Tampereen yliopisto

Jami Kulmala

MODERNI WEB-KEHITYS TEKOÄLYN AVULLA

Informaatioteknologian ja viestinnän tiedekunta
Kandidaattitutkielma
Marraskuu 2023

TIIVISTELMÄ

Jami Kulmala: Moderni web-kehitys tekoälyn avulla
Kandidaattitutkielma
Tampereen yliopisto
Tietojenkäsittelytieteiden tutkinto-ohjelma
Marraskuu 2023

Tässä tutkielmassa tarkastellaan tekoälyn hyödyllisyyttä, asemaa ja käyttöä modernissa web-kehityksessä. Tarkoituksena on selvittää aiempien tutkimusten perusteella, millaisilla tavoilla tekoälyä hyödynnetään ja onko sen käyttö välttämätöntä web-kehittäjälle, joka haluaa pysyä tämän nopeasti kehittyvän ohjelmoinnin alan kehityksessä mukana. Sekä yksilön, että yrityksen tavoite on tehdä mahdollisimman hyvää koodia mahdollisimman pienin kustannuksin, joten on mielenkiintoista tutkia, miten hyvin tekoäly toimii tässä työkaluna. Tutkielmassa perehdytään ensin web-kehitykseen ja tekoälyyn yleisesti, minkä jälkeen siirrytään tarkastelemaan tekoälyn käyttöä ja merkitystä. Lopuksi esitellään sen erilaisia sovelluskeinoja web-kehityksen eri vaiheissa.

Tutkielman artikkeleita ja konferenssipapereita on kerätty Tampereen yliopiston kirjaston Andor -palvelusta, ACM-tietokannasta ja Google Scholarista. Lähteitä on kerätty vuosien 2012 ja 2023 väliltä, ja varsinkin uusimpia tekstejä on pyritty painottamaan tekoälyn uusimpien kehitysharppauksien vuoksi. Valitut lähteet käsittelevät web-ohjelmistokehitystä ja erilaisia tapoja hyödyntää tekoälyä sen eri vaiheissa.

Tutkimuksen tulokset tukevat tekoälyn käytön hyödyllisyyttä apuvälineenä web-kehityksen eri vaiheissa. Rutiininomaisia tehtäviä ja vaikeasti ymmärrettäviä rakenteita pystytään toteuttamaan tekoälyn avulla huomattavasti pelkkää ihmisen työtä nopeammin ja tehokkaammin. Sitä voidaan soveltaa kaikissa web-kehityksen vaiheissa ja eri osissa toimivaa web-sovellusta. Ihmisen ja tekoälyn yhteistyön merkitys korostuu tutkielman aikana, sillä ilman ihmistä tekoälyn hyöty web-kehityksessä on hyvin rajallinen. Sekä web-kehitysprosessi, että tekoälymallien kehitys ovat vaativia ja aikaa vieviä prosesseja, joten oikeiden työkalujen valinta on tärkeää. Tekoälypohjaisia työkaluja on lukuisia erilaisia, ja niitä on saatavilla eri käyttötarkoituksiin. Niiden vahvuutena on erityisesti nopeus ja tehokkuus, mutta heikkoutena epäluotettavuus ja harjoitusdatan mahdollinen huono laatu. Tekoälyn kehitys on jatkuvaa, ja sen kehitys näkyy varmasti myös tulevaisuuden web-kehityksessä.

Avainsanat: tekoäly, web-kehitys, tekoälypohjainen web-kehitys, koodin generointi

Sisällysluettelo

1 Johdanto

Web-kehitys on ohjelmistokehitystä, jossa sovellus suoritetaan selaimessa. Nettisivu tai -sovellus on jokaiselle yritykselle lähes välttämättömyys ja siksi se on yksi suosituimmista ohjelmoinnin muodoista (Dzhangarov et al., 2021). Käyttäjäystävällinen ja toimiva verkkosivu on oleellinen keino sekä houkutella uusia asiakkaita että pitää vanhat asiakkaat tyytyväisinä. Siksi web-sovelluksia kehitetään jatkuvasti. Mahdollisimman hyvin toimivat palvelut ovat lopulta jokaisen etu. Web-ohjelmointi kehittyy jatkuvasti ja uusia työkaluja ja sovelluksia tulee markkinoille jatkuvasti.

Web-sovelluksen toiminta perustuu asiakasohjelman esittämiin pyyntöihin ja palvelimen vastauksiin eli web-sovellus on käytännössä näiden välinen vuorovaikutusmalli, joka toimii selaimessa (Dzhangarov et al., 2021). Sovellus jaetaan selainpuoleen eli front-endiin ja palvelinpuoleen eli back-endiin. Selainpuolella tarkoitetaan koodia, joka ajetaan selaimessa eli siitä näkyvät käyttäjälle sivun rakenne, tyyli ja toiminnallisuus. Puolestaan palvelinpuolella tarkoitetaan kaikkea piilossa suoritettavaa koodia, kuten pyyntöjen käsittely, tietoturva ja tietokantaoperaatiot. Usein back-end koodi on ohjelman kannalta raskasta ja onkin tyypillistä, että palvelimina käytetään erilaisia pilvipohjaisia ratkaisuja, jolloin raskas laskenta tapahtuu pääasiassa siellä (Dzhangarov et al., 2021). Tämä tehostaa sekä ohjelmaa, että ohjelman toteutusta huomattavasti.

Toimivan ja käyttäjäystävällisen web-sovelluksen luominen on hidas ja vaativa prosessi. Sopivien työkalujen ja metodien valitseminen ja löytäminen on tärkeää mahdollisimman sujuvan prosessin ja odotuksia vastaavan lopputuloksen kannalta. Ohjelmointiympäristön ja -kielen lisäksi web-kehittäjän tulee valita esimerkiksi tietokanta, pilvipalvelu, tyylikirjasto ja versionhallinta. Ohjelmointikielistä suosituimpia ovat web-ympäristössä JavaScript, Java ja Python, joista jokaisesta löytyy lukuisia erilaisia kirjastoja web-kehitykseen (Dzhangarov et al., 2021).

