Nimi:	**Syy**	**Pisteet (p)**
Oliopohjainen ohjelmointi	Olio-ohjelmoinnin kurssi	pakollinen
Vähintään 5 erilaista luokkaa & oliota	Pakollinen vaatimus tehtävänannossa	pakollinen
Vähintään yhtä APIa käytetty	Pakollinen vaatimus tehtävänannossa	pakollinen
Sovellus tallentaa käyttäjän toiminnan	Pakollinen vaatimus tehtävänannossa	pakollinen
Käyttäjä voi katsella tallentamiaan tietoja myöhemmin	Pakollinen vaatimus tehtävänannossa	pakollinen
pakolliset ominaisuudet		**13p**
Sivuvalikko	Helpottaa sovelluksessa siirtymistä ja luo ohjelmasta järkevän kokonaisuuden	4p
Applikaation rakentamisessa hyödynnetty fragmentteja	Osasimme käyttää fragmentteja paremmin. Mielestämme ohjelman tuottaminen niillä oli helpompaa	2p
Responsiivinen käyttöliittymä	Halusimme ettei ohjelman käyttö rajoitu vain yhdelle laitteelle	1p
Viikoittaiset tartuntamäärät kaikilta alueilta ja muutos edelliseen viikkoon	Sovelluksen idea on auttaa ihmisiä tarkastelemaan covid-19 tartuntoja, jolloin on luonnollista, että käyttäjälle ilmoitetaan koko Suomen tartunnat. Halusimme sovelluksen näyttävän myös muutoksen edelliseen viikkoon tuomaan käyttäjälle lisää informaatio pandemian etenemisestä	2p
Kaikki sairaanhoitopiirit listattu, käyttäjä voi tarkastella yksittäisen sairaanhoitopiirin tartuntamääriä ja saa tiedon muutoksesta edelliseen viikkoon	Halusimme antaa käyttäjälle mahdollisuuden tarkastella tartuntoja omalta lähiseudultaan. Datasta oli saatavavilla tartunnat sairaanhoitopiireittäin. Halusimme sovelluksen näyttävän myös muutoksen edelliseen viikkoon tuomaan käyttäjälle lisää informaatio pandemian etenemisestä	2p
Pop-up ilmoitukset	Pop-Up ilmoitukset lisäävät käyttäjälle informaatiota hänen toiminnastaan, jolloin sovelluksen käytöstä tulee mielekkäämpää	3p

suosikkien tallentaminen	Suosikkien tallentaminen oli mielestämme hyvä ja realistinen ominaisuus tallaiselle sovellukselle sillä tällöin käyttäjän ei tarvitse joka kerta käyttäessään sovellusta kerätä haluamiaan kaupunkeja uudestaan.	3p
Käyttäjä voi poistaa suosikeista kohteita	Mikäli käyttäjän mieli muuttuu eikä hän haluakkaan katsoa aikaisemmin valitsemiaan kohteita hän voi poistaa suosikkejaan ja lisätä uusia suosikkeja	1p
Sovellus on rakennettu hyvin suunniteltuista komponenteista	Mielestämme sovelluksen suunniteltu hyvin. Käyttämämme komponentit toteuttavat niille annetut tehtävät ja ovat selvästi ymmärrettävissä ulkopuolisen silmin lisäksi ymmärtämistä on helpotettu koodin kommentoinnilla.	5p
singletonia käytetty	Singletonin käyttö oli mielestämme järkevää esimerkiksi datan lukemisessa, sekä tiedostojen kirjoittamisessa, niin pystyimme estämään turhat ongelmat, kuten tiedostojen ylikirjoittamiset, sekä datan lukemisen useaan kertaan.	2p
hyödylliset linkit tietoa välilehdellä	Covid-19 on suurimmalle osalle käyttäjistä vielä suhteellisen uusi asia, eikä kaikilla ole välttämättä vielä tietoa, miten siihen pitäisi varautua, jolloin on hyvä, että tällaisesta sovelluksesta löytyy hyödyllisiä linkkejä, josta	3p
Näppäimistön automaattinen sulkeutuminen	Näppäimistön käytöstä tekee mukavamman se, että se katoaa sen käyttämisen jälkeen automaattisesti, eikä sitä tarvitse joka kerta itse sulkea. ominaisuutta käytetty kirjautumisen ja rekisteröitymisen yhteydessä	1p
kirjautuminen applikaatioon	Käyttäjä voi kirjautua applikaatioon ja tallettaa käyttäjälleen henkilökohtaiset	3p

