Linköpings universitet | Insitutionen för teknik och naturvetenskap
Kandidatarbete | Medieteknik
Vårterminen 2023 | LiU-ITN-TEK-G–23/007–SE

Virtual Guide: Analyzing the Implementation and Applications of Virtual Assistants in Exhibition Settings

Filip Malm-Bägén
Erik Dahlström
Filip Hamrelius
Viktor Larsson
Dasmit Sethi
Jim Wahlström

Examinator: Daniel Jönsson

LINKÖPINGS UNIVERSITET

Linköpings universitet
SE-581 83 Linköping
013–28 10 00, www.liu.se

Sammanfattning

Denna rapport utreder interaktionen mellan dator och människa i en utställningsmiljö. Utredningen har genomförts genom att utveckla en virtuell assistent. I sin tur har det använts en fokusgrupp med användartester för att bedöma implementeringen. Systemet Virtuell Guide är en helhetslösning för besökare på en utställning, med syfte att ersätta mänskliga guider inom fakta, frågor och andra interaktioner mellan besökare och utställningen. Systemet är interaktivt, dynamiskt och erbjuder användaren muntlig och visuell återkoppling till frågor eller påståenden angående utställningen i fråga. Utredningen hanterar grafisk implementation, språk, röster och olika faktorer som kan påverka användarupplevelsen som bland annat fördröjning av svar från assistenten.

I det stora hela utvärderades användarupplevelsen och i sin tur hur ett system av denna natur skulle implementeras och hanteras. Projektgruppen utredde och beprövade olika modeller inom språk, samtal och fakta samt diskuterade och undersökte grafiska alternativ för användarupplevelsen. För att möjliggöra utredningen genomfördes projektet i en grupp om sex personer med hjälp av det agila ramverket Scrum. Samtliga i gruppen agerade utvecklare och två individer hade ansvaret som Scrum-master respektive Produktägare.

I slutändan implementerades två olika system. Det första systemet grundades på Furhat och utvecklades med fokus på den virtuella guiden och därav minimal funktionalitet i gränssnittet. Det andra systemet utvecklades med ett mer invecklat gränssnitt inkluderande knappar, chatt och bildanalys, med hjälp av Python och PyQt6.

Med de färdigkonstruerade systemen genomfördes ett användartest med hjälp av ett flertal olika situationer som en fokusgrupp fick utvärdera och i sin tur svara på ett antal frågor. Genom användartester har gruppen utrett ett flertal olika faktorer och parametrar tillhörande system av denna natur. Användartesterna inkluderade fördröjning, grafisk implementation, språk, röst och språkliga formuleringar.

Resultatet visade att representationer av interaktiva system med ett mänskligt utseende är en svår process som bör undvikas om projektgruppen saknar erfarenhet, tid eller resurser. Trots detta visade resultatet att virtuella assistenter var fullt möjligt att utveckla utan tidigare erfarenheter med hjälp av färdiga modeller, tekniker och hjälpmedel. Det ska dock bemärkas att en tydlig systemarkitektur är viktig för att undvika kompatibilitets problem som ofta förekommer när system använder ett flertal färdiga modeller.

Innehåll

Figurer

Kapitel 1

Introduktion

Rapporten presenterar en detaljerad genomgång av undersökningen kring virtuella assistenter med avseende på användarupplevelsen och tillhörande faktorer. Den presenterar även hur olika bibliotek och externa lösningar kan påverka utvecklingen av komponentbaserade system. I samband med undersökningen presenteras implementationen av systemen som möjligjorde utredningen. Det här kapitlet presenterar syfte, bakgrund och avgränsningar, men även de frågeställningar projektet har formats efter och som i sin tur har skapat den utredning som gruppen och projektet har genomfört.

1.1 Bakgrund

Projektet som beskrivs har en specifik inriktning mot att sammanfoga befintliga metoder och tekniker inom artificiell intelligens och maskininlärning. Det är viktigt att poängtera att projektet inte syftar till att utveckla helt nya metoder eller tekniker, utan istället dra nytta av befintliga processer som redan har visat sig vara effektiva. Detta kan effektivisera implementeringsprocessen och i sin tur möjliggöra utredningen. Detta sparar tid och resurser som annars skulle ha behövts för att utveckla nya modeller eller liknande från grunden. Därför kan mer fokus och tid ägnas åt de specifika frågor och utredningar som projektet syftar till att utforska.

En annan fördelaktig aspekt med projektet är att det bygger på välbeprövade tekniker och metoder som har visat sig vara både tillförlitliga och effektiva. Detta minskar risken för potentiella problem eller buggar som kan uppstå vid utvecklandet av nya modeller. Projektet har emellertid inte som syfte att utveckla nya metoder inom artificiell intelligens eller maskininlärning från grunden. Den fokuserar istället på utredningen mellan dator och människa. Systemet är således metoden för att utreda frågeställningarna.

1.2 Syfte

Projektet ska undersöka interaktionen mellan dator och människa med avseende på utställningsmiljöer. En del i denna undersökning innefattar hur olika färdigimplementerade bibliotek, metoder och algoritmer kan sammankopplas för att bilda ett totalt system och helhetslösning för ett nytt applikationsområde. Frågeställningarna riktar sig mot interaktionen mellan dator och människa och hur den på bästa sätt kan anpassas till en utställningsmiljö. Mer specifikt ska det utredas vilken grafisk representation som är bäst lämpad för denna typ av system, även hur standarder och bibliotek påverkar system som använder ett flertal olika externa färdigimplementerade lösningar. Systemet har därför i syfte att förverkliga denna interaktion för att i sin tur utredas.

Sammanfattningsvis är syftet med projektet att undersöka interaktionen mellan dator och människa i en utställningsmiljö och således utveckla en helhetslösning för en virtuell assistent genom att sammanfoga olika färdigimplementerade bibliotek, metoder och algoritmer.

1.3 Frågeställningar

- Vilken grafisk representation är bäst lämpad för att representera ett system för interaktion mellan en mänsklig användare och dator i en utställningsmiljö? Hur påverkar den grafiska representationen användarupplevelsen och känslan systemet förmedlar?
- Hur kan färdiga modeller inom bildanalys, textanalys, talanalys och diskussionsmodeller användas för att skapa en interaktiv virtuell assistent, och hur påverkar skillnaden mellan lokalt processade modeller och externt baserade modeller implementeringen och utvecklingsmöjligheterna?
- Hur kan standarder och externa bibliotek påverka eller hämma arbetet och implementationen av ett system starkt beroende på färdiga lösningar och tidigare implementerade funktioner och modeller? Hur påverkar en anpassad systemarkitektur och planering produktens utveckling ur detta perspektiv?

1.4 Avgränsningar

Systemet avser endast kommunikation på engelska.

Systemet tar inte hänsyn till kontexten utan det ska användaren göra. Den virtuella guiden är inte utvecklad för att svara på irrelevanta frågor eller påståenden.

Systemet kräver att klienten där den körs på har en mikrofon, kamera och skärm.

