Expertise
and insight
for the future

Teemu Rytsölä

Optimal Wear OS application for diabetes care

Metropolia University of Applied Sciences

Bachelor of Engineering

Information and Communication Technology

Bachelor's Thesis

30 October 2018

Author	Teemu Rytsölä
Title	Optimal Wear OS application for diabetes care
Number of Pages	42 pages + 1 appendices
Date	30 October 2018
Degree	Bachelor of Engineering
Degree Programme	Information and Communication Technology
Professional Major	Mobile Solutions
Instructor	Petri Vesikivi, Principal Lecturer

This thesis was made as a research and development project for a Finnish software development company focusing on health and medical software. The goal was to research what makes an optimal diabetes care application for wearable devices such as smartwatches, more specifically for Wear OS platform, and develop a prototype application based on this research.

The research included insight on the diabetes care market, current technologies and solutions for diabetes care along with future expectations and technologies that are being developed. Thorough user inquiry about smart care and wearable technologies in diabetes self-care was performed to find out information directly from diabetics. Wearable UI and UX guidelines and technologies were also studied as a part of this process.

As a result of research from not only medical technologies and solutions but also from wearable technologies, a minimum viable application was developed on Android Wear 2.0 platform on a Motorola 360 smartwatch. This application summed up the user desires and improvements based on the user research and testing.

The outcome of the thesis is a detailed documentation of research for what an optimal Wear OS application for diabetes care should possess, what it takes to develop a diabetes care wearable application and how it was developed and finally a prototype of the application. It also contains information and speculation about future possibilities in terms of wearable technology in diabetes care and health care, such as non-invasive blood glucose measurements and electrocardiogram measurements.

Keywords	Wear OS, Android, Smartwatch, Diabetes, Smart care

Tekijä Otsikko Sivumäärä Aika	Teemu Rytsölä Optimaalinen Wear OS -sovellus diabeteksen hoitoon 42 sivua + 1 liite 30.10.2018
Tutkinto	Insinööri (AMK)
Tutkinto-ohjelma	Tieto- ja viestintätekniikka
Ammatillinen pääaine	Mobile Solutions
Ohjaaja	Yliopettaja Peri Vesikivi

Insinöörityö tehtiin tutkimus- ja kehitysprojektina suomalaiselle yritykselle, joka keskittyy terveys- ja lääketieteen alan ohjelmistokehitykseen. Työn tarkoituksena oli tutkia, mitkä ominaisuudet tekevät älykellosovelluksesta optimaalisen sovelluksen diabeteksen omahoitoon Wear OS -alustalla, ja toteuttaa prototyyppisovellus käyttäjätutkimukseen perustuen.

Tutkimukseen kuului markkinatutkimus, jossa selvitettiin olemassa olevia teknologioita ja ratkaisuja diabeteksen omahoitoon sekä odotuksia ja teknologioita, jotka ovat yhä kehityksessä. Tutkimuksessa toteutettiin kattava käyttäjäkysely älykkäistä hoitotavoista ja älykelloteknologioista liittyen diabeteksen omahoitoon, jotta saataisiin tietoa itse diabeetikoilta. Myös älykellojen käyttöliittymä ja käytettävyys toimintatavat sekä teknologiat olivat osa taustatutkimusta.

Terveysteknologioiden, sovellusten sekä älykellojen taustatutkimuksen perusteella itsenäinen vähimmäisarvoinen sovellus kehitettiin Android Wear 2.0 -alustalle Motorola 360 -älykellolle. Tämä sovellus sisälsi käyttäjien toiveet ja parannukset perustuen käyttäjätutkimukseen ja testaukseen.

Työn lopputuloksena oli yksityiskohtainen ja dokumentoitu insinöörityö siitä, mitä optimaalinen Wear OS -sovellus diabeteksen omahoitoon vaatii, mitä tämänkaltaisen sovelluksen kehittäminen vaatii sekä miten sovellus on kehitetty ja lopuksi itse prototyyppisovellus. Työ sisältää myös tietoa ja pohdintaa tulevaisuuden mahdollisuuksista liittyen älykellon käyttöön diabeteksen omahoidossa ja terveydenhuollossa, kuten ei-invasiiviset verensokerimittaukset ja elektrokardiogrammimittaukset.

Avainsanat	Wear OS, Android, älykello, diabetes, älykäs hoito

Contents

List of Abbreviations

ORM	Object-relational mapping. The set of rules for mapping objects in a programming language to records in a relational database, and vice versa.
DBMS	Database management system. Software for maintaining, querying and updating data and metadata in a database.
UI	User interface. The interface that lets the user interact with the software.
API	Application Programming Interface. Set of subroutine definitions, communication protocols, and tools for building software.
SPOT	Smart Personal Object Technology. Microsoft's first take on technology related to smartwatches.
CPU	Central Processing Unit. Electronic circuitry that performs the calculations.
BG	Blood glucose. Levels of glucose in blood stream.
LTE	Long Term Evolution. Advanced type of 3G connection.
Wi-Fi	Radio technology for wireless local area networking.
SQL	Structured Query Language. Programming language used to manage databases.
MVP	Model View Presenter. Architecture pattern where presenter acts as an intermediate between view and model.
IDE	Integrated Development Environment. Environment to write code in and compile it to an application.
DAO	Data Access Object. An object that provides an abstract interface to a database.
ID	Identifier. Identifier used in computer science

CEO	Chief Execute Officer. The highest-ranking officer in a company, answers to the board.
CGM	Continuous Glucose Monitoring. Continuous insight into glucose levels.
BGM	Blood Glucose Measurement. Single reading of blood glucose levels.
HbA1c	Hemoglobin A1C. Long term measurement of blood glucose levels between 2-3 months.
NFC	Near Field Communication. Radio technology used to transmit information.
UX	User Experience. A person's experience in using a product.
GPS	Global Positioning System. A satellite-based radio navigation system.
AI	Artificial Intelligence. Intelligence demonstrated by machines.
OS	Operating System. The platform's main software that everything else is run on.
LCD	Liquid Crystal Display. Display technology used to create slim and light displays.
REST	Representational State Transfer. Architectural style for implementing HTTP APIs.
HTTP	Hypertext Transfer Protocol. Application protocol used for browsers and WWW-servers.

1 Introduction

Diabetes or Diabetes mellitus is a group of metabolic disorders in which there are high blood sugar levels over a prolonged period. It has become one of the most serious health care problem all over the world. In the year 2000 the estimated number of people suffering from diabetes was approximately 171 million and was expected to rise to 366 million by 2030. (1, p. 1)

Diabetes self-care has not only become a business but a necessity. With its growing numbers more and more people are affected by diabetes and aren't getting the care that they need. In recent years the market for diabetes self-care applications, hardware, and for all kinds of health and medical applications has risen significantly. Due to digitalization, we are advancing quickly to a stage were most services and products can be enhanced or made easier to access on a mobile platform such as smartwatches or smartphones.

In this thesis, I studied current software and hardware solutions concerning health, medical and diabetes safe-care applications to find out what makes them so efficient and useful. I also studied some aspects related to the business side of diabetes care and smart care technologies. For my own implementation of an optimal standalone Android wear application, I wanted to utilize my research not only from other applications but also get feedback from users who would be using these applications, users who knew what the actual struggles were when it comes to living with diabetes. The user research inquiry was conducted in collaboration with Topi Penttilä.

The implementation was done as a standalone Android wear application on a Motorola 360 smartwatch with Android Wear 2.0 and development was also done on an Android wear emulator. This smartwatch would allow me to gather user health data such as heart rate and activity using integrated sensors and would also provide a round user interface for users to manually enter data and interact with the software. Based on my results I would design and create an optimal Android wear application for diabetes self-care. After finishing development of this application, I performed user testing with it to receive feedback and open up more discussion about the future of wearables in healthcare.

For developing this software, I used a free IDE, or Integrated Development Environment, Android Studio by Google. Development was done with programming language Java version 8.

2 Market Research

The market for diabetes care is huge and constantly growing, in 2013 the expenditures were approximately 548 billion USD, and had a growth rate of almost 23% in just two years, up to 673 billion USD. According to the global diabetes community, the estimated number of diabetics right now in 2018 is 415 million (2, p. 51). For reference, 415 million people with diabetics is estimated to be 1 in 11 of the global adult population. This number of diabetics already overshadows previous estimates made in year 2000 for year 2030 of 366 million. Global diabetes community also estimates that 46% of people with diabetes are undiagnosed. By 2040, there is expected to be 642 million diabetics worldwide. (3)

In 2010 the diabetes care device market in Brazil, Russia, India and China was estimated to be around 489.7 million USD and was expected to grow up to 1.1 billion USD in 2015. This care device market includes devices such as insulin pumps, continuous glucose monitoring systems and self-monitoring blood glucose systems (4). The care devices are not the only expenditure. Image 1 shows that the diabetes-related healthcare expenditures are massive and average approximately 727 billion USD.

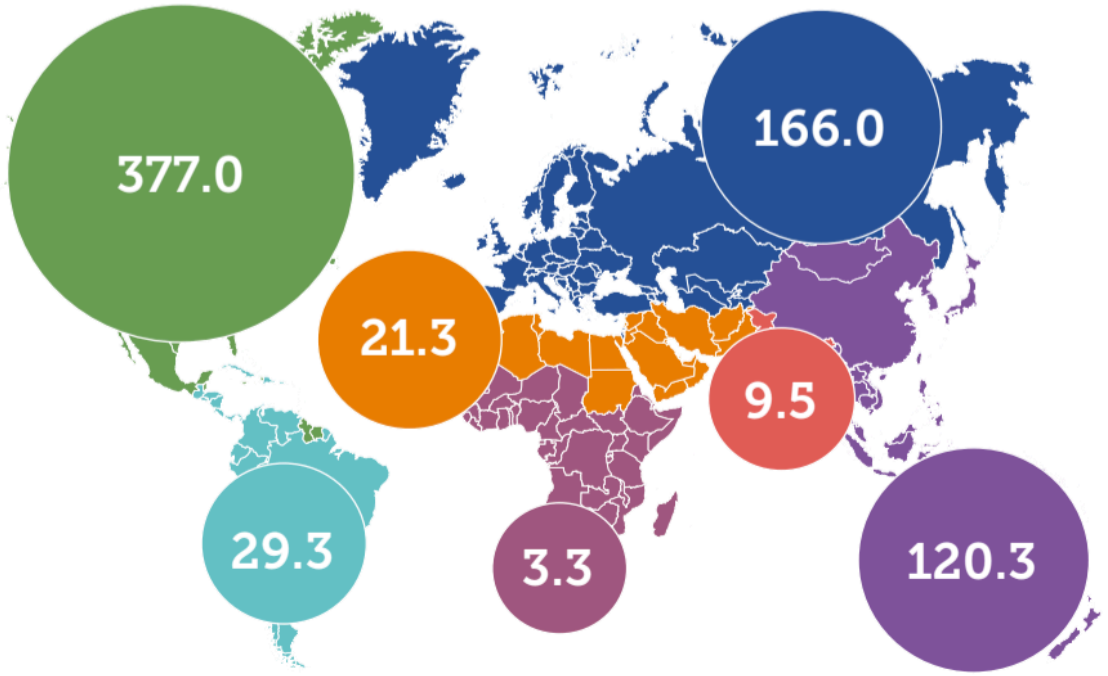

Image 1. Diabetes-related healthcare expenditure in adults in 2017. USD in billions. (2, p. 40)