	suosikit kts. *suosikkien tallentaminen*	
Sovelluksella voi olla useampia käyttäjiä	Applikaatio pystyy käsitellä useampia käyttäjiä ja pystyy esittämään sisään kirjautuneen käyttäjän suosikki SPH:t suosikit-välilehdellä	3p
Kirjautumisen salasana noudattaa hyvän salasanan sääntöjä (sisältää vähintään yhden numeron, erikoismerkin, ison ja pienen kirjaimen, on vähintään 12 merkkiä pitkä)	Turvallisten salasanojen käyttäminen on nykypäivänä yksi tärkeimmistä asioista. On tärkeää, että sovellukseemme kirjatuvat käyttäjät voivat pitää tietonsa salassa eikä heidän salasanojansa ole helppo arvata.	2p
Lisäominaisuudet yhteensä		**37p**
Pisteet yhteensä		**13p + 37p = 50p ≈ 40p**
Työtehtävät		tunti (h)
Aleksi Haapalainen: Suunnittelu, toteutus		Suunnittelu 5h, toteutus 40h = 45h
Jaakko Moisio: Suunnittelu, toteutus		suunnittelu 5 h, toteutus 38 h = 42h
Severi Salonen: Suunnittelu, toteutus		suunnittelu 5h, toteutus 30= 35h

Esipuhe

Olemme luoneet mielestämme kattavan applikaation. Toteutimme suunnitelmaamme mielestämme kohtuullisella tarkkuudella. On kuitenkin todettava, ettemme pystyneet loppukevään kiireisestä aikataulusta johtuen tekemään kaikkia lupaamiamme ominaisuuksia. Totesimme osan ominaisuuksien ominaisuuksista jäävän lian vähälle käytölle sovelluksemme todelliseen käyttötarkoitukseen nähden. Ajatuksemme muuttui ohjelmaa tehdessämme ja korvasimme osan ominaisuuksista uusilla toiminnallisuuksilla. Kaikki lopulliset ohjelmamme sisältävät toiminnallisuudet on esitelty yllä olevassa taulukossa.

Ohjelma

Toteutimme ohjelman, jonka toiminnallisuudet ovat nähtävillä yllä olevasta taulukosta. Ohjelman perusideana on Covid-19 tartuntojen tarkastelu THL:n avoimen datan palvelusta. Käyttäjä näkee sovelluksen aloitussivulla kokonaistartuntamäärän Suomessa todetuista koronavirustartunnoista.

Ohjelmamme käyttää Android API 28 ja emulaattorina käytimme google Pixel 5. Ohjelmamme toteutettiin pääasiassa Android Studiolla. Käytimme apunamme myös Visual Studio Codea. Ryhmätyökaluina käytimme Microsoft Officen Wordia raportin, suunnitelman tekoon sekä muistiinpanoihin. Githubia käytimme projektin jakamiseen toisillemme. Viimeisenä ryhmätyökaluna käytimme Discordia etätyöskentelyyn. Dokumentointityökaluna käytimme niin ikään Microsoft Officen Wordia. Testaustyökaluina käytimme Pixel 5 API 28 emulaattoria sekä OnePlus Nord CE 5G puhelinta.

