Hot Soup

Tekijät Petteri Raina, Eemil Aspholm, Santeri Hynninen

Kuvaus ohjelmasta

Ohjelmassa kirjaudutaan sisään omalla laitteelle tallentuvalla käyttäjällä, minkä jälkeen voidaan muokata profiilin tietoja, luoda ruokapäiväkirja ja lisätä siihen Apista haettuja annoksia. Tämän lisäksi käyttäjä voi tutkia tupakoinnin riskitekijöitä, seurata painoaan sekä laskea hiilijalanjälkensä.

Näillä osiolla pyrimme tekemään ohjelmasta käytännöllisen ja realistisen, mutta samalla keräämään mahdollisimman paljon pisteitä kurssia varten.

Tekijät

Tekijä	Eemil Aspholm	Santeri Hynninen	Petteri Raina
Omat osuudet	Ruoan hakeminen ja päiväkirja Riskitekijät, graafit	Käyttäjä Kirjautuminen Profiilin muokkaus	Hiilijalanjälki Ja Siihen tiedon hakeminen
Roolit	Toteuttaja, todella aikaansaava ja toteutti monia haastavia osia.	Valmentaja, valmensi muita työn tekemisessä ja toteutti oman osansa.	Yhteydenpitäjä ja viimeistelijä, hoiti ryhmän viestinnän ja viimeisteli ja testasi työtä.

Ohjelman toteutus

Ulkopuoliset kirjastot:

`'com.google.android.material:material:1.3.0'`	parempia ui komponentteja varten
`'com.github.PhilJay:MPAndroidChart:v3.1.0'`	MPAndoridChart, Datan visualisointiin ja kuvaajien luontiin
`'com.github.yesterselga:password-strength-checker-android:v1.0'`	Password strenght checker, luomaan salasanan vahvuus mittari

Sekä paljon androidin itsensä tarjoamia kirjastoja

- Mitä työkaluja on käytetty?
    - Ryhmätyökalut

Discord ja snapchat, jossa teimme oman ryhmän tämän projektin tekemiseksi ja jossa pähkäilimme ongelmia yhdessä. Github toimi välineenämme koodin jakamisessa.

- Ohjelmistokehitystyökalut

Käytimme ohjelmistokehitykseen Android Studiota.