2.1 Smart healthcare services

Diabetes-related healthcare expenditures are massive and the number of patients keeps growing. This is where using smart healthcare services such as mHealth, or mobile health, come in, since they can potentially be very effective for diabetic patients and cut costs. Using mHealth services and systems comes at a cost however, patients and health care professionals must be trained to use these tools accordingly. Traditionally these services and systems refer to patients that are using a smartphone or notebook input their medical data, be it manually or through automated sources like Bluetooth to access them. The professionals caring for the patients can monitor patient data and communicate with the patient through the service's website. After 6 months of applying these types of smart care service methods in Daegu, Korea, all 56 patients that were under the program had lower blood glucose levels and HbA1c, which is your average blood glucose levels over the past two to three months. However, differences in body weight and body mass index were insignificant. The conclusion was that smart care services can definitely be effective when it comes to lowering HbA1c and blood glucose levels. (5)

Mobile technologies such as smartphones, smartwatches, laptops and tablets can offer excellent opportunities for using mHealth. Especially type 2 diabetics can find these smart mobile healthcare services to be significantly more helpful. Mobile healthcare services were able to improve the HbA1c values up to 0,3% for patients that suffer from type 1 diabetes and 0,8% for patients with type 2 diabetes, which can be significant improvement for the patient. Interventions through mHealth provide a promising approach when it comes to self-care in diabetes. (6)

For people suffering from diabetes the future can definitely be bright in terms of digital solutions for managing the disease. Diabetes is an exceptionally good target for innovations in digital health due to its high prevalence and morbidity from complications that can be prevented, not to mention the reliance on data and lifestyle monitoring. In the last three years fitness and health applications have grown by 330%, but the evidence is still small in terms of actual benefits to people with long-term conditions such as diabetes. (7, p. 911)

2.2 Smartwatches

There has been a lot of commercial activity concerning smartwatches in the recent years, but smartwatches started already back in 2002 when Microsoft announced their SPOT or Smart Personal Object Technology watches. The SPOT watch became available later in the year 2004. While the SPOT watch was not a widespread success and was eventually pulled from the market due to not being able to address the real user needs, it was still an important stepping stone in the path of developing wearable technology. Technology in smartwatches has come far from that thanks to smartphones becoming more relative and advanced with the app store model. Smartwatches today still use the same infrastructure from smartphones giving opportunities for third parties to develop software and also deploy it to huge audiences. Bluetooth is used to link a smartphone with a smartwatch, so that it can leverage apps and cloud services. (8)

There are a few approaches one can entertain when viewing the development of smartwatches, disruptive innovation, sustaining innovation and radical sustaining innovation. The disruptive innovation theory, which means that while smartwatches are unlikely to be low-end disruptions, meaning cheaper and worse versions of current products. They can, however, be a market disruption, in the sense that they would be considered as taking a technology and applying it in to a different domain to help solve a new problem, thus developing a new market. In this case the technology is referring to smartphone technologies such as small touch screens and powerful computing platforms in small form factors. (8)

The sustaining innovation in the context of smartphones, as previously mentioned, a lot of smartwatches depend on a smartphone for its operation. However, some smartwatches are becoming more independent, with direct Wi-Fi connections and even cellular connection. In sustaining innovation, a smartwatch is considered to be an extension of the smartphone, something to make the experience of a smartphone better, rather than being an experience of its own. For example, smartwatch notifications are a lot more accessible to the user, being able to quickly glance at something tied to your wrist rather than pulling out your phone from your pocket or bag. You can use your watch as a remote for your phone apps, like by controlling the music player or respond to messages with preselected responses. But there are arguments against this as well. Is making the information more accessible justified considering the tradeoffs of potentially worse user experience through, for example, worse interface or slower processor? Or can the watch

notifications be considered as even more aggressive disruptions since some people express that they already have too many notifications from their phone alone? (8)

When considering a smartwatch, another aspect that can be taken is how do these devices perform as a watch. In radical sustaining innovation, a watch would be performing the tasks of a traditional watch, while bringing in a lot more capabilities like applications and notifications. Taking the initial values of a basic watch, and enhancing them with a radically different technology, that is what radical sustaining innovation means in the context of smartwatches. This view might seem really general, but the point is to accomplish a watch with more advanced technology without sacrificing the values of a traditional watch. This means things such as not needing to charge the watch daily and having long battery life, this is achievable with more conservative CPU for less complex tasks, low power consumption LCDs and hardware buttons instead of a touchscreen. (8)

A report from 2016 by the International Data Corporation estimated that smartwatch sales will reach 17.8 billion USD by the year 2020. The most common operating systems in the field are WatchOS, Wear OS, Tizen and Real-Time Operating System. Out of these four, Google's Wear OS, Apple's WatchOS and Tizen are the most popular with market shares of 22,9%, 52,3% and 12,7% in 2016. The benefits of a smartwatch product in a customer's perspective are generally increased by software, hardware and appearance and therefore the factors indirectly affect the purchase intention. Potential customer's positive attitude and perceived values towards smartwatches also directly affects purchase intention. (9)

When trying to imagine the ideal smartwatch I would consider it to be something in between a radical sustaining innovation and a disruptive innovation. I believe that we are currently at a point where smartwatches are in a sustaining innovation stage, being merely extensions of a smartphone. As time goes on and technology advances further I believe eventually smartwatches are going to be making smartphones more and more obsolete, stealing away their responsibilities one by one. Google and Apple are already pushing for artificial intelligence voice command assistants like Siri and Google Assistant on smartphones and smartwatches. These assistants don't require huge computing power on the device or a large screen to interact with therefore nullifying the advantages of smartphones. However, they do require internet connection to access their cloud services, which can not quite yet be taken granted in today's smartwatches.

Wear OS is an operating system designed for smartwatches and wearables by Google. It was initially called and published as Android Wear in 2014 but was later rebranded to Wear OS. LG Electronics and Motorola Mobile were among the first to confirm that they would be releasing their own watches on this new software platform. According to Sundar Pichai in 2014, Google's former senior vice president of Android and Chrome, current CEO, wearables are just starting to grow and that we have barely scratched the surface of what they are capable of. Google is also pushing hard with to make user interactions with wearables a lot easier by providing the voice-controlled Google Assistant. (10)

While Apple has been ahead of Google when it comes to wearables most notably with the Apple Watch and WatchOS, Google has the resources to challenge them and provide a great wearable experience with continued development on Wear OS and upcoming google wearable products. It is even rumored that Google will finally bring to market its own Wear OS powered smartwatch, that the industry has been begging for. (11)

Having Google create both the hardware and software for this product is something that has already happened with Google's pixel smartphones, that are often considered as one of the best flagships in the Android market. This idea of working to create one product by one company is ideal with large amounts of resources and shows us why Apple has been so successful. Companies are able to push the limits of software and hardware when they are the ones that are providing both, optimize both platforms to create one efficient product.

2.3 Google Fit

Google Fit is a very common and favored health, fitness and lifestyle application. This simple and efficient application developed by Google LCC, is among the most used with over 10 million installs. It also averages a rating of 3,9 in the Google Play Store. Google Fit is free, it provides the user with effortless background tracking of your daily activities like running, walking or bicycling on your Android smartphone or smartwatch. The application also lets you view your previous activities, measures approximate calorie burns throughout your day. It also sets goals for the user to reach their daily activity amount (12). The most valuable thing about using Google Fit, however, is in my opinion the fact that developers can integrate Google Fit in to their own applications and use the data it gathers automatically.

While Google Fit is not a direct medical or diabetes care application, I decided to include it since a lot of the medical and health applications take advantage of its API. From my own experiences, it has become essential in monitoring my daily activities and motivating me to complete my goals of 10 000 steps a day for example. I also found it extremely helpful that it can calculate my calories burned by just staying awake for the day and for all my exercise. It seamlessly imports my data to other apps at my wish, my meal monitoring app now has access to how many calories I have burned through the day and helps me gain or lose weight, whatever I have set as my goal. Multiple Diabetes applications also integrate Google Fit for this exact same reason, it provides them with the data that lets their application be a lot more effective while not cluttering it with too many features and settings.

2.4 Diabetes:M

Diabetes:M is a popular diabetes self-care application developed by Sirma Medical Systems. It has over 100 000 installs in the Google Play store with an average rating of 4,6 across 15 621 reviews (13). Diabetes:M is free to install but displays ads to the user in the application, however ads can be removed by subscribing to their service. The subscription in Diabetes:M costs 4.99 USD a month, with discounts for 6 and 12-month subscriptions (14). The service also includes some new features such as Bluetooth integration for the most popular glucose meters, smart assistant that utilizes artificial intelligence to help you with your diabetes care among many other things. Diabetes:M also supports Google’s Wear OS and Apple’s WatchOS. (15)

After testing out the Diabetes:M app myself, while I can not fully criticize the effectiveness of this application for diabetics, I can attempt to test the features and UI designs of the application and make conclusions from that. I like their approach of a subscription model-based application, the idea is to lure in users through free application, make users realize how useful it is, and then have them feel like they want and need the paid subscription version. However, the problem with this kind of business model is that a lot of the applications using this model end up overdoing it. Stripping way too many core features from your free users can lead to them being unsatisfied and frustrated with the application itself or not limiting it enough so that users will not convert to paying customers (16).

In my opinion Diabetes:M does the subscription model fairly well, they limit the following features; smart assistant, additional profiles, expanded food database, blood glucose pattern analysis, data synchronization through different devices, Bluetooth integration for most popular blood glucose meters, additional lab results, no advertisements, medical reports, export and import options (15). Out of all these features that are locked behind the paid subscription, I feel only the Bluetooth integration for blood glucose meters is out of place and should be standard for all users.

