CHALMERS

Sketchagram
- Kommunikationsapplikation för Android Wear

Kandidatarbete inom Data- och informationsteknik

Fredrik Holmdahl
Magnus Huttu
Alexander Härenstam
Alexander Jaballah
Danny Lam
Olliver Mattsson

Institutionen för Data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2015

Sketchagram
Communication application for Android Wear

FREDRIK HOLMDAHL
MAGNUS HUTTU
ALEXANDER HÄRENSTAM
ALEXANDER JABALLAH
DANNY LAM
OLLIVER MATTSSON

Examiner: JAN SKANSHOLM

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover: Logo for Sketchagram.
Department of Computer Science and Engineering
Göteborg, Sweden June 2015

Abstract
This bachelor thesis aims to explore how human digital communication can be simplified, given recent technological advancements. The primary objective is studying how communication can be optimized by developing a messaging application for smartphones and smartwatches.

Design guidelines from well-known sources have been followed to create a graphical user interface that can be evaluated. In order to learn more about the interaction with smartwatches user tests have been carried out. Standards have been followed during the implementation to produce a familiar user interface and thereby the best possible product.

A messaging application has in light of this been created for smartwatches with an extended application on smartphones. The functionality includes sending drawings and emoticons. Conclusions drawn from the user testing indicate that the applications provide an efficient way of communicating, even though the application for smartwatches has shown to be the greater choice.

Sammandrag
Kandidatarbetet ämnar att undersöka hur mänsklig digital kommunikation kan förenklas, givet nyligen gjorda tekniska framsteg. Det primära målet är att studera hur kommunikationen kan optimeras genom att utveckla en meddelandeapplikation för smarta telefoner och smarta klockor.

Riktlinjer för design från välkända källor har använts för att skapa ett grafiskt gränssnitt vilket kan utvärderas. För att dra lärdomar om interaktionen med smarta klockor har användartester genomförts. Standarder har följts under implementationen i syfte att producera ett bekant gränssnitt och därmed den bästa möjliga slutprodukten.

En meddelandeapplikation har i detta avseende skapats för smarta klockor med en utökad applikation för smarta telefoner. Funktionaliteten inkluderar att skicka målningar och uttryckssymboler. Slutsatserna från användartesterna indikerar att applikationen erbjuder ett effektivt sätt att kommunicera, även om applikationen för klockan har visat sig vara det föredragna alternativet.

Förord

Innehållet i rapporten beskriver skapandet av kommunikationsapplikationen Sketchagram, utfört av sex stycken tredjeårsstudenter från institutionerna Data- och informationsteknik vid Chalmers Tekniska Högskola, vårterminen 2015.

Arbetet har utförts under 16 veckor och projektgruppen önskar att tacka Alex Gerdes för hans engagemang och agerande som handledare.

Ordlista
Android: Open source operativsystem tillhandahållet av Google.

Android Wear: Open source operativsystem för smarta klockor vilket utökar operativsystemet Android.

JSON: Ett standardiserat format för att temporärt hantera logiska kodstrukturer, i textformat.

Klasser: Beståndsdel av ett program i Java, med specifikt definierad funktionalitet, för ett avgränsat område.

Openfire: Open source mjukvara för en XMPP-server.

Ramverk: Återanvändbar mjukvara vars syfte är att stödja utvecklingen av nya tillämpningar, inom respektive tillämpningsområde av ramverket.

Raspberry Pi: Liten dator som bygger på en ARM-processor.

Scrum: Cyklisk arbetsprocess för mjukvaruutveckling där endast en undergrupp av de totala arbetsuppgifterna ingår i varje cykel och avslutas med en utvärdering.

Structured Query Language, SQL: Ett språk använt till att skapa relationsdatabaser.

UML: Ett sätt att rita upp en modell enligt ett standardiserat tillvägagångssätt.

XMPP: Nätverksprotokoll vilket används för chatt-applikationer.

Innehåll

1. Inledning

Smarta klockor har gjort sitt intåg på teknikmarknaden i form av Android Wear och Apple Watch, därmed behövs nya smidiga lösningar för att dessa på bästa möjliga sätt ska kunna nyttjas [1, 2]. En av de största utmaningarna ligger i hur nya enheter utvecklas och anpassas för att interagera med användare.

Nya tekniker utvecklas för att förenkla moment för användarna. Smarta telefoner har revolutionerat samhället sedan en tid tillbaka och många människor äger en sådan nuförtiden. Smarta telefoner används dagligen och tack vare en stor variation av applikationer, tillgängliga för innehavaren, blir enheten väldigt personlig [3].

Idag torde därmed finnas en stor potential för de applikationer vilka stödjer interaktionen med smarta klockor. Klockan kan kommunicera med telefonen och nyttja dess funktionalitet, vilket öppnar upp för skapandet av nya innovativa applikationer.

1.1. Bakgrund

Kommunikationstekniken har utvecklats med en enorm fart, från den klassiska hemtelefonen till en bärbar smart telefon vars storlek passar i byxfickan. Telefonen används dagligen för att skicka textmeddelanden och för att ta emot samtal, men eftersom användningen är frekvent ställs krav på smidighet och snabbhet. Problemet med bärbara datorer och telefoner är att de måste plockas fram varje gång det är dags att inleda en dialog med en annan person, vilket är en tidskrävande process.

Modern teknik går i stora drag ut på att förenkla för användaren och om personen i fråga slipper ta upp sin telefon för att skicka meddelanden, kan detta leda till en stor effektivisering av momentet. Här kommer smarta klockor in i bilden, en vardaglig accessoar vilken både är lättillgänglig och konsumenter redan börjat nyttja [4].

1.2. Syfte

Avsikten är att skapa en applikation vilken kan skicka meddelanden med hjälp av smarta telefoner och klockor. Applikationen skall vara anpassad till förutom att kunna utföra uppgifter på den smarta telefonen, även kunna komponera meddelanden på klockans skärm. Intentionen är därför att dessutom undersöka vilket tillvägagångssätt som lämpar sig bäst för dessa uppgifter samt även vilken design som främjar användarvänligheten mest.

Ändamålet är därmed främst att dra lärdom om hur gränssnitt bör utformas för att möjliggöra enkel och intuitiv kommunikation via smarta klockor, trots den begränsade skärmytan. Detta inkluderar att ta hänsyn till hur kommunikationen via klockor kan förenkla människans vardagliga dagsbehov. Momenten inkluderar omständigheter där personen befinner sig på resande fot, tränar, handlar och andra tillfällen där tillgängligheten till telefonen är begränsad.

1.3. Avgränsningar

Projektet avgränsas till att skicka enkla uttryckssymboler och skisser mellan två klockor, samt till att utvärdera interaktionen mellan användaren och applikationen. Kravet för att möjliggöra detta, likt beskrivet i syftet är att skapa en applikation till telefonen vilken klockan kan nyttja för funktionalitet. Anledningen till detta beror på att applikationer för Android Wear kräver att det finns en korresponderande applikation på telefonen, vilken

den använder sig av för funktionalitet [5]. En bidragande effekt av att göra en mobilapplikation är att kravet på smartklocka dessutom kan elimineras. Användningsmiljön avgränsas därför till att baseras på ett scenario där två personer kommunicerar mellan varandra med hjälp av en klocka tillsammans med en telefon baserade på operativsystemet Android. En annan avgränsning vilken gjordes med hänsyn till tidsbegränsningen var att inte implementera en egen servermjukvara utan istället nyttja en redan existerande.

1.4. Problem

Smarta klockor är en teknik vars intåg till teknikmarknaden innebär nya utmaningar för de utvecklare, eller pionjärer vilka ligger i framkant. Största problemet med klockan är skärmbegränsningen, då väsentlig information skall få plats på skärmen utan att det blir rörigt. Visuella meddelanden innebär en stor mängd information och förutom att presentera innehållet bör även avsändare, klockslag samt status för anslutningen också presenteras. Detta för att nämna några få exempel på information som kan återfinnas i dagens meddelandeapplikationer. Dessa krav behöver därmed tillgodoses på en mycket mindre yta än tidigare.

Förutom ovannämnda finns även begränsningar i det medium använt för interaktionen, vilket i detta fall motsvarar skärmen vilken återfinns på den smarta klockan. Fingergester täcker vanligtvis majoriteten av synfältet då användaren interagerar med klockan och därmed förhindras användaren från att få en bild av vad rörelsen faktiskt åstadkommer.

1.5. Vision

Grundidén för applikationen bygger på den idé Apple tidigare presenterat och deras sätt att kommunicera via smarta klockor, vilket introducerades under deras årliga utvecklarkonferens 2014 [1]. Även om inspirationen ursprungligen härstammar från Apple, är det viktigt att betona att funktionaliteten ej skall vara densamma. Sketchagram skall fungera för Android-enheter samt att fokus genomgående ligger på att göra det enklare för användaren att kommunicera via snabba enkla meddelanden.

Applikationens omfång definierades i begynnelsen till att vara begränsat till att möjliggöra målande med hjälp av fingergester, på både smarta klockor och telefoner. Vid kandidatarbetets start sammanställdes gruppens gemensamt önskade funktionalitet i form av milstolpar, se bilaga A. Milstolparna inkluderade dock funktionalitet vars användningsområde redan täcktes av gedigna applikationer då exempelvis kommunikation via textmeddelanden redan är ett utbrett användningsområde. Därmed avgränsades projektet ytterligare efter hand till att vara mer definierat och endast inkludera att skicka meddelanden innehållande antingen en målning, eller en uttryckssymbol.

1.6. Förstudier

Projektet bygger på en dagsaktuell idé och ett nytt sätt att kommunicera, vilket inneburit inlärning av nya kunskaper innan projektets mer tekniska delar kunnat påbörjas. Majoriteten av denna förstudie har genomförts på Googles egna utvecklarguide kallad Android Developers, där information finnes för relevanta kreationer vilka kan tänkas skapas med operativsystemet Android [6].

Designen på applikationen beslutades vara högt prioriterad och därför krävdes stor för-

djupning inom riktlinjer från Android Developers. Hela konceptet smarta klockor var vid projektets början ännu nytt vilket inneburit att Android Developers var till stor hjälp för designen. Där återfinns flertalet olika standarder för hur designen genomgående bör se ut och exempelvis hur väl olika färger passar ihop med varandra.

Förutom det ovannämnda krävdes kunskaper kring standarden för kommunikation vid namn XMPP vilken skulle användas för serverkommunikation [7]. Potentiella servermjukvaror undersöktes därmed för att hitta en mjukvara med lämplig funktionalitet. Kravet var då att dessa skulle implementera XMPP-protokollet. Utbudet på nätet omfattar ett flertal olika servermjukvaror vilka använder sig utav XMPP-standarden och dessa mjukvaror är skrivna i olika programmeringsspråk, men erbjuder näst intill identisk funktionalitet.

Studier kring hur kommunikation kan ske mellan smarta klockor och telefoner var även en viktig del av inlärningen, lyckligtvis finnes detta väl dokumenterat på Android Developers [8]. Kommunikationen mellan olika användare var däremot ett svårare beslut, då Googles rekommendationer ej tillfredsställde de existerande behoven för servern. Istället undersöktes andra alternativa verktyg med krav på god dokumentation och utbredd användning.

2. Teknisk Bakgrund

Utvecklingen av applikationen krävde kunskaper inom främst Android-programmering men även inom kommunikationstekniker för meddelanden, databasspråk samt design för smarta telefoner. Sedermera är avseendet med detta avsnitt att ge en inblick i dessa olika delar.

2.1. Tidigare projekt

Apple Watch demonstrerade en funktion för att kommunicera med hjälp av smarta klockor, hösten 2014 [1]. Denna funktion är endast tillämpad för produkter från Apple och kan därför ej användas fristående från företagets produkter. Applikationens idé grundar sig i detta sätt att kommunicera, men trots detta faktum finns skillnader i hur Apples produkt och hur projektets utvecklade applikation är tänkt att fungera.

2.2. Smarta telefoner

Smarta telefoner är idag ett av de verktyg vilka används dagligen och vanligen sköter delar av vardagen. Därmed har dessa tekniska hjälpmedel blivit en blandning mellan en handdator samt en ursprunglig telefon, vilka tidigare endast kunnat användas till att ringa enkla telefonsamtal. Således finnes en stor del av vad en privatperson skulle kunna tänkas behöva inom den tekniska världen inbyggt i dagens telefoner [9].

Dagens telefoner har olika funktioner och utseende, däremot baseras alla i grunden på ett operativsystem. Operativsystemet fungerar specifikt för vissa modeller och tillåter användaren att göra telefonen mer personlig genom att installera olika applikationer. Bluetooth-tekniken används för att en smartklocka med samma operativsystem som telefonen skall kunna få åtkomst till funktionalitet [10]. Således kan funktionaliteten i en mobilapplikation utökas till att även stödja interaktioner från en smartklocka.

2.3. Smartklocka

Armburna digitaliserade enheter kan med hjälp av små pekskärmar göra det möjligt att genomföra simpla kommandon vilka underlättar användarens behov till telefonen. Med den smarta klockan behöver därmed inte användaren längre ta fram telefonen ur fickan eller väskan för att kunna svara på ett samtal eller för att få se innehållet i en notifikation.