Tekoälyllä tarkoitetaan järjestelmän kykyä tulkita sille annettua dataa oikein ja siitä saadulla tiedolla oppia suorittamaan erilaisia tehtäviä joustavasti ja ihmismäisesti. Varsinkin suurien datamäärien ja kasvaneen laskentatehon myötä, tekoäly on noussut valtavaksi liiketoimintamahdollisuudeksi ja puheenaiheeksi yleisesti. Tällä hetkellä tekoäly on erityisen hyödyllinen yksinkertaisissa ja rutiininomaisissa tehtävissä. Kuitenkin jo pitkään on uskottu sen pystyvän alkaa esittämään kognitiivisia, emotionaalisia ja sosiaalisia kykyjä tuotoksissaan, ja näin mukailemaan paremmin ihmisen tekemää työtä. (Haenlein & Kaptan, 2019) Tekoäly jaetaan eri kategorioihin, joista tärkein sen soveltamiseen web-kehityksessä on neuroverkot. Neuroverkoilla tarkoitetaan järjestelmiä, jotka simuloivat älykkyyttä, jäljittelemällä aivoissa tapahtuvaa asioiden yhdistelyä. (Pareek, 2012)

Tekoälyä hyödynnetään web-kehityksessä jo laajalti, ja sen käytön voidaan odottaa vain kasvavan tulevaisuudessa. Tekoälyteknologioiden yhteenlaskettu markkina-arvo oli arviolta noin 7.35 miljardia Yhdysvaltojen dollaria vielä vuonna 2018, mutta vuonna 2025 sen oletetaan kasvavan noin kymmenkertaiseksi, mikä kuvastaa hyvin tekoälymarkkinoiden kehitystä. Tekoälyn ja koneoppimismallien levittyä kaikkien käyttöön, uskotaan sen vaikuttavan moderniin web-kehitykseen. (Gupta, 2019)

Tekoäly toimii web-kehittäjän tukena monella tavalla. Web-sovelluskehitys on turha aloittaa tyhjästä, sillä tekoälyn kehityksen myötä on olemassa lukuisia keinoja saada käyttöön valmiita, toimivia ja testattuja komponentteja, mikä tekee sovelluksen kehityksestä nopeampaa ja tehokkaampaa. (Gupta, 2019) Automaattiset ominaisuudet ja generoitu koodi voivat toimina suurena apuna web-sovellusta toteuttaessa, mutta voivat aiheuttaa myös ongelmia. Luvussa 2.2 lisää tekoälyn haasteista ja ongelmista.

Datan määrän kasvaessa sen hyödyntäminen, mutta myös turvallisuus on suuressa osassa käyttäjäystävällisen web-sovelluksen toteutusta. Tekoälypohjaiset koneoppimismallit ovat vahvistaneet sivujen tietoturvaominaisuuksia, ja näiden jättäminen pelkästään ihmisen toteutettavaksi olisi turhaa ja riskialtista. Data, jota hyödynnetään muun muassa web-sovellusten jatkuvaan kehitykseen pysyy näin yksityisenä. Tekoälyn avulla datan kerääminen ja analysointi voidaan automatisoida, ja näin kehittää web-sovelluksia paremmiksi tehokkaasti. Tekoälyä hyödyntää myös testaus, joka on erittäin tärkeää julkaistavan web-sovelluksen toiminnan kannalta. (Gupta, 2019) Tekoälyä voidaan siis hyödyntää eri web-kehitysprosessin vaiheissa ja tehtävissä. Yksin se ei kuitenkaan toimi, vaan tarvitaan erilaisia osaajia luomaan toimiva ja valmis käyttäjäystävällinen sovellus, ja yhdistämään sen tarjoamat palat toisiinsa.

Tämä tutkielma on kirjallisuuskatsaus, jossa perehdytään tekoälyn hyödyllisyyteen ja merkitykseen modernissa web-kehityksessä aiempien tutkimustuloksien perusteella. Aloitin tutkielman lähteiden haun Andor -palvelusta hakusanoilla ”artificial intelligence” ja ”web development” ja rajasin hakutulokset vuodesta 2015 vuoteen 2023 ja vertaisarvioituihin lehtiin. Lähteitä aiheesta löytyy paljon, mutta tärkein rajaava tekijä niiden valinnassa on ollut se, tarjoaako lähde relevanttia ja luotettavaa tietoa tutkielman tärkeimmistä teemoista vai jotain aiheen vierestä.

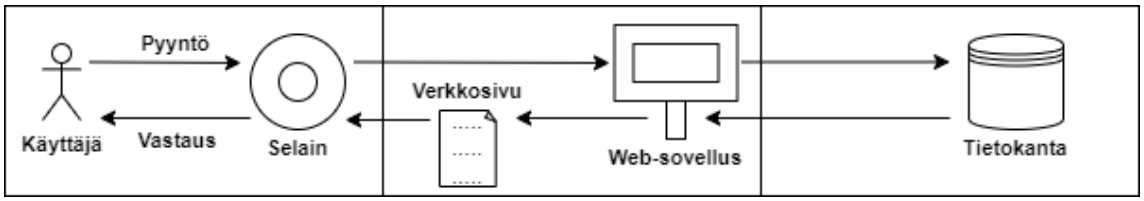
Kirjallisuuskatsauksen edetessä ja tutkielman rakenteen hahmotuttua hakuni ovat kohdistuneet tarkemmin tiettyihin tekoälyn sovelluksiin ja eri web-kehityksen vaiheisiin, joissa tekoälyä hyödynnetään. Hakusanoina ovat toimineet esimerkiksi ”bug”,”UI”, ”information security”, ”data”, ”automation”, yhdistelemällä näitä hakusanoihin ”artificial intelligence” ja ”web”. Artikkeleita on tutkielman edetessä haettu Andorin lisäksi ACM-tietokannasta ja Google scholarista.

2 Tekoäly ja web-kehitys

Tässä kappaleessa esitellään web-sovelluksen ja web-kehityksen rakenteet, joihin tekstissä viitataan myöhemmin. Kappaleessa esitellään myös erilaisia tekoälyn sovelluskohteita ja web-kehityksessä sovellettavia tekoälyn tekniikoita. Lisäksi otetaan kantaa tekoälyn vahvuuksiin, heikkouksiin ja merkitykseen web-kehityksessä.