Kirjastoja ja teknisiä toteutuksia koodissa on käytetty laajasti eri osa alueilta ja eri lähteistä löytyneiden vinkkien avulla. Kirjastoja ohjelmaan on sisällytetty paljon, mutta lähtökohtaisesti suurin osa ovat peruskirjastoja, joita tuodaan peruskomponenttien käytön yhteydessä. Esille kuitenkin haluamme tässä nostaa org.json kirjastot, joiden käyttö helpotti JSON datan lukemista huomattavasti, kun merkkijonoksi luettu data pystyttiin muuntamaan suoraan JSON olioksi ja edelleen JSON taulukoksi. Lisäksi apua ja uutta tietoa saimme paljon internetistä. Suurimmaksi avun lähteeksi ilmeni stackoverflow sivusto, josta yleensä löytyi apua aiheisiin, jotka olivat meille uusia tai vaikeita. Valmiita komponentteja emme kuitenkaan suoraan löytäneet vaan kirjoitimme pääosin kaiken itse, myöskin siitä johtuen, että useimmiten löytämämme asiat eivät olleet suoria ratkaisuja, vaan antoivat suuntaa, miten voisimme ratkaista omat ongelmamme. Kuitenkin kaksi asiaa ohjelmaamme on tehty pitkälti suoraan kopioiden. Näistä ensimmäinen on navigation drawer, jonka toteutukseen käytimme guides.codepath.com sivustolta löytyvää navigation drawer tutoriaalia. Lisäksi kirjautumiseen tehdyt komponentit ovat tehty geeksforgeeks.com sivuston tutoriaalin, sekä sourcecodester.com sivuston tutoriaalin avulla.

Sivuvalikosta käyttäjä voi siirtyä aloitussivun, sairaanhoitopiirien ja suosikkien välillä. Sairaanhoitopiirit sivulta käyttäjä pystyy tarkastelemaan yksittäisen sairaanhoitopiirin tartuntamääriä. Samalla käyttäjälle ilmoitetaan tartuntamäärämuutokset edelliseen viikkoon nähden. Käyttäjä voi valita pudotusvalikosta haluamansa viikon, jota hän haluaa tarkastella. Sivun alalaidasta käyttäjä voi lisätä haluamansa sairaanhoitopiirit suosikkeihin, joita käyttäjä pääsee tarkastelemaan suosikit-välilehdeltä sivuvalikosta.

Suosikeista käyttäjä pääsee sairaanhoitopiiriä klikatessaan suoraan katsomaan suosikkiensa tartuntamääriä. Käyttäjä voi lisäksi poistaa valitsemiaan suosikkeja erillisen painikkeen avulla. Käyttäjä toimista ilmoitetaan käyttäjälle Pop-Up ilmoituksien avulla.

Toteutus ja työnjako

Suunnittelimme ensin ohjelmaamme haluamamme toiminnallisuudet. Käytännössä toteutimme tämän listaamalla mielestämme mielenkiintoisia toiminnallisuuksia paperille. Toiminnallisuuksien pohjalta lähdimme tekemään luokkakaavioita ja piirtämään sovelluksemme ulkoasua paperille. Hyviin ratkaisuihin päädyttyämme lähetimme mielestämme hyvän suunnitelman harjoitustyön ensimmäiseen palautukseen.

Kaikki ryhmämme jäsenet suunnittelivat sovellusta. Pystyimme järjestämään työllemme sen verran aikaa, että pystyimme suunnittelemaan työtämme kaikkien ollessa koolla läsnä fyysisesti. Jokainen ryhmämme jäsen sai kertoa vapaasti mieliteensä yksityiskohtiin ja saimme mukavasti aina yksimielisyyden jokaisen osan suunnitteluun. Varmasti kaikki olimme hyvin tyytyväisiä luomaamme suunnitelmaan.

Suunnitelman jälkeen lähdimme olio-ohjelmoimaan suunnitelman pohjalle perustunutta sovellustamme. Kävimme läpi useita työtapoja muun muassa tutkimme live-jakamista Android Studiossa. Pitkän harkinnan jälkeen päätimme työtavaksi sellaisen, jossa varsinainen koodi valmistuu yhden ryhmän jäsenen tietokoneelle kerrallaan, ja muut tukevat hänen työtänsä jakamalla pienempiä komponentteja GitHubin välityksellä. Ratkaisuun päädyttiin, koska ajattelemme tämän olevan kaikista tehokkain ja nopein tapa saada vaadittu ohjelma tehtyä. Mielestämme koodin rakentuminen samaan aikaan olisi tuonut huomattavia lisähaasteita yhteensovittamisvaiheessa, koska versionhallinta ei ole erityisen tuttua ryhmän jäsenille, joten totesimme tämän ratkaisun olevan meille kaikista paras.