I mainly wanted to focus on the UI of the Diabetes:M Wear OS application, for it was more in line with this project's theme. When designing UI layouts for Wear OS, different devices must be considered. Wear OS can have watches that utilize a more traditional round watch face, or a square watch face based on the smartwatch's screen (17). Looking at the two watch faces provided by Diabetes:M for your Wear OS, they provide an analog style clock and a digital style clock. Providing of course an analog and a digital style is very traditional in Wear OS applications, but as seen in image 2, the watch face also tries to display other information like your last measured blood glucose levels and at what time it was taken. It also displays your active insulin and your last insulin dose with the time it was taken at. Seems like your last carbohydrates and the time they were consumed are only displayed in the analog watch face.

Image 2. Diabetes:M Wear OS circle watch faces. (18)

When considering the way these watch faces indicate time, they are not perfect. The color palette seems like an awful choice, especially with the analog watch face. Having black background is good for contrast but having light blue notches for hour and minute

indicators in the dial, with red and white hands indicating the time gives the watch face a messy look (19). User should be able to take a glance at the watch face and receive simple information from an uncluttered screen (20). Diabetes:M also displays way too much information on its watch face, even if the containers displaying the data are according to Wear OS material guidelines. In my opinion parsing the carbohydrates out of the watch face and creating a clearer way of indicating the data would make the experience a lot better.

The ways of entering data through the watch interface in Diabetes:M is also very confusing. After some trial and error, I managed to be able to save my inputs. Designing something like this, where you input multiple different values can't be easy. But the key is to try and separate each input into its own view, which Diabetes:M manages to do. I do think the way of saving your data input was too vague and could be separated to its own screen.

2.5 Continuous glucose monitoring applications

Dexcom is a company that manufactures CGM, or Continuous Glucose Monitor, sensors & insulin pumps. Dexcom also provides an application, Dexcom G5 Mobile, in the play store & app store to view this data on Android and iOS operating systems. Applications for smartwatches also exist, for Wear OS from Google and WatchOS from Apple. Unlike a traditional blood glucose meter, or BGM, which provides a single reading of a patient's glucose level, CGM systems can provide dynamic blood glucose level information every five minutes. Maximum amount of readings per day is 288. The benefits of CGM according to Dexcom are a reduction in A1C, reduction in hypoglycemia and better monitoring of glucose levels at night with alerts. (21)

Dexcom G5 Mobile has over 10000 downloads in Google's play store, with an average rating of 3.029 out of 5 (22). The smartwatch application provided by Dexcom to Wear OS isn't a standalone application, meaning you must download the smartphone application in order to use the smartwatch application for Android. Showing continuous glucose on your smartwatch sounds like a good idea, but ultimately feels a bit unnecessary since a smartwatch is not really the correct platform to be viewing large amounts of data in graphs due to small screen sizes and limited interactions. As seen in image 3, Along with

the latest glucose reading, Dexcom does provide a trending arrow for your blood glucose levels which is essential information for diabetics. (23)

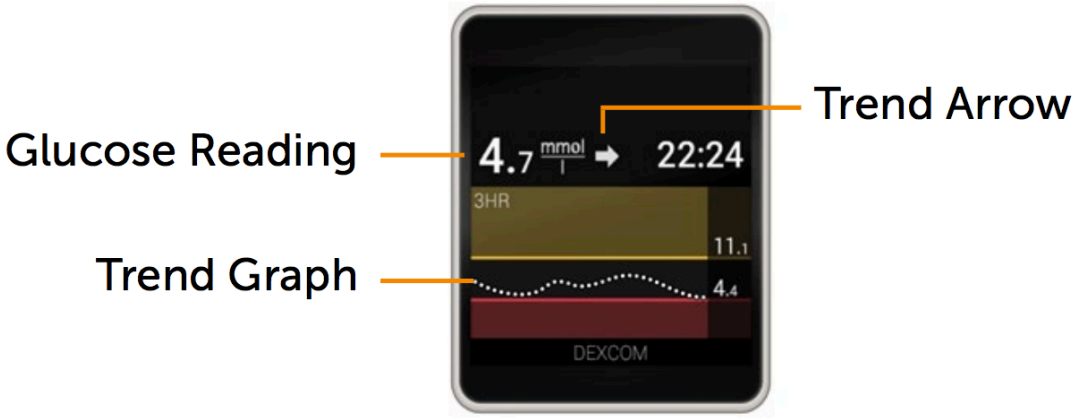

Image 3. Dexcom's CGM application on Android wear. (23)

Another thing to note about this application not being a standalone wearable application is that it requires you to have your smartphone with you at all times, if you wish to receive information from your CGM sensor to your phone. The sensor data is then forwarded to your Android Wear smartwatch and displayed there, so the sensor can't be directly connected to your watch. This also means your configured settings from CGM sensor only directly alert your smartphone or separate monitor, however similarly to the CGM data, these can be forwarded to your smartwatch as notifications. Dexcom also has an application, Dexcom Share, that allows users to share their glucose levels with other users. This is especially helpful for parents that have children suffering from diabetes. (23)

One popular sensor is also Abbott's FreeStyle Libre. The way this sensor works differently from Dexcom's CGM sensor is that instead of sending continuous information about blood glucose levels to your monitor or smartphone, it instead uses NFC, Near Field Communication, to transmit data to your smartphone or FreeStyle Libre reader. When the user scans their FreeStyle Libre sensor, all the blood glucose level data gets from the last 8 hours gets transmitted to the device. (24)

As of this moment FreeStyle LibreLink, which is an application for smartphones on Android or iOS does not yet support wearables. However, the application has approximately over 90000 installs over multiple countries (25). It provides very similar ways to view your blood glucose measurements within a smartphone application as Dexcom's G5 Mobile.

It supports Android version 5.0 and higher and requires near field communication sensor in that device. (26)

3 User research

The aim of this user research was to survey the actual and most essential needs that diabetics require in a wearable application.

3.1 Participants and data collection

Me and Topi Penttilä designed an inquiry the purpose of which was to be short and straight forward enough so that the respondents would feel inclined to answer the questions properly and accurately. This kind of inquiry would also let me reach a large amount of people and give me more insight to what the most important factors are when it comes to diabetes self-care and technology concerning it. The inquiry was aimed at diabetics and health care professionals that work with diabetics. I managed to reach 146 people through my employer's social network channels, Finnish diabetes union and diabetes association. The inquiry was done in Finnish. The questions that were asked in this inquiry can be seen in appendix 1.

Starting with basic information such as gender, age and the part that diabetes plays in that respondent's life, for example if they are diabetic themselves, working with diabetics, or both, even if they are just interested in the subject. It is worth nothing that based on the respondent's answers, they would be asked different questions in the inquiry, so a non-diabetic could not respond to all the same questions as a diabetic.

If the respondent chose to answer the inquiry as a diabetic, they would be followed up with more questions about their diabetes. I wanted to find out what type of diabetes they are diagnosed with, for how long and what they would say are the biggest challenges when it comes to living with diabetes.

I also wanted to know the respondents' experiences and knowledge when it comes to intelligent diabetes self-care. Key here was to keep the questions simple and understandable since I expected most of the respondents to be elderly people that aren't necessarily aware of all the possible technologies out there. Basic information on whether they are using a smartphone or smartwatch at all, if so what operating system does it run.

I focused on mapping what other self-care tools the respondents were using; for example, if they have experience with intelligent self-care solutions, insulin pump or Bluetooth blood glucose meters. Ending the inquiry with more specific questions related to smartwatch features, wearable apps, and possible beneficial features that a diabetes self-care app would have.

Respondents could also leave their name, email and phone number in case they wanted to possibly be involved in developing this application further in the future.

3.2 Results and analysis

The results of this inquiry show approximately 73% respondents were female and 27% male, this however does not reflect the gender spread in terms of diabetes since generally there are more males with diabetes than females with diabetes (2, p. 75). However, this could perhaps be explained due to sociological reasons and activity in social media between males and females.

As seen from image 4, the majority of respondents were between ages 36 and 45 (27%). This age division of the diabetics in this inquiry is not reflecting the actual data, since majority of people with diabetics are people between ages 60 and 80 (2, p. 75). The age distribution is more likely due to the ways this inquiry was conducted, since social media channels are more desirable and familiar to younger audiences.

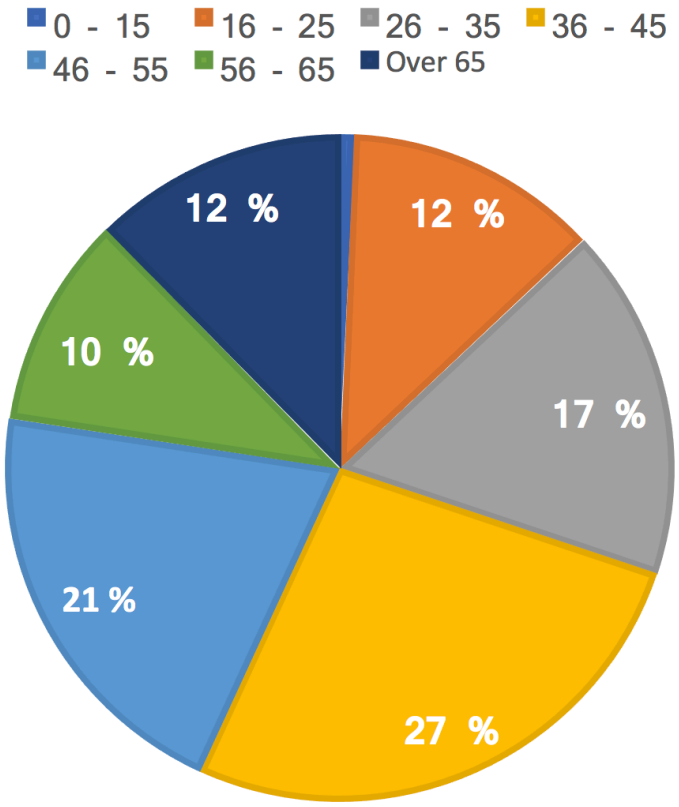

Image 4. The age disparity of respondents in the inquiry.

Over 87% of respondents were diabetics, but answers were also received from health care professionals that work with diabetics up to 8% of all the respondents. 71% of the diabetics were diagnosed with type I diabetes, and up to 78% have been living with diabetes for over 5 years. Among the biggest issues that these diabetics had was controlling their carbohydrate intake and therefore managing inconsistent blood sugar levels. Almost 69% of the respondents used an Android smartphone and 27% were using iOS. 4% of people didn't use smartphones, could not tell what their operating system was or that they were using some other operating system. Smartwatch users were few since 125 respondents didn't use smartwatches at all. The few 13% of respondents that said they were using smartwatches included operating systems from Android wear, Apple watch and other health or lifestyle watches.