2.1 Tekoälyn soveltaminen web-kehityksessä

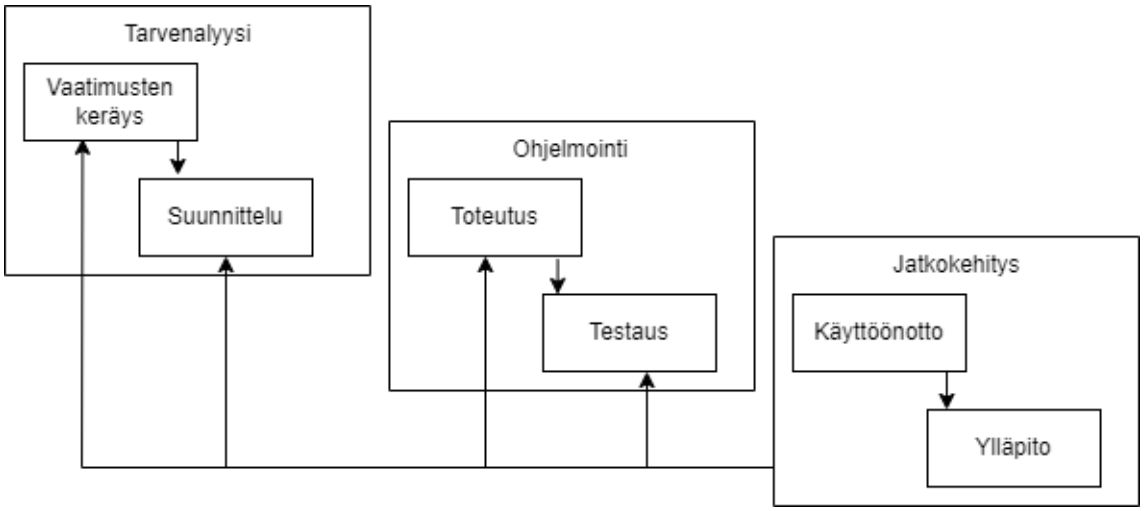
Web-sovelluksia on dynaamisia ja staattisia, joiden erona on dynaamisen sovelluksen jatkuva vuorovaikutus palvelimen kanssa. Staattiselle web-sovellukselle riittää sen lataus, kun sivu avataan, mutta dynaaminen sivu päivittyy jatkuvasti käyttäjän syötteiden perusteella. Dynaaminen web-sovellus toimii kolmessa tasossa alla olevan kaavion mukaan. (Tekerek & Bay, 2019)



Kuva 1 Dynaamisen web-sovelluksen toiminta (Lähteestä: Tekerek & Bay, 2019)

Suurin osa web-kehityksestä keskittyy toiseen tasoon, jossa itse web-sovellusta kehitetään. Tämän lisäksi myös tietokannan rakentaminen ja suunnittelu vaatii työtä. Tutkielmassa keskitytään lähinnä tekoälyn soveltamiseen dynaamisissa web-sovelluksissa niiden yleisyyden ja moderniuden vuoksi.

Web-kehitys sisältää samat vaiheet kuin mikä tahansa muu ohjelmistokehitys: tarveanalyysin, suunnittelun, toteutuksen, testauksen ja jatkokehityksen (Fadhil et al., 2020). Se lähtee käyntiin vaatimusten keräyksellä, jossa kartoitetaan esimerkiksi sovelluksen käyttötarkoitus, käyttäjäkunta ja tärkeimmät ominaisuudet. Sen jälkeen siirrytään ohjelmiston suunnitteluun, jossa muun muassa päätetään sovelluksen arkkitehtuurista, designista ja komponenteista, jotka sitten toteutetaan sovelluksen toteutusvaiheessa. Toteutuksesta siirrytään testaukseen ja jatkokehitykseen, joissa varsinkin sovelluksen käytettävyyden ja toimivuuden takaaminen on olennaista. Web-sovelluskehitys on iteratiivinen prosessi, ja sen vaiheet menevät usein päällekkäin sekä pysyvät mukana koko sovelluksen elinkaaren ajan. Tekoälyä voidaan hyödyntää web-kehityksen jokaisessa vaiheessa.



Kuva 2 Sovelluskehityksen elinkaari (Lähteestä: Dehaerne et al., 2022)

Sekä käyttäjän näkökulmasta että sovelluksen laadun ja turvallisuuden kannalta tekoäly on web-kehittäjälle olennainen työkalu. Tekoälypohjaiset tietoturvaratkaisut, automatisoidut testausmenetelmät ja koodin sekä sovelluksen laadun parantaminen ovat sen suoria käyttöesimerkkejä web-kehityksessä. (Gupta, 2019) Kaikessa ohjelmistokehityksessä on haasteena varsinkin sovelluksen laatu ja kehityksen nopeus. Tekoälypohjaiset työkalut pyrkivät vastaamaan varsinkin näihin haasteisiin ja niitä kehitetään eri sovellusalueihin jatkuvasti. (Fadhil et al., 2020)

Web-kehittäjän tulee pysyä mukana tekoälyn kehityksessä, sillä sen avulla pystytään toteuttamaan ja parantamaan web-sovelluksia huomattavasti pelkkää ihmisen työtä tehokkaammin. Sovelluskehityksen lisäksi tekoälyä hyödynnetään web-sovelluksissa esimerkiksi sisällön ja mainosten personoinnissa ja apuvälineiden, kuten chatbottien ja äänentunnistuksen, muodossa (Gupta, 2019). Tekoäly toimii myös mahdollisuutena hyödyntää internetistä peräisin olevaa dataa entistä paremmin varsinkin käyttäjäkokemuksen parantamiseen (Pareek, 2012). Näin esimerkiksi sovelluksen personointi ja ulkoasu kehittyvät tekoälyn mukana.

Tärkeitä tekoälyn tekniikoita web-kehityksessä ovat koneoppiminen, syväoppiminen ja konenäkö. Koneoppimisessa on kyse hahmontunnistuksesta eli aiemmasta datasta pyritään oppimaan kaavoja, joita voidaan sitten soveltaa uuteen dataan esimerkiksi luokitteluun ja regressioon. Syväoppimisessa pyritään kehittämään algoritmeja, jotka pystyvät löytämään datasta piilossa vaikuttavia yhteyksiä sekä kaavamaisuuksia hierarkisoimalla dataa. Konenäkö puolestaan tarkoittaa tekoälyn kykyä analysoida ja ymmärtää kuvia ihmisen tavoin. (Stocco, 2019)

2.2 Tekoälyn heikkoudet ja haasteet

Tekoäly ei pysty samanlaisiin kognitiivisiin toimintoihin kuin ihminen vaan mukailee niitä opetusdatastaan tunnistamiensa kaavojen ja yhteyksien perusteella (Huang & Zheng, 2023). Se tekee siis johtopäätöksiä sille syötetyn datan perusteella yhdistelemällä ja tunnistamalla asioita kaavamaisesti, eikä oikeasti kokemalla niitä. Virheiden ja epätoivottujen seurauksien välttämiseksi, on erityisen tärkeää opettaa malleja laadukkaasti ja syöttää niille riittävän laadukasta dataa. Tietoisuuden ja subjektiivisen kokemuksen ansiosta tekoäly ei pärjää ihmiselle kekseliäisyydessä ja luovuudessa (Huang & Zheng, 2023). Se ei myöskään pysty ymmärtämään ihmistä yhtä hyvin kuin toinen ihminen, joten ihmisen rooli pysyy vielä suurena kaikessa web-kehityksessä.