Kapitel 2

Relaterat arbete

Det här kapitlet diskuterar och utvärderar tidigare arbeten i linje med den undersökning som genomförts. Kapitlet går igenom rapporter som faller i linje för denna studie. Kapitlet går även igenom de viktigaste delarna för att utveckla en virtuell guide så som hur AI representeras grafiskt, vilka talmodeller som används och de chattbotar som vanligtvis används inom området. Den diskuterar dessutom specifika bibliotek, tekniker och modeller som är relevanta för detta projekt. Kapitlet ska ge läsaren en bakgrund för kommande kapitel med bättre förståelse till de val som har gjorts.

2.1 Grafisk representation av AI

Konceptet med en virtuell guide är inte nytt, utan det kan hittas på ett flertal andra ställen men paketerat annorlunda. Det är viktigt att titta på liknande lösningar för virtuella assistenter för att utredningen ska bli så välgrundad och gedigen som möjligt. Några exempel på kommersiella chattbotar är Amazons Alexa eller Apples Siri.

Dessa system har tagit ett direkt avstånd från att representera interaktionen med en person, trots att interaktionen är utvecklad för att agera som mänsklig. Att gå vägen och representera system med en mänsklig representation kan göras på ett flertal sätt. Harvard Buisness review menar i deras artikel *AI with a human face* att det finns fyra alternativ och tillvägagångsätt för detta: Virtual assistant, Virtual Companion, Virtual agent och Virtual influencer [1]. Projektet i fråga faller under Virtual assistant, eftersom systemet riktar sig mot konversationer och mer specifik hjälp inom ett givet område. Harvard buissness review menar att ett mänskligt ansikte i dessa sammanhang kan ge goda fördelar och tar som exempel Digital Domains, Zoey, som är en komplett chattbot med mänskligt utseende. Systemen i fråga har dock tagit år att utveckla och som användar fundamentalt avancerade koncept och ofta är patenterade [1].

En studie från *Association for Information Systems* genomförde en kvalitativ studie baserad på 41 intervjuer som fokuserar på användarens uppfattning om AI baserade chattbotar. chattbotarna i fråga var representerade med ett mänskligt utseende. Resultaten visar att en del användare uttryckte osäkerhet i om den aktuella chattboten var mänsklig eller inte. När fokusgruppen var under uppfattningen att de interagerade med en människa ansåg de interaktionen vara äkta, värdefull och pålitlig. Om de var medvetna om att det var en AI-modell uppfattade de interaktionen som opålitligt och värderade den sämre. Användarna föredrog interaktionen men en människa framför den klart tydliga AI-modellen. Baserat på dessa resultat föreslår studien att äkthet är en avgörande faktor för användares acceptans av AI-baserade chattbotar [2].

Implementationer för att representera AI-lösningar grafiskt varierar mycket. Det är tydligt att det finns fördelar med en mänsklig representation om den kan göras korrekt och på ett representabelt sätt. Dock

är det många företag med AI-lösningar använder alternativa representationer som till exempel figurer, abstrakta former eller andra animationer. Dessa alternativa representationer kan skapas i flera olika verktyg. Därför har gruppen valt att utveckla olika grafiska representationer i Furhat, Blender och Omniverse. Målet är att konstruera bästa möjliga grafiska representation som är bäst lämpad för att representera ett system för interaktion mellan en mänsklig användare och dator i en utställningsmiljö.

2.1.1 Furhat

Furhat är en social robot [3]. Det innebär att den är utvecklad för att förmedla konversationer mellan människor och datorer. Robotens yttre består av ett föränderligt människohuvud som är kapabel till att visa komplicerade ansiktsuttryck. Dess inre, alltså mjukvaran, består av färdigutvecklade AI-modeller menade för konversationer med människor. De modeller roboten har tillgängligt är för språkigenkänning, talsyntes (text-till-tal), ansiktsigenkänning och animerad munrörelse utefter tal. Med dessa modeller och ett mänskligt yttre kan systemet efterlikna mänskliga konversationer med kontinuerlig ögonkontakt och med känslor vilket kan vara lämpligt inom applikationsområden som terapi, underhållning och studier .

Utveckling mot roboten sker genom dess *Software Development Kit* (SDK) och utförs på tre olika vis. Det första sättet är via det grafiska gränssnittet *Blockly* och de två andra sätten är genom *Kotlin Skill API* och *Remote API*. Blockly uppnår snabbutvecklade program med begränsad funktionalitet. Remote API använder begränsade funktioner men har möjligheten programmera på ett av 50-tal olika programmeringsspråk, varav Python är inkluderat. Kotlin Skill API är det sätt man väljer om man vill uppnå komplicerade kommunikationsflöden i kombination med gestikuleringar, men man måste programmera i det Java-liknande programmeringsspråket Kotlin.

2.1.2 Blender

Blender är en gratis 3D programvara som används för att animera visuella modeller och special effekter. Verktyget gör det möjligt att skapa olika 2D- eller 3D-animationer utifrån utvecklarnas egna kreativitet. Blender är primärt känt för dess förmågor inom 3D-animering, medan 2D-animeringen är något mer begränsade i vad för resultat som kan produceras. Gruppen kommer stegvis att utforska båda modellerna.

2.1.3 Omniverse

Omniverse är en plattform för realtids 3D-grafik som enkelt kan implementeras med andra 3D-animeringsprogram som till exempel Blender, 3DS Max eller Maya. Med Omniverse kan användare även jobba samtidigt i en virtuell miljö, vilket gör det till en bra lösning för ett team som arbetar på ett projekt samtidigt. Det är också ett bra presterande plattform som kan snabbt och effektivt hantera och rendera komplexa 3D-modeller.

Omniverse har också ett användarvänligt gränssnitt, som gör det enkelt för användare att skapa och animera 3D-modeller och animeringar för olika projekt [4].

Audio2Face är en funktion i Omniverse som gör det möjligt för en användare att enkelt, med hjälp av en AI, skapa realistiska ansiktsanimeringar och läppsynk direkt från en ljudkälla. Detta betyder att utvecklare inte behöver göra någon egen animering, utan enbart en fil för 3D-modellen [5].

2.2 Talmodeller

Talmodeller är viktiga komponenter i många olika program, exempelvis Google Home [6], Siri [7] och Alexa [8]. En välutvecklad talmodell kan skapa en mer positiv upplevelse för användaren genom att kunna förstå och återge tal på ett mer naturligt sätt. För att skapa en effektiv talmodell krävs vanligtvis stora mängder data för att modellen ska kunna förstå och lära sig olika röster, uttal och språkmönster.

2.2.1 Tal-till-text

Taligenkänning och textgenerering från ljud är ett växande område inom artificiell intelligens, och det finns många färdiga modeller tillgängliga för python. En sådan modell är *Huggingface Wav2Vec2* som är enkel att implementera och ger hög prestanda[9]. Wav2Vec2-modellen är speciellt utvecklad för att konvertera tal till text och är därför särskilt lämplig för denna typ av projekt.

En av fördelarna med Wav2Vec2-modellen är dess självövervakningsteknik, vilket innebär att modellen kan träna sig själv utan extern data. Detta gör att implementeringen kan vara enklare och mer effektiv eftersom mindre data behöver hanteras. Dessutom kan modellen tränas för specifika uppgifter, såsom att hantera specifika ljudförhållanden och identifiera specifika personers röster [9]. Viktigt att notera är att Wav2Vec2-modellen endast är ett exempel på en modell som kan användas för taligenkänning och textgenerering från ljud. Det finns många andra modeller som även kan vara lämpliga för detta ändamål.