2.4. Android

Open Handset Alliance står bakom Android även om Google har huvudansvaret för utvecklingen av operativsystemet [11]. Google lägger därmed ett stort fokus på sina egna applikationer och att erbjuda ett helhetsintryck, genom att bland annat ge utvecklare tillgång till deras grafiska riktlinjer.

Applikationerna utvecklas i Java mot ett ramverk, förutom när det kommer till hur gränssnittet utvecklas. Androids grafiska gränssnitt byggs upp av Activities och Fragments. En Activity är likt ett nytt program öppnat i form av ett fönster där grafik kan placeras direkt inuti. Grafiken kan även placeras inuti en Activity i form av ett eller flera Fragment. Ett Fragment utgör sedan en grafisk funktionalitet och flertalet olika grafiska funktionaliteter kan då placeras inuti en Activity. De vyer vilka framställts under arbetets gång har således baserats på ett Fragment, placerat inuti en Activity.

Andra viktiga begrepp i operativsystemet utgörs utav bland annat SharedPreferences, en databas lokalt på den smarta telefonen, ämnad för att framförallt hantera gemensamma

data mellan olika applikationer [12]. Därtill finns ett Manifest, vilket är en fil där utvecklare tillåts definiera mer övergripande egenskaper, exempelvis vilken funktionalitet applikationen kommer att behöva tillgång till på användarens telefon.

2.4.1. XML

Android använder sig av standard-formatet XML för att definiera utseendet på grafiska statiska element och animationer. Varje vy har förutom logiken skriven i Java, en XML-fil där utseendet definieras och dessa importeras via ramverkets metoder. XML-filerna kan även referera till varandra och därmed återanvända redan definierade kodstycken. Detta leder till att separation av kodstycken möjliggörs och därmed kan komplexiteten på filerna minskas [13].

2.4.2. Android Studio

Google har framställt utvecklingsverktyget vid namn Android Studio vilket används till Android-utveckling. Verktyget har många funktioner vilka underlättar utvecklingen av applikationer. Programmet baseras på IntelliJ vilket är en IDE, verktyg för utveckling av mjukvara, använd utav Java-utvecklare. Detta i kombination med Gradle, en automatiserad byggmiljö för att inhämta bibliotek till utvecklingen, gör programmet väl anpassat till projektets behov [6].

2.4.3. Android Developers

Ett hjälpsamt verktyg vid Android-utvecklingen under projektets gång har varit Android Developers, vilket är den officiella guiden för Androidutvecklare och är skapad av Google själva [8]. Guiden innehåller specifika steg och guider för hur en utvecklingsmiljö skall se ut och fungera. Android Developers beskriver även hur kopplingen mellan olika specifika klasser i Android ska göras och hur dessa fungerar. Guiden beskriver också hur lagring hanteras i form av en SQLite databas för Android.

2.5. Databas och server

Språket SQLite härstammande från ett C-bibliotek valdes enligt riktlinjer från Android Developers. SQLite är, till skillnad från andra SQL-språk, ett databassystem vilket är inbyggt i slutprogrammet [14]. Google själva rekommenderar att SQLite används då en databas skapas för att användas tillsammans med applikationer för Android. De har även en guide på Android Developers för hur en databas skall skapas vilket förenklar utvecklingen av denna, istället för att utvecklare skall integrera något annat bibliotek.

En server är en mjukvara vilken körs på extern hårdvara vars uppgift är att agera knytpunkt för ett antal klienter. Till applikationen har XMPP-standarden valts för implementation av att skicka meddelanden då det är standard för kommunikationsapplikationer. Standarden är framtagen specifikt för att kunna skicka olika former av meddelanden mellan användare [7]. Många standarder inom XMPP definierar även hur kontakter ska hanteras samt även hur meddelanden, vilka skickas till otillgängliga kontakter, hanteras. Detta gör standarden mycket lämplig för applikationer där meddelanden skickas.

2.5.1. Bluetooth

Vid överföring av information via Bluetooth krävs att enheterna är parkopplade, vilket innebär att användaren har bekräftat sitt ägande av enheterna [10]. Parkopplade enheter

sparar noder till varandra för att veta var informationen skall skickas. När enheterna har noder vilka pekar till varandra kan de utan avbrott skicka information i form av meddelanden.

2.6. Design

Till designen användes Photoshop, från företaget Adobe, vilket är en mjukvara för framtagning av grafiska komponenter såsom bilder, ikoner och animationer [15]. Android Developers erbjuder ett bibliotek av ikoner vilka är standarder i operativsystemet och har möjligheten att importeras in till Photoshop för redigering eller användas direkt i Android Studio.

I version 5.0, Android Lollipop, introducerades en ny designstandard vid namn Material Design, med tanken att skapa ett uniformt utseende för hela Googles ekosystem [16]. Genom att ge riktlinjer för färgval och val av grafiska element, utgör dessa ett varumärke i pixelform för operativsystemet Android, utöver deras logotyp.

3. Metod

Arbetsmetodiken som tillämpats beskrivs genomlöpande. Både ur programmerarens perspektiv, men även utifrån designperspektiv och genomförandet av användartester.

3.1. Arbetssätt

SCRUM är utvecklingsmetoden vilken använts genom projektets gång för att bibehålla en god struktur och för att hålla fast vid en frekvent dialog mellan gruppmedlemmar [17]. Detta för att hålla reda på vad som gjorts, behövde göras samt vad som gruppen i stunden funderade på. Arbetsmetoden användes även för att övriga medlemmar ska upptäcka om någon i gruppen behövde hjälp.

Projektet delades in i tre olika sprintar där varje sprint innehöll olika användarscenarion vars deadline var satt till en brytpunkt för början av nästkommande sprint. Under denna tid förekom även ett flertal dagliga möten där vardera medlem redovisade vad som hade gjorts och vad som skulle göras samt om det fanns några hinder på vägen. Förutom detta ägde även ett större möte rum innan varje sprint påbörjades där föregående sprint utvärderades och nästkommande sprint planerades.

För att nå en hög effektivitet i gruppen har det skett olika arbetsfördelningar, där huvudfördelningarna var att jobba i grupper om två och två. Grupperna har tilldelats var sitt ansvarsområde, en gruppering för design, en för databas och server samt en för klientsystemet. Anledningen till detta var främst att underlätta för gruppen genom att vara två i varje team, då mycket var nytt och på detta sätt skulle det alltid finnas en gruppmedlem att diskutera med.

Under projektet har parprogrammering, efter pair-programming standarden, till stor del använts för att effektivisera kodning och minimera fel. Pair-programming är ett koncept vilket bygger på att två personer skriver kod samtidigt som de diskuterar eventuella problem. Med hjälp av metoden kan tillkommande fel vid programmering upptäckas tidigare vilket leder till färre komplikationer. Detta ger ett effektivare och mer korrekt arbete än om personerna i fråga hade suttit på egen hand [18].

För att säkerställa att produkten uppnått målet om att vara enkel och smidig att använda har det genomförts användartester. Gruppen har även använt applikationen dagligen för att testa att det inte uppstår applikationsfel. Då fel uppstått har en rapport om felet framställts via en inbyggd rapporterinsfunktion på Github, en hemsida för versionshantering. Därmed gjordes en person ansvarig för att lösa problemet.

3.1.1. Versionshantering

Verktyg för versionshantering användes för att upprätthålla en bra struktur genom utvecklingen och valet föll på verktyget Git [19]. Detta verktyg valdes främst av anledningen att de flesta medlemmarna i gruppen redan använt det tidigare. Genom att skapa olika förgreningar har en fungerande version av projektet alltid varit tillgänglig och då ny funktionalitet implementerats skapades en ny förgrening från den fungerande versionen. Då denna funktionalitet var färdig och felfri sammanfogades förgreningarna till en ny fungerande version och kvarvarande förgrening togs bort. Git har även varit ett nyttigt redskap när det gällde att se historiken för projektet och fortlöpande kunna publicera delmål.

3.1.2. UML

Modelleringen av projektet utfördes med hjälp av UML (Unified Modelling Language), vilket är en standard för visualisering av modeller till mjukvara [20]. Modellen har gemensamt inom gruppen tagits fram med hjälp av mindmapping vilket innebär en öppen idékläckning där alla idéer är accepterade.

Genom att först ta fram väsentliga koncept vars betydelse är passande till applikationens logik fås en naturlig ordrymd. Logiken kan sedan konverteras till kod, i form av klasser och variabler. Ordrymden kan sedan förbättras ytterligare genom att skapa relationer mellan koncepten och lägga till eventuella egenskaper. Varje iteration av idékläckningen har goda förutsättningar att förenkla logiken i ordrymden. Eftersom detta sedan utgör grunden för det UML-diagram, vars definition utgör applikationens modell, är det en viktig och tidssparande process. Mindre tid behöver sedan spenderas på att flytta kod när logiken är i behov av en förändring. Detta är dock en hårfin avvägning gällande hur mycket förebyggande arbete som är rättfärdigat, beroende på applikationens komplexitet.

3.2. Användartester

Teoretiska och empiriska utvärderingar i form av användartester behövdes för att förbättra designen och dess användbarhet. Användbarhet innebär hur effektiv, funktionell samt tillfredsställande applikationen är [21] . Genom att sammanställa användartesternas data kan flera nödvändiga funktioner läggas till eller prioriteras bort.

Ett verktyg vid namn JustInMind användes för att skapa en digital prototyp i form av klickbara mockups, vilket innebär en skalenlig illustration av designen i form av bilder. Verktyget erbjöd därmed förenklad testning av designen genom att det ursprungligen kan skapas ett bildflöde istället för att flödet skall programmeras. Detta besparar tid då nya flöden kan skapas och justeras utan att skriva någon kod. Användarna kan då få en känsla för hur applikationen fungerar, trots att prototypen i detta skede endast består utav ett navigationsflöde av statiska bilder.

3.2.1. Teoretiska metoder

Två olika teoretiska metoder användes för att utföra användartesterna. Den första metoden heter Hierakisk uppgiftsanalys, förkortat HTA. Metoden används för att strukturera upp vilka delar systemet innehåller samt vilka steg användaren behöver gå igenom för att slutföra de valda uppgifterna [16]. Metoden formulerades först till ett huvudmål vilket bröts ner i mindre delmål tills en önskad abstraktionsnivå erhölls. På den lägsta abstraktionsnivån beskrivs även den handling vilken behöver utföras för att delmålet skall uppfyllas [22]. Denna metod ger en god överblick över systemets funktioner samt deras uppbyggnad.

Den andra metoden kallas för Cognitive Walkthrough och dess syfte är att undersöka om användaren försöker utföra uppgiften på rätt sätt och varför [16]. Till varje handling ställs fyra frågor vilka beskriver hur väl systemet fungerar:

- Kommer användaren att försöka uppnå rätt effekt?
- Kommer användaren att upptäcka att rätt handling finns tillgänglig?

- Kommer användaren att associera rätt handling med önskat mål?
- Om rätt handling utförs, får användaren återkoppling om detta?

3.2.2. Urval

Det är viktigt att inkludera testpersoner med olika teknikvanor och livserfarenheter eftersom Android Wear är en ny teknik ute på marknaden. Syftet är att få en spridning av olika mentala modeller, hur personer tänker sig att saker fungerar, samt få en mer korrekt bild av hur en testperson i praktiken kommer att använda det grafiska gränssnittet.

3.2.3. Testmiljö

Användartesterna utfördes i Användbarhetslabbet på Chalmers, Johanneberg. Denna miljö är till för att säkerställa lika förutsättningar, till exempel att ljus- och ljudförhållanden är lika i de olika testerna för att förbättra tillförlitligheten. Genom att ha användartesterna inomhus finns det ingen risk att prototypen tar skada från regn och blåst. Dessutom blir dokumentationen och kontrollen av användartesterna enklare inomhus då exempelvis buller från trafik ej kan störa.

3.2.4. Uppgifter

Till båda användartesterna användes sju testpersoner, vilket är precis lagom och en bra marginal för att få ett helhetsintryck. Enligt Jakob Nielsen behövs det endast fem testpersoner för att finna de flesta felen [23]. Testpersonerna skulle utföra ett visst antal huvuduppgifter. Dessa uppgifter bestod av mindre delar, vilket exempelvis skulle kunna innebära att söka efter en kontakt och skicka en uttryckssymbol. Det såg ut på angivet sätt:

1. Skapa ett konto
2. Skicka ett meddelande (starta konversation)
3. Vänta på svar.

Dessa uppgifter följdes upp av scenarion, det vill säga en kort beskrivning för att testpersonerna ska veta vad de skall göra. För vidare information se bilaga D.

3.2.5. Användartest 1

Deltagarna bjöds enskilt in i labbrummet och testet genomfördes med endast två projektmedlemmar närvarande, en vars uppgift var att agera försöksledare och en vars uppgift var att anteckna samt ta hand om inspelningen. Framför deltagaren fanns två smarta telefoner, där den ena representerade telefonens vy, och den andra klockans vy. Se figur 3.1.
Testet bestod av tre scenarion och såg ut enligt följande:

Scenario A:
“Du har precis skaffat en smart watch. Det första du gör är att installera Sketchagram. För att börja kunna använda den måste du skapa ett konto och lägga till en vän. Lägg till din bästa vän “Anna123”.”