Tekoälymallien kehitys on vaativa ja aikaa vievää prosessi kehityksen monimutkaisuuden vuoksi. Esimerkiksi vaikeasti ymmärrettävä rakenne, ja malliin herkästi syntyvä harha, ovat kehittäjien riesana (Stocco, 2019). Moni asia voi johtaa heikkolaatuiseen dataan, joka päätyy tekoälymallin käyttöön. Esimerkiksi virheellinen, riittämätön tai epäedustava data on tekoälymallien jatkuvana riesana. Datasta saattaa löytyä virheellisiä korrelaatioita, tai esimerkiksi jokin ryhmä saattaa olla yliedustettuna tai suosittuna. (Reyero et al., 2023) Datan ja koneoppimismallien tulee siis olla puolueettomia ja virheettömiä, että niistä on oikeasti hyötyä. Web-kehitykseen soveltuvien tekoälymallien luomisessa ongelmaksi saattaakin koitua riittävän laadukkaan datan heikko saatavuus. Data on usein rikkonaista, ja sen käsittelyyn menee huomattava määrä aikaa ja resursseja. Usein ongelmaksi koituu myös datan yksipuolisuus, missä tapauksessa korrelaatiot ovat vääristyneitä ja myös malli saattaa helposti toimia väärin (Reyero et al., 2023).

Tekoälypohjaisten työkalujen heikkouksiin ja rajoituksiin lukeutuvat myös tekoälyn generoiman koodin mahdollinen heikko laatu, epäjohdonmukaisuus, epätarkkuudet sekä mahdollinen epäajantasaisuus (Huang & Zheng, 2023). Tekoäly ei esimerkiksi välttämättä osaa ottaa suoraan huomioon kaikkia turvallisuus- ja tyyliseikkoja, joten ihmisen tulee valvoa sen tuottamaa sisältöä jatkuvasti. Tekoälyn tiedot ja taidot perustuvat vain sille syötettyyn dataan, mikä ei kata kaikkea (Huang & Zheng, 2023). Tämä saattaa joissain tilanteissa johtaa ongelmiin esimerkiksi juuri vanhentuneen tiedon ja sen tuottamien väärien johtopäätöksien vuoksi.

2.3 Tekoälyn vahvuudet ja mahdollisuudet

Tekoälymallit ovat web-kehityksessä hyödyllisiä varsinkin sovelluksen tietoturvaominaisuuksien parantamisessa, testauksen automatisoinnissa, datan keruussa ja analysoinnissa sekä koodin ja käyttöliittymäsuunnitelmien laadun parantamisessa (Gupta, 2019). Näiden lisäksi tekoälyä voidaan hyödyntää ohjelmointiprosessissa ohjelmoinnin tukena erilaisten tekoälypohjaisten työkalujen, kuten OpenAI:n ChatGPT:n ja GitHub Copilotin avulla. Nämä työkalut pystyvät tällä hetkellä esimerkiksi selittämään,

uudelleenjärjestämään, yksinkertaistamaan, korjaamaan ja täydentämään koodia sekä testaamaan sitä ja löytämään ohjelmointivirheitä. (Lau & Guo, 2023) Näin tekoälypohjainen ohjelmointi pystyy säästämään varsinkin aikaa ja vaivaa koodia kirjoittaessa sekä toimimaan apuna erilaisten ominaisuuksien toteutuksessa.

Verrattuna ihmiseen tekoäly on tällä hetkellä tehokkaimmillaan rutiininomaisissa tehtävissä, ja kun käsittelyssä on erilaisia laajoja datajoukkoja. Tekoälypohjaisista työkaluista esimerkiksi Copilotia ohjelmistokehittäjät käyttävät ohjelmoinnissa varsinkin mahdollisuuksien kartoittamiseen ja työn nopeuttamiseen (Lau & Guo, 2023).

Internetin sekä sieltä saatavan datan määrän kasvaessa mahdollistuu uudenlaisten palveluiden ja työkalujen kehitys (Pareek, 2012). Esimerkiksi äänen-, kuvan- ja tekstintunnistuksen hyödyntäminen, käyttäjäkokemuksen parantaminen, koodin muutoksista oppiminen sekä testien automaattinen luonti ja ylläpito ovat tekoälyn kehityksen mahdollisuuksia (Stocco, 2019). Tekoälyn kehityksen uskotaan olevan vasta alkuvaiheessa, ja uusien tekoälypohjaisten mallien ja työkalujen kehitys on jatkuvaa (Gupta, 2019). ChatGPT ja Copilot toimivat esimerkkeinä käyttäjäystävällisistä työkaluista, jotka ovat kaikkien saatavilla ja käytettävissä. Vastaavien työkalujen kehityksen voi uskoa kasvavan tulevaisuudessa, ja näin web-kehityksen kehittyvän sen mukana.

3 Tekoälyn sovellustavat web-kehityksessä

Tässä kappaleessa esitellään erilaisia olemassa olevia ja tutkijoiden visioimia tekoälyn sovelluskeinoja. Kaikkia sovelluskeinoja ei ole voitu sisällyttää tutkielmaan, mutta mukana on web-kehitysprosessin kannalta olennaisimmiksi arvioidut.

3.1 Käyttöliittymien suunnittelu ja toteutus

Tekoälyä voidaan hyödyntää käyttöliittymien suunnitellussa ja luomisessa. Käyttöliittymän suunnittelu alkaa usein ideoinnista ja hahmottelusta. Kun suunnitelma on saatu valmiiksi, aletaan sitä kokeilemaan itse ohjelmassa. Tekoälyä on ehdotettu tämän prosessin nopeuttamiseen generoimalla koodipohjia suoraan piirretyistä suunnitelmista. Tällaiset tekoälymallit hyödyntävät varsinkin neuroverkkoja ja konenäköä ymmärtämään kuvaa graafisesta käyttöliittymästä, siirtääkseen sen valmiiksi koodielementeiksi kuten tekstiksi ja nappuloiksi itse ohjelmaan. (Stocco, 2020) Näin käyttöliittymien suunnittelu ja toteutus tehostuisi sillä perusominaisuudet täyttävien käyttöliittymäpohjien luominen ja testaus siirrettäisiin tekoälyn vastuulle. Tällöin web-kehittäjälle jää enemmän resursseja vaativampien ominaisuuksien toteuttamiseen.

Tekoälyllä voidaan parantaa myös käyttöliittymän suunnitelman laatua ja poistaa paineita suunnittelijan harteilta. Tekoälyn avulla saadaan helposti luotua perusvaatimukset täyttävät pohjat komponenteille, sekä myös kehitettyä suunnitelmia

eteenpäin ja aina innovatiivisemmiksi. Tekoälypohjainen käyttöliittymien suunnittelu toimii ihmisen ja tekoälyn vuorovaikutuksena. (Huang & Zheng, 2023)

Käyttöliittymien suunnittelu ei siis toimi pelkästään tekoälyn avulla. Tekoälyä voidaan käyttää esimerkiksi olemassa olevien design-pohjien kopiointiin ja parantamiseen, sekä erilaisten komponenttien generointiin (Huang & Zheng, 2023). Tekoäly oppii sille annetusta datasta, eikä pysty ihmisen silmin tarkastelemaan kopioimiaan tai luomiaan käyttöliittymiä. Täten ihmistä tarvitaan sekä mallin opettamiseen, että datan keruuseen, jonka on oltava riittävän laadukasta (Huang & Zheng, 2023). Ilman ihmistä tekoälypohjainen malli pystyy tuottamaan lukuisia piirroksia ja koodipohjia, mutta ihminen tarvitaan, samoin kuin datan valitsemisessa, rajaamaan näistä hyvät ja huonot (Huang & Zheng, 2023).