Ett ytterligare alternativ för att konvertera tal till text är den relativt nylanserade modellen från OpenAI, *Whisper*. Modellen lanserades i september 2022 och är en *sekvens till sekvens* modell med andra ord en *Transformer baserad encoder-decoder modell*, alltså en modell som iterativt kartlägger lager för lager av indatan för att tillslut konvertera den muntliga datan till en text fil. Whisper är tränad på 680 000 timmar röstinspelning och är en av de bättre modellerna på marknaden, enligt Open-AI [10].

2.2.2 Text-till-tal

Text-till-tal-modeller är en viktig teknologi som används för att skapa realistiska röster från text. Det finns flera olika modeller att välja mellan, inklusive text-Till-tal från *Huggingface* [11] och *gTTS* [12]. Huggingfaces modell är komplicerad att implementera och är inte lika utvecklad som andra modeller på marknaden [11]. gTTS är en enklare modell att implementera och som använder Google översätts text-till-tal API och stödjer då många olika språk [12].

För att skapa en mer engagerande upplevelse med en virtuell guide kan det vara fördelaktigt att använda en röst som låter som en naturlig människa. *Resemble AI* är en modell som använder AI för att i samma hastighet som de tidigare nämnda modellerna kunna skapa sådana röster från text. Modellen tränas med hjälp av data från röstinspelningar för att kunna skapa en en röst som låter en naturlig människa [13]. Resemble AI gör det även möjligt att skapa egna röster, som låter som sig själv, genom att använda egen data från röstinspelningar.

2.3 Diskussionsmodeller och chattbotar

Följande kapitel beskriver olika diskussionsmodeller och chattbotar, inklusive deras användning inom företag och utställningsmiljöer. Rapporten tittar närmare på samtalsbaserade system och deras tidiga utveckling, samt moderna tillämpningar som kombinerar talsyntes, chattbotar och virtuella avatarer.

2.3.1 Samtalsbaserade system

Samtalsbaserade system med förmågan för dialog har funnits länge i olika former. En av de tidigaste av denna typ av system är den virtuella psykoterapeuten *ELIZA*, som var utvecklad av Joseph Weizenbaum i 1966 [14]. Den är idag känd som en av de första typen av ”chattbot”, som är en term för programvara med förmågan för mänsklig konversation. Även om ELIZA var regelbaserad, det vill säga att den genererade text baserad på en algoritm, och att Weizenbaum förväntade att dialog med ELIZA skulle vara ytliga, var den i stället effektiv då den lyckades få patienter att framkalla känslomässiga reaktioner och fick dem känna att de pratade med en intelligent robot [14].

2.3.2 Utställningsmiljö och chattbot

Användning av en chattbot inom företag har potentialen att förbättra kundupplevelsen och minska kostnader för företaget [15]. Samma gäller för utställningsmiljöer där en hjälpsam chattbot kan ta emot frågor och generera relevanta svar för utställningen likt en guide [16]. Med den senaste utvecklingen inom språkteknologi och AI finns potential för mer avancerade samtalsbaserade AI-chattbot inom utställningsmiljöer, där många framgångar har gjorts under den senaste tiden. Ett exempel är projektet *Tinker* som är en virtuell guide utvecklad av Duguleana et. al. [17] som kombinerar talsyntes, chattbot och virtuell avatar för att skapa en intelligent samtalsbaserad virtuell guide som interagerar med besökare. Resultatet av projektet utfördes i form av användartester där användare fick ge åsikt om deras upplevelse med Tinker vilket gav ett positivt resultat [17]. En mer AI-baserad tillämpning av chattbot är *IRIS+* [18] projektet för museet ”Museu do Amanhã” i Rio de Janeiro, Brazil vilket använder IBMs Watson [19]. IRIS+ är en interaktiv implementation av ett frågebesvarande system där chattboten introducerar en utställning och avslutar med frågan *”After everything you learned in the main exhibition, what are your concerns in today’s world?”*, som ett sätt att engagera sig i konversation med besökare [18].

Kapitel 3

Beskrivning av metoden

I det här kapitlet beskrivs den metod som användes för att besvara frågeställningarna. Kapitlet diskuterar källkritik men ger framförallt en utförlig genomgång av de system som utvecklats för att utreda ärendet och i sin tur besvara frågeställningarna.

3.1 Källkritik

Valet av källor har noga övervägts där majoriteten av dem har hämtats från *Google Scholar*. Att hämta källor från Google Scholar är ingen garanti på trovärdighet, men det gallrar bort de generiska faktakällor och artiklar som annars skulle använts genom att endast använda sig av publikationer av olika grader. På detta sätt att gruppen totalt undvikt källor som Wikipedia. I projektet har gruppen alltså valt att förlita sig på mer specialiserade och vetenskapligt granskade källor som Google Scholar erbjuder. Utöver det har vissa mer vardagliga artiklar använts. Dessa har dock valts med omsorg från trovärdiga källor med författare specificerade inom det givna området. För de mer tekniska källorna har dokumentation eller utgivarens hemsida använts för att hämta fakta och referera till.

3.2 Implementering av system

Följande avsnitt beskriver implementationen av de systemen som användes för utredningen i projektet. Kapitlet omfattar talmodeller, diskussionsmodeller och chattbotar, Blender för 3D-modellering, bildanalys och grafisk representation, desktopprogram, samt Furhat. Kapitlet ger en bakgrund till kommande diskussion och lägger grunden för användartesterna.

3.2.1 Talmodeller

Den slutgiltiga implementationen av tal-till-text och text-till-tal använder sig av olika lösningar och modeller för att uppnå det resultatet som önskats. Den virtuella guiden har utvecklats för att starta genom att användaren säger en förutbestämd fras. Denna fras kan anpassas för att passa den miljö systemet är placerad i.

Tal-till-text
Tal-till-text modellen använder sig av Whisper som är en modell för tal-transkribering. Modellen är tränad på engelska. När modellen är laddad, spelas ljudet in fyra sekunder och transkriberas sedan genom att jämföra det inspelade ljudet med modellen för att producera en textsträng. Gruppen valde att spela in ljudet i fyra sekunder då det märkes att det var en lämplig tidslängd för användaren att tala in sin fråga. Genom att ladda modellen i samband med att applikationen laddas kan systemet optimeras avservärt.

Text-till-tal
Text-till-tal modellen använder *Tacotron2*. Tacotron2 fungerar genom att ta in en text som inmatning och konverterar sedan den till en ljudfil i WAV-format. Denna ljudfil kan sedan spelas upp för användaren. Genom att ladda modellen i samband med att applikationen laddas kan systemet optimeras avsevärt.

Tacotron2 är en maskininlärningsmodell som bygger på neurala nätverk för att generera tal från text. Modellen tränas på stora mängder röstdatabaser för att kunna generera naturligt klingande tal. När Tacotron2 får en inmatningstext bearbetar den texten genom flera neurala nätverkslager för att generera en sekvens av spektrogram. Därefter används *WaveNet* för att generera ljudvågen från spektrogrammet [20]. WaveNet använder en generativ modell för att skapa en sekvens av ljudvågor som låter naturliga. Slutligen spelas ljudfilen upp för användaren.