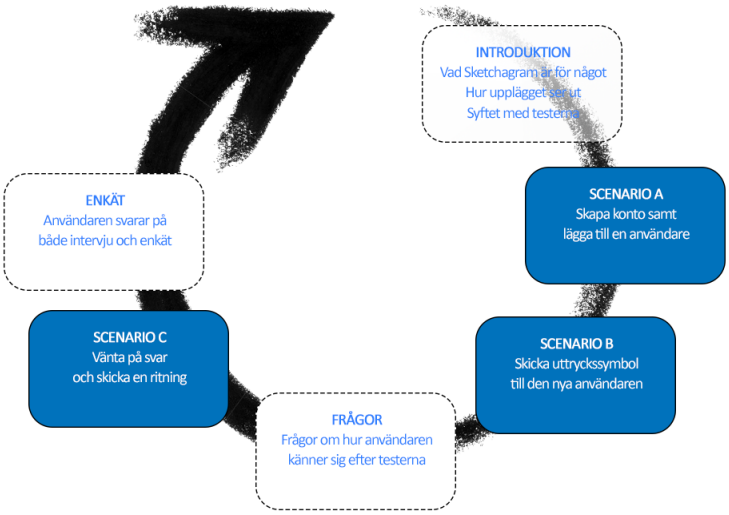
Scenario B:
“Nu när du lagt till din bästa vän, vill du börja kommunicera med henne. Detta vill du

Figur 3.1: Användartest 1 med deltagare och försöksledare till vänster. Förstorad vy på applikationen till höger.

göra genom att skicka en glad uttryckssymbol från applikationen”

Scenario C:
“Du har nu fått svar från din bästa vän. Kolla vad hon skrev och svara med att måla en tumme upp och skicka det till henne från klockan.”

Utöver dessa scenarion ställdes frågor om hur deltagaren kände sig efter varje uppgift i form av “Hur självsäker känner du dig just nu?”. Deltagaren fick då välja på en skala från 1, väldigt osäker till 5, väldigt säker. De fick även svara på varför de tyckte så. Efter samtliga scenarion svarade deltagarna dessutom på ett formulär. Formulärets frågeställningar om hur det var att använda två skärmar samt upplevelsen i sin helhet och i detalj. Detta inkluderade då frågor om design, layout, färg och font. Se figur 3.2 användartesternas testprocedur samt bilaga B för mer information.



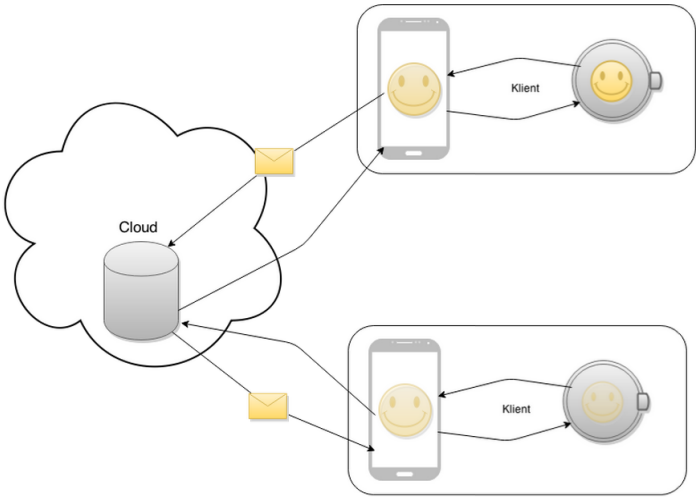
Figur 3.2: Testprocedur för användartest 1 och 2.

3.2.6. Användartest 2

Användartest 2 blev till skillnad från föregående test förflyttat till ett vanligt grupprum på Chalmers eftersom behovet av ett anpassat rum ej var lika stort. Framförallt på grund av att designen vid detta skede var implementerad i applikationen och att testdeltagarna då kunde interagera med telefonen precis som i en vardaglig miljö. Deltagarna fick denna gång även samma uppgifter och frågor som i användartest 1. Anledningen är att kunna jämföra med data från användartest 1. För vidare information, se bilaga C.

4. Implementation/Design

Projektet resulterade förutom i en slutgiltig produkt även i tre prototyper på vägen. Dessa prototyper presenteras i detta kapitel för att ge en inblick i applikationens utveckling. Varje prototyp är resultatet av en genomförd sprint och målet för vardera var att uppnå respektive milstolpar, se bilaga A för fler detaljer. Figur 4.1 illustrerar applikations flöde då ett meddelande skickas.



Figur 4.1: Grundläggande illustration av systemets uppbyggnad.

4.1. Utkast design

Parallellt med utvecklingen togs först basutseendet för applikationen fram genom pappersskisser. Syftet med skisserna var främst att se vilka vyer som var nödvändiga samt hur navigationen skulle ske emellan dessa. Efter detta applicerades dessa skisser på Photoshop prototyper där grafikbiblioteket tillhandahållet via Android Developers användes. I dessa prototyper utvärderades designen och ändringar gjordes för att vyerna skulle representera den slutgiltiga designen mycket bättre. Se bilaga E för bilder på utkast.

4.2. Prototyp 1

Vid första sprintens avslut hade dem utsatta milstolparna färdigställts. Fokus låg till största del på funktionalitet samt att applikationen skulle ha möjligheten att skicka ett statiskt textmeddelande från en klocka till en annan. Detta skedde då via en server givet att två specifika användarkonton användes gentemot servern. Dessutom hade mobilapplikationen samma möjligheter eftersom klockan utökar den smarta telefonens funktionalitet, likt nämnt tidigare i Problem 1.3.

Milstolparna för prototyp 1 bestod av att det dels skall finnas en applikation på klockan samt en applikation på telefonen vars funktionalitet sköter kommunikationen mellan klockan och servern. Även någon typ av funktionalitet skulle existera på klockan samt att kommunikation mellan två klockor skulle finnas och fungera. Samtliga av dessa uppnåddes. Se figur 4.2 för en tidig design av inloggningsskärmen.

Figur 4.2: Loginskärm för prototyp 1.

4.2.1. Server

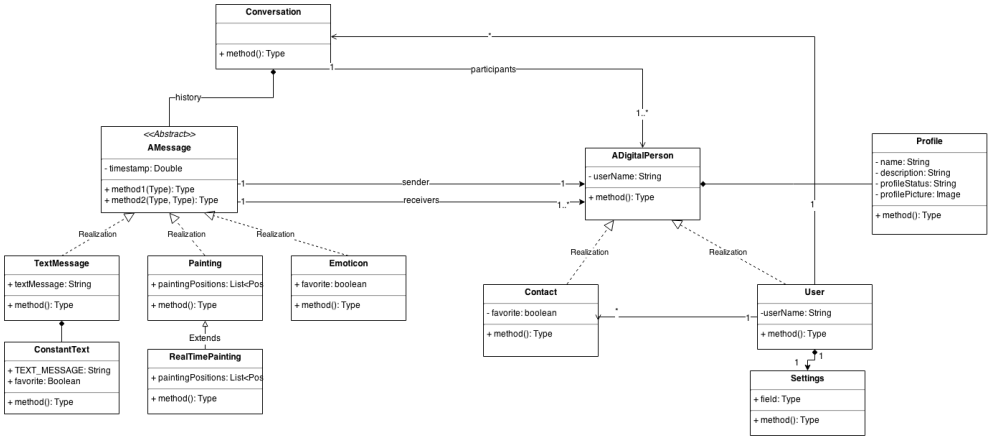
Servern hade installerats och var redo att börja användas då första prototypen var färdigställd. I detta tidiga skede installerades servermjukvaran på en Raspberry Pi, främst för att uppnå en låg energiförbrukning, då den endast konsumerar 0,5 till 1 watt [24]. Mjukvaran vilken valdes att agera server kallas Openfire och har öppen källkod samt bygger på protokollet XMPP [25]. Openfire tillhandahåller allt från att skapa konton till att hantera kontakter mellan användarna. Dessutom håller den även reda på vilka användare som är tillgängliga samt sköter kommunikationen mellan dem. Servern tar därmed emot data från klienter och skickar detta vidare till rätt mottagare samt skickar uppdateringar om förändrad status av kontakter, till de användare vilka prenumererar på dessa.

Mjukvaran har även en egen arkitektur för att kunna konfigurera en MySQL-databas. Databasen lagrar information relaterad till servern, vilket omfattar bland annat användare, kontakter och meddelanden. Stöd finns även för tillägg vilka kan utöka funktionaliteten på servern. Dessa tillägg innehåller funktionalitet skapad utifrån XMPP-standarden och tillhandahålls för att Openfire valt att ej inkludera dessa från start.

Funktionaliteten implementerad på klienten vilken använder servern inkluderar att skapa ett konto samt att skicka ett enklare meddelande. Detta gjordes för att förenkla implementationen av prototyp 2 då den både skulle kunna skicka mer avancerade meddelanden och hantera kontakter kopplade till ett visst konto. I denna prototyp fanns dock ingen funktionalitet för att hantera kontakter eller för att kunna skicka meddelanden mellan fler än två förskapade konton. Till detta ändamålet användes biblioteket Smack vilket är utvecklat av organisationen Ignite Realtime som även producerat Openfire [26].

4.2.2. Modellering i UML

Till prototyp 1 hade ett UML-diagram över modellen tagits fram för att kunna diskutera kopplingar mellan de olika klasserna och få ett logiskt flöde. Modellen var näst intill komplett redan efter prototyp 1, även om detaljer fortfarande fortlöpande skulle komma att falla på plats. Se figur 4.3 för en översikt över UML-diagrammet.



Figur 4.3: UML-diagram över första prototypens modell.

4.2.3. Smartklocka

Endast två knappar fanns på den smarta klockan, en knapp för att skicka meddelanden och en knapp vilken i framtiden skulle komma att visa användarens konversationer. Då användaren dessutom klickade på “New Message” visades en lista med alla kontakter. Längst ned i kontaktvyn fanns sedermera knappen ”Send” och när användaren klickade på denna skickades ett meddelande till vald kontakt. Se figur 4.4 för överblick över startskärmen.

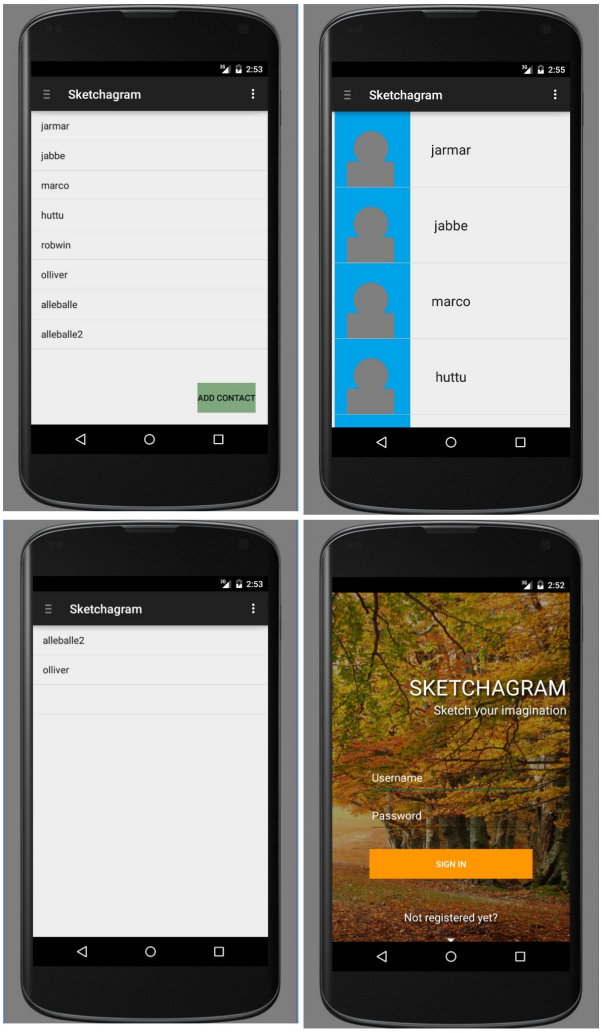
Figur 4.4: Startskärm för fyrkantig och rund smartklocka.

Fokus för klockans applikation var att få den att kommunicera med den smarta telefonen. Eftersom klockan har lägre prestanda än telefonen går det mycket snabbare att endast samla information på den och låta telefonen utföra alla beräkningar. Den smarta klockan och telefonen kommunicerade först med hjälp av textsträngar, vilket var smidigt när informationen endast bestod av strängar men detta tillvägagångssätt användes ej efter prototyp 1.

4.3. Prototyp 2

Grundläggande grafiskt gränssnitt var implementerat och möjliggjorde för användare av mobilapplikationen att kunna skicka målningar mellan varandra, vilka sorterades in i olika konversationer. Detta var även möjligt från klockans applikation. Förutom detta gick det även att skapa konton, lägga till kontakter, nå vyn för inställningar samt en informationsdialog om applikationen. Användaren möttes dessutom av en inloggningsskärm vid start av applikationen där det krävdes inloggningsuppgifter eller att användaren registrerade sig. Informationen gick även via servern för att verifiera uppgifterna. Målningar spelades upp likt originalet målades, efter varje gest, istället för att ge en statisk bild. Vid

detta skede hade också servermjukvaran istället installerats på en vanlig dator på grund av vissa begränsningar i operativsystemet på tidigare enhet. Figur 4.5 visar de olika vyerna för applikationen.