Puhalsimme kaikki niin sanotusti yhteen hiileen. Vaikka varsinainen koodi oli käytännössä vain yhdellä koneella, saimme merkittävästi hyötyä kahdesta muusta koneesta sekä niiden takana olleista aivoista. Pystyimme aina kokeilemaan komponentteja toisilla koneella ja niiden toimittua lisäämään ne lopulliseen työhömme. Eteen tulleet ongelmat ratkesivat tehokkaasti, sillä pystyimme hakemaan tietoa kolmella koneella aina samaan ongelmaan. Näin saimme tehostettua toimintaamme ja edettyä työssämme tehokkaasti.

Työskentelimme lähtökohtaisesti päivisin aina yhdessä tilassa, jolloin pystyimme keskustelemaan työhön liittyvistä asioista moitteettomasti. Iltaisin jatkoimme usein työtämme Discordin välityksellä näytönjaon avustuksella.

Ryhmänä hyväksymme työtapamme, ja jokainen ryhmän jäsen oli tyytyväinen työnjakoon. Työtavan valitsemisessa helpotti myös se, että olimme koko kurssin tehneet yhteistyötä toistemme kanssa ja osasimme hyvin tulkita toistemme koodia. Pystyimme vaihtamaan sulavasti sitä, kuka vuorollaan oli kirjoittamassa lopullista työtämme, ja ketkä etsivät tietoa seuraavista kohdista ja auttoivat mahdollisten virheiden ratkaisemisen kanssa. Olimme määritelleet ryhmänä jokaisen päivän tavoitteet, jotka pystyimme saavuttamaan. Tavoitteiden saavuttamisten avulla saimme palautettua mielestämme hyvin suunnitellun, toteutetun ja ajallaan valmistuneen harjoitustyön.

Toiminnallisuuksien toteutus

Mielestämme ensimmäiset ja suurimmat haasteet koimme datan lukemisesta avoimen rajapinnan palvelusta, sillä mielestämme palvelun tarjoama data oli aluksi suhteellisen vaikeasti hahmotettavissa. Datan sisäistämisen jälkeen pystyimme kuitenkin suhteellisen vaivattomasti hyödyntämään sitä työssämme.

Datan jatkuvan päivittymisen takia, sekä erittäin hankalan muotoilun takia koimme takaiskuja, kun data päivittyi. Muotoilu oli erittäin hankala saada toimimaan responsiivisesti, sillä dataa piti paljon pilkkoa, mutta esimerkiksi datassa esiintyvät viikot on kirjattu avoimeen dataan pidemmälle, kuin viikkokohtaiset tartunnat. Lisäksi testaaminen osoittautui erittäin hankalaksi, kun pystyimme käytännössä kaksi kertaa työn aloittamisen jälkeen testaamaan, toimiiko ohjelmalle responsiivisesti datan päivittymiseen nähden.

Ohjelmassa on edelleen mahdollisena riskinä, että THL:n päivittäessä dataa meidän ohjelmamme ei pysty vastaamaan muuttuvaan dataan ja datan lukemisessa tapahtuu virhe. Ja kuten edellä mainittiin emme tätä datan muuttumista pystyneet testaamaan haluamallamme tavalla. Olemme kuitenkin luottavaisia, että lopullinen toteutuksemme on responsiivinen muuttuvaan dataan. Jos aikaa olisi ollut enemmän, olisimme varmuudella pystynyt vielä tarkastelemaan tätä enemmän ja saanut varmuuden datan lukemiseen. Myöskin vuoden vaihtuessa ohjelmamme ei pysty vastaamaan dataan, sillä datan formaatti tulee muuttumaan, kun vuosi vaihtuu.

Tiedoston lukeminen ja kirjoittaminen fragmenttien avulla tuotti meille aluksi harmaita hiuksia, sekä datan kuljettaminen fragmenttien välillä. Ongelmien ratkaisun jälkeen pystyimme hyödyntämään ominaisuuksia muissa samanlaisissa kohdissa, joten uuden oppimisesta oli todellakin meille paljon hyötyä.

Singletonin käyttö oli mielestämme erittäin hyvä ominaisuus ohjelmaamme, sillä se on merkittävä osa oliopohjaista ajattelua. Lisäksi meidän ohjelmassamme näimme tarvetta singleton toteutukselle, sillä näin pystyimme välttämään mahdolliset ongelmat päällekirjoituksessa, sekä datan lukemisessa moneen kertaan.