Tekoäly mukailee ihmisen älykkyyttä ja pystyy avustamaan päätöksenteossa. Se pystyy tukemaan päätöksentekoa ja nopeuttamaan sitä tunnistamalla ainakin väärät päätökset, mutta päätöksen lukitseminen jää viimekädessä ihmisen harteille. Tekoälyn nykytilassa ei voida puhua vielä älykkyydestä, joten varsinkin kriittisissä ja tärkeissä linjauksissa sen apua kannattaa hyödyntää maltillisesti. (Huang & Zheng, 2023)

3.2 Virheiden tunnistaminen ja ennustaminen

Ohjelmointivirheiden käsittely on kriittisessä osassa modernia web-kehitystä. Niiden pääseminen valmiiseen tuotteeseen aiheuttaa helposti yritykselle mainehaittaa ja siksi ne pyritään poistamaan kokonaisuudessaan jo sovelluksen tuotannon aikana. Lisäksi virheiden tapahtuminen ohjelmiston kehityksen aikana on lähes väistämätöntä ja niiden käsittely aina ehkäisemisestä löydökseen ja ratkaisuun vie huomattavan osan kehityksen resursseista. Virheiden tunnistaminen ja ennustaminen ovat monimutkaisia tehtäviä ja siksi menetelmiä virheiden tunnistamiseen ja ennustamiseen hyödyntäen tekoälyn keinoja pyritään kehittämään. (Fadhil et al., 2020)

Virheiden tunnistusta voidaan lähestyä monin eri tekoälyn keinoin. Esimerkiksi tiedonlouhinta-, neuroverkko- ja koneoppimispohjaisia malleja on saatavilla ja kehityksessä. Nämä mallit pystyvät tunnistamaan ja luokittelemaan erilaisia vikoja, haavoittuvuuksia ja virheistä suuristakin ohjelmistoista. (Fadhil et al., 2020) Virheentunnistajille, jotka perustuvat näihin tekoälyn keinoihin on tyypillistä regression, klusteroinnin ja luokittelun hyödyntäminen. Erityisen hyödyllistä on ennen testausta löytyvien virheiden tunnistus ja paikantaminen ja tätä lähestytään usein varsinkin tiedonlouhinnan keinoin (Fadhil et al., 2020). Monet ohjelmointivirheet huomataankin vasta testausvaiheessa tai sovelluksen elinkaaren myöhemmissä vaiheissa, joten tekoälyn merkittävänä etuna on virheen mahdollisimman aikainen tunnistaminen, jolloin koodi on vielä ohjelmoijan muistissa ja virhe korjattavissa merkittävästi pienemmällä vaivalla ja resursseilla.

Ohjelmointivirheitä voi tapahtua myös koodin konfiguraation muuttuessa sovellusta ajettaessa, jolloin seuraukset varsinkin suurissa sovelluksissa voivat jatkua pitkälle ja ovat usein vaikeammin korjattavissa. Tekoälypohjaisia keinoja on etsitty ennustamaan virheitä konfiguraation muuttuessa. Erityisen hyödyllisenä pidetään riskialttiiden muutoksien ja koodimoduulien tunnistamista. Näin voitaisiin tunnistaa osat, jotka tarvitsevat eniten tarkastelua esimerkiksi testausvaiheessa ja koodia muokatessa. Tämänhetkinen tutkimus keskittyy lähinnä vikojen tunnistamiseen ja ennustamiseen, eikä esimerkiksi erilaisten konfiguraatioiden tuottamiseen. (Fadhil et al., 2020) Myös virheiden ennustamisen tavoitteena on virheiden mahdollisimman aikainen paikantaminen ja löytyminen. Tapoja ennustajien toteuttamiseen on monia, joista esimerkkeinä tunnistajien tavoin erilaiset luokittelijat ja neuroverkot sekä syväoppimisen -mallit ja bayesilaiset metodit (Fadhil et al., 2020).

Myös tekoäly tekee virheitä, ja se ei välttämättä pysty tunnistamaan omia virheitään. Pelkkään tekoälyyn nojautuminen ohjelmaa testatessa olisi hyvin riskialtista, mutta apuvälineenä se toimii erittäin hyvin ja on lähes välttämätön suuria ohjelmia toteuttaessa.

3.3 Sovelluksen tietoturva

Web-sovellukset monimutkaistuvat ja niiden käsittelemän datan määrä kasvaa jatkuvasti. Siksi tietoturvasta tulee aina vain suurempi osa modernia web-kehitystä, jossa tekoälyä hyödynnetään varsinkin vahvistamaan käyttäjätietojen suojausta ja sovelluksen turvallisuutta (Gupta, 2019). Sovelluksen turvallisuudesta pystytään huolehtimaan suunnittelemalla ja toteuttamalla koodi oikein, mutta on tyypillistä, että monella web-kehittäjällä on riittämättömät tiedot ja taidot koskien mahdollisia uhkia web-ympäristössä. (Tekerek & Bay, 2019). Perinteiset web-sovelluksen palomuurit pystyvät estämään verkosta tulevat hyökkäykset tehokkaasti, mutta varsinkin HTTP eli hypertekstin siirtoprotokolla sisältää haavoittuvuuksia, joihin perinteiset suojaukset eivät välttämättä kaikissa tapauksissa toimi.

Tekerek ja Bay (2019) esittelevät artikkelissaan tekoälypohjaisen palomuurin, jossa HTTP-hyökkäykset tunnistetaan opettamalla palomuuri tunnistamaan HTTP-liikenteestä tulevia poikkeavuuksia hyödyntämällä neuroverkkoja. Käytännössä palomuuri opetettiin siis tekoälyllä tunnistamaan epätavalliset HTTP-pyynnöt sekä pyynnöt, jotka eivät sovi rakenteeltaan sovelluksen hyväksymiin pyyntöihin. Tutkimuksessa vertailtiin toteutusta aiempiin tutkimustuloksiin ja todettiin tekoälypohjaisen ratkaisun sekä estävän verkosta tulevat hyökkäykset, että suojaamalla sovelluksen HTTP-hyökkäyksiltä perinteisiä palomuureja paremmin. Lisäksi todettiin, että pelkkä HTTP-pyyntöjen suojaus säännöillä ei tule riittämään. Tässäkin tapauksessa on hyvä huomata, että tekoäly ymmärtää ainoastaan kaavat, joita se oppii sille syötetystä datasta, eikä oikeasti ymmärrä mitä on tekemässä. Siksi ihmisen rooli on yhä suuri tietoturvan suunnittelussa ja toteutuksessa.

