1. Which of the following is NOT a version control system?

☐ Git
☐ SVN
☑ Docker
☐ Mercurial

2. What is Continuous Integration (CI)?

☐ Automating deployment to production
☑ Frequently integrating code into a shared repository
☐ Managing production infrastructure
☐ Running security tests

3. Which of these tools is used for monitoring and logging in DevOps?

☐ Jenkins
☑ Prometheus
☐ Git
☐ Terraform

4. Which of the following best describes Kubernetes?

☐ A version control system
☐ A CI/CD pipeline tool
☑ A container orchestration platform
☐ A security scanning tool

5. What does the ‘shift-left’ approach in DevOps mean?

☐ Delaying testing until later stages
☐ Automating infrastructure provisioning
☑ Integrating testing and security earlier in the development lifecycle
☐ Deploying software without review