Over 56% were aware of intelligent solutions such as smartphone or smartwatch applications that could be used in diabetes self-care. Respondents were able to choose mul-

tiple options in a question about diabetes self-care devices. 30% of respondents mentioned using an insulin pump and a smartphone application to help with the diabetes self-care was up to 40%, but only around five respondents said they were using a smartwatch or smartwatch application for their diabetes care.

When asked what experiences diabetics had had with smartwatch applications geared towards diabetes care, most of them had no experiences at all. A few of the respondents mentioned Balansio, which is another growing diabetes care application for smartphones and smartwatches, along with Diabetes:M and FreeStyle LibreLink. LibreLink is an application for smartphones, that lets you scan your blood glucose from a small sensor attached to you that is constantly measuring your blood glucose levels (26). Seeing their blood glucose levels from their smartwatch within a glance was also a popular answer. The trend of wanting to see your latest blood glucose levels and whether it's rising or not continues throughout other questions of what they would like to see in the smartwatch application as well. Features that would also be important to respondents were automatic data input from things like sensors and Bluetooth meters for blood glucose levels or active insulin, activity, step counts, and calories burned were also requested. Being able to set reminders or alarms was another popular request, mainly these would be used for notifying the user when to take medicine or measurements.

The following user needs were identified based on the answers to the open questions about the optimal wearable application. Diabetics want to be able to see their latest blood glucose levels on the smartwatch at a glance, they want to have reminders for taking medicine or measurements and if possible automatic data inputs from other sources like Bluetooth meters or Abbott's Libre sensor along with automatic activity tracking from other applications like Google Fit or Apple's Health Kit.

3.3 Limitations of research

A few issues with this inquiry were the fact that the sample size of respondents while relatively large for a thesis could still have been larger, 146 respondents is not enough to make 100% correct conclusions based on the answers when there are 415 million diabetics in the world. Another one is that this inquiry was done in Finnish and all the respondents were presumably Finnish as well, meaning this poll is not necessarily relevant to other countries with perhaps different cultural or economical standards.

To make inquiry more accurate it would have to be translated to English or preferably to every respondent's native language and be distributed to a much larger audience that would still have the desire and time to give legitimate answers. On improving the inquiry, I also received feedback from respondents saying that the option for choosing age should be more precise and that some people use multiple smartphones that may have different operating systems as well.

4 Diabetes care application

After researching other applications including their features and UI solutions, performed a thorough survey on what the wishes of diabetics were for an optimal diabetes smartwatch application. Studied the market of diabetes self-care and smartwatches in order to justify an application for a device that is going to becoming even more relevant in the future in terms of health care, I can decide the requirements for this project's implementation.

4.1 Defining requirements

Based on user feedback, most important feature that I decided should be on this smartwatch application is inputting your blood glucose levels manually and if possible, through a blood glucose meter or a sensor. Manual input would not be too hard to implement but designing an intuitive way of inputting the values could be more difficult. Automated sources for measuring blood glucose could be a bit trickier to implement, but without the complications of creating an intuitive UI for inputting the data.

Possibility for attaining automated inputs could be through Dexcom. Dexcom does provide a free to use REST API for retrieving CGM data that has been backed up to their cloud services. However, this would require the user to be using Dexcom's own application first on your smartphone to back up the data to the cloud, and then retrieve that information from the cloud to the smartwatch, ultimately defeating the purpose of CGM. There would also need to be way for users to view this data input history, a diary view that would list all previous inputs and order them based on when the input was made.

Among the popular requests were also reminders or alarms for taking medications or measurements, so implementing something like this was also very important. It also would make a lot of sense for a smartwatch to be able to do something like that. Indicating what the reminder or alarm is for could be difficult, gaining access to all different types of medications that diabetics could use would be hard as well. Instead of having a database full of different medications, user could simply set an action for a reminder, like 'take medicine' or 'take measurement'. Optional note field could also be added so that the user can specify which medicine and so forth, although the complications of inputting text into smartwatch become relevant here.

Displaying the latest blood glucose levels on the watch face is a must. I do not necessarily think anything else is as important to show on the watch face aside from the actual time. One idea could be showing the reminder alarms in the watch face by highlighting the time in the analog watch or displaying the time of the upcoming reminder. Color coding things correctly here will be crucial for making a clear and concise user experience.

Feature that would be nice to have is automated data integration with the Google Fit to get activity measurements. With activity data and carbohydrate data input the application could predict lowering or rising blood glucose values and show the user which way it's trending.

Implementing all the features into this would require too many resources but they are definitely something to consider in the later versions of the application. In conclusion, the optimal application would have the following features in an order of priority:

- Automated data input
- Manual data input
- Reminders
- Latest input value
- Blood glucose level trend
- Google Fit integration for activity
- Personalized settings

4.2 Designing UI

When designing for Wear OS, there are few things that must be taken into consideration. The system is revolved around the core actions of suggest and demand minimizing the user interaction. Suggest meaning providing the user with information or suggesting actions without prior interaction. Demand as in asking for information or action from the user when it's needed. Wearable devices are supposed to deliver glanceable information the user in real time, allowing them to stay more connected to the online world as well as the real one. The interfaces should be clear, easy to read and the information should be organized using a clear hierarchy. Due to limit screen size, your interactions spots or tap targets should be well-spaced and easy to interact with, avoid crowding the with too much user input. The application should also be a time saver. Creating efficient flows

that make the users accomplish their tasks quickly is essential. When designing flows, you should avoid too many steps and complicated experiences for doing a task. (27)

When creating UI designs for this application, I decided to focus on usability and efficiency, rather than just style. The most optimized color palette for Wear OS would be a black and white color palette (19). Therefore, I chose a darker color palette with light blue accents, see image 5, to keep the UI simple and efficient while also taking into consideration red-green color blindness. I won't be displaying all the different views of this application to limit the number of images, but I will show the most important views and describe them all.

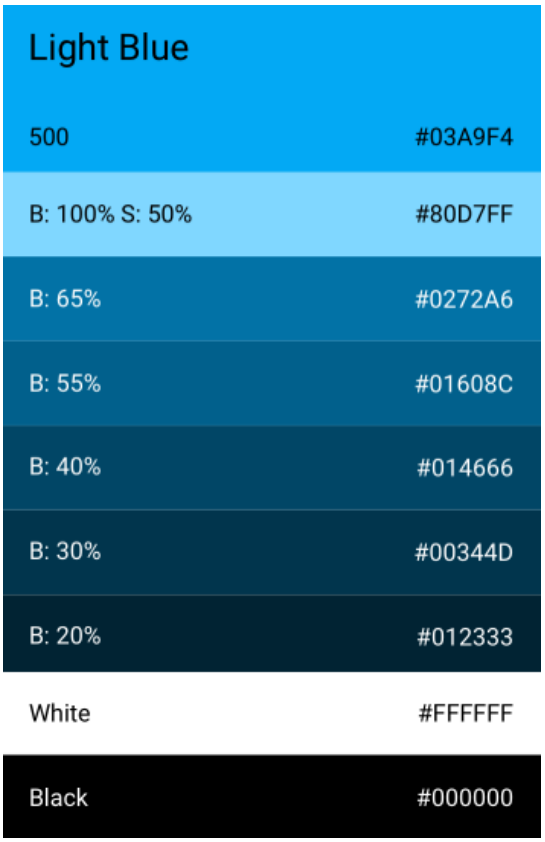

Image 5. Light blue color palette with standard black and white to be used in this project. (19)

For the watch face I wanted to make just a digital one for this prototype. As seen in image 6, upper half of the circle watch face holds the time with hours in bold and minutes in regular Roboto font. Using Roboto as font is standard according to Google's guidelines (28). Using black and a dark blue background for more optimal battery life along with white text that provides good contrast and makes it easy to tell what the time is and what your last blood glucose measurement was. I separated the time and the blood glucose

value to their own sections with a light blue color line, in addition to making the blood glucose value's background dark blue. For interactions, if user were to tap lower section of the watch face, they would be able to see full information of their last input such as time and other inputs that were submitted at the same time. Another possible idea would be to add a progress circle around the watch face indicating today's activity in percentages, full circle being 100% and so forth, if automated activity data source like Google Fit was integrated.

Image 6. Simple digital watch face design for the project.

Navigation between the views of inputting data, reminders, settings and diary would be done with the upper navigation menu commonly used in wear applications as seen in image 7 (29).

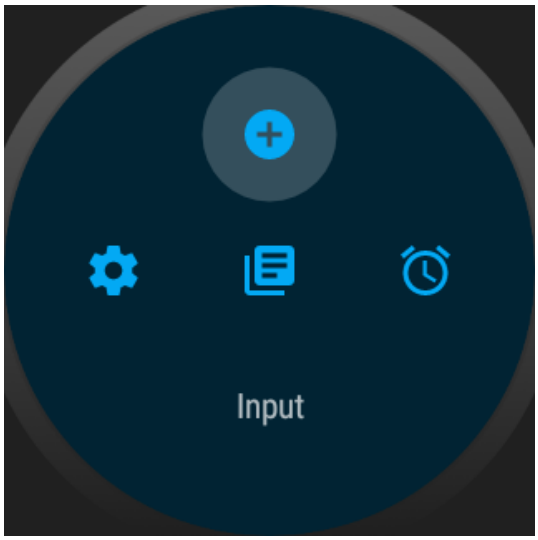

Image 7. Navigation menu containing the most requested functionality.

For data input, a simple plus and minus signs that add or subtract the value by one as default would be optimal, saving the value would always be at the bottom navigation menu. To choose between setting the decimal number or the integer, user would be able to tap the value to switch between the two. Selected number would be highlighted with a different color indicating that it is editable. This view can be seen in image 8.

Image 8. Simple input view for entering blood glucose levels.

For settings view, displaying the datatypes in a list would be enough, user could then tap each item to reveal more information about that datatype's settings and edit them. Similar hierarchy would exist in the diary view, all data entries that have been inputted by the

user would be shown in a list, ordered by the time of input, showing the most recent up top. Important thing to further visualize these lists to the user is to add datatype specific icons to indicate their difference, so that the user does not necessarily have to rely just on the name of the datatype, for example blood glucose datatype having a blood drop icon and carbohydrates having food related icon. By tapping entries, user could see more information about that entry and perhaps in edit that data entry.

Reminders view is hard to design since there is a lot of data that has to be inputted, generally you want to keep screens less crowded. My initial reminders view can be seen in image 9. User would be able to select the weekdays up top from the row of letters that represent each day of the week. Below the letters there's a native time selector widget and finally the traditional save action that exists in all the views that want to save data.

Image 9. Initial view design for creating reminders.