Röstaktivering
I programmet användes även biblioteket *SpeechRecognition* för att identifiera när en användare säger hälsningsfrasen för att starta en konversation med chatbotten. SpeechRecognition användes över Whisper för att SpeechRecogntion är snabbare, men inte lika noggrann. Det tillåter hälsningsfrasen att aktiveras snabbare än om Whisper hade använts. Funktionen "*RecognizeGoogle()*"i SpeechRecognition användes för att transkribera ljudsignalen till text och om biblioteket känner av att användaren säger hälsningsfrasen, startar chatbotten upp en konversation och börjar lyssna efter vilka frågor användaren vill ställa.

3.2.2 Diskussionsmodeller och chattbotar

För att undersöka lämpliga förtränade AI-modeller användes plattformen *Hugging Face* som erbjuder moderna förtränade modeller inom språkteknologi [21, 22]. För syftet av projektet behandlas fråge-

besvarande modeller, QA, som används för att extrahera relevant information från en textmängd för att svara på en fråga [23]. Det finns olika typer av frågebesvarande modeller: *Extractive* QA där systemet extraherar svaret på frågan från textmängden och *generative* QA där systemet genererar fritext baserad på textmängden [23] [24]. Modeller som *BERT* används inom *extractive* QA och modeller som *GPT* används inom *generative QA* [24]. Vidare skiljer frågebesvarande system sig baserade på hur svaret hämtas från textmängden, där *open* QA genererar svar utifrån en kontext och *closed* QA genererar svar utan kontext [24].

Modeller

Flera frågebesvarande modeller av olika typer undersöktes för projektet i syftet att jämföra för- och nackdelar att de ska kunna användas i en utställningsmiljö. Bland de QA modellerna som undersöktes var av typen BERT, GPT och T5 [23] [25]. Vidare undersöktes svenska varianter av de olika modellerna där det var möjligt. De engelska varianter av modellerna valdes, eftersom modeller inom språkteknologi är mer fokuserade på engelska och då svenska varianter inte var tillgängliga.

- **BERT-base-swedish-cased**. Denna modell är av typen *extractive* QA utvecklad av KBLab för svenska. Modellen hittar relevant information i en textmängd för att hitta svaret på en fråga. Denna modell är användbart där det krävs hög prestanda med mindre resurser än de andra modellerna. Denna modell är dock begränsad då den inte genererar text vilket gör den olämplig för en chattbot [26].
- **GPT-sw3**. Denna modell är av typen *generative* QA skapad av AI Sweden för att generera text i nordiska språk och finns i olika storlekar, det vill säga parametrar, där större antal parametrar kräver mer beräkningsminne. Då det är en closed QA är det möjligt att finjustera modellen för anpassning till open QA. Då modellen är en GPT-modell och inte tillgänglig via API krävs det väldigt mycket beräkningsminne för att använda de större varianterna av modellen vilket är utanför ramen för detta projekt. Därmed användes varianten **GPT-sw3-126m** som är den minst krävande modellen [27].
- **GPT-3.5**. Denna modell är, precis som GPT-sw3, skapad av OpenAI. Då GPT-sw3 är tränad på nordiska språk är denna modell tränad på flera språk, med fokus på engelska. Skillnaden mellan GPT-3.5 och GPT-sw3 är att GPT-3.5 är tillgänglig via *OpenAI API* och kostar per användning medan GPT-sw3 är tillgänglig via Hugging Face och är därmed gratis. Endast internetuppkoppling krävs för denna modell och är den är inte begränsad av hårdvara.
- **T5-base-finetuned-question-answering**. Denna modell är en kombination av extractive och generative QA, och är därför mer flexibel [28]. Då modellen är mer komplex än BERT är denna modell mer resurskrävande. Varianten av modellen som undersöktes är finjusterat för open QA [29].

Valet av rätt modell för den virtuella guiden är avgörande för en användarvänlig och effektiv interaktion med besökare. För att undersöka chattbotens upplevelse med besökare utfördes det användartester där användarna av systemet fick berätta om deras interaktion med den virtuella guiden och om användarna föredrar en *extractive* eller *generative* typ av chattbot.

3.2.3 Grafisk implementation i Blender

Implementationen av 2D- och 3D-animeringen gick till enligt följande. Med tanke på Blenders primära förmågor inom 3D-animeringen kunde kundens initiala önskemål att avatarens ansikte och rörelsemönster gjordes en första prototyp av ansiktet i 3D. Samtliga utvecklare från gruppen har erfarenheter inom Blender men relativt tidigt i processen insåg den ansvarige utvecklaren att kundens önskan av utseende av avatarens ansikte skulle bli för svårt att konstruera utifrån den givna tidsramen. Något som ledde till att arbetsprocessen för en 3D modell avslutades och gruppen tittade på möjliga lösningar för en 2D-animering istället.

Till skillnad från 3D modellen motsvarar inte 2D-animeringen kundens ingångsmål för det visuella utseendet. Däremot kunde projektgruppen konstruera en enklare variant av en 2D-animering som alternativ. Den första versionen av 2D-modellen blev en smiley med ögon och mun. Munnen läppsynkade till ljudet av ljudfilen som spelades upp för användaren. Utöver munnen skulle bildanalysen applicera på animationens ögon för att följa användaren. Däremot implementerades detta aldrig då både läppsynkade munnen och bildanalysen av ögon på samma Blender-fil blev ogenomförbart, även denna modell avslutades. Den färdiga modellen för 2D-animeringen visas i Figur 1.

2D-animering, när smileyen inte pratar. 2D-animering, när smileyen pratar.

Figur 1: Grafiska representationer av 2D-animeringar, konstruerade i Blender.

3.2.4 Bildanalys och grafisk representation

Visuell interaktion med en produkt är en viktig faktor för användarupplevelsen och vital komponent för utredningen och i sin tur frågeställningarna. Genom att använda bildanalys och ansiktsigenkänning kan applikationer skapa en mer mänsklig interaktion med användaren. Analysen går ut på att en kamera och en dator samarbetar för att identifiera mänskliga ansikten och logga deras position. Med denna position kan sedan gränssnittet anpassas på olika sätt. Detta ger användaren ett mer familjärt intryck av systemet då den lyssnar, talar och ser på dig. Detta leder till en mer personlig och engagerande användarupplevelse.