Suurin osa hyökkäyksistä verkossa tapahtuu sovelluksen tasolla verkkoliikenteessä. Web-sovelluksen palomuurilla tarkoitetaan järjestelmää, joka rajaa ja valvoo sovelluksen ja palvelimen välillä tapahtuvaa liikennettä. Perinteinen palomuuri toimii tunnistamalla tuntemattomia pyyntöjä tai tunnistamalla tunnettuja hyökkäystyyppejä, joko suoraan palvelimessa tai sovelluksen ja palvelimen välissä. Perinteisen palomuurin heikkoutena on, että jokainen hyökkäystyyppi pitää lisätä siihen erikseen ja uudet hyökkäystyypit joudutaan aina lisäämään siihen, tai se olisi tehoton niitä vastaan. (Tekerek & Bay, 2019)

Tekoälypohjaisen hybridipalomuurin vahvuutena on, että se pystyy sille syötettyjen sääntöjen lisäksi tunnistamaan epätavallisia HTTP-pyyntöjä ja päivittämään itseään sille syötetyn datan perusteella. Tutkimuksen palomuurin tekoäly toteutettiin neuroverkkona, joka tarkastelee syötteen kirjaimia ja pituutta, pyrkien tunnistamaan hyökkäyksiä oikeasta palvelimen liikenteestä. Mallille syötettiin HTTP-dataa, josta se oppi tunnistamaan hyökkäykset perinteisiä malleja nopeammin. (Tekerek & Bay, 2019)

3.4 Sovelluksen testaus

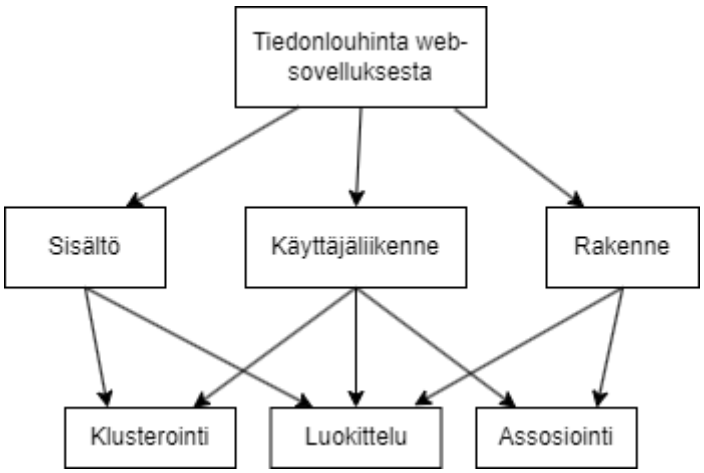
Testaus on tärkeä osa web-sovelluksen elinkaarta. Testaamalla sovellusta jatkuvasti riittävän tarkasti varmistetaan sovelluksen tehokkuus ja luotettavuus sekä erilaiset virheet sovelluksen toiminnallisuudessa voidaan havaita. (Alenzi et al., 2022) Manuaalinen testaus ja testien luominen on aikaa vievää ja siksi automaattiset työkalut voivat olla hyvin hyödyllisiä sovelluksen testausta kehittäessä.

Automaattiset testityökalut ja metodit kehittyvät teknologian mukana kovaa. Esimerkkejä markkinoilla olevista web-sovellusten testaustyökaluista ovat Selenium, Watir, TestComplete, SahiPro, QTP, JMeter ja LoadRunner. Käytettävät työkalut valitaan pääasiassa tuettavien kielien, ominaisuuksien, järjestelmän ja saatavilla olevan tuen perusteella. Nämä työkalut vaativat monipuolista ohjelmointiosaamista ja usein kokemusta niiden käytöstä, jotta niitä päästään hyödyntämään käytännössä. (Alenzi et al., 2022) Ne kuitenkin mahdollistavat jatkuvan ja monipuolisen ohjelman testauksen ja siksi automaattinen testaus on erityisen hyödyllistä.

Automaattiset testityökalut eivät ole täydellisiä ja niiden käyttö sisältää ongelmakohtia. Esimerkiksi testien kehitys on yhä työlästä ja aikaa vievää sekä testien ylläpidon kustannukset nousevat suuriksi sovelluksen monimutkaistuessa. Ongelmia voidaan lähestyä esimerkiksi konenäön ja koodin indeksoinnin yhdistelyllä tavoitteena nopeuttaa ja tehostaa testien luomista, ylläpitoa ja korjausta. Testityökalut toimivat pääasiassa vain kooditasolla, eivätkä ota huomioon sovelluksen visuaalisia piirteitä. Konenäöllä pystytään vastaamaan tähän ongelmaan. Myös testien tapauksessa tekoälymallien kehitys on haastavaa ja sisältää erilaisia kehityskohtia. (Stocco, 2019)

3.5 Web intelligence

Internetissä tapahtuu valtavasti liikennettä jatkuvasti, josta jäävää tietoa voidaan kerätä hyödynnettäväksi eri tarkoituksiin. Web intelligencellä tarkoitetaan tämän internetistä peräisin olevan tiedon tutkimista, keräämistä ja analysointia tekoälyn keinoin (Pareek, 2012). Web intelligence toimii siis mahdollisuutena kehittää ja tehostaa palveluita aina vain paremmiksi hyödyntäen verkosta louhittua tietoa, jota pelkkä ihminen ei välttämättä pystyisi hyödyntämään ja käsittelemään ilman tekoälyn apua.



Kuva 3 tiedonlouhinta web-sovelluksesta (Lähteestä: Pareek, 2012)

Tietoa louhitaan internetistä eri tavoin, mutta pääasiassa tämän kaiken työn tavoitteena on tuottaa mahdollisimman hyviä palveluita käyttäjille. Dataa voidaan louhia verkkosivun liikenteestä, rakenteesta ja sisällöstä hyödyntäen eri tekoälyn tekniikoita. Esimerkiksi kuluttajatietojen ja -tottumusten tarkastelu on erittäin hyödyllistä sovellusten kehittämisessä ja personoinnissa. Puolestaan sisältöä ja rakennetta tarkasteltaessa voidaan löytää näkemyksiä sovellusten parantamiseen entisestään. Myös internetin älykkäät järjestelmät ovat esimerkki web intelligencen hyödyntämisestä. Varsinkin verkossa toimivat toimijat kuten myyjät, mainostajat ja rahoittajat hyötyvät verkon tiedonlouhinnasta erityisen paljon kehittäessään palveluitaan. Verkosta saadun tiedon avulla kuluttajan tarpeita voidaan ennustaa. Näin web-kehitys ja sovellusten suunnittelu pysyvät kuluttajan edellä ja jatkuva kehitys voidaan taata. (Pareek, 2012)

3.6 Koodin generointi

Tekoälyn avulla voidaan generoida koodia käyttäjän syötteen perusteella. Tekoälypohjaiset laajat kielimallit kuten ChatGPT ja Copilot perustuvat internetistä louhitulla datalla harjoitettuihin neuroverkkomalleihin. Nämä työkalut pystyvät tuottamaan toimivaa koodia osana erityyppisiä web-sovelluksia. (Lau & Guo, 2023) Niiden tuottamaa sisältöä voidaan hyödyntää web-kehityksen eri vaiheissa, pelkän koodin tuottamisen lisäksi ideointiin ja testaukseen. Ne eivät pysty luomaan kokonaisia

sovelluksia eivätkä osaa ajaa koodia, jota ne tuottavat. Sen sijaan tukena erilaisten yksittäisten funktioiden, komponenttien ja sivujen luomisessa ne suoriutuvat hyvin.