4.3 Improvements

After iterating through the designed views with colleagues like Ella Kaila-Kamath, a professional UX/UI Designer, I received feedback on what should be improved and what was good. I had immediate improvements for my initial watch face design. In image 10 of my new watch face design, the latest blood glucose entry also indicates whether the value is within the user's set settings. For example, if the value was higher than 12 the background would be yellow to indicate high blood glucose or hyper, and if the value is

below 4, background is colored as red to indicate hypo. Values in between are considered acceptable and display green background color.

Image 10. Improved watch face design with color indicator for value.

Improvements to the icons that I was currently using also came up. For example, using a diskette icon to represent saving is not necessarily as relevant anymore, and perhaps could be replaced with just text such as ‘save’. Indicating to the user in which view they currently are was also something that could be improved by adding a title or an icon up top. It also was not completely clear that the navigation menu is up top and can be swiped down. The navigation menu has a peek functionality that gets triggered whenever user interacts with the view, where it show’s a bit of itself so that the user can drag it down or tap it to show it as seen in image 11.

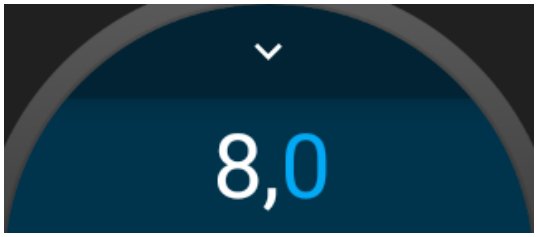

Image 11. Navigation menu’s peek functionality

The peek functionality could perhaps be improved with by adding that view’s specific icon instead of a downward facing arrow indicator, that would also resolve the issue of not indicating to the user which view they are in.

Improving the reminders view was also straight forward, input methods were good, but the screen is way too crowded. Dividing day selection, time selection and a note selection to three different tabs is essential. Indicating that the view has more than one tab would be done by showing arrow indicators in the right or left, user could then swipe left or right to switch between tabs or tap the arrows as seen in image 12.

Image 12. New reminders view with day selection divided into its own view.

I also had an idea of changing the way the week days are shown in the view but could not get a clear and concise resolution if they weren't ordered in one row. One option could also be to list the weekdays and allow selecting of those list items since spacing between the letters is still narrow.

The final navigation flow of the application can be seen in image 13, there are still few more views that could've been added to make it even more complex. For instance, editing in the settings view, and a time view on the watch face that on tap expands the time to full screen and shows a more detailed time and date.

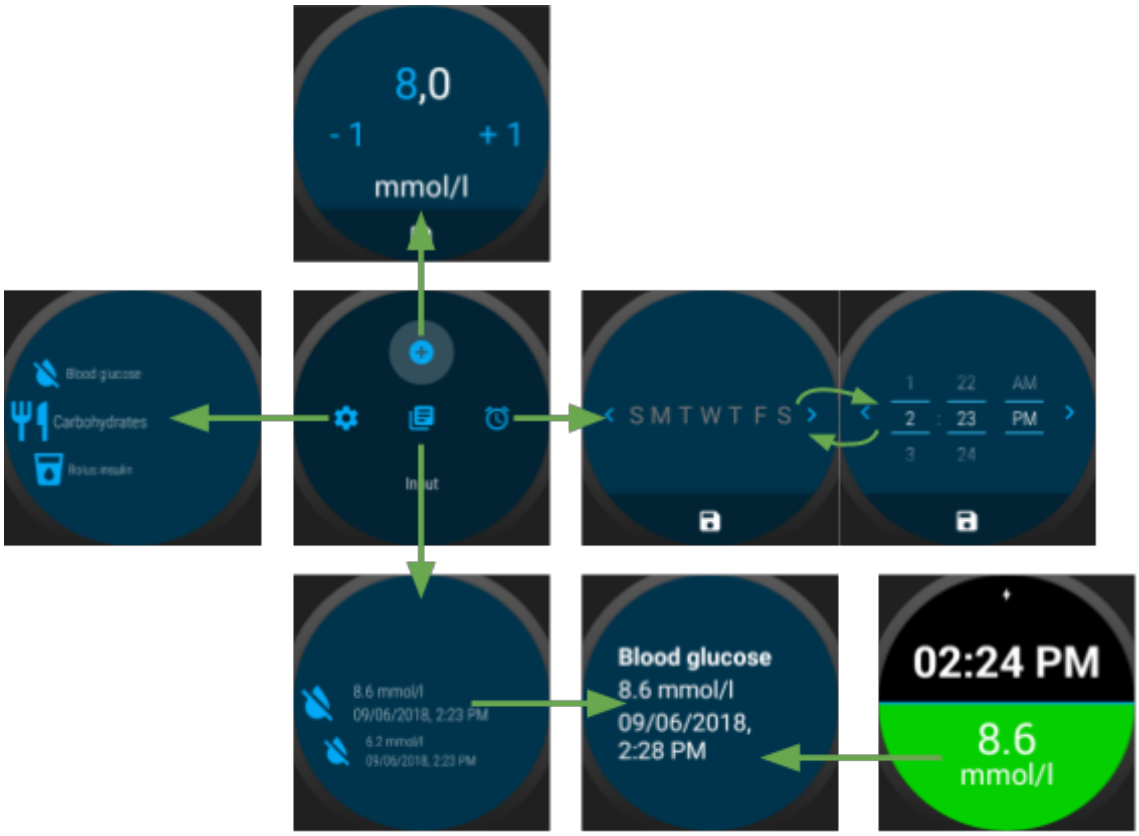

Image 13. Final navigation flow of the application.

5 Application implementation

Due to the CamelCase naming scheme commonly used in Java, when I explain my project's code in this chapter, I will follow the naming conventions in plain text as well. Instance variables and methods start with lower case letters and with words written together, for example data entry would be dataEntry, and a get method for data entry would be getDataEntry() with parentheses, as it is in code. When referencing Java classes and object types, names start with a capital letter and are also written together, such as in the case of data entry, it would be DataEntry (30). One last naming method that is used in .xml type files is that they are all lower case and instead of writing words together, they are separated by underscores. For example, data_entry_view.xml.

5.1 Architecture

Designing application architecture for this wearable project brought up some design and implementation issues. One issue that came into mind was dealing with a wearable device networking, it can be difficult and inconsistent since they don't usually support LTE also known as Long Term Evolution, aside from the most expensive and newest wearable devices. LTE is a technology used by wireless phone and smartphone companies around the world to provide a high-speed network in their devices (31). Therefore, most wearable devices rely on WiFi to connect to the internet or Bluetooth to receive data from the smartphone that is connected to the internet.

For this project, I decided against a networking capability due to the cost of implementing a good backend solution, and networking solution for the wearable device. In this case it would take away from designing and implementing the actual core features and functions of what makes an optimal Android wearable diabetes self-care application. However, implementing cloud support for a project like this is an obvious improvement that could and should be done in the future when the resources are available.

Since deciding against a networking capability for this project, implementing a reliable local storage solution was the correct thing to do. Android provides support for SQLite databases and this works well in Wear OS as well (32). I chose to use a dependency called OrmLite to help me manage and initialize my database. I also added Butterknife library to make binding views in fragments and activities more convenient.

For my actual application architecture, I went with a very MVP like approach. MVP, or model-view-presenter, is an architectural pattern that has derived from MVC, or model-view-controller. It is mostly used for building user interfaces where the presenter is the supervising controller that handles all the presentation logic between model and view. Application class, DiabetesWearApplication is responsible for maintaining global application state and holds other singletons that have access to the database. I also use just one activity, MainActivity, that creates a view with navigation. Inside that activity view's container, I populate a fragment view based on where the user navigates through the navigation menu. These fragments also usually hold modules responsible for handling that view's heavier tasks.

5.2 Development

When I started developing this application they key thing was that it was supposed to be a standalone application, meaning it should be able to perform independently without a smartphone. This means that generally your application should target at least API level 25 (33). This can be done in your application's build.gradle file. Standalone wearable application must also be declared in your manifest file as seen in listing 1.

```
<meta-data
      android:name="com.google.android.wearable.standalone"
      android:value="true" />
```

Listing 1. Declaring application as a standalone in AndoidManifest.xml. (33)

As for my development methods, after having my features and UI designs roughly figured out I started the development. I wanted to divide my features into their own tasks of watch face, navigation between fragments, and separate each fragment to be their own task. Mirroring the division of fragments, I also divided creating my models and accessors into their own tasks. I estimated this project having approximately around 4000 lines of code not counting generated files. After finishing development of my tasks to a reasonable extent, I started working all around the application, making small improvements incrementally to each part of it. Even though I developed this application alone I used git to handle the version control and back it up in my private Bitbucket repository.

Performing this project on a new platform was a challenge but also a good experience. I was able to use my current knowledge to great extent and still learn many new things

like how working with different screen types should be handled and the other limitations of a wearable platform. In terms of biggest challenges that I faced must have been designing an intuitive user interface and implementing it to work on a real wearable device correctly. On top of that I felt that I perhaps could have went more in depth with complications and customizability of the watch faces since I believe they are the key feature of any smart watch application.

5.3 Models

I have four types of data models that get persisted in the local database, Unit, DataType, DataEntry and Settings. All these classes are subclasses of an abstract class called BaseModel. Purpose of BaseModel is to offer a level of abstraction in accessing these models through another subclass of an abstract BaseAccessor class. BaseAccessor offers some basic query and storing methods for its subclasses, these BaseAccessor subclasses manage the database table for that specific BaseModel subclass.

All the subclasses of BaseModel hold a variable field id type of UUID, but for DataEntry it is automatically generated when the object gets stored in the database. DataType has been intentionally named in an abstract way, since it can be many different things. For example, a DataType object can be blood glucose levels, weight or carbohydrates consumed and so on, it's supposed to represent different types of inputs and each type has its own unique id to represent that DataType. Alongside its id, DataType holds variables like name, units and settings. Name is of type String and describes what the dataType is. Units is a Collection of Unit objects since a dataType's entry values could be displayed in multiple different units. Settings field holds an object of type Settings, settings are supposed to represent unique settings for that specific dataType, like maximum or minimum inputs for example.

Unit represents the unit types that can exist for dataTypes. It holds a String variable name to describe itself, and the dataType it belongs to. DataType and Unit have a many to one relationship, but this could be refactored in the future to support many to many relationships, if more dataTypes would be sharing the same units. Similar to DataType, the id of Unit is used to identify which type of unit is.

Settings is specific to a dataType, it holds variables such as unit for which one the user prefers to use, dataType which it belongs to, limits for high and low input values, default values and increment that could also be edited by the user to make input feel more personal. Settings can be defined for each dataType. For now, DataType, Unit and Settings get generated at the start of the application with some default values, but ultimately if networking were applied to the software, these would be defined in the cloud and synchronized to the device. This would allow adding different types of DataTypes, Units and Settings a lot easier, since cloud would provide them and a software update to the application wouldn't even be necessary.