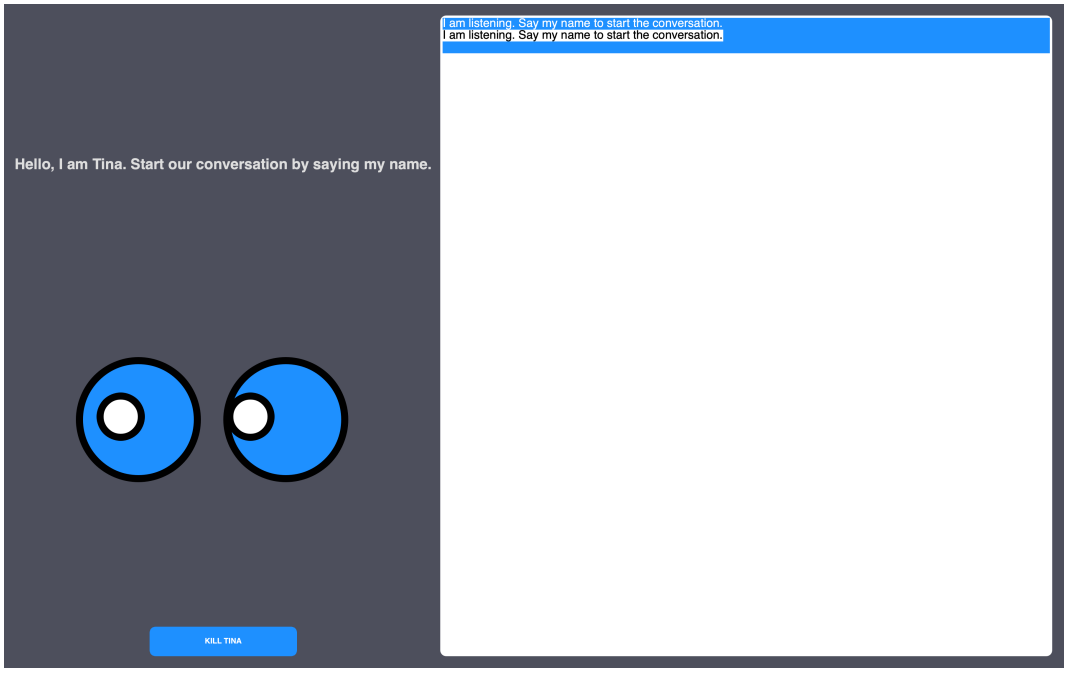
Dessutom kan denna typ av interaktion även öka användarens förtroende för produkten och skapa ett starkare känslomässigt band mellan användaren och produkten än vad applikationen hade gett om ansiktet inte fanns [30].

Utvecklingen av ett ansiktsigenkännande program kan göras med hjälp av Python och OpenCV. OpenCV är ett bibliotek med öppen källkod, biblioteket innefattar 2500 algoritmer för datorseende och bildbehandling. OpenCV har tillgång till datorseendealgoritmer för att detektera ansikten i både bild eller video [31].

Den grafiska representationen innebär två simpla ögon som först implementerades i PyGame. En pythonbaserad plattform utvecklad för implementering av multimedieappar och spel. PyGame är byggt på SDL-biblioteket somstödjer både Windows, macOS och Linux [32]. Däremot konverterades grafiska representationen till en implementering i PyQt6 istället. Med hjälp av bildanalyskomponenten följer ögonen efter användarens position. Designen är implementerad för att matcha resterande gränssnitt.

Bildanalyskomponenten innefattar en ansiktsspårning som identifierar om ett ansikte befinner sig framför kameran eller inte. Utöver detta kan ansiktsspårningen även följa efter ansiktet. Med denna metod sparas koordinater efter hur ansiktet rör sig. Dessa används i sin tur för att förflytta ögonen i gränssnittet. Om användaren inleder med att vara framför kameran men sedan rör sig utanför kamerans räckvidd kommer ögonen att kvarstå i den senaste angivna koordinaten från ansiktsspårningen. Här kommer ögonen kvarstå fram tills att användaren är inom kamerans räckvidd igen.

Den bildanalys som görs är baserad på biblioteket OpenCV, ett programvarubibliotek riktat mot datorseende och bildanalys. Biblioteket har en färdigtränad modell med ett dataset tränat på ansikten. Bildanalyskomponenten har åtkomst till kameran på den givna datorn. På varje stillbild av videosignalen används OpenCV-verktyget för att skanna den givna bilden, då för att avgöra om ett ansikte finns i bilden eller ej. Om den gör det aktiveras koordinaterna och passas till den grafiska representationen för att förflytta den.



Figur 2: Slutgiltiga desktopapplikationen utvecklad i PyQt6.

3.2.5 Integration av system och sammankoppling av komponenter

Utveckling av desktopprogram, inklusive *Graphical User Interface* (GUI), det vill säga gränssnitt, i Python är en väletablerad process och en relativt vanlig tillvägagångssätt för att utveckla generiska program för både Windows, MacOS och Linux.

Vanligtvis väljer utvecklare andra programmeringsspråk för avancerade applikationer, till exempel Java eller C++. Python kan dock föredras för projekt av mindre till medelstor skala, där slutmålet inte är kommersiellt bruk. Detta beror på att Python som programmeringsspråk är relativt enkelt med hög användarvänlighet [33]. Varför det kan vara fördelaktigt att välja något annat alternativ för större

skala eller kommersiellt bruk beror på hastigheten hos programmeringspråken, där Python är relativt långsamt då det är ett tolkat språk, inte ett kompilerat språk som exempelvis C++ [34].

Utvecklingen av program i Python kan göras direkt med *toolkits* som *TKinter*, det vill säga att inget externt bibliotek behövs användas. Tkinter är ett bra val för simpla program och testing. Utvecklingen av program av större skala med avancerade element görs inte med endast Python, utan kräver ofta externa bibliotek, liksom *Frameworks*. Där de vanligaste inkluderar någon version av *PyQt*. Biblioteket erbjuder ofta fler möjligheter men också större utbud av komponenter att inkludera i programmet, även nyare och mer moderna gränssnitt med större möjlighet till design. Det finns ett flertal välkända kommersiella program som är utvecklade i PyQt6, bland annat Dropbox och Ninja-IDE [33].

För att tillåta den grafiska utredning frågeställningarna menar på, består det slutgiltiga systemet av en desktopapplikation, och kan ses i Figur 2. Applikationen sammanfogar samtliga komponenter och binder samman det till det totala system som användaren integrerar med. Det står det också för den grafiska implementationen utöver bildanalysen, det vill säga gränssnittet. Gränssnittet består av enkel struktur som ger användaren möjlighet att följa konversationen i text och även skrolla upp och ner i konversationen för retroaktiv läsning. Utöver chattfunktionen har gränssnittet knappar som kan hjälpa till att kontrollera interaktionen eller stänga programmet. Gränssnittet och systemet i sin helhet är byggt med PyQt6 och Python 3.8.16. För att tillåta både konversationen och videoanalysen parallellt har trådning applicerats. Trådning innebär att två uppgifter kan köras parallellt i en och samma fil. De andra komponenterna körs från andra filer och kräver inte någon threading.