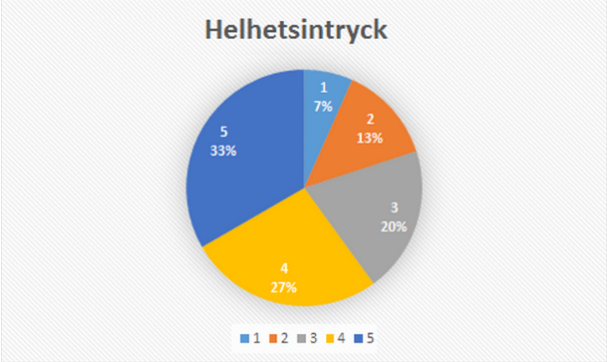


Figur 4.5: Översta raden till vänster, konversationsvy, till höger, vyn för val av kontakt. Nedre raden till vänster, kontaktlista, till höger, inloggningsvy.

Kraven för milstolpe 2 hade därmed uppfyllts, se bilaga A. Dessa bestod av följande: en användare skulle kunna ha kontakter, det skulle gå att skapa ett konto, samt att det skulle vara möjligt att kommunicera mellan två olika konton. Det skulle även gå att måla bilder och skicka dessa mellan klockor.

4.3.1. Användartest 1

Deltagarna vid användartest 1 ombads att svara på olika typer av frågor som respons till de scenarion de precis utfört, där en stor del av dessa var kopplade till självsäkerhet. Se figur 4.6.



Figur 4.6: Resultat av helhetsintrycket på applikationen från första användartestet.

Utmärkande var framförallt att de testpersoner med tidigare erfarenhet från Android Wear hade lättare att sätta sig in i hur klockans applikation fungerade än andra. Framförallt berodde detta på att designen följde de riktlinjer vilka återfinns för Android. Se bilaga B för detaljer om resultaten.

4.3.2. Kontakter

Prototyp 2 hade även stöd för kontakter men saknade en kontroll på servern som verifierade om användaren verkligen fanns. Det innebar att även om användaren som lades till inte fanns skapades en bindning mellan dem i databasen, dessa sparades både lokalt och på servern. Lokalt för att minska trafiken till servern då det ej ansågs väsentligt att fråga vilka kontakter användaren hade, varje gång en interaktion med dessa utfördes.

4.3.3. Måla

Vid slutförandet av den andra prototypen hade ritfunktionen färdigställts och det gick att skicka målade bilder till varandra. Målningarna sparades ner till bytefält för att inte uppta stor plats i applikationen. Databasen kunde orsaka problem i form av att applikationen fått slut på minne samt att databasen inte kunde hantera oändligt mycket data. Eftersom målningarna gjordes om till bytefält följde en minimal effekt på hur lång tid det tog att skicka fälten över servern. Detta ger ingen märkvärd skillnad vid låg belastning men är ändå värt att nämna då det kan ge stor effekt vid avsevärt fler förfrågningar till servern då fler användare ansluter sig.

Innan en målning skickas behövs dessutom en konvertering till JSON-objekt för att kunna överföra den via nätverket, detta för att XMPP inte stödjer att skicka Java-objekt utan endast klarar av att skicka text. Eftersom målningen gjordes om till bytefält hjälpte detta även till att snabba upp konverteringen samt återställningen av JSON-objekt.

4.3.4. Databas

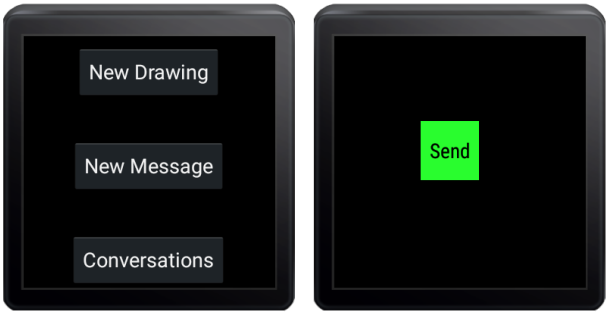
När den andra prototypen var klar hade databasen för klienten färdigställts. Det krävdes enbart en simpel databas med tre tabeller. En tabell lagrade kontakter med de fält vilka applikationen krävde. Nästa tabell lagrade alla meddelanden, dessa lagrades endast lokalt och inte på servern vilket innebär att om enheten byts följer inte meddelanden med. Meddelandena sparades som JSON-objekt i databasen, då det skulle kräva en större mängd

processorkraft att skapa speciella tabeller och spara varje enskilt värde på målningarna. Sista tabellen innehöll sedan de olika konversationerna. Detta gjordes genom att ha ett konversationsid kopplat till en eller flera kontakter och på detta sätt identifiera vilka som pratade med varandra.

4.3.5. Smartklocka

Prototyp 2 innebar för den smarta klockan ingen ny design men en helt ny knapp, nämligen "New Drawing". Klickar användaren på knappen kommer denne till en vit skärm där en målning kan målas. När målningen är klar kommer användaren vidare till en lista med alla kontakter och där väljer denne vilka kontakter meddelandet skall skickas till.

Målningen i prototyp 2 krävde att stora mängder data skickades mellan enheterna vilket ej längre gick att hantera med hjälp av endast textsträngar. Informationen som skickas mellan en smart telefon och klocka består därför hädanefter av bytefält. En illustration av klockan efter prototyp 2 finnes i Figur 4.7.



Figur 4.7: Vänstra bilden visar huvudvyn, högra vyn visar knappen för att skicka ett meddelande.

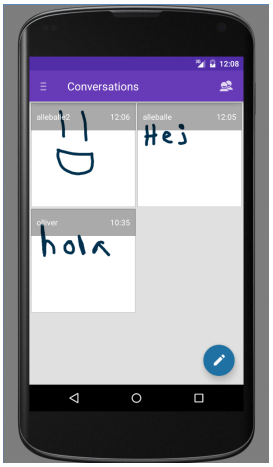
4.4. Prototyp 3

Sista prototypen gav medvetna avvikelser från ursprungliga milstolpar då exempelvis kommunikation i realtid, även om tekniskt möjligt, frångicks. Anledningen var att målningen först skickas då den är färdigmålad. Mottagaren upplever målningen vara likt realtid då den målas upp precis som sändaren målat den. Målningen upplevs även mer levande eftersom att den målas upp framför mottagarens ögon. Detta blir ej riktig realtid, där användare ser vad som målas samtidigt som det sker, dock är det väldigt likt utan att det blir alltför tekniskt komplicerat och tidskrävande. Se figur 4.8 för en överblick över konversationsvyn.

4.4.1. Användartest 2

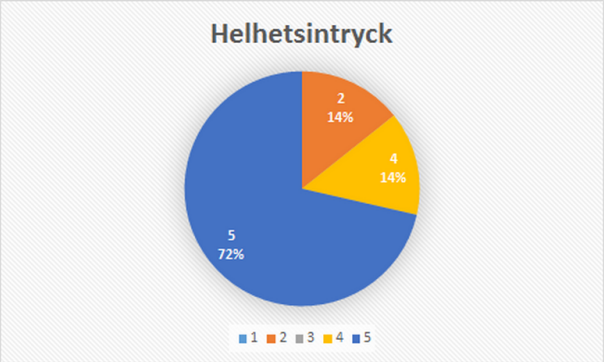
Under andra användartestet var det ett urval av individerna från första användartestet tillsammans med nya personer, detta säkerställer att ett resultat från olika perspektiv uppnås. Processen var likadan som första användartestet förutom att i andra användartestet användes en fullt navigeringsbar applikation, på både mobilen och klockan. Muntliga intervjuer samt skriftliga formulär hölls efter användartest 2 om upplevelsen.

Responsen vid detta test var mer positiv jämfört med användartest 1. Tips på möjliga



Figur 4.8: Konversationsvy.

förbättringar bestod av detaljer gällande vissa funktioner, exempelvis “mindre uttrycks-symboler” eller “mer information i kontaktlistan”. Genom att jämföra resultaten från användartest 1 och 2 kan det ses en stor skillnad på responsen. Användartest 2 fick bättre betyg i nästan alla kategorier, vilket syns extra tydligt i helhetsintrycket (se figur 4.9). Det finns fortfarande utvecklingsmöjligheter för effektiviteten, exempelvis att kunna skicka meddelande direkt efter att ha lagt till en kontakt. Se bilaga C för detaljer om resultaten.



Figur 4.9: Sammanställning av helhetsintrycket för applikationen.

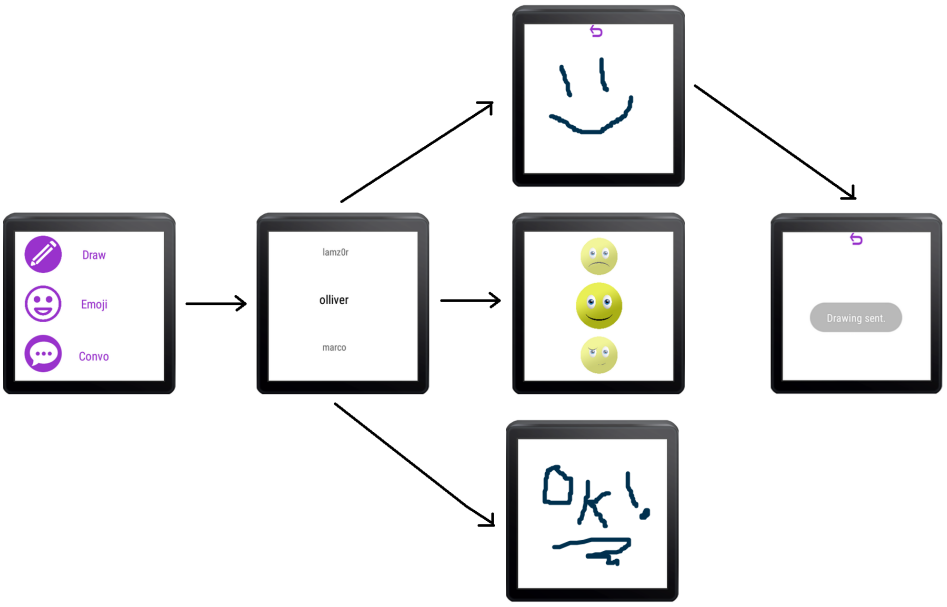
4.4.2. Notifikationer

Liksom många andra applikationer med meddelandefunktioner använder sig Sketchagram av notifikationer för att uppdatera användaren om att ett meddelande har mottagits. En stor del av alla meddelanden vilka skickas mellan användare är målningar, därför innehåller notifikationerna även målningen som skickats. Alla notifikationer är länkade med en konversation, vilken nås om notifikationen klickas på.

4.4.3. Smartklocka

Inför prototyp 3 implementerades den grafik som designerna arbetat fram under föregående sprintar. Flödet i klockans applikation illustreras i Figur 4.10. Huvudvyn är skärmen till vänster och där kan användaren välja att skicka en målning, uttryckssymbol eller se konversationer. När användaren väljer ett av alternativen kommer den vidare till en vy med

en lista bestående av alla sina kontakter. Efter att en kontakt är vald fortsätter applikationen till vyn som motsvarar användarens val i huvudvyn. Flödet på klockans applikation är tänkt att imitera det som är implementerat i telefonens applikation.



Figur 4.10: Illustration av flödet på smartklockan.

5. Resultat

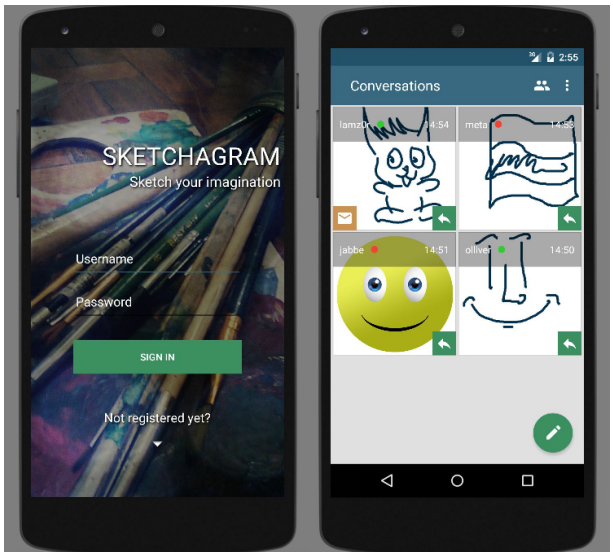
Projektets mål har varit att framställa en produkt som med hjälp av en smart telefon och klocka simplifierar för användarens behov av att kommunicera. Avsnittet nedan presenterar resultatet. Inledningsvis introduceras mobilapplikationens olika vyer, följt av vyerna för applikationen på klockan och avslutningsvis detaljer kring designen.

5.1. Smarttelefonen

Applikationens vyer för telefonen presenteras här nedan och funktionaliteten beskrivs i ord. Texten beskriver genomlöpande applikationen likt det flöde en användare normalt skulle genomgå.

5.1.1. Startskärm

Vid första installationen av applikationen på telefonen möts användaren av en logotyp med applikationens namn. Strax under finner användaren en ruta för att logga in och två alternativ ges. Om användaren har ett konto kan denna logga in med detta och annars erbjuds denne att registrera sig, se figur 5.1. Då användaren har ett tidigare konto och försöker logga in på applikationen kommer indatan användaren matar in att jämföras med informationen på servern. Detta för att kontrollera att användaren tidigare skapat ett konto och därmed i detta skede skriver in ett korrekt lösenord. Vid fem misslyckade försök att logga in måste användaren vänta i fem minuter innan det går att prova igen. Spärren avser att blockera upprepade felaktiga försök och därmed motverka elakartade handlingar då användare testar alla möjliga kombinationer av lösenord för att hitta lösenordet för en användare, enligt processen känd som bruteforce.