Laajat kielimallit pystyvät toimimaan apuna koodauksessa pääasiassa kolmella eri tavalla: tuottamalla koodia kuvauksesta, tuottamalla kuvausta koodista tai tuottamalla koodia koodista. Koodin tuottaminen kuvauksella toimii käyttäjän syötteellä esimerkiksi tekstinä tai kuvana. Tästä tekoälymalli sitten pyrkii parhaansa mukaan tuottamaan halutun lopputuloksen koodina. Puolestaan tuottaakseen kuvauksen koodista, mallin tulee ymmärtää koodia ja eri ohjelmointikieliä niin hyvin, että se pystyy tuottamaan siitä huomioita luonnolliselle kielelle. Koodista koodia tuottamalla malli pystyy esimerkiksi korjaamaan vikoja ja lisäämään uusia ominaisuuksia koodiin. (Dehaerne et al., 2022)

Koodingeneroijia on saatavilla monia tunnetuimpien lisäksi ja erilaisia tutkijoiden visualisoimia malleja on kehitteillä. Jotkut generaattorit toimivat hyvin joissain tietyissä tehtävissä, mutta huonosti yleisesti. Toiset taas toimivat yleisesti hyvin, mutta saattavat tietyssä kontekstissa aiheuttaa kömpelöjäkin virheitä. Tällaisten laajojen kielimallien kehitys vaatii erittäin paljon dataa ja siksi ne vaativat myös erittäin paljon laskentatehoa ja energiaa. Malleja kehitetään jatkuvasti ja uusia tekoälyn tekniikoita hyödynnetään niiden ominaisuuksien parantamiseen. (Dehaerne et al., 2022) Todennäköistä siis on, että tulevaisuudessa ne selviävät aina vain monimutkaisemmista tehtävistä.

4 Keskustelu

Tutkielman tavoitteena oli löytää erilaiset tekoälyn sovelluskeinot ja tutkia sen soveltuvuutta web-kehitykseen. Erilaisten sovelluskeinojen lisäksi esille nousi web-kehityksen haasteet ja tekoälyn vahvuudet ja heikkoudet. Luokittelu, luonnollisen kielen käsittely ja regressio ovat esimerkkejä tekoälyn ja koneoppimisen keinoista, jotka eivät aina toimi täydellisesti. Näitä hyödynnetään web-kehityksessä laajasti, joten varsinkin käyttäjän tulee olla varautunut, ettei malli toimi aina täydellisesti. Mallit vaativat siis jatkuvaa ylläpitoa tekniikan kehittyessä ja ympäristön muuttuessa. Tekoälyn avulla tai ilman, web-sovelluksen luonti ja ylläpito jää monimutkaiseksi ja aikaa vieväksi prosessiksi. Tekoäly on kuitenkin korvaamaton apu jo sen nykymuodossaan eri ominaisuuksia toteuttaessa.

Monimutkaiset web-sovellukset toimivat siten, että jopa eri yritykset vastaavat sovelluksen eri osista. Erilaisia yritysten ja henkilöiden tarjoamia kirjastoja ja palveluita käytetään sovelluksen eriosissa. Olennaista ei ole tietää miten jokin kirjasto tai sovelluksen osa on toteutettu vaan, miten sitä käytetään. Esimerkiksi front-end kehittäjän ei välttämättä onnistuakseen tarvitse tietää miten back-end tai tietty kirjasto on toteutettu, mutta on kuitenkin hyvä olla tietoinen niiden toiminnasta ongelmilta välttyäkseen. Sama pätee tekoälyssä ja sen käytössä, jossa kehitys on ylittänyt yksittäisen ihmisen ymmärryksen tason. Web-sovellukset ovat alttiita virheille, uhille ja ongelmille, joihin

tekoälyllä voidaan yrittää vastata. Se ei kuitenkaan pysty ymmärtämään koko sovelluksen rakennetta, joten sitä tulee käyttää harkiten vaikeampien seurauksien välttämiseksi. Sovellusta muokatessa tapahtuu herkästi tarkoittamattomia virheitä ilman riittävää ymmärrystä web-kehityksestä. Tekoälyä hyödyntäessä olisi olennaista tiedostaa edes sen rajoitteet ja toimintaperiaate, jotta epätoivotuilta seurauksilta voitaisiin

välttyä. Tärkeää on ymmärtää sen luovan ainoastaan parhaaksi näkemänsä ratkaisun mahdollisesti riittämättömän opetusdatan perusteella.

Tekoäly on kaikkien saatavilla tekstipohjaisten laajojen kielimallien kautta ja kysymys tiedon oikeudesta ja puolueettomuudesta tulee vastaan kaikkialla. Web-kehityksessä suurimmaksi haasteeksi nousee kuitenkin tekniikan nopea kehitys ja sovellusten monimutkaisuus. Tekoälyn yleisiä ongelmia voidaan lähestyä esimerkiksi säätelyllä ja datan laadun tarkkailulla (Haenlein & Kaptan, 2019), mutta web-kehitykseen soveltuvien mallien kehittämiseen dataa on melko huonosti saatavilla. Tekoäly ei vielä kehitä itse itseään, joten varsinkin herkästi haavoittuvaisten web-sovellusten tapauksessa tekoälypohjaiset tietoturvaominaisuudet saattavat nopeasti menettää tehonsa. Tämä itse itseään kehittävä tekoäly on kuitenkin tulevaisuudessa mahdollista ja nähtäväksi jää, mihin se tulee pystymään web-kehityksen saralla.

Tutkielmassa on esitelty monia keinoja hyödyntää tekoälyä web-kehityksessä, mutta niitä löytyisi eri hakusanoilla ja luovuudella varmasti lisääkin. Web-ympäristö aiheuttaa erityisiä vaatimuksia ohjelmistoille, mutta monet yleisenkin ohjelmistokehityksen työkalut ja keinot soveltuvat siihen. Tutkielman tavoitteen sekä rajallisen pituuden vuoksi eri tekoälyn sovelluksiin on tutustuttu lähinnä pintapuolisesti, eikä yksittäisiin sovelluskeinoihin paneuduta liian tarkasti. Jokaisesta sovelluskeinosta löytyy varmasti valtavasti lisää tietoa, joka tähän tutkielmaan ei olisi mahtunut.