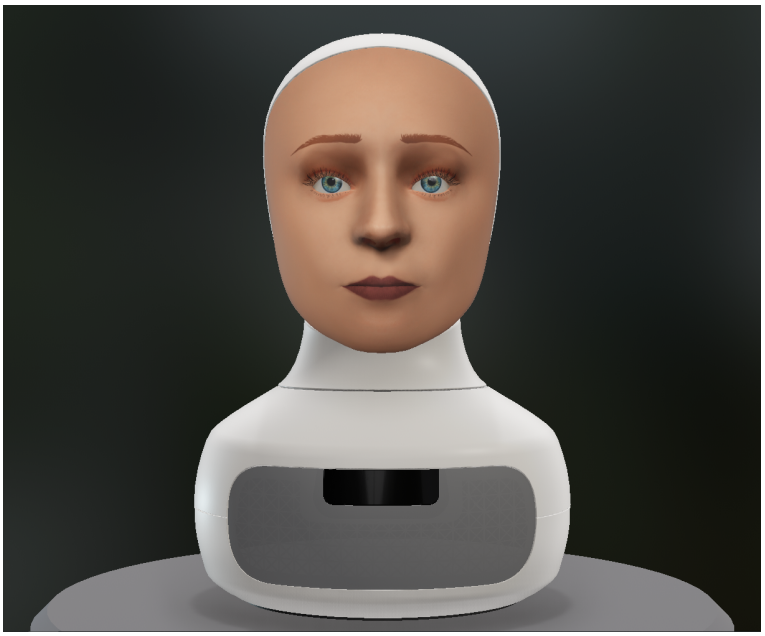
3.2.6 Furhat

Virtuella guidens mänskliga representation utvecklades med Furhat Robotics digitala avatar, Virtual Furhat, och Furhat SDK som styr och ger form till avataren. Alla komponenter som den virtuella guiden skulle bestå av, förutom chatbotten, fanns tillgängliga genom inbyggda funktioner i verktyget. Den slutgiltiga grafiska representationen av virtuella guiden ses i Figur 3 och består av ett grafiskt gränssnitt där ett människoansikte är i centrum.

Till en början undersöktes de tre olika utvecklingssätten för att se vilket som skulle passa projektet bäst. När det tillkom mer information om att AI-modellen för chattboten använde sig av Python valdes utvecklingssättet Python Remote API som grund för det underliggande systemet. Detta eftersom de andra utvecklingssätten, som beskrivs i avsnitt 2.1.1, inte programmeras på Python och därmed skulle kräva extra arbete för att ta fram en lösning för chattbot-integrationen.

Systemet bestod då av begränsade men enkla funktioner som skickar kommandon direkt till verktyget genom en lokal nätverksserver. Funktioner som beordrar verktyget till att lyssna och tolka språk genom en mikrofon, eller som beordrar den till att tala hade redan egna inbyggda gestikuleringar. Därmed krävdes det inte alltför många kommandon för att få avataren att uppföra sig som människolikt.

Det arbete som skedde därefter var att sammanföra chattboten med kommunikationsflödet. Den virtuella guiden ber användaren ställa en fråga. Därefter lyssnar verktyget och skickar det tolkade språket direkt in i modellen för chattboten. Det genererade svaret skickas därefter till Virtual Furhat som svarar tillbaka till användaren i form av tal tillsammans med gestikuleringar som höjda ögonbryn, blinkande ögon eller leende i varierande mån. Avataren ställer därefter frågan om användaren vill veta något mer, och då har användaren valet att avsluta konversationen genom att svara med ”nej”, ”nej tack”, eller ”hej då”.



Figur 3: Virtual Furhat med ansiktet "Isabel".

Kapitel 4

Resultat

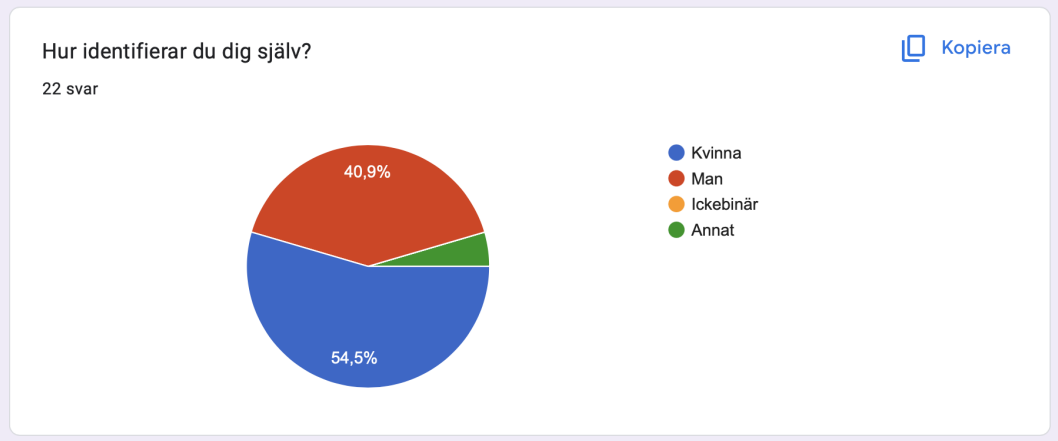
Det här kapitlet presenterar de resultat som har framtagits i projektet. Kapitlet presenterar och sammanfattar de resultat som tagits fram genom användartester tillsammans med det systemen.

4.1 Användartester

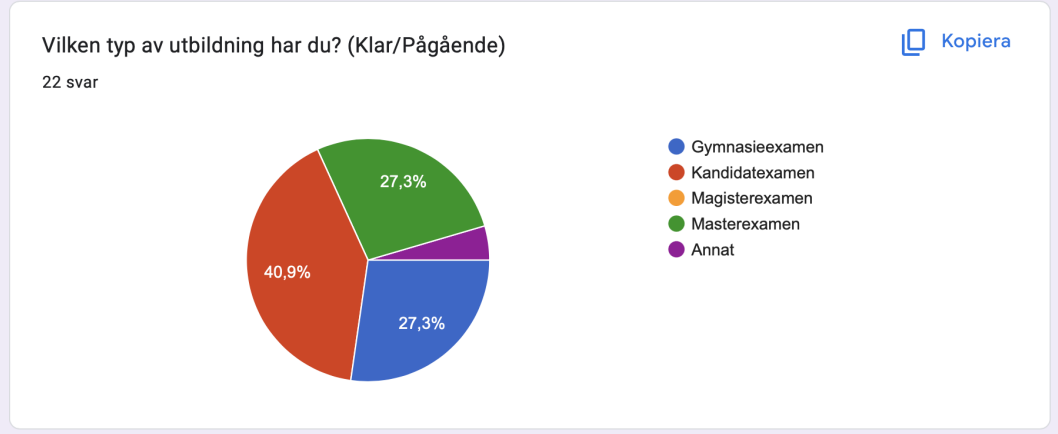
Metoden för användartester grundade sig på färdigkonstruerade situationer presenterade som ett antal videofilmer. Detta genomfördes med färdiginspelade interaktioner mellan en person och den givna virtuella guiden. Konversationen var menad att vara vardaglig och generisk då användartesterna inte skulle utvärdera konversationen utan endast utomstående parametrar. Gruppen valde att spela in konversationerna med respektive virtuell guide för att kontrollera att diskussionen och svaren var densamma mellan båda systemen.

Genom att ändra olika parametrar mellan interaktionerna fick fokusgruppen uppleva olika varianter av den virtuella guiden och utifrån upplevelsen svara på frågor. Områdena som undersöktes var fördröjning, chattbotens formuleringar, den grafiska representationen och rösten som den virtuella guiden talade med. Fokusgruppen fick svara genom ett färdigkonstruerat formulär. För att få ett statistiskt representabelt resultat har gruppen följt SCB's riktlinjer för att välja rätt metod, det vill säga statistikguiden [35]. SCB påstår att ett slumpmässigt urval är att föredra, genom att lotta fram deltagare och sedan genomföra undersökningen. Gruppen följde denna metod genom att tilldela samtliga gruppmedlemmar med uppgiften att samla svar. Metoden är ingen motsvarighet till en riktigt slumpmässig fokusgrupp, men bör ge ett godtyckligt resultat. Att använda sig av personer i sin direkta närheten var en fördel på grund av tidseffektiviteten, dock avstod gruppen från detta i all mån för att inte få en för smal fokusgrupp.