Figur 5.1: Till vänster, vy för inloggning i slutprodukten. Till höger, första sidan som visas efter inloggning är konversationslistan.

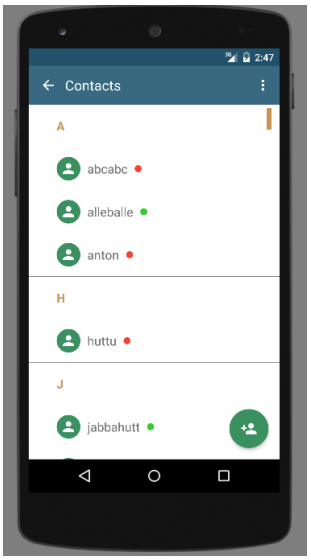
Om användaren är ny behöver denne istället skapa ett konto med e-post, användarnamn och lösenord som inmatningsdata. Den nya användaren kommer då lagras i serverns databas, vilket tillåter användaren att logga in på olika enheter utan att behöva skapa ett nytt konto.

Startskärmen är simpel och lätt för användaren att förstå vid första användning, vilket resultatet från användartester indikerar i bilaga C. Det finns inte någon överflödig information vilken kan förvirra för användaren och stämmer därför väl överens med syftet. Om användaren väljer knappen för att registrera sig dras startskärmen nedåt med en animerad effekt, vilket leder till registreringsvyn. Se återigen figur 5.1. för en överblick över startskärmen och konversationsvyn.

5.1.2. Kontakter

Applikationen innehåller en kontaktlista där användaren kan se sina kontakters namn och status om de är tillgängliga eller ej. Det finns även ett alternativ att lägga till flera kontakter under denna vy. De kontakter vilka läggs till lagras sedan i den lokala databasen och på servern. Status kan däremot enbart erhållas från servern. För att en användare ska visas som tillgänglig måste applikationen vara igång, antingen aktivt eller i bakgrunden på den smarta telefonen. Dessutom måste användaren vara inloggad mot servern.

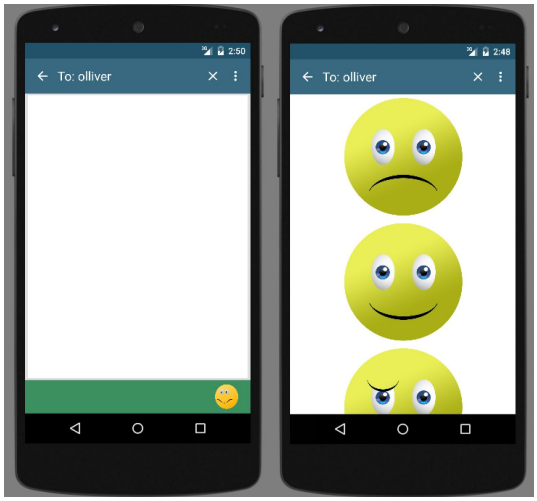
För att snabbt kunna hitta vilken eller vilka kontakter som är inloggade sorteras dem efter första bokstaven i användarnamnet. När rullningslistan används visas den bokstav användaren är på i sin kontaktlista. Se figur 5.2. för kontaktlistan.



Figur 5.2: Vy för kontakter i slutprodukten.

5.1.3. Målning

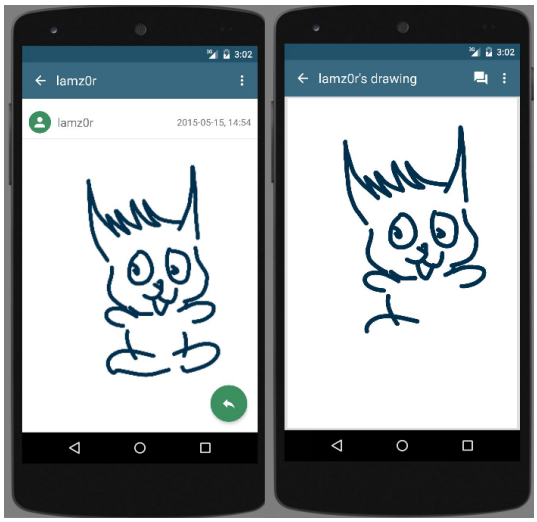
Användaren väljer vilken kontakt målningen skall skickas till, innan den målas. Användarna hämtas från den lokala databasen och sedan visas dem upp i en lista. Efter några små modifikationer och beslut i gruppen avgränsades chattfunktionen, se 1.2. Syfte och Avgränsningar, till att endast klara av målningar och uttryckssymboler. Efter att användaren målat klart dröjer det två sekunder innan den skickas vidare. Anledningen till tidsgränsen är för att användaren skall kunna fortsätta måla på bilden även om fingret lyfts från skärmen. Vyn består till synes av en kanvas där användaren kan måla med hjälp av fingergester, samt även en knapp för att kunna nå vyn för uttryckssymboler. Se figur 5.3. för illustration av vyn.



Figur 5.3: Till vänster, vy för målning med en knapp i högra nedre hörnet som leder till en lista med uttryckssymboler. Till höger, en lista över tillgängliga uttryckssymboler.

5.1.4. Konversationer

Konversationsvyn är det första användaren möts av och målet är att ge en smidig överblick över den funktionalitet applikationen erbjuder. Denna vy visar alla de konversationer som har påbörjats med olika kontakter och visar även det sista meddelandet vilket hör till varje konversation. Detta för att ge användaren en snabb överblick över de senaste händelserna. Användare kan även svara genom att klicka på svara-knappen i det nedre högra hörnet på varje konversation. På motsatt sida finns även en plats för en symbol när en notifikation fås. Även här syns användarens status bredvid dess namn. Se figur 5.4. för att se hur en målning ser ut inne i en konversation samt hur den ritas upp.



Figur 5.4: Till vänster, vyn för en konversation där bilderna visas i en lista. Till höger, vyn för en bilds uppritning.

5.1.5. Notifikationer

Vid inkommande meddelanden får användaren en notifikation på sin enhet. Denna innehåller bland annat information om vem som skickat meddelandet samt själva meddelandet. Om användaren trycker på notifikationen öppnas applikationen och meddelandet visas. För att användaren ska kunna få notifikationer måste status på användaren vara aktiv. Alla tidigare konversationer lagras sedan på den lokala databasen, detta gör det då möjligt för användaren att se tidigare skickade målningar. Varje skickat meddelande är då definierat med ett datum, en tidpunkt samt vem avsändaren är.

5.2. Smartklockan

Android Wear lägger störst vikt på att presentera notifikationer från telefonen och att tillåta interaktion med dessa. Det krävs en installerad applikation på klockan om specialanpassade funktioner, som att måla, skall användas. Den implementerade applikationen i klockan ger användare möjlighet att välja bort interaktion med telefonen. Användning av klockan sparar tid och användaren undviker otymplighet, vilken tillkommer vid användning av telefonen, vilket beskrivs i Teknisk Bakgrund 2.

När användaren navigerat till den smarta klockans installerade applikationer och startat Sketchagram finner den sig i huvudvyn. Första gränssnittet består av tre olika alternativ vilket följer standarden i Android [21]. Alternativen är kombinerade med en beskrivande symbol och interaktion med dessa leder vidare till funktionerna måla, skicka uttryckssymbol och en vy för att se tidigare konversationer. Användaren kan klicka på hela raden och inte bara på texten eller ikonen vilket underlättar navigeringen på klockans lilla gränssnitt. En illustration av startskärmen syns i Figur 5.5.

Figur 5.5: Huvudvy.

5.2.1. Bluetooth

Applikationen utnyttjar tekniken och skickar anpassade meddelanden med information beroende på vad användaren vill göra på den smarta klockan. Meddelandet består utav en textsträng och ett bytefält. Textsträngen använder applikationen för att urskilja vilken sorts information bytefältet innehåller. Java har en teknik vilken tillåter bytefält att konverteras till en nyckel-värde tabell, kallad Map. Från denna går det sedan att hämta information som tal, strängar och listor. Då en Map gör det enkelt att utvinna information på ett smidigt sätt är tekniken att föredra när bytefält används.

5.2.2. Välja kontakt

När användaren valt ett av alternativen på startskärmen visas kontaktvyn vilken innehåller alla användarens kontakter, se figur 5.6. I kontaktvyn väljer användaren en kontakt den vill interagera med. Detta extra steg infördes för att simplifiera för användaren och ett likadant flöde är implementerat på den smarta telefonen. Var gång användaren kommer till kontaktvyn hämtar klockan alla kontakter på nytt för att alltid hålla användarens kontakter uppdaterade och sorterade. Kontaktlistan sorteras efter när användaren senast interagerade med varje kontakt.

Figur 5.6: Kontaktvy.

5.2.3. Målningar

Gränssnittet består här utav en vit bakgrund där användaren tillåts interagera genom att måla med fingret på klockan. Vyn startar en tidtagning i bakgrunden vilken har till uppgift att upptäcka när användaren är klar med målningen. Tiden återställs var gång användaren målar på skärmen. Först då tidsgränsen på två sekunder passerats, skickas målningen eftersom att applikationen då uppfattar den som färdigritad.

I målningsvyn erbjuds alternativet att gå tillbaka genom att trycka på retur-symbolen högst upp i gränssnittet. Om användaren klickar fel eller ångrar sig, skall det gå att avbryta målningen. Standard för att gå tillbaka i Android Wear är att svepa med fingret från vänster till höger men då detta uppfattas som att användaren vill fortsätta måla var en bakåtknapp tvungen att implementeras. Målningar på klockan är implementerade i stort sett likadant som på den smarta telefonen, dock med lite färre funktioner eftersom att ej alla kan användas på klockan. Målningsvyn illustreras i Figur 5.7.

Figur 5.7: Målningsvy efter att användaren målat en glad gubbe.

5.2.4. Uttryckssymboler

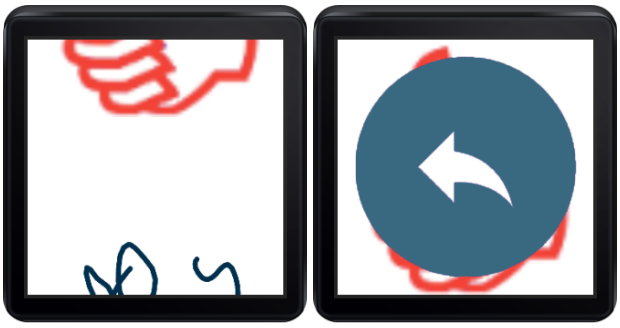
Applikationen innehåller även en lista med olika uttryckssymboler. Genom att trycka på en uttryckssymbol skickas den till mottagaren som valdes i den föregående vyn. Uttrycks-symbolsvyn illustreras i Figur 5.8.

Figur 5.8: Uttryckssymbolsvy där alla uttryckssymboler representeras i en lista.

5.2.5. Konversationer

På klockan visas en simpel vy av användarens konversationer. Om kontakten användaren valt att interagera med i kontaktvyn ej innehar en startad konversation kommer ett textmeddelande upp på skärmen vilken förklarar varför ingen konversation visas. Skulle kontakten däremot ha meddelanden mellan sig och användaren fortsätter applikationen till en anpassad konversationsvy. Genom att svepa uppåt eller nedåt med fingret kan användaren sedan enkelt navigera mellan olika målningar och uttryckssymboler, skickat av antingen användaren själv eller mottaget från kontakten.

Vill användaren svara inuti en konversation trycker denne en gång på meddelandet och klickar sedan på knappen för svar, vilken genast dyker upp. Interagerar användaren med svarsknappen dyker målningsvyn upp och målningen skickas till kontakten i konversationen. En skärmbild tagen när användaren går mellan två skickade meddelanden i konversationsvyn och svarsknappen illustreras i Figur 5.9.



Figur 5.9: Till vänster, användaren växlar mellan två meddelanden. Till höger, en svarsknapp visar sig när användaren tryckt på ett meddelande

5.3. Design

Googles Material Design fanns i åtanke redan från ett tidigt stadium för att få applikationen att kännas ny och stilren. Detta innebär att allt från komponenter i gränssnittet till mått har utgått från deras standarder. För att få en fulländad Material Design-känsla har även ikoner inhämtats från Googles egna auktoriserade källa, istället för Androids standardsymboler vilka ingår i utvecklingsverktyget [16].

5.3.1. Användartester

På första användartestet reagerade majoriten på designen. Färgerna ansågs som tråkiga och behövde bli mer inbjudande, med exempelvis accentfärger, det vill säga färger som kompletterar och balanserar varandra, för att göra formgivningen mer personlig. Startvyn på telefonapplikationen var otydlig då den vita texten inte syntes ordentligt gentemot bakgrundsbilden. Även storleken på vissa ikoner och uttryckssymboler behövde göras större. Vissa ikoner var till och med förvirrande för användaren, exempelvis Sketchagramlogotypen som i detta skede var ett stort S. Flertalet trodde att S:et betydde “SEND”, då den visades i samband med målarvyn. Undersökningarna visade även att testdeltagarna tyckte “Lägga till kontakt”-ikonen ej var tydlig, vilket resulterade i en ny design för ett senare skede. Självsäkerheten i användartest 1 var dessutom blandad. På scenario A var det 17% av användarna som svarade 5, vilket är det högsta betyget, se bilaga B. Om självsäkerheten sedan jämförs med användartest 2 är det tydligt att det har blivit en förbättring. På scenario A i andra användartestet var det 100% som svarade 5, se bilaga C.