DataEntry is a sum of DataType, Unit and user inputted value. For example, DataEntry gets created and stored in to the database whenever user inputs new blood glucose levels. It fills in the following fields; timeStamp, dataType, value, unit and groupId. Timestamp representing the time of input, dataType representing what type of dataType this input made for, unit representing what type of unit was used to make this input. Value being the actual value of the input that was made, and groupId being an identifier that gets applied to all the entries that get made at the same moment.

Another model that could be added in the future is one for reminders. It would hold variables for time, more specifically hour and minute integers to represent the time of alarm. Also, which days of the week the alarm should ring, as an array of integers between 1 and 7 each number representing a day of the week, as well as a string note representing what the reminder is for.

5.4 OrmLite

This is where using OrmLite comes in, using annotations to easily create database tables out of java classes. Looking at listing 2, annotating a class as DatabaseTable with a name makes creating database models straight forward and easy to understand. These classes also require an empty constructor that OrmLite uses to reflect class variables into table columns. Columns in the table are achieved with simple DatabaseField annotation before the variable declaration. (34)

```
@DatabaseTable(tableName = TABLE_NAME)
public class Unit extends BaseModel {
      public static final String TABLE_NAME = "unit";

      public Unit() {}

      @DatabaseField(columnName = "id", id = true)
      private UUID id;
      ...
}
```

Listing 2. Annotating a database table with OrmLite.

OrmLite also helps a lot in defining previously mentioned Accessor classes. When looking at code in listing 3, this class defines a Dao or Data Access Object for our Unit class. The constructor receives a ConnectionSource from our DiabetesAppDbHelper class that is a subclass of OrmLiteSqliteOpenHelper, a class provided by OrmLite to help initialize the database and handle database version upgrades. Dao provides many convenience methods like createOrUpdate(), queryForAll() and queryForId() for example. (34)

```
public class UnitAccessor extends BaseAccessor {
      private final Dao<Unit, UUID> unitDao;

      public UnitAccessor(ConnectionSource connection) {
            unitDao = DaoManager.createDao(connection, Unit.class)
      }
      ...
}
```

Listing 3. Accessor class with Data Access Object.

Now that it has been covered how accessor classes are initialized and how models are annotated as database tables with columns, we can look at our OrmLiteSqliteOpenHelper subclass DiabetesDbAppHelper. This class overrides two methods from its parent class, onCreate() and onUpgrade(), a super constructor is also required with database name and version set as parameters alongside context. A few more parameters are included in the constructor for a cursor factory and an input stream to the optional configuration file. In onCreate() we create all the tables in our database by calling a method seen in listing 4.

```
private void createV1Tables() throws SQLException {
     TableUtils.createTable(connectionSource, DataType.class);
     TableUtils.createTable(connectionSource, DataEntry.class);
     TableUtils.createTable(connectionSource, Unit.class);
     TableUtils.createTable(connectionSource, Settings.class);
}
```

Listing 4. Method for creating database tables.

TableUtils class provided by OrmLite allows us to create tables of our model classes effortlessly and without writing any SQL. Worth noting is that creating tables requires a try catch block for an SQLException, and after creating the tables, connection to the source should be released as well.

onUpgrade() method's purpose is exactly what the name entails, what should happen when your database version increments. For example, if you desire to add more tables in new version releases or make changes to existing tables by adding or removing columns. When database upgrades happen, database version is traditionally incremented by one. When onUpgrade() gets called, it has four parameters, the SQLiteDatabase that is getting upgraded, ConnectionSource that you can use to make changes to the database with, old version integer and the new version integer of the database. When making changes to the database for newer application versions, while you can use tools provided by OrmLite to create and alter tablets and many other things, it is traditionally more reliable to write raw SQL commands to precisely execute what needs to be done in each version upgrade.

5.5 Application

DiabetesWearApplication class is a subclass of Application, it is the singleton to organize application wide tasks since it is accessible from nearly anywhere within our application. Singleton is a software design pattern that restricts the instantiation of a class to one object, it suits the purpose of coordinating the actions of one object across the whole system. In overridden method onCreate() we populate our database with the preset of DataTypes, Units and Settings that get generated from ModelDataGenerator class. The application is also a great place provide our accessor classes from, as seen in listing 5 we make sure that the accessors only get created one, resulting in only one instance ever being created. This is important but can often be an oversight, we would not want

two instances of the same accessor making changes to the same table at the same time, possibly resulting in corrupted data or memory leaks.

```
public DataEntryAccessor getDataEntryAccessor() {
      if(dataEntryAccessor == null) {
            try {
                  dataEntryAccessor = new DataEntryAccessor(getDbHelper());
            } catch (SQLException e) {
                  throw new RuntimeException("Failed to initialize accessor",e);
            }
      } else {
            return dataEntryAccessor;
      }
}
```

Listing 5. Get method that also initializes dataEntryAccessor.

Along with accessor get methods, application also has a get method for the DiabetesAppDbHelper, that we use to initialize our accessor classes with.

As seen in image 14, the application holds and initializes our accessor classes that handle querying and changes to the database. It also provides our activities and fragments these accessor instances through its get methods, then our activities can use that same instances to directly use the accessor objects.

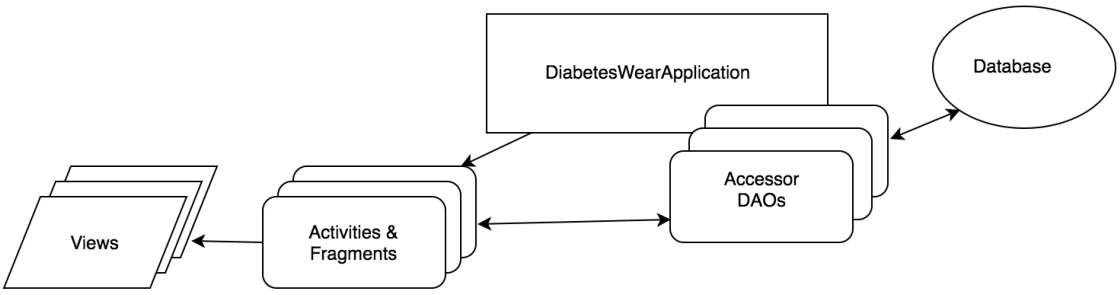

Image 14. Diagram displaying the application logic.

5.6 Views

Regarding views of this application, I decided to use only one activity coupled with multiple different fragments. Activities are classes in Android applications and they are an important part of the application's lifecycle, they are also generally the ones that handle management of the view and its contents (35). MainActivity is responsible for displaying

the navigation menu up top and the possible action menu down at the bottom, based on what fragment is displayed in the MainActivity's fragment container, we show, hide or display a different action menu at the bottom of the screen. Each item that is included in the navigation menu has its own fragment, upon clicking the item, that item's fragment is loaded into the container. All these fragments are a subclass of an abstract class called NavigationFragment that is a subclass of Fragment. Fragments are similar to activities but more light weight, they represent a behaviour or a portion of user interface in a FragmentActivity, which is an activity that can contain Fragments (36). NavigationFragment has some methods that are convenient for all the different fragments, like showing or hiding the action drawer menu, or showing a confirmation notification that changes or input have been saved for example.

In views like the input view, where logic that manages other things than the view itself need to be used, extracting that logic to another class is important. In this case, it's called InputData. InputData class holds the BigDecimal value that gets set and changed by user interactions, and the fragment just displays the value. It handles the logic concerning subtracting from the value and adding to the value, making sure that users can not input negative values for example. InputData also handles storing the DataEntry once the user taps save in the action drawer menu. First InputData builds the entry out of the all the attributes it holds, such as datatype, unit, timestamp and value. Then it proceeds to start an asynchronous background task to store this entry with dataEntryAccessor class that we pass to it in the constructor. Once the entry has been successfully stored, a save notification is triggered indicating to the user that the entry has been now saved. It's important to note that database tasks should not be done in the main UI thread, but always be ran in the background thread.

Settings view and diary view are both similar fragments due to them both having recycler views with items, but they possess different adapters and view holders for their items. For settings view the adapter takes a list of settings objects in the constructor and displays them in a simple item view with just an icon specific to the datatype and its name. For the diary view items are a little more complex since they also display the time of input in the item and the inputted value with its unit next to the datatype specific icon. For now, these views support only viewing information, but ultimately these items open a detailed view of that item and allowing the editing of it, this should be implemented in later iterations of the application.

The view for setting reminders, while it does not persist any data at this moment it can be designed. The way this would work is very similar to the input view, a separate object that handles logic and background tasks for storing all our reminders objects into our database, outside the fragment.

Watch face is something every Wear OS application is encouraged to provide. While Apple doesn't allow third party watch faces for WatchOS, for Wear OS it is considered essential due to the always on screens on Wear OS devices. I decided to create my own watch face for this project but wanted to focus on just a digital watch face, providing an analogue watch face in the future is a possibility. The reason I chose digital over analogue was that I prefer it myself, they are easier to implement as a watch face, also adding complications to the watch face is easier and makes the screen have a lot cleaner look.

Adding a watch face to your project is very straightforward. Like with many other components Google has provided good documentation and tutorials for creating your own watch faces. When creating a wearable project in Android Studio you can choose to add digital or analogue watch face, this will add a simple looking watch face service to your project. It will also automatically add the correct permission requests to your project's android manifest file such as wake lock, ability to receive complication data and the service itself. You can also choose to create your own watch face class that extends CanvasWatchFaceService, but you have to override few methods and also add your own Engine class and EngineHandler within that service. Using the generated code from Android Studio creating watch face classes for you is far more efficient, and then editing them to fulfil your needs. (37)

In my DigitalWatchFace class nothing is specifically out of the ordinary. In the inner class Engine that is a subclass of CanvasWatchFaceService.Engine, I set a custom background image from resources to a Bitmap instance variable in the overridden onCreate() method. Other things that also get initialized in onCreate() are fonts that I use for displaying time and latest blood glucose measurement value and unit, along with Paint instance variables for unit, value and time, that are responsible for drawing the texts in the correct fonts, colors and sizes. Overridden onSurfaceChanged() method allows us to save the dimensions of the surface we're going to be drawing on into instance variables.