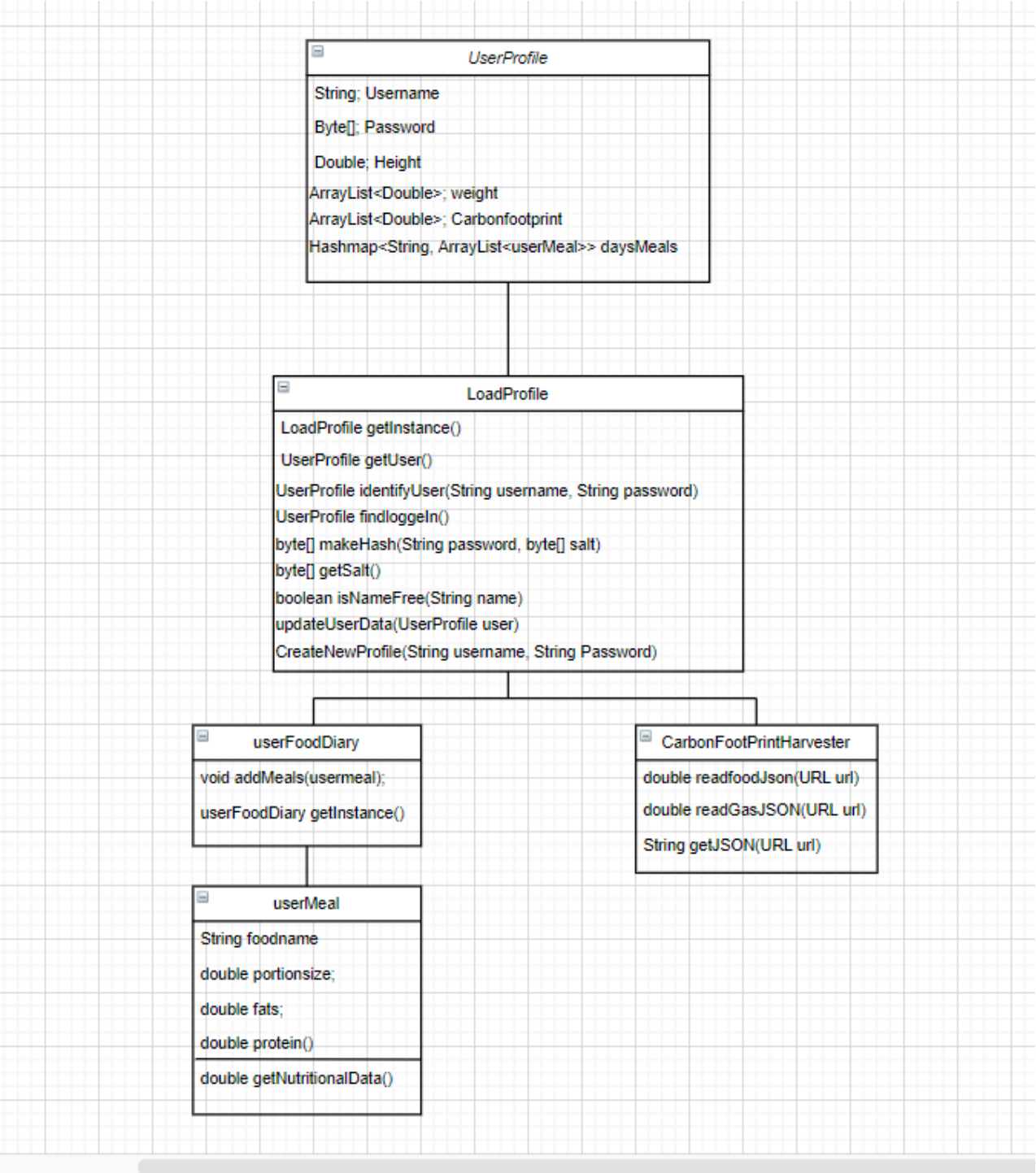
- Testaustyökalut

Emme käyttäneet erillisiä testaustyökaluja, vaan testasimme ohjelmaa itse Android Studiossa.

- Dokumentoitityökalut

Käytimme dokumentointiin Wordia ja Draw.io -nimistä ohjelmaa.

Luokkakaavio



Toteutetut ominaisuudet

Ominaisuus		Pisteet
Olio-ohjelmoitu	Käytetty kattavasti eri luokkia ja hyödynnetty näitä	Pakollinen
Vähintään viisi erilaista luokkaa & oliota (käyttöliittymäluokkia ei lasketa)	Esimerkiksi: LoadProfile, UserProfile, CarbonFoodPrintDataHarvester, UserMeal, UserFoodDiary	Pakollinen
Vähintään yhden APIn käyttö, esim. Ilmastodieetti: https://ilmastodieetti.ymparisto.fi/ilmastodieetti/swagger/ui/index	Hyödynnetty Ilmastodieetti API:tä.	Pakollinen
Sovellus tallentaa käyttäjän toiminnan (käyttäjän syöttämät arvot / tulokset) logiin (JSON, XML jne.)	Tiedot tallennetaan käyttäjään olioon ja tämä olio tallennetaan tiedostoon, mistä tulokset voidaa edelleen lukea	Pakollinen
Logia on mahdollista tarkastella (puhtaana tekstinä, graafisilla käppyröillä jne.), eli voidaan tutkia arvojen (esim. oma massa) kehitystä kirjausten edetessä	Käyttäjä voi tarkastella tietojaan esimerkiksi painonhallinta- tai ruokavalioaktiviteetissa kuvaajista	Pakollinen
Ohjelma on rakennettu hyvin suunnitelluista UI-komponenteista	Ohjelmassa on hyödynnetty kuvitusta ja Material.io ui-komponentteja(Esimerkiksi Profiili kohdassa ja kirjautumisessa) Tämän lisäksi erillaisia ikone ja muotoja on hyödynnetty. Myös väriskaala on suunniteltu	5 pistettä
Kirjautuminen applikaatioon	Sovelluksessa vaaditaan kirjautuminen	3 pistettä
Sovelluksella voi olla useampi käyttäjä (ja niiden luominen), tietojen tallennus järkevästi jonnekin	Sovellukseen voi luoda useamman käyttäjän ja heidän tietonsa tallennetaan erikseen omiin tiedostoihinsa	3 pistettä
Kirjautumisen salasana noudattaa hyvän salasanan sääntöjä (sisältää vähintään yhden numeron, erikoismerkin, ison ja pienen kirjaimen, on vähintään 12 merkkiä pitkä)	Käyttäjän luomisessa ohjelma vaatii salasanan täyttävän hyvän salasanan vaatimukset, eikä anna luoda profiilia ennen sitä.	2 pistettä