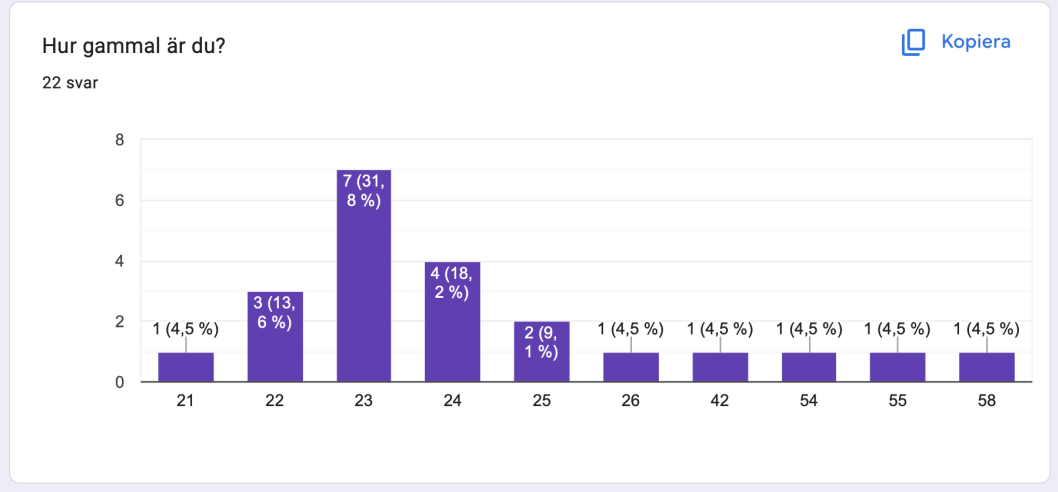
Totalt 22 personer genomförde användartestet med tillhörande enkät. I bilaga C presenteras undersökningen i sin helhet. Urvalsgruppen innefattade 40.9% män, 54,5% kvinnor och 4.5% övrigt. Ålderspannet sträckte sig mellan 21 till 58 år, med en majoritet av 23-åringar, som ses i Figur 4 och Figur 6. Majoriteten av korrespondenterna, 40.9%, besitter en kandidatexamen, vilket syns i Figur 5, och en minoritet av urvalsgruppen har som vana att gå på museer eller andra vernissagemiljöer, vilket kan läsas ur Figur 7. Majoriteten anser sig själva som tekniskt intresserade och deras användande av ChatGPT, Siri, Alexa och andra virtuella assistenter är jämt fördelat med lika många röster på alternativet ännvänder oftasom alternativet ännvänder sällan". Detaljer finns i Figur 8 respektive 9.



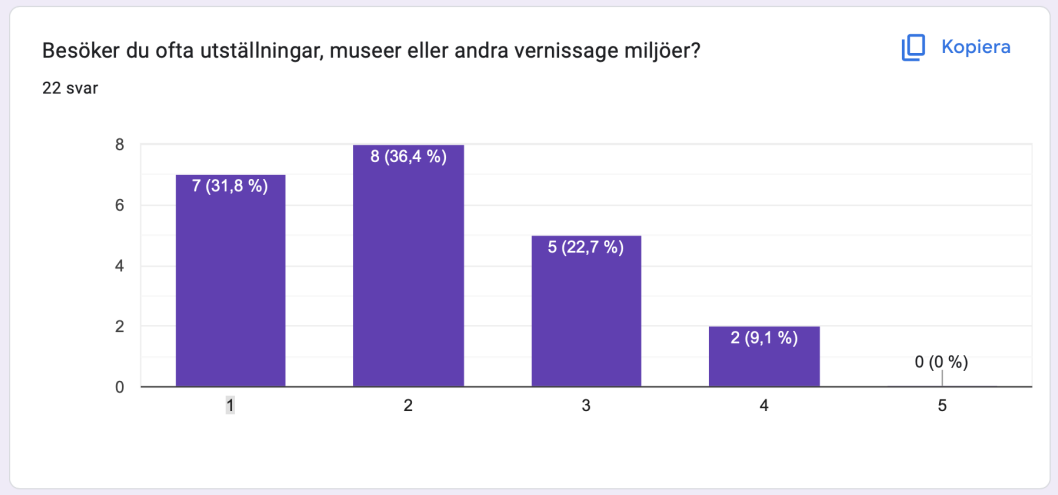
Figur 4: Könsfördelning på användartester.



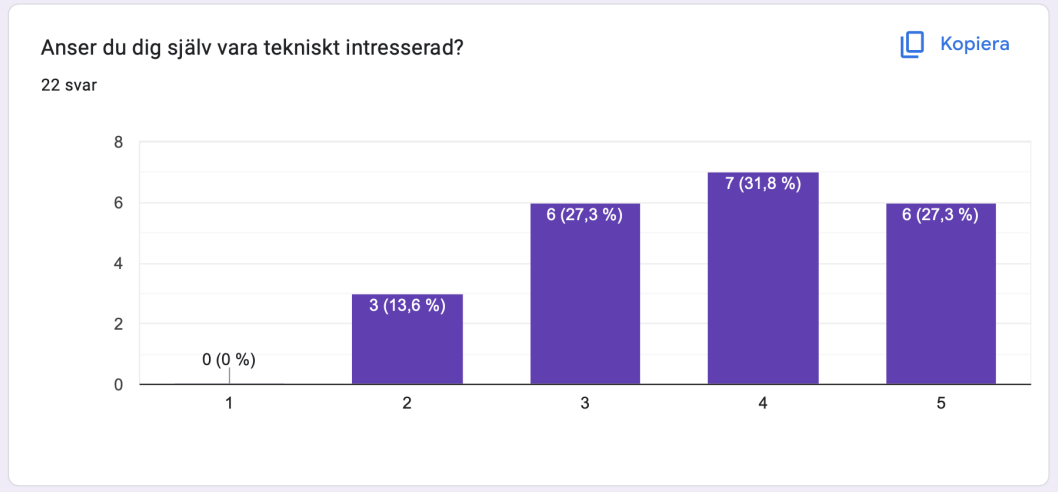
Figur 5: Utbildingsfördelning på användartester.