In Engine's onDraw() method is where the initialized background bitmap actually gets drawn along with the paint instance variables for all the different texts. Overridden onDraw() method has two parameters, Canvas and Rect, Canvas will allows us to call its draw methods, and Rect will provide the bounds. As seen in listing 5, we're passing the initialized paint objects to the canvas' drawText() and drawRect() methods along with dimensions and content. The latestDataEntry variable gets retrieved through application's dataEntryAccessor everytime this watch face gets created. Since input can only be done manually for now, we don't need to be concerned about checking for new values, but this could be done with an observer pattern dependency such as EventBus to notify us whenever a new blood glucose value gets stored into our database.

```
public void onDraw() {
      ...
      canvas.drawText(time, timeXOffSet, timeYOffSet, timePaint);
      canvas.drawRect(0L, (screenHeight/2) + 2L, screenWidth, screenHeight,
      vaueStatePaint);
      if(latestDataEntry != null) {
            canvas.drawText(latestDataEntry.getValue().toPlainString(), val-
            ueXOffSet, valueYOffSet, valuePaint);
            canvas.drawText(latestDataEntry.getUnit().getName(), unitXOffSet,
            unitYOffset, unitPaint);
      }
}
```

Listing 6. Drawing on canvas in onDraw() method.

Note that the x and y offset variables for time, value and unit were calculated in another overridden method onApplyWindowInsets(). Another important thing is to make sure that you don't do any heavy tasks in the onDraw() method, since it runs on UI thread. There are other methods as well that the developer can take advantage of, one example being onTapCommand(). This is where you can add screen interactions, in my case I added opening the latest entry's information when lower half of the screen, where the latest blood glucose value is displayed, is tapped.

6 Future of smart healthcare

6.1 Testing feedback

I tested the prototype of an optimal diabetes care application that I had developed for Wear OS to type 1 diabetics fair on September 29th, 2018. I also wanted to provoke discussion and gather more ideas about what the optimal wearable application for diabetics can and should be. The fair took place in Helsinki, Hietaniemenkatu 14, and approximately 200 people attended. It consisted of multiple presentations that were all themed around making diabetes a more manageable illness all the way from having a so-called hypo alert dog, that could be classified as a service dog to help you realize your low blood glucose levels, to innovative wearable software and sensor technologies. I was also able to have small presentation about my thesis where I urged people to come and try out my application, give me feedback and discuss the future of wearables in not just diabetes care but also in general improvement of health. In this chapter I will be discussing the feedback and improvements for my application along with future expectancies for the industry.

Most common questions I was asked when presenting my application to people at the fair was; “so, what sensors does this application work with?” to my embarrassment I had to explain that currently it only relies on manual input. That even further convinced me of the essentiality of automated data input. However automated inputs from a sensor like Abbott Libre for example require contracts between companies and multiple regulations for using each other’s technology. Dexcom’s free network API implementation while a good idea wouldn’t make a lot of sense in a standalone Wear OS application, since providing CGM data inconsistently isn’t ideal. Best situation would be to have a direct connection between the sensor and the smart watch.

The overall reception for my small application was good, and people were excited to see where this goes next. People especially liked the way the watch face was clear and easy to understand, while it indicated whether their last value was in range of their personal blood glucose settings, they also said that they would definitely like to see which way their blood glucose was trending towards. Showing a trending indicator is possible but difficult with manual input, since the number of data sets is relatively small. With a sensor’s automated inputs this would be a lot easier since an algorithm could be comparing

the latest values and therefore predict if the blood glucose level is rising or not. In regard to the input view people felt that it was well implemented functionality and while it was definitely necessary, since if all else fails in terms of automated sources there still needs to be a way to input data, ideally the data would come from a sensor.

Having reminders for medication or taking measurements was considered to be very useful by many of the testers. Especially the fact that it was the watch that had the reminders was what impressed people, almost like a daily calendar for diabetes care on your wrist. Reminders could also fire different types of alarms based on what the its set action was. For example, vibration or different alarm sounds. Automated reminders were also a subject of interest, having a reminder to take measurements if none have been taken for a while, or notifying user of lowering blood sugar levels. In terms of settings view people felt that it would indeed be good to have a possibility to edit your ranges of acceptable glucose levels or maximum and minimum values for different types of datatype inputs.

6.2 Wearables in healthcare

While receiving good feedback about my application, I also took on the opportunity of discussing more about smartwatches and diabetes care with people in the fair. To some with active lifestyle a watch with sensors like GPS, heart rate monitor, altitude meter and more is an excellent choice and worth the purchase if it is also be able to aid them in managing their diabetes. Others insisted that watch should be stylish and correct size for them to wear so that they would want to use it daily, but the idea of a watch with a small form factor that could replace the monitors that some diabetics carry everywhere was considered to be an excellent solution.

With Apple releasing the new Apple Watch Series 4, one of the new features was FDA-approved EKG, or electrocardiogram, to monitor heart rhythms (38). Features like this while controversial, show that especially Apple is trying to push for Apple Watch becoming a digital health device and we can hope that Google wants to follow that direction with its own upcoming device releases. This feature can potentially help reveal serious medical conditions but could also lead to false positives and over diagnosis (38). Another interesting feature that was added is a method of recognizing when a wearer of the watch falls down, the watch will alert them with the option to place an emergency call through

the watch, it will even automatically initiate the call if the user remains unresponsive for one minute (38). This could be especially good for elderly people or people that work under circumstances where falls are common.

Something to expect from the smartwatch manufacturers like Apple, Google and even Samsung is even more new ways to monitor your health. One feature that has been rumored is of course the ability to measure blood glucose levels non-invasively (39). This is a feature that would change the industry of diabetes care, being able to measure your blood glucose levels with your smartwatch while also having it do everything else like activity monitoring or insulin pump management.

Among the other discussions I had with diabetics, one of the most interesting topics was AI, or artificial intelligence, and machine learning in terms of diabetes self-care. Having an application that would adapt to the lifestyle of each person and make the intelligent care it provides personal to the patient. For example, diabetics with different sleep schedules due to working night shifts, active lifestyle or other perhaps limitations from disabilities could be addressed better with adaptive intelligent care. Machine learning could also be used in identifying carbohydrate amounts in a diabetic's meals by having them take pictures of their meals, this kind of image recognizing machine learning already exists to an extent but could also be extended to medical use cases. Using AI would also result in lower cost of medical services, instead of a patient having to go to the doctor to receive a diagnosis and a care plan, the patient and the AI could instead perform those (40, p. 4).

Machine learning and artificial intelligence in diabetes care open up a lot of opportunities. However, there's possibilities for significant health risks, regulatory issues and ethical questions of whether an artificial should be responsible for taking care of a person's chronic illness, but under the right circumstances I believe this could be a huge advancement in diabetes self-care.

Diabetes could be managed with digital health in the future but there still some things that need to be addressed to make that completely possible. Usability for devices or software that is used to manage diabetes needs to get better, and completely satisfy the patients. There also needs to be a way to make digital health solutions desirable to health care professionals, so that they provide considerable benefit compared to traditional methods. Not forgetting about the market, there needs to be economic benefit to satisfy

the people who are willing to pay for it. Finally comes security so that health safety can be preserved, and regulators of products will also accept these changes. With all the controversy of recent data leaks, information security is even more important when it comes to patient specific medical data. (7, p. 912)

7 Summary

The goal of this thesis was to find out what makes an optimal Wear OS application for diabetes care and implement a prototype application. It was also important to justify the reasons behind developing such application not just in terms of medical benefits but also in terms of economic benefits to all parties involved. To find out what the diabetics consider as essential features and what they desire from a smartwatch application an inquiry was done reach as many diabetics as possible to base the features on. Research on existing technologies and applications was also relevant in justifying the features and designs of the application.

The result was a standalone Android wear application for diabetes self-care that matches the minimum viable product requirements. While not ready for commercial use the application possesses primitive versions of the key features described by users including data input, latest value and its state and customizability in an intuitive user interface. There would still definitely be room to develop this application further in terms of achieving automated inputs from CGM sensors from manufacturers like Dexcom. However, this requires more resources since support for full networking would have to be implemented on a device that has completely unreliable networking. Providing cloud support for the application is another key improvement that could be done in the future. It would add another level of complexity into the development but with great benefit to the end user by allowing backing up of their medical data and even providing that data to their personal doctors or health care professionals.

More importantly this thesis provides insight into what goes into researching, designing and developing a wearable diabetes application. For example, this study analyses what dependencies to use, what kind of architecture the application should have, and what the real challenges in terms of development and designing for wearable platforms such as Wear OS are. Also, how to attain constructive feedback for your project and analyze it is discussed. To create something great, huge amounts of research must be carried out to understand the illness that is diabetes on top of complicated designs, benefits and limitations of wearable devices.

This thesis also discusses and hopefully encourages more discussion about wearables being and becoming even more related to medical devices. One aim of this study was to analyse useful health data to the users by allowing them to make improvements on their

lifestyle. To help the users notice oddities that they otherwise would not have, should be considered to be the future of wearable technology. The possibilities of making the patients, doctors and health care professionals' lives easier with wearable technology and smart healthcare in general are endless and should be taken into serious consideration if not already into action.

References

1 Fonseca, Vivian. 2008. Diabetes: Improving Patient Care. E-Book. Oxford University Press.

2 IDF Diabetes Atlas. 2017. E-book / internet source. International Diabetes Federation. <http://www.diabetesatlas.org/>.

3 World diabetes prevalence. Internet source. Diabetes Digital Media ltd. <https://www.diabetes.co.uk/diabetes-prevalence.html>. Read 23.8.2018

4 Diabetes Care Devices Market In Brazil, Russia, India, China (BRIC) Worth $1.1 Billion In 2015. 2011. Internet source. PR Newswire Association LLC. <https://search-proquest-com.ezproxy.metropolia.fi/docview/861372393>. Read 23.8.2018.

5 Young-Soon, Hung & Yongsuk, Kim & Chang Hee, Lee. 2014. Effectiveness of the smart care service for diabetes management. Internet source. Healthcare informatics research. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4231179/>. Read 23.8.2018.

6 Kitsiou, Spyros & Paré, Guy & Jaana, Mirou & Gerber, Ben. 2017. Effectiveness of mHealth interventions for patients with diabetes: An overview of systematic reviews. Internet source. <http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0173160>. Read 23.8.2018.

7 Kerr, David & Gabbay, Robert A. & Klonoff David C. 2018. Finding Real Value From Digital Diabetes Health: Is Digital Health Dead or in Need of Resuscitation? Journal of Diabetes Science and Technology. Volume: 12, issue: 5, pages: 911-913. <http://journals.sagepub.com/doi/full/10.1177/1932296818771200>. Read 4.10.2018.

8 Lyons, Kent. 2016. Smartwatch Innovation: Exploring a Watch-First Model. Internet source. <https://ieeexplore-ieee-org.ezproxy.metropolia.fi/document/7389267/?arnumber=7389267>. Read 24.8.108.