Muut ominaisuudet, joita käytimme, olivat huomattavasti helpompia toteuttaa. Osan valmiiksi listatuista ominaisuuksista näimme jäävän sen verran vähälle käytölle, että näimme järkevämmäksi toteuttaa itse kehittelemämme paremmat toiminnallisuudet kuin annetut esimerkkitoiminnallisuudet.

Yhteenvetona mielestämme harjoitustyö oli erittäin mielenkiintoinen, ja se yhdisti kurssilla käytyjä asioita monipuolisesti. Erityisesti harjoitustyön mahdollistamat vapaudet saavat ryhmältämme kiitosta.

Ensi vuoden harjoitustyössä toivomme ajankohtaisten aiheiden jatkuvan, sillä me opiskelijat saamme siitä varmasti lisämotivaatiota harjoitustöiden tekemiseen.

Ryhmäläisten kommentit

Jaakko Moisio: Mielestäni harjoitustyön sisältö oli erittäin mielenkiintoinen ja ajankohtainen. Harjoitustyö tuotti ryhmälle sopivia haasteita, joista kaikista kuitenkin selvisimme ryhmänä mallikkaasti. Harjoitustyö oli omalle osaamiselleni alkuun suuri haaste, josta mahdollisesti en yksin olisi selvinnyt. Hyvän ryhmähengen avustamana kohtaamani ongelmat alkoivat ratkeamaan. On hauska huomata, miten osaamiseni kehittyi merkittävästi harjoitustyön aikana ja olen paljon valmiimpi seuraaviin Javan haasteisiin ja olio-ohjelmointiin. Mielestäni tällainen työ on hyvä mahdollisuus kerryttää kokemusta Olio-ohjelmoinnin parissa, ja tästä on hyvä ponnistaa työelämään jatkamaan olio-ohjelmointitaitojen kartuttamista. Kiitos mukavasta harjoitustyöstä!

Severi Salonen: Harjoitustyötä oli mielestäni mukavaa tehdä, kun sai valita aiheen itse sekä oli annettu valmiiksi kiinnostavia sekä ajankohtaisia aiheita kuten covid-tartunnat. Harjoitustyön pohjalta pystyi itse nähdä isomman kokonaisuuden aina datanhakemisesta käyttäjälle näkyviin komponentteihin. Olio-ohjelmointi ideologiaa kertyi harjoitustyön yhteydessä vielä enemmän, vaikka välillä jotkin asiat tuntuivat haastavilta toteuttaa.

Aleksi Haapalainen: Harjoitustyö oli erittäin suuri ponnistus omalle osaamiselle, sillä tässä pääsi oikeasti avartamaan ohjelmointi osaamista, kun tekeminen ei rajoittunut esimerkiksi yhteen ohjelmointioppaaseen vaan omien ideoiden pohjalta piti etsiä uusia toteutustapoja ja niiden avulla osaaminen laajeni huomattavan paljon. Olen tyytyväinen, miten hyvin pystyimme toteuttamaan ideoitamme ja suunnitelmaamme ja emme kokeneet kovinkaan suuria esteitä työn aikana ja ne ongelmat mihin törmäsimme, saimme ratkaistua kohtuullisessa ajassa. Tämä konsepti harjoitustyönä oli erittäin mukava ja tätä pystyisi jatkamaan ja laajentamaan erittäin paljon ja omaa koodia pystyisi koko ajan myös parantamaan, kun silmä harjaantuu. Olen kuitenkin tyytyväinen lopputulokseen ja valmiiseen tuotteeseen. Ensi vuodelle kehitysideana voisin mainita avoimen datan miettimisen tarkemmin. Tällä kertaa tuntui, ettei esimerkiksi THL dataa ollut mietitty kovinkaan tarkkaan. Avoimet datat vaihtelivat erittäin paljon vaikeustasoiltaan, sillä THL:n data oli erittäin haastava verrattuna esimerkiksi finnkinon dataan. Lisäksi

kurssimateriaaleissa ei ollut kovinkaan tarkkaan opetettu JSON datan lukemista, vaan kaikki keskittyi pitkälti XML dataan.