Salasanan tallennus käyttää jonkinlaista hash-menetelmää ja suolausta (esim SHA-512 + salt)	Salasanan tallennus käyttää sekä hash-menetelmää että suolausta. Hash algoritmi on SHA-512 ja suola tallennetaan käyttäjälle.	2 pistettä
Jokin toinen datalähde fiksusti implementoituna (näitä löytää esim. https://www.avoindata.fi/, https://www.europeandataportal.eu/fi tai https://thl.fi/fi/tilastot-ja-data/aineistot-ja-palvelut/avoin-data/avoimet-rajapinnat) eli sovellus käyttää siis useampaa datalähdettä kerralla	Sovellus käyttää api.triptocarbon.xyz -sivuston api:tä jonka avulla käyttäjä saa laskettua omia autoilunpäästöjä. Käyttäjältä kysytään esimerkiksi mikä on auton käyttövoima ja ajokilometreja. Myös THL tarjoamasta finelin apista haetaan eri ruoka ja annoskokoja mistä käyttäjä saa määrittää omia syömisiään ruokapäiväkirjaan	5 pistettä
Ohjelmaan on mahdollista syöttää perustiedot (esim. pituus, paino, ikä(/syntymävuosi), kuva, asuinkunta) käyttäjästä ja näitä arvoja käytetään jossakin	Sovelluksessa on mahdollista syöttää kaikki mainitut tiedot ja niitä käytetään eri kaavioissa	2 pistettä
Ohjelma kerää käyttäjän massan kehityksestä dataa ja näyttää muutokset graafisesti havainnollistaen ruudulla	Painohallinta kohdassa käyttäjälle luodaan graaffi, hänen asettamista painoista sovellukseen	3 pistettä
Ohjelma näyttää graafisesti ilmastodieetin tarjoamien arvojen muutokset käppyröillä (esim. kuinka lihan kulutus ja hiilijalanjälki on muuttunut aikojen saatossa)	Kyllä, hiilijalanjälkikohdassa.	3 pistettä
Ohjelma kertoo asuinkunnan, ikäluokan yms. riskitekijät (esim. kunnassa X tupakoitsijoita on Y%) pohjautuen THL:n dataan omien syötteiden lisäksi	Riskitekijät kohdassa esitellään tupakoinnin vaaratekijöitä	2 pistettä
Ohjelma peilaa eri datalähteidä ristiin ja muodostaa uutta tietoa (THL:n dataa höystettynä tilastokeskuksen datalla ja saldona saadaan ulos kaavio Z)	Ohjelma laskee ruuan ja autoilun päästöt eri apeja hyväksikäyttäen ja muodostaa näistä suhteita piirakka diagrammiin ja samalla eri otoksista kuvaavaan pylväsdiagrammin	3 pistettä
Ohjelma muistaa käynnistämisen / kirjautumisen jälkeen missä	Kyllä. Käyttäjällä on arvo “lastActivity”, johon tallennetaan viimeisimmän luokan nimi. Kun käyttäjä	2 pistettä