5 Yhteenveto

Tekoälyä osataan soveltaa jokaisessa web-kehityksen vaiheessa. Sovellusta suunniteltaessa sitä voidaan hyödyntää ideointiin, vaatimusten kartoitukseen ja itse suunnitteluprosessissa design- ja koodipohjien tuottamiseen. Toteutusvaiheessa tekoäly toimii apuna muun muassa komponentteja tuottamalla ja koodin laadun parantamisessa sekä tietoturvaominaisuuksien toteuttamisessa se on erittäin arvokas. Testien toteutuksessa tekoäly vähentää varsinkin ihmisen manuaalista työtä ja monia web-kehitykseen soveltuvia automaattisia testikehyksiä on saatavilla. Tekoäly selviää laajojen datamassojen käsittelystä huomattavasti ihmistä paremmin ja esimerkiksi käyttäjäliikenteen seuranta ja sovelluksen ylläpito tehostuu. Näin se toimii myös pohjana viedessä moderneja web-sovelluksia eteenpäin.

Tekoälypohjaisia työkaluja on saatavilla web-kehitykseen lukuisia, joista merkittävimpiä ohjelmointiin ovat ChatGPT ja Copilot. Näiden heikkoutena on

mahdollinen epävarmuus ja epätarkkuus varsinkin edistyneemmissä tapauksissa, mutta ohjelmointiprosessin tehostamiseen ne toimivat ensiluokkaisesti. Automaattiset testityökalut, chatbotit, äänen- ja liikkeentunnistus, koodipohjageneraattorit, virheiden tunnistus ja ennustus, tietoturvatyökalut ja datan automaattinen keräys ja analysointi ovat muita esimerkkejä tekoälyn sovelluksista ja mahdollisuuksista web-kehityksen maailmassa.

Tekoäly vähentää web-kehitysprosessiin kuluvaa aikaa ja sovellukseen liittyviä inhimillisiä virheitä. Erityisen tärkeää on käyttää laadukasta dataa mallin opetuksessa, jotta sen suorituskyky on mahdollisimman hyvä. Vaikka tekoälymallien luominen on haastavaa ja kuluttavaa, niiden käyttämättä jättäminen lisäisi web-sovellusten kustannuksia merkittävästi. Tekoäly selviää rutiininomaisista tehtävistä ja yksinkertaisten ominaisuuksien toteuttamisesta varsin hyvin, mutta monimutkaisten web-sovellusten toteutuksessa ihmisen ja tekoälyn yhteistyö on välttämätöntä.

Web-kehitys ja tekoäly ovat molemmat haastavia ja nopeasti kehittyviä haaroja. Ajoittaisilta virheiltä ei voida välttyä, mutta ihmisen tulee toimia vastuullisesti kummallakin alalla. Web-kehitys tulee varmasti kehittymään tekoälyn mukana, ja tulevaisuudessa yhä parempi käyttäjäkokemus ja tehokkaampi kehitys ovat sen avulla mahdollisia.

Lähdeluettelo

Alenzi, A., Alhumud, W., Bryce, R., & Alshammari, N. (2022). A survey of software testing tools in the web development domain. J. Comput. Sci. Coll. 38, 2 (October 2022), 63–73. *https://doi.org.libproxy.tuni.fi/10.5555/3575846.3575854*

Dehaerne, E., Dey B., Halder S., De Gendt S., & Meert W. (2022) "Code Generation Using Machine Learning: A Systematic Review," in IEEE Access, vol. 10, pp. 82434-*https://doi.org/10.1109/ACCESS.2022.3196347*

Dzhangarov, A., Pakhaev, K, & Potapova, N. (2021). Modern web application development technologies. IOP conference series. Materials Science and Engineering 1155.1 (2021): 12100. *https://iopscience.iop.org/article/10.1088/1757-899X/1155/1/012100*

Fadhil, J. A., Wei, K. T., & Na, K. S. (2020). Artificial Intelligence for Software Engineering: An Initial Review on Software Bug Detection and Prediction. *Journal of Computer Science*, *16*(12), 1709–1717. *https://doi.org/10.3844/jcssp.2020.1709.1717*

Gupta, A. (2019). How Artificial Intelligence is reshaping web designing and web development? Express Computer.

https://andor.tuni.fi/permalink/358FIN_TAMPO/176jdvt/cdi_proquest_reports_2170946759

Haenlein, Michael, Kaplan, Andreas. (2019). A brief history of artificial intelligence: on the past, present, and future of artificial intelligence. California management review, 2019, Vol.61 (4), p.5-14. *https://doi.org.libproxy.tuni.fi/10.1177/0008125619864925*

Huang, L., & Zheng, P. (2023). Human-Computer Collaborative Visual Design Creation Assisted by Artificial Intelligence. ACM Trans. Asian Low-Resour. Lang. Inf. Process. 22, 9, Article 221 (September 2023), 21 pages.

https://doi.org.libproxy.tuni.fi/10.1145/3554735

Lau, S., & Guo, P. (2023). From "Ban It Till We Understand It" to "Resistance is Futile": How University Programming Instructors Plan to Adapt as More Students Use AI Code Generation and Explanation Tools such as ChatGPT and GitHub Copilot. In Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 1 (ICER '23), Vol. 1. Association for Computing Machinery, New York, NY, USA, 106–121. *https://doi.org.libproxy.tuni.fi/10.1145/3568813.3600138*

Pareek, R. (2012). Web intelligence-an emerging vertical of artificial intelligence. *International Journal of Engineering And Computer Science*, *3*, 9430-9436.

https://www.researchgate.net/publication/287489726_Web_Intelligence-An_Emerging_vertical_of_Artificial_Intelligence

Reyero Lobo, P., Daga, E., Alani, H., & Fernandez, M. (2023). Semantic Web technologies and bias in artificial intelligence: A systematic literature review. *Semantic Web*, *14*(4), 745-770. *https://content.iospress.com/articles/semantic-web/sw223041*

Stocco, A. (2019). How artificial intelligence can improve web development and

testing. In Companion Proceedings of the 3rd International Conference on the Art, Science, and Engineering of Programming (Programming '19). Association for Computing Machinery, New York, NY, USA, Article 13, 1–4.

https://doi.org.libproxy.tuni.fi/10.1145/3328433.3328447

Tekerek, A. D. E. M., & Bay, O. F. (2019). Design and implementation of an artificial intelligence-based web application firewall model. *Neural Network World*, (4). *https://doi.org/10.14311/NNW.2019.29.013*