9 Kuo-Lun, Hsiao & Chia-Chen, Chen. 2018. What drives smartwatch purchase intention? Perspectives from hardware, software, design, and value. Internet source. <https://www.sciencedirect.com/science/article/pii/S0736585317305804?via%3Dihub>. Read 24.8.2018.

10 Goldstein, Phil. 2014. Google unveils Android Wear software for smart watches, other wearables. Internet source. Fierce Wireless. <https://search.proquest.com/docview/1508607213>. Read 28.9.2018.

11 Swider, Matt &Rogerson, James. 2018. Wear OS: Google's new name for Android Wear explained. Internet source. Tech Radar. <https://www.techradar.com/news/wear-os>. Read 1.10.2018.

12 Android Rank Google Fit. 2018. Internet source. Android Rank. <https://www.androidrank.org/application/google_fit_fitness_tracking/com.google.android.apps.fitness?hl=en>. Read 19.8.2018.

13 Goole Play Store Diabetes M. Internet source. Google Play Store. <https://play.google.com/store/apps/details?id=com.mydiabetes>. Read 19.8.2018.

14 Why Diabetes:M had to switch to a subscription based service. Internet source. Sirma Medical Systems. <https://www.diabetes-m.com/blog/news/why-diabetesm-had-to-switch-to-a-subscription-based-service/>. Read 24.8.2018.

15 Diabetes M. Internet source. Sirma Medical Systems. <https://www.diabetes-m.com/>. Read 23.8.2018.

16 Cohn, Chuck. 2.7.2018. Should You Consider A Freemium Model For Your Business? Forbes. Internet source. <https://www.forbes.com/sites/chuckcohn/2015/07/02/should-you-consider-a-freemium-model-for-your-business-pros-cons/#45c9c23d24a7>. Read 10.10.2018.

17 Define layouts on Wear. Internet Source. Android Developers. <https://developer.android.com/training/wearables/ui/layouts>. Read 24.8.2018.

18 Diabetes:M User's Guide Android Wear. Internet Source. Sirma Medical Systems. <https://sites.google.com/view/diabetes-m-userguide/android-wear?authuser=0>. Read 24.8.2018.

19 Wear OS by Google Style – Color. Internet source. Google. <https://designguidelines.withgoogle.com/wearos/style/color.html#color-darker-color-palettes>. Read 24.8.2018.

20 Wear OS by Google Patterns – Interactive watch faces. Internet source. Google. <https://designguidelines.withgoogle.com/wearos/patterns/interactive-watch-faces.html#interactive-watch-faces-creative-vision>. Read 24.8.2018.

21 Dexcom continuous glucose monitoring. Internet source. Dexcom, Inc. <https://www.dexcom.com/continuous-glucose-monitoring>. Read 2.10.2018.

22 Android Rank Dexcom G5 Mobile. 2018. Internet source. Android Rank. <https://www.androidrank.org/application/dexcom_g5_mobile/com.dexcom.cgm?hl=en> Read 1.10.2018.

23 Dexcom G5 Mobile System User Guide. 2017. Internet source. Dexcom, Inc. <https://s3-us-west-2.amazonaws.com/dexcompdf/OUS+Specific+PDFs/Regulatory+UG+mmol+English+080117/LBL013137+Rev+008+AW%2C+G5+Mobile+UG+OUS+Eng+mmol+web.pdf>. Read 1.10.2018.

24 FreeStyle Libre. 2018. Internet source. Abbott Oy. <https://freestylediabetes.fi/tuotteemme/freestyle-libre> Read 1.10.2018.

25 Android Rank FreeStyle Libre. 2018. Internet source. Android Rank. <https://www.androidrank.org/application/freestyle_librelink_de/com.freestylelibre.app.de?hl=en> Read 1.10.2018.

26 FreeStyle Libre. 2018. Internet source. Abbott Oy. <https://www.freestyle.abbott/fi/fi/libre/index.html>. Read 25.8.2018.

27 Wear OS by Google – Creative vision. Internet source. Google. <https://designguidelines.withgoogle.com/wearos/wear-os-by-google/creative-vision.html#>. Read 30.8.2018.

28 Wear OS by Google Style – Typography. Internet source. Google. <https://designguidelines.withgoogle.com/wearos/style/typography.html>. Read 31.8.2018.

29 Wear OS by Google Components – Navigation drawer. Internet source. Google. <https://designguidelines.withgoogle.com/wearos/components/navigation-drawer.html#>. Read 16.9.2018.

30 Java naming conventions. Internet source. Oracle. <https://www.oracle.com/technetwork/java/javase/documentation/codeconventions-135099.html>. Read 1.10.2018.

31 Ante, Spencer & Troianovski, Anton. The iPhone 5 Supports LTE – What Is That? 2012. Wall Street Journal. <https://search-proquest-com.ezproxy.metropolia.fi/docview/1039194597/?pq-origsite=primo>. Read 14.10.2018.

32 Android Developers – Databases. Internet source. Google. <https://developer.android.com/guide/topics/data/data-storage> Read 20.9.2018.

33 Android Developers – Standalone Wear apps. Internet source. Google. <https://developer.android.com/training/wearables/apps/standalone-apps>. Read 28.9.2018

34 OrmLite – Lightweight Java ORM Supports Android and SQLite. Internet source. <http://ormlite.com/sqlite_java_android_orm.shtml>. Read 20.9.2018.

35 Android Developers – Activity. Internet source. Google. <https://developer.android.com/reference/android/app/Activity>. Read. 14.10.2018.

36 Android Developers – Fragments. Internet source. Google. <https://developer.android.com/guide/components/fragments>. Read. 14.10.2018.

37 Android Developers – Build a watch face service. Internet source. Google. <https://developer.android.com/training/wearables/watch-faces/service>. Read 2.10.2018.

38 The new Apple Watch can conduct an EKG from your wrist. 2018. American Health Line. First Look. Internet source. <https://search.proquest.com/docview/2103334896?pq-origsite=primo>. Read 4.10.2018.

39 Bhagat, Hitesh Raj. 2017. What's next for Apple: Blood sugar monitoring or a tie-up with Adidas & Nike for smart clothing? [Panache]. New Delhi: The Economic Times. <https://search.proquest.com/docview/1917489846/?pq-origsite=primo>. Read 4.10.2018.

40 Aittoskoski, Mortti. 7.3.2018. Tekoäly apuna diabeteksen itsehoidossa. Metropolia UAS. Pages: 3-4. <http://www.theseus.fi/handle/10024/141873>. Read 4.10.2018

User research form

*Pakollinen

Sukupuoli *

- ○ Nainen
- ○ Mies
- ○ Muu
- ○ En halua vastata

Ikä *

- ○ 0-15
- ○ 16 - 25
- ○ 26 - 35
- ○ 36 - 45
- ○ 46 - 55
- ○ 56 - 65
- ○ Yli 65

Olen *

- ○ Diabetesta sairastava
- ○ Diabeteksen parissa työskentelevä
- ○ Diabeteksen parissa työskentelevä diabeetikko
- ○ Muuten vain kiinnostunut aiheesta

SEURAAVA

Diabeteksen älykäs omahoito

Perustietoja diabeteksestasi

Diabeteksen tyyppi

- ○ Tyyppi I
- ○ Tyyppi II
- ○ Muu: _____

Kauanko olet sairastanut diabetesta?

- ○ alle 1 vuoden
- ○ 1 - 5 vuotta
- ○ yli 5 vuotta

Mitkä ovat sinulle suurimmat haasteet diabeteksen kanssa elämisessä?

Oma vastauksesi

TAKAISIN SEURAAVA

Diabeteksen älykäs omahoito

*Pakollinen

Älykäs omahoito

Käytätkö älypuhelinta? Jos käytät, niin mikä käyttöjärjestelmä siinä on? *

- ○ Android (Samsung, Huawei, OnePlus, Sony, LG yms.)
- ○ iOS (Apple iPhone)
- ○ Muu
- ○ En osaa sanoa
- ○ En käytä älypuhelinta

Käytätkö älykelloa? Jos käytät, niin mitä? *

- ○ Android wear
- ○ Apple watch
- ○ Muu älykello
- ○ En käytä älykelloa

TAKAISIN SEURAAVA

Diabeteksen älykäs omahoito

Omahoitotyökalut

Oletko tutustunut diabeteksen omahoidossa käytettäviin älykkäisiin ratkaisuihin, kuten älypuhelin- ja älykellosovelluksiin?

- ○ Kyllä
- ○ En

Millaisia omahoitotyökaluja käytät?

- ☐ Insuliinipumppu
- ☐ Bluetooth-verensokerimittari
- ☐ Puhelinsovellus diabeteksen omahoitoon
- ☐ Älykelloa tai siinä olevia sovelluksia omahoitoon
- ☐ Muu: _____

Jos et käytä älykelloa, mutta huomaisit siitä olevan apua päivittäiseen elämääsi diabeetikkona, käyttäisitkö sitä?

- ○ Kyllä
- ○ En
- ○ Ehkä

TAKAISIN SEURAAVA

Diabeteksen älykäs omahoito

Älykello

Älykello on rannekello joka tuo älypuhelimen ominaisuudet ranteeseesi. Se pystyy lisäksi mittaamaan erilaisia arvoja tarkemmin kuin älypuhelimet, esimerkiksi pulssi, askelmäärä, energiankulutus yms.

Onko sinulla kokemuksia diabeteksen omahoitoon suunnatuista älykellosovelluksista? Millaisia? Mitä sovelluksia olet käyttänyt?

Oma vastauksesi

Mitkä olisivat sinulle tärkeitä tai hyödyllisiä ominaisuuksia diabeteksen omahoitoon suunnatussa älykellosovelluksessa?

Oma vastauksesi

Haluaisitko kellon muistuttavan sinua jostakin? Mistä?

Oma vastauksesi

Mitä asioita haluaisit kellon näyttävän sinulle nopeasti tai yhdellä vilkaisulla?

Oma vastauksesi

Mitä haluaisit kellon tai sovelluksen seuraavan?

Oma vastauksesi

TAKAISIN SEURAAVA

Diabeteksen älykäs omahoito

Yhteystiedot

Kiitos kyselyyn vastaamisesta!

Mikäli haluat jatkossa olla mukana kehittämässä tätä tai muita diabeteksen älykkäitä omahoitotyökaluja, voit jättää yhteystietosi.

Yhteystietoja ei julkaista tai jaeta. Yhteystietojen jättäminen on vapaaehtoista.

Nimi

Oma vastauksesi

Sähköpostiosoite

Oma vastauksesi

Puhelinnumero

Oma vastauksesi

TAKAISIN LÄHETÄ