Svaren från användartest 2 innehöll mycket positiv feedback. Responsen var att färgerna därmed var tydliga och höll en enhetlig stil. Framsidan var smidig, logisk och enkel att navigera. I klockan slopades däremot logotypen i målarvyn för att ej förvirra användaren. Ikoner och uttryckssymboler har blivit förstorade för att de skall synas extra tydligt, vilket uppskattades väldigt mycket av användarna. Gällande förbättringsmöjligheter önskade en majoritet en knapp för att manuellt kunna skicka meddelanden i applikationens målarvy, istället för att dessa skickas automatiskt.

5.3.2. Färger

Färgerna för applikationen valdes till att bestå utav blått som primär färg, grönt som sekundär färg samt en tredje kompletterande färg, gult. Även en mörkare färgton för var och en av dessa inkluderades för att bland annat simulera olika tillstånd, vilket exempelvis uppkommer då en knapp är nedtryckt. Valet av färger gjordes utefter färgtips från Cameron Chapman gällande hur en färgpalett bör skapas [27]. För att få en lugnare känsla av applikationen rekommenderas det att välja tre liknande färger i ett färghjul totalt sammansatt av tolv färger [28]. På detta sätt skapas inga skarpa kontraster mellan olika komponenter i gränssnittet. Eftersom applikationen är liten, till skillnad från stora webbapplikationer, valdes istället endast två liknande färger och en kompletterande färg åt notifikationerna.

Både färgen grön och blå är kända färger från välkända applikationer med framförallt Instagram, Facebook, Whatsapp och Hangouts i åtanke, vilket gör att användaren känner igen sig. Användningen av den primära färgen är i form av bakgrund till den övre Action Bar, vilket är namnet för verktygsfältet högst upp i gränssnittet och kan nås i hela applikationen. Den sekundära färgen används främst till knappar, medan den kompletterande

färgen nyttjas till utstickande element, vilket bland annat inkluderar färg till notifikationerna.

5.3.3. Mönster och komponenter

Alla vyer representerar listor av olika slag, därför behövdes ett bra mönster för att representera dessa. Kontaktlistan sorteras efter bokstavsordning där varje kontakt har en profilikon som standard samt en statusikon. I konversationsvyn däremot representeras relevant information vid en snabb överblick. Eftersom meddelanden alltid är en bild eller uttryckssymbol valdes ett rutnät där varje ruta innehåller en illustrativ bild, vilken representerade det sista meddelandet från personen. Här används de standarder Material Design definierat för rutnät, vilket vanligtvis representerar fotografier i ett bildbibliotek [16]. Detta ger applikationen en unik känsla inom konversationsapplikationer där meddelanden vanligtvis representeras utav en lista med rader. Till varje ruta infogades också två ikoner i de nedre hörnen och även dessa enligt interaktionsmönstret för rutnät. Dess funktioner beskrivs i konversationsdelen under resultatet. Väl inne i en konversation representeras sedan meddelandelistan som ett nyhetsflöde med nya meddelanden längst ner.

Ett designmönster vilket endast använts på telefonen är Floating Action Button, vilket innebär en svävande knapp ovanför de andra grafiska komponenterna. Knappen befinner sig i det nedre högra hörnet och används endast för att utföra primära interaktioner [29]. I detta fall nyttjas det för att exempelvis skicka ett nytt meddelande i konversationsvyn samt för att lägga till nya kontakter i kontaktvyn. Övriga designkomponenter inkluderar hur knappar, menyer, dialoger, textfält och Toasts vilket är Androids sätt att presentera en textsträng för användaren, bör användas. Gällande dessa standarder hänvisas dock till Material Design [16].

6. Diskussion

Gruppen har arbetat med att få ihop en innovativ applikation vars syfte är att simplifiera kommunikationen för människan i dagsrutinerna. Gruppen har inte alltid varit enig i alla beslut, men en fullständig applikation har skapats i slutändan och nått upp till förväntningarna.

6.1. Arbetsprocessen

Det finns vissa förändringar gruppen skulle göra om ett liknande projekt skulle utföras i framtiden. Något som bestämdes tidigt var vilket arbetssätt gruppen skulle använda och valet föll på SCRUM. Problemet vilket uppstod för gruppen med användningen av SCRUM kom från svårigheterna att hålla reda på vilka användarberättelser som var kvar eller vilken prioritering de skulle ha. Ett återkommande problem vilket uppstod när varje sprint avslutades var att olika användarberättelser inte alltid var tillräckligt klara för att nästa sprint skulle kunna påbörjas. Efter att gruppen analyserat vad som skulle kunnat göras annorlunda kom det fram olika förslag på förändringar. De största förändringarna vilka föreslogs var att använda kortare sprintar istället för långa. Ännu en lämplig förändring hade varit att lägga mer fokus på de mest relevanta användarberättelserna direkt, vilket hade resulterat i att projektet aldrig stått stilla på grund av ofullständig funktionalitet.

En del uppgifter gruppen önskat utföra har ej blivit genomförda främst på grund av för höga ambitioner och att dessa inte varit lika högt prioriterade som andra mål. Att implementera tester, både grafiskt och datamässigt för applikationens användargränssnitt och modell, var ett av dessa. Detta mål blev bortprioriterat då alla gruppmedlemmar hade andra uppgifter vilka ansågs viktigare att utföra. Gruppen hade inte heller tillräckligt med kunskap inom exempelvis testning vilket medförde att det blev bortprioriteringar. Även den hårda kopplingen mellan mjukvarumodell och server gjorde det svårt att skriva tester som enbart påverkade modellen och inte servern då de flesta metoder denna kan utföra även går via servern. Ytterligare en komplikation med testerna var att buggarna framförallt orsakades utav grafiska implementationer i Android och sällan av modellen.

Under projektets gång har det varit svårt att få ett bra flöde mellan design och implementation. På grund av det valda arbetssättet har gruppen behövt börja implementera programmet och lägga in tillfällig design eftersom att det inom tidsramen ej tillåtits att vänta på färdigställandet av slutgiltig design. Detta ledde till att efter designteamet blev klara med att designa på papper gick de över till implementering av denna design i applikationen parallellt med att de andra medlemmarna var tvungna att fortsätta implementera övriga funktioner för att hinna. Om mer tid varit tillgänglig hade det varit önskvärt att först göra klart designen av applikationen innan den börjat implementeras. Detta innebär att det redan under implementeringsfasen varit möjligt att förverkliga rätt design direkt och dubbelarbete hade kunnat undvikas.

Det fanns dessutom under projektets gång ingen tydlig struktur över hur uppgifter från dokumentet med önskade användarscenarion skulle fördelas. Om en projektledare hade utsetts vars ansvar var att fördela dessa uppgifter till de olika projektmedlemmarna hade detta problem kunnat undvikas. Genom att göra på det här sättet hade det ej funnits några tvivel om vilka uppgifter som hade högst prioritet och skulle lösas först. Detta hade även gjort att en medlem inte fått någon ny uppgift förrän den pågående uppgiften var klar. Detta för att förhindra att en viktig funktion blir halvfärdig till följd av att en annan uppgift

tagits an innan den tidigare var fullständig.

6.2. Lärdomar

Under tillfällen då servern inte fungerade blev resultatet att applikationen kraschade och förhindrade utvecklingen. Lärdomen från detta är att gärna öppna upp för att kunna köra applikationen utan krav på server. Detta skulle åstadkomma att applikationen blir mer löskopplad och kan användas till olika serverlösningar på ett smidigare sätt. Om applikationen istället haft en struktur baserad på att hantera varje modul utan att de har kännedom om varandras innehåll, skulle en sådan nätverkskomponent kunnat uppnås.

Resultaten från användartesterna gav en inblick i hur användarna navigerar och trycker sig fram i applikationen. Att ha ett användartest innan implementationen är viktigt då det indikerar om utvecklandet av vår applikation går i rätt riktning eller ej. Mycket potential för förbättringar identifierades redan efter första användartestet och det togs väl tillvara. Genom att diskutera resultaten från datan som samlades in, kunde projektet följa en tydlig väg mot en förbättrad och mer användarvänlig applikation. För att se om applikationen uppnått sitt mål är det ännu viktigare att ha ett användartest i slutet av projektet för att kunna jämföra med ursprunglig data. Stora skillnader var tydliga, då resultaten från användartest 2 var helt klart mycket bättre och de flesta felen vilka identifierades under användartest 1, löstes till användartest 2.

Testdeltagarna som både deltagit i det första samt det andra användartestet beskrev den nya prototypen som tydligare och mer effektiv, se bilaga C. Majoriteten ansåg att färger och stilrena ikoner gör applikationen mer användarvänlig. Testerna resulterade även i en bättre och stadigare applikation. De fel som identifierades under användartest 2 var främst relaterade till detaljer och extrafunktioner. Det bör dock genomföras i fler iterationer innan applikationen kan släppas till konsumentmarknaden. Många funktioner som skulle kunna göra applikationen ännu mer attraktiv behövs för att få användarna att stanna och vilja fortsätta använda Sketchagram. Några exempel inkluderar fler val av färger till målningarna och fler uttryckssymboler.

6.3. Snabb utveckling

Visionen förändras under tidens gång då nya idéer uppkommer eller då en idé visar sig vara sämre i praktiken. Dessutom gäller det att vara först och snabb på att komma igång, då utvecklingen hos konkurrenterna vanligtvis är betydligt snabbare. Under projektets gång har Facebook släppt ett API för tredjeparts-appar i Facebook messenger och Apple har släppt sin smarta klocka [1, 30]. Detta är både positivt och negativt, då det är tillfredsställande att applikationen har stora möjligheter att vara attraktiv, även om flera företag satsar på liknande idéer. Det är svårt att konkurrera med dessa jättar, då de redan är etablerade och har överlägset mycket mer resurser.

I takt med att nya klockor med Android Wear introduceras utvecklas också operativsystemet. Google har under projektets gång introducerat en ny uppdatering till operativsystemet vilken innehåller bland annat stöd för Wi-Fi. Detta betyder att applikationen har potentialen att utöka funktionalitet på den smarta klockan samt bli mindre beroende av telefonen. På detta sätt blir klockan mer självständig då de vitala funktionerna som att skicka ett meddelande över nätverket inte längre skulle kräva en telefon. Google har även

introducerat stöd för att måla uttryckssymboler vilka användaren kan använda till att besvara meddelanden. Denna funktion fungerar på ett sådant sätt att den känner igen vilken uttryckssymbol användaren målar och byter ut till en motsvarande uttryckssymbol. Även detta visar på att vår vision är attraktiv, då flera uppdateringar tillkommer med liknande funktionalitet. Med hänsyn till de uppdateringar vilka är på intåg skulle användaren endast behöva klockan och kravet på telefon elimineras då eftersom klockan får tillgång till Wi-Fi [31].

6.4. Miljömedvetenhet

I ett tidigt skede av projektet installerades servermjukvaran på en Raspberry Pi. Den främsta anledningen till detta var för att gruppen önskade ha en strömsnål serverlösning. Denna lösning fick dessvärre överges under sprint 2 då det inte gick att installera tillägg på den installerade mjukvaran medan den kördes på en Raspberry Pi. Det tillägg som behövde installeras möjliggjorde genomsökning av användare på servern, vilket användes då användaren skulle lägga till en kontakt för att kontrollera att denna verkligen fanns. Problemet som uppstod vid installationen av tillägget var att Raspberry Pi krävde att servermjukvaran kördes under Java 7. Dessvärre när Java 7 installerats på hårdvaran startade inte mjukvaran, utan kraschade direkt. Tillföljd av detta startades servern på en vanlig dator istället och där gick det att både köra servern samt installera tillägget. Detta gjorde att gruppen fick använda sig utav en vanlig dator med högre strömförbrukning vilken stod igång dygnet runt. Energiförbrukningen blev därmed i slutändan avsevärt högre än förbrukningen från en Raspberry Pi.

6.5. Framtida arbete

Realtid mellan två enheter kräver rimligen direkt kommunikation då detta ställer krav på att exempelvis varje hundradels sekund skicka nya uppdateringar och en sådan utväxling är ej lämplig via en server. Därmed kan beslutet att frångå äkta realtid ifrågasättas då det skulle vara tekniskt möjligt, men möjligtvis på allt för stor bekostnad. För att få realtid att fungera hade en direktuppkoppling mellan två klienter behövt upprättas. Denna koppling mellan två användare, vilka önskade kommunicera med varandra, hade behövts upprättas innan en användare faktiskt börjar måla. Det hade tekniskt kunnat likställas med att ett telefonsamtal upprättas med en annan användare vilken då får acceptera realtidskommunikationen. När detta är gjort kan de två användarna börja måla bilder till varandra sinsemellan. Ett annat problem som uppstår då detta skall implementeras blir hur det rent designmässigt löses om de båda användarna målar samtidigt. Detta skulle då leda till ännu djupare studier, hur användaren skulle uppfatta att exempelvis behöva vänta på sin tur innan denne kan börja kommunicera. Därmed valdes att bortse från detta eftersom att det varken påverkar syftet eller de övriga avsatta kraven för projektet.