näkymässä käyttäjä oli ennen ohjelman sulkemista	kirjautuu sisään avataan tähän tallennettu activity. Kirjautuessa on myös kohta “muista minut” Tällöin ohjelma muistaa käyttäjän aina kun ohjelma avataan eikä käyttäjän tarvitse kirjautua sisään. Profiili kohdasta käyttäjä voi kirjautua ulos jolloin ohjelma taas kysyy tietoja käynnistäettäessä.	
Asynkronisten HTTP-kutsujen käyttö dataa haettaessa	Kun tietoja haetaan riskitekijöihin ohjelma tallentaa nämä myös tiedostoon. Jos esimerkiksi tähän käytetty APi on alhaalla tai ei ole nettiä ohjelma hyödyntää näitä aikaisemmin tallennetuja tietoja.	2 pistettä
Fragmenttien hyödyntäminen aktiviteettien sijasta käyttöliittymiä rakennettaessa	Fragmentteja on hyödynnetty mm. painonhallinta - aktiviteetissa, jossa on kaksi eri fragmenttia joita voi vaihtaa ruudun yläosasta	2 pistettä
Scoped storagen käyttäminen tiedon tallennuksessa (ei vaadi käyttäjän myöntämiä oikeuksia laitteen massamuistiin, vaan toimii omassa “hiekkalaatikossaan”)	Kyllä. Ohjelmalle ei luovuteta oikeuksia ulkopuoliseen muistiin. Kaikki tallennetaan ohjelman alle(käyttäjät ja data)	2 pistettä
Responsiivinen käyttöliittymä (toimii siis erikokoisilla ruuduilla sulavasti)	Käyttöliittymä mukautuu käyttäjän laitteen ruudun koon mukaan. Komponenttien koko ei muutu mutta niiden asettelu mukautuu jokaiseen näyttöön erikseen.	2 pistettä
Jokin oma hieno ominaisuus tai toiminto (tai useampi)	Popup –ikkunat, jotka pomppaavat ruudulle ja kertovat virheestä. Näitä on hyödynnetty myös kirjautumisessa(Käyttäjänimi on varattu tai tunnukset ovat väärät), sekä ruokadatan kanssa. Työhön on myös sisällytetty esteettinen toolbar, jota klikkaamalla pääsee päävalikkoon.	6

Summa		47->40

Työmäärät

Tekijä	Tehtävät	Tunnit
Santeri Hynninen	Devaus, UI, suunnittelu	60-70h
Petteri Raina	Devaus, viimeistely, testaus	50-60h
Eemil Aspholm	Devaus, suunnittelu	60-70h
Summa		Noin. 185h

Mitä opin harjoitustyöstä?

Santeri:

Opin paljon Android studiosta, Javasta, mobiilikehittämisestä sekä varsinkin tiedonhausta tämän kaltaisiin projekteihin. Mielestäni projekti oli mielenkiintoinen toteuttaa, ja sen parissa oppi paljon. Hieman harmittaa, ettei kaikkia olio-ohjelmoinnin perusasioita kuten periyttämistä tullut käytettyä ohjelmassa, koska sille ei suoranaisesti ollut tarvetta ja ryhmätyöskentelyn ansiosta se olisi vaatinut paljon enemmän työtä.

Petteri:

Opin olio-ohjelmoinnista, fragmenttien ja aktiviteettien käytöstä, ja etenkin käyttöliittymän rakentamisesta ja APIn käytöstä paljon. Pidin siitä, että harjoitustyö ei ollut vain teoreettista, vaan rakensimme käytännössä hyödyllisen sovelluksen, joka käytti avointa dataa.

Eemil:

Opin aktiviteeteista, fragmenteistä ja kirjastoista sekä niiden mahdollisuuksista android studiossa. Myös Apien käyttäminen ja sieltä saadun datan hyödyntäminen tuli tutuksi

projektia tehdessä. Harjoitustyötä oli mukava tehdä sillä siinä oppi mielestäni enemmän asioita kuin itse kurssilla yhteensä.