Potentiella vidareutvecklingar och förfiningar utav applikationen kan vara att meddelanden krypteras och därmed är säkra för avlyssningar. Användaren skall kunna känna sig säker när denne startar en konversation att externa parter inte är involverade. Applikationen saknar även säkerhet gällande krypterade lösenord för användarkonton. Detta är en funktion vars prioritet är hög, speciellt när applikationen eventuellt ska ut på konsumentmarknaden. Ännu en vidareutveckling vilken gruppen samtalat kring är hur röststyrning skulle kunna användas för navigering och framförallt med tanke på individer med begränsad rörlighet. Dessutom för att lättare få mer kontroll och kunna anpassa applikationen

efter specifika behov skulle en välbehövlig förfining vara att skapa en egen servermjukvara. Alla dessa idéer är genomförbara men skulle leda till betydligt mer studier, arbete och resurser.

7. Slutsats

En applikation har skapats för att förenkla kommunikationen genom smarta klockor. De flesta planerade funktioner implementerades och endast ett fåtal valdes bort. Detta gjordes av tidsbegränsning samt för att få ett skarpare fokus på applikationen. Användartesterna tyder på att designen i applikationen gör den smidig att använda vilket förenklar kommunikationen för de användare vilka innehar klockor.

Efter att användartesterna utförts har applikationen formats till att bli effektiv och användarvänlig. Responsen från användare gällande applikationen är positiv och slutprodukten skulle kunna sättas i bruk för konsumenter efter ytterligare iterationer. Applikationen löser ett behov av att förenkla kommunikationen genom att använda en klocka och är enligt testdeltagarna ett mycket effektivt sätt att kommunicera.

För liknande projekt rekommenderas att satsa på att göra designen fullständig innan implementationen påbörjas. Detta skulle leda till en bättre struktur och mer fokus ägnas åt funktionaliteten. Även om den avsatta tidsgränsen nått sitt slut har gruppen fortfarande planer på att utveckla applikationen och därmed göra den klar att släppas till konsumentmarknaden.

Litteraturförteckning

[1] (2015) Apple - apple watch - films. [Online]. Available: http://www.apple.com/watch/films/

[2] M. Nardone and V. Silva, "Discovering android wear," in *Pro Android Games*. Apress, 2015, pp. 339–368. [Online]. Available: http://link.springer.com/chapter/10.1007/978-1-4842-0587-7_11

[3] O. Westlund, "Mångfacetterad och oumbärlig: om mobilens roll i vardagslivet," pp. pp. 101–110, 2014. [Online]. Available: http://oscarwestlund.com/wp-content/uploads/2014/02/MedieSverige-2014-Westlund-PRE-PRINT.pdf

[4] J. Minney, "FIRST APPLE WATCH DATA: ONE JUST ISN't ENOUGH," 2015. [Online]. Available: http://intelligence.slice.com/first-apple-watch-data-one-just-isnt-enough/

[5] Google inc. (2015) Creating wearable apps. [Online]. Available: https://developer.android.com/training/wearables/apps/index.html

[6] M. Wolfson and D. Felker, *Android Developer Tools Essentials: Android Studio to Zipalign*. O'Reilly Media, Inc., 2013.

[7] P. S.-A. psaintan@cisco.com. (2015) Extensible messaging and presence protocol (XMPP): Core. [Online]. Available: http://tools.ietf.org/html/rfc6120

[8] Google inc. (2015) Build for a multi-screen world. [Online]. Available: http://developer.android.com/

[9] L. Barkhuus and V. E. Polichar, "Empowerment through seamfulness: smart phones in everyday life," *Personal and Ubiquitous Computing*, vol. 15, no. 6, pp. 629–639, Dec. 2010. [Online]. Available: http://link.springer.com/article/10.1007/s00779-010-0342-4

[10] Bluetooth SIG, Inc. (2015) The smart way to connect. [Online]. Available: http://www.bluetooth.com/Pages/Bluetooth-Brand.aspx

[11] F. Feinbube, "Android," 2011. [Online]. Available: http://www.citemaster.net/get/4c5f7a02-b81c-11e3-b9c7-00163e009cc7/EmbeddedOS_MobileAppDevelopment_Android.pdf

[12] Google inc. (2015) Storage options. [Online]. Available: http://developer.android.com/guide/topics/data/data-storage.html#pref

[13] M. Butler, "Android: Changing the mobile landscape," vol. 10, no. 1, pp. 4–7, 2011.

[14] J. Kreibich, *Using SQLite*. O'Reilly Media, Inc., 2010.

[15] Adobe. (2015) Adobe photoshop. [Online]. Available: http://www.photoshop.com/products/photoshop

[16] Google inc. (2015) Introduction - material design. [Online]. Available: http://www.google.com/design/spec/material-design/introduction.html#

[17] K. Schwaber and J. Sutherland, “The scrum guide,” 2015. [Online]. Available: http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-US.pdf

[18] L. Williams, R. Kessler, W. Cunningham, and R. Jeffries, “Strengthening the case for pair programming,” vol. 17, no. 4, pp. 19–25, 2000.

[19] Software Freedom Conservancy. (2015) About. [Online]. Available: http://git-scm.com/about

[20] OMG. (2015) Introduction to omg’s unified modeling language. [Online]. Available: http://www.omg.org/gettingstarted/what_is_uml.htm

[21] e. a. Osvalder, A-L., *Metoder i Arbete och teknik på människans villkor*. M. Bohgard, et al., vol. 2010.

[22] L.-O. Bligård, “Teoretisk utvärdering av användarvänlighet.” 2014, föreläsning i kursen Usability: Teoretisk utvärdering av användarvänlighet.

[23] J. Nielsen, *A mathematical model of the finding of usability problems*. Nielsen, Jakob, and Landauer, Tomas K., vol. 1993.

[24] RASPBERRY PI FOUNDATION. (2015) Raspberry pi 1 model b+. [Online]. Available: https://www.raspberrypi.org/products/model-b-plus/

[25] Ignite Realtime. (2015) Openfire. [Online]. Available: http://www.igniterealtime.org/projects/openfire/

[26] Ignite realtime. (2015) Smack API. [Online]. Available: http://www.igniterealtime.org/projects/smack/index.jsp

[27] C. Chapman. (2015) Color theory for designers: Creating your own color palettes. [Online]. Available: http://www.smashingmagazine.com/2010/02/08/color-theory-for-designer-part-3-creating-your-own-color-palettes/

[28] Paletton. (2002) Paletton - the color scheme designer. [Online]. Available: http://paletton.com/#uid=1000u0kllllaFw0g0qFqFg0w0aF

[29] Google inc. (2015) Buttons: Floating action button. [Online]. Available: http://www.google.com/design/spec/components/buttons-floating-action-button.html#

[30] Facebook, “Facebook Messenger,” 2015. [Online]. Available: https://developers.facebook.com/docs/messenger

[31] “Android Wear: wear what you want, get what you need,” 2015. [Online]. Available: http://googleblog.blogspot.com/2015/04/android-wear-wear-what-you-want-get.html

A. Bilaga - Milstolpar

A.1. Milstolpar

För att lösa alla problem/uppgifter har ett antal milstolpar i form av projekttid framtagits. Detta används till största del för att strukturera upp arbetet.

Milstolpe 1

- **Applikation** - Det ska finnas en applikation på den smarta klockan samt en kompanjonsapplikation på den smarta telefonen som sköter all kommunikation mellan den klockan och serverdatorn
- **Funktioner på den smarta klockan** - Det ska finnas någonting att göra på applikationerna
- **Kommunikation mellan smarta klockor** - Möjliggöra att kommunicera mellan två smarta klockor.

Milstolpe 2

- **Kontakter** - det ska gå att skapa ett konto, samt. kommunicera mellan två olika konton
- **Kommunikation** - det ska gå att måla bilder och skicka dessa mellan smarta klockor.

Milstolpe 3

- **Realtidskommunikation** - Kunna se på sin smarta klocka vad den andra målar, samtidigt som det händer
- **Skicka förinställda textmeddelanden**
- **Få notiser när någon kommunicerar med ditt konto**

A.2. Slutprodukt

Till slutprodukten finns en del krav som behöver vara uppfyllda. Dessa är uppdelade i funktionella och icke-funktionella krav. Funktionella krav innefattar allt som applikationen ska kunna göra och icke-funktionella krav består av resterande.

A.2.1. Funktionella krav:

- **Kontaktlista** - Användaren ska ha en kontaktlista där den sparar kontakter som användaren ska kunna skicka och få meddelanden från.
- **Skicka förinställda textmeddelanden** - Användaren ska kunna skicka olika förvalda textmeddelanden för att effektivisera konversationen.
- **Skicka meddelanden från telefonen** - Användaren ska kunna svara på meddelanden även om denne inte har en smart klocka till hands.
- **Skicka uttryckssymboler** - Användaren ska enkelt och smidigt kunna skicka uttryckssymboler till andra klockor/användare.
- **Skicka egenritade bilder** - Användaren ska kunna illustrera och skicka egna bilder på både den smarta telefonen och den smarta klockan.

- **Meddelanden ska skickas i realtid** - Användaren ska kunna se skrivna meddelanden i realtid.
- **Första gången applikationen öppnas ska en wizard erbjudas, dvs. en genomgång på hur applikationen fungerar** - Användaren ska kunna snabbt komma igång med applikatonen.

A.2.2. Icke-funktionella krav:

- **GUI ska vara enkelt att använda** - Det ska vara simpelt att navigera runt.
- **Applikationen ska inte krascha** - Det ska vara ett stabilt system.
- **Applikationen ska flyta på vid användning** - Det ska inte vara någon fördröjning
- **Applikationen ska ha säkerhet för profiler och lösenord** - Det ska vara säkert för användaren att ha ett konto.

A.3. Problembeskrivningar

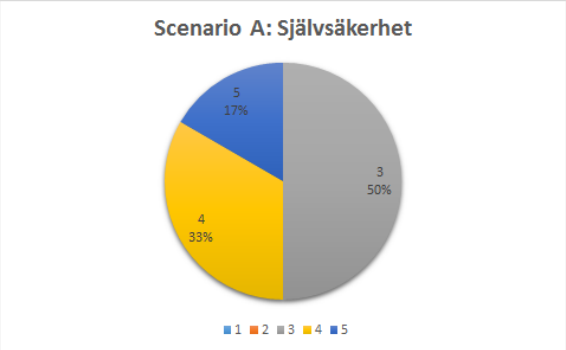
Följande problemställningar kommer att behöva lösas för att slutprodukten ska kunna genomföras.

- **Kontakter** - Hur identifieringen av mottagare och avsändare genomförs.
- **Databas** - Hur strukturen bör se ut i systemet för att hantera kontakter, meddelanden etc.
- **Design** - Hur en enkel och användarvänlig design som fungerar på olika klockor/skärmar tas fram.
- **Socialt** - Hur verifiering vid skapande av konton går till, t.ex. använda mobilnummer, Facebook, Google Plus etc för konto och kontaktinformation.
- **Server/klient** - Tekniskt löser meddelanden i realtid.
- **Databas** - Vilken data som behöver sparas, vilken databas som ska användas.

B. Bilaga - Användartest 1

Nedan följer resultaten från användartest 1.

Scenario A (skapa konto och lägga till kontakt)
Nedan följer svaren från scenario A.

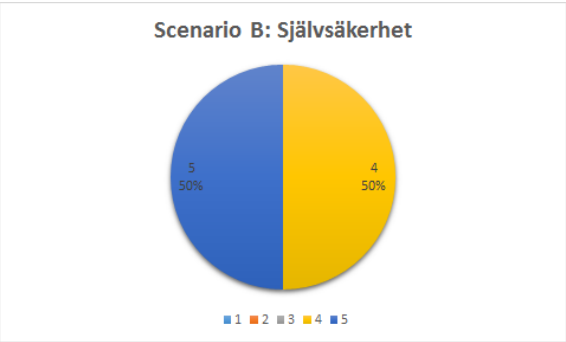


Figur B.1: Cirkeldiagram över användarnas svar för scenario A, där betygskalan var satt från 1-5.

Varför?

- Trodde att man redan var medlem.
- Ingen feedback direkt när man sökte på Anna123
- Svårt att hitta “lägga till” -knappen
- Ovan med Android
- Logiskt, kunde gissa sig fram på ikonerna
- Svårt att se med grå text på inloggningsvyn

Scenario B (skicka uttryckssymbol)
Nedan följer svaren från scenario B.

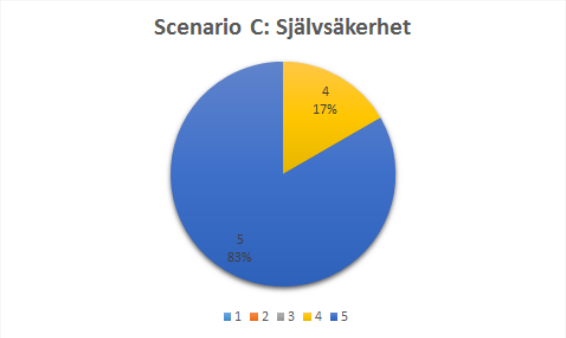


Figur B.2: Cirkeldiagram över användarnas svar för scenario B, där betygskalan var satt från 1-5.

Varför?

- Simpelt, kopplade ikonerna
- Tryckte på text istället för ikon
- Flöt på bra, trots ovan vid klocka
- Ville hellre rita direkt utan att välja pennan
- Såg varje steg, ville också måla direkt.
- Mer swipe interaktion och större knappar
- Enkelt

Scenario C (vänta på svar/svara)
Nedan följer svaren från scenario C.



Figur B.3: cirkeldiagram över användarnas svar för scenario C, där betygskalan var satt från 1-5.

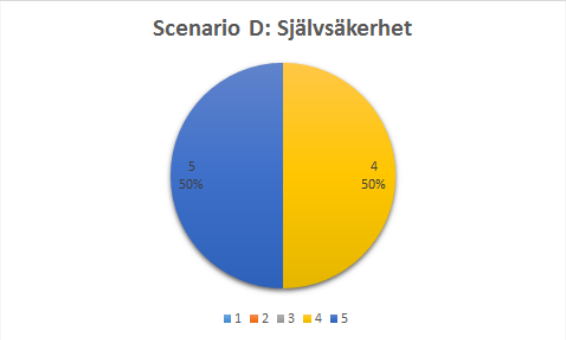
Varför?

- Kände igen ikonerna
- Tryckte på ej implementerade knappar för att nå målet
- Förväntat resultat

Scenario D (svara via klockan)
Nedan följer svaren från scenario D.

Hur självsäker kände du dig?
Varför?

- Förväxlade S:et i loggan till “svara”-knapp
- Learnability
- Ville ha större knappar



Figur B.4: Cirkeldiagram över användarnas svar för scenario D, där betygskalan var satt från 1-5.

- Svårt att veta att man ska dra uppåt

Övrigt
Deltagarna fick även delge sina generella intryck från testerna och sammanställningen ser ut enligt följande.

Färger

- Lite svårt i början (password).
- Kanske göra texten vit i början? Svårt att se.
- Stilrent.
- Kanske mer inbjudande? Mer personligt. Accentfärg, bilder, som framsidan.

Font

- Loggan känns inklistrad i första vyn på appen.
- Svårt att se på Input.

Storlek

- Större text på draw eller ta bort.
- Lagom stort.
- Ikoner behövs göra större, välja smileys för små.

Layout

- Naturligt i appen.
- Swipe-meny, först välja smiley, sen rita? Välj tom smiley annars om ej vill ha smiley.

Övrigt

- Trodde Sketchagram-symbolen betydde SENT.
- När ritar - ta bort text.

Svar från undersökning om den smarta telefonen
Nedan följer en sammanställning av svaren på frågorna som ställdes kring navigation på den smarta telefonen.

Hur var det att använda två skärmar?

- Relativt enkelt. Måste man gå genom mobilen för att skicka nytt meddelande?
- Funkade ganska smidigt
- Helt ok, ovärt att göra allt på den lilla ju.
- Jag tänker att samma inställningar påverkar både telefon och klocka, alternativt olika inställningar som kan sättas i telefon.
- Funkar bra, hade velat kunna skicka via mobilen också.
- Lite förvirrande men de följer konventioner för layout vilket är bra.

Hur smidigt var det att navigera olika sidor?

- Tydliga ikoner och bra feedback när man kommit till fel skärm, dvs lätt att hitta tillbaka.
- Enligt standard-Android så bra.
- Smidigt, men gör det enklare att lägga till en ny kontakt. Första gången appen öppnas kan jag hänvisas direkt till söka användare.
- Såg ej plustecknet när jag skulle lägga till vänner.
- Skicka till kontakt: hade hellre sett en overlay av kontakterna alltså när man klickar på en kontakt, en liten overlay med alternativ ex. skicka meddelande.

Vad kan/behöver förbättras?

- Tydligare placeholders på login. Väldigt stilren app men gillar appar som är personliga och kanske “pratar” med mig.
- Det var mycket på settings men så behöver det kanske vara. Den var lite överväldigande.
- Det skulle kunna vara mer färger så att det känns lite mer lekfullt (om man nu vill uppnå det) annars tycker jag att det såg väldigt bra ut!
- Text i username/password, konstig färg.
- Minska antalet sidbyten vid kontakt -> skicka meddelande.

- Möjlighet till default bakgrund.

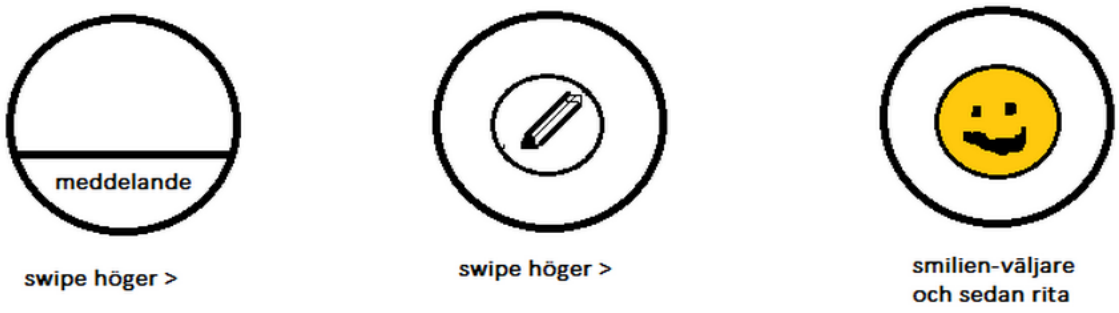
Svar från undersökning om smart klocka
Nedan följer en sammanställning av svaren på frågorna som ställdes kring navigation på den smarta klockan.

Hur smidigt var det att navigera olika sidor?

- Relativt enkelt, symbolerna var lätta att förstå. Lite oklart vad man förväntar sig när man får ett meddelande, blir det autorespond eller visas meddelandet/symbolen på helskärm?
- Ej smidigt, svårt om man inte använt såna klockor tidigare.
- För små knappar.

Vad kan/behöver förbättras?

- Möjlighet att använda en default bakgrund oavsett vem som skickar meddelandet
- Större ikoner ex vid skicka ett meddelande och tydligare riktning för swipe vid svar.
- Det var smidigare att använda klockan eftersom det är färre sidor att jobba med.
- Lite otydligt hur man skulle göra i början, men det kanske bara var för att jag aldrig använt smart watch innan.
- Actions med höger swipe, t ex enligt bilden:



Figur B.5: Skiss från testare.

- Större ikoner och göra lättare att förstå hur nya element läggs till på vyn. Kanske en välkomsttext första gången som pekar på ikonerna.
- Jag tror att texten är redundant när vi har smiley och pennikonerna och tar onödig plats. (vad gör den lila ikonen???). Sen är de små, kanske ska de försvinna när man ritar (som snapchat).

C. Bilaga - Användartest 2

Nedan följer resultaten från användartest 2.

Scenario A (skapa konto/lägga till kontakt)
Nedan följer svaren från scenario A.

Hur självsäker kände du dig?

Nivå	Antal
1	0
2	0
3	0
4	0
5	6

Tabell C.1: Självsäkerhet under Scenario A

Varför?

- Smidigt, logisk layout
- Enkelt att navigera
- Simpelt

Scenario B (skicka uttryckssymbol)
Nedan följer svaren från scenario B.

Hur självsäker kände du dig?

Nivå	Antal
1	0
2	0
3	0
4	1
5	5

Tabell C.2: Självsäkerhet under Scenario B

Varför?

- Inga problem
- Lite för stora symboler, men annars bra

Scenario C (vänta på svar/svara via klockan)
Nedan följer svaren från scenario A.

Hur självsäker kände du dig?

Nivå	Antal
1	0
2	0
3	0
4	0
5	6

Tabell C.3: Självsäkerhet under Scenario C

Varför?

- Notifikationer borde dyka upp på skärmen som sms
- Enkelt att svara, endast en klick
- Bra val av ikoner
- Det var tydligt

Övrigt

Färger

- Appen har en enhetlig stil med färger för knapparna.
- Klara och tydliga färger, ikonerna borde följa färgschemat bättre och vara enklare (mindre detaljer, som klockans emoji)
- Lagom tydligt med färgerna.
- Bra
- Ja, de var tillräcklgit starka och tydligt.
- Ful färg haha.

Font

- Bra
- Störde mig inte alls
- Stilren

Storlek

- Inga problem
- Det var OK
- Lagom stort

Layout

- Det är naturligt. Kanske för stora meddelande-bilder i konversationen som gör det jobbigare att scrolla och få en överblick.
- Känns bra i allmänhet
- Naturligt

Svar från undersökningar av smart telefon
Nedan följer en sammanställning av svaren om den smarta telefonen.

Hur var det att använda två skärmar?

- Inte så van vid det, men det funkade bra
- Det var kul!
- Inga problem
- Ej märkvärdigt, tycker det är lätt

Hur smidigt var det att navigera olika sidor?

- Smidigt

Vad kan/behöver förbättras?

- Onlinestatus vid skicka till kontakt. Alternativt mer info i kontaktlistan, t.ex. att kunna skicka meddelande därifrån?
- Starta konversation genom att klicka på kontakt i kontaktlistan, hade vart kul att kunna rita på ett skickat meddelande (smiley eller ritning) och skicka tillbaka, kunna vrida på skärmen när tangentbord används, Lägg till kontakt-alternativ i menyn i "Kontaktvyn", knapp för att lägga till kontakt "konversationsvyn", gruppkonversation vore trevligt, kunna lämna konversation, kunna ta bort enskilt meddelande ur historik
- Kanske så att man kan skriva till en kontakt direkt efter man har lagt till en kontakt?? Inget som jag kommer på. Kanske mindre smileys?
- Jag vill ha en skicka-knapp när man ritar. Letade ett tag men förstod sen att den skickades automatiskt.
- Inget speciellt, det gick att utföra sina uppgifter enkelt.

Svar från undersökningar av den smarta klockan
Nedan följer en sammanställning av svaren om den smarta klockan.

Hur smidigt var det att navigera olika sidor?

- Smidigt!
- Inget speciellt här heller :p det var smidigt Inte van vid klockan så förstod inte hur man går bakåt osv, men det är ju inte appens fel.

- Som appen, eller till och med smidigare!

Vad kan/behöver förbättras?

- Inget speciellt, det var fina ikoner!
- Man lärde sig från appen att ritningen skickas automatiskt, så det är ju bra att de följer samma mönster.

D. Bilaga - Användarscenarion

Nedan följer de olika scenarion vilka användartesterna baserats på. **Scenario A:**

1. Starta Sketchagram.
2. Skapa användare.
3. Lägga till Anna123.

"Du har precis skaffat en smart watch. Det första du gör är att installera Sketchagram. För att börja kunna använda den måste du skapa ett konto och lägga till en vän. Lägg till din bästa vän **Anna123."**

Scenario B:

1. Skicka en uttryckssymbol till Anna123.

"Nu när du lagt till din bästa vän, vill du börja kommunicera med henne. Detta vill du göra genom att skicka en glad smilies från mobilen."

Scenario C:

1. Kolla aviseringarna.
2. Starta klockan.
3. Svara med att rita en tumme-upp till Anna123.

Du har nu fått svar från din bästa vän. Kolla vad hon skrev och svara med att rita en tumme upp och skicka det till henne."

Scenario D:

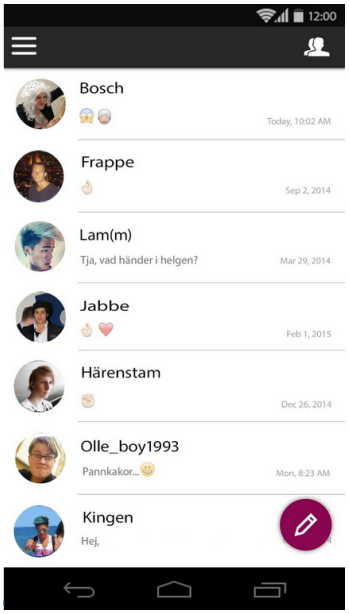
1. In på inställningar.
2. Ändra profilbild och favoritmarkera två smiliesar.
3. Skicka smiliesen till Anna123.

"Du har använt Sketchagram ett tag nu. Du vill favoritmarkera smiliesen med solbrillor och smiliesen som pussar. Skicka en puss-mun till **Anna123."**

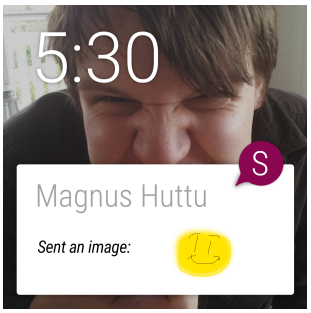
E. Bilaga - Utkast första design

Designutkast för den smarta telefonen

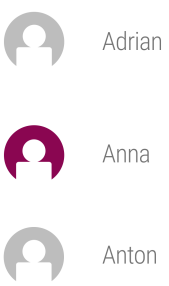
Nedan presenteras bilder för designen för prototyp 1.



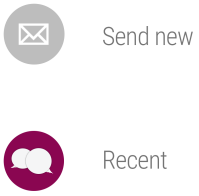
Figur E.1: Vy för överblick av aktiva konversationer. Designutkast för den smarta klockan Nedan presenteras bilder på designen för första prototypen.



Figur E.2: Vy för inkommen notifikation från kontakten Magnus Huttu.



Figur E.3: Vy för val av kontakt.



Figur E.4: Flervals vy, vilket är den första användaren möter.