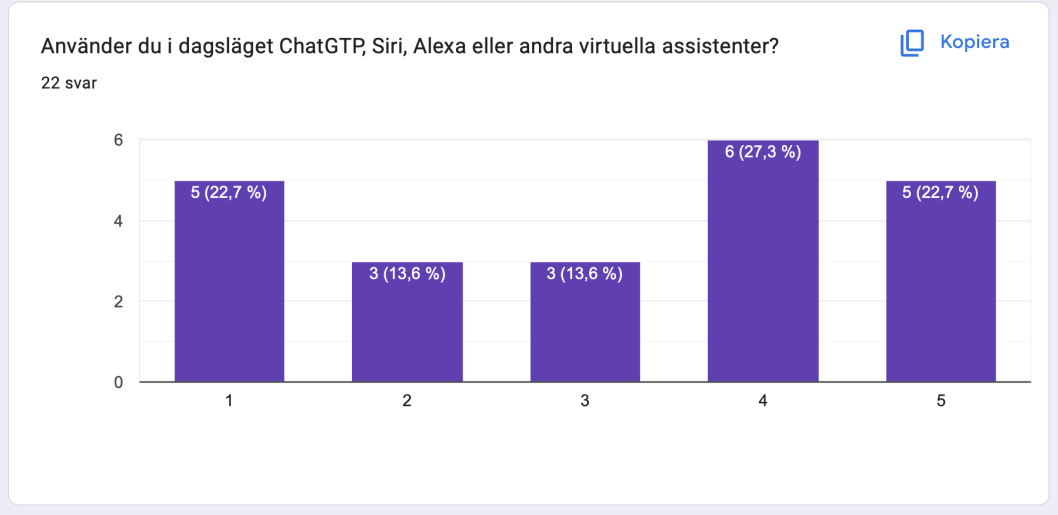
Figur 6: Åldersfördelning på användartester.



Figur 7: Intresse i utställningar hos användarna.



Figur 8: Tekniskt intresse hos användarna.



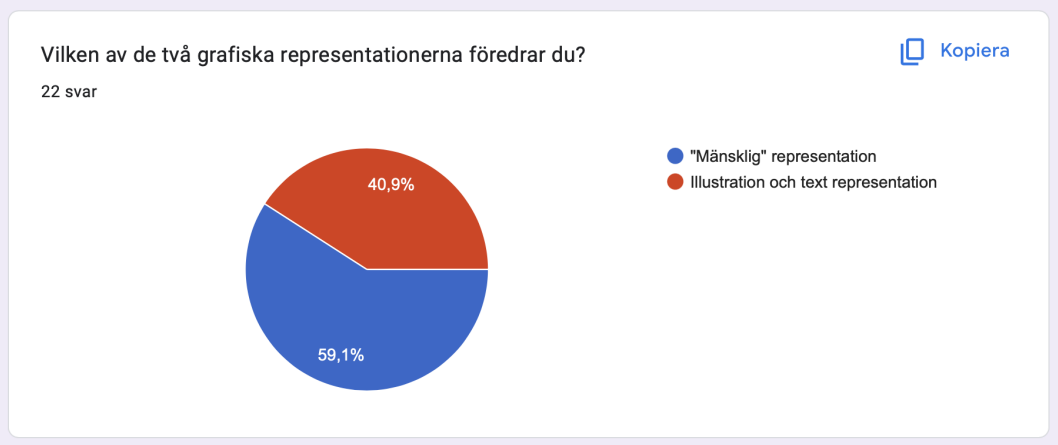
Figur 9: Användning av tekniska instrument i assistentform hos användarna.

4.2 Grafisk representation

I Figur 10 presenteras fördelningen bland respondenterna huruvida Furhat eller PyQt6-gränssnittet föredrogs. Den mänskliga representationen, och därmed Furhat, föredrogs med 59.1% av rösterna gentemot 40.9% . Motiveringarna som löd var blanda annat att den känns mer verklig, ögonen från PyQt6 var för stirriga och för att det upplevdes som en vanlig konversation med Furhat. Dessa motiveringar kan ses i Figur 11 till 13.

Från Figur 14 syns även att det är mer viktigt att konversationen loggas med text, än att texten inte skulle synas. En stor majoritet anser dessutom att det inte känns positivt att ögonen från PyQt6 följer användarens blick, vilket kan avläsas från 15. Avslutningsvis hade 50% av respondenterna föredragit en abstrakt representation, exempelvis en vågform eller liknande animation som påminner av Apples Siri eller Google Assistant. 36.4% är osäkra i frågan och 13.6% hade inte föredragit det, visar Figur 16.

56% upplevde någon typ av obehag av systemen. Fem personer ansåg PyQt6-gränssnittet som stirrigt och obehagligt och där åtta personer tyckte Furhat var obehaglig. Det vill säga 22% respektive 36%. Detaljer kan ses från Figur 17 till 20.



Figur 10: Alternativ grafisk representation.

4.3 Chattbot

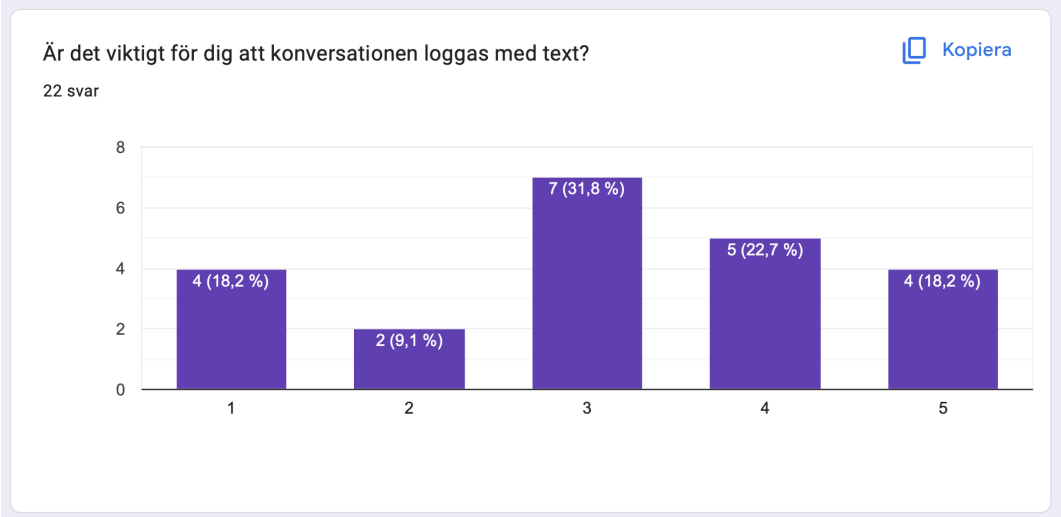
Alla modeller undersöktes och testades. Medan BERT-modellen gav hög precision i att hitta relevanta svar med lite fördröjning kan den i andra hand inte generera text som liknar en mänsklig text. Båda GPT-modellen gav mer utförliga svar och finjusterades ytterligare för ett bättre upplevelse, men är begränsade där modellen **GPT-sw3-126m** genererade olämpliga och felaktiga svar på grund av begränsad beräkningsminne och att **GPT-sw3** kostar för att använda. Detta gjorde **T5** modellen mest lämplig med tillräcklig hög precision, låg fördröjning och utförliga svar för att kunna användas i projektet.

I Figur 21 presenteras det resultatet av användartesterna där användarna av systemet fick ge sin åsikt om hur de vill att systemet ska besvara frågor. Det gavs tre alternativ med tre olika grader av utförligt svar. 54.5% föredrog ett konverserande och omfattande svar från chatbotten. 9.1% föredrog när endast svaret och inget annat gavs och 36.4% föredrog när ytterligare fakta gavs utöver svaret på den givna frågan. Sammanfattningsvis föredrog majoriteten ett mer utförlig svar på frågan som uppfylls av *generative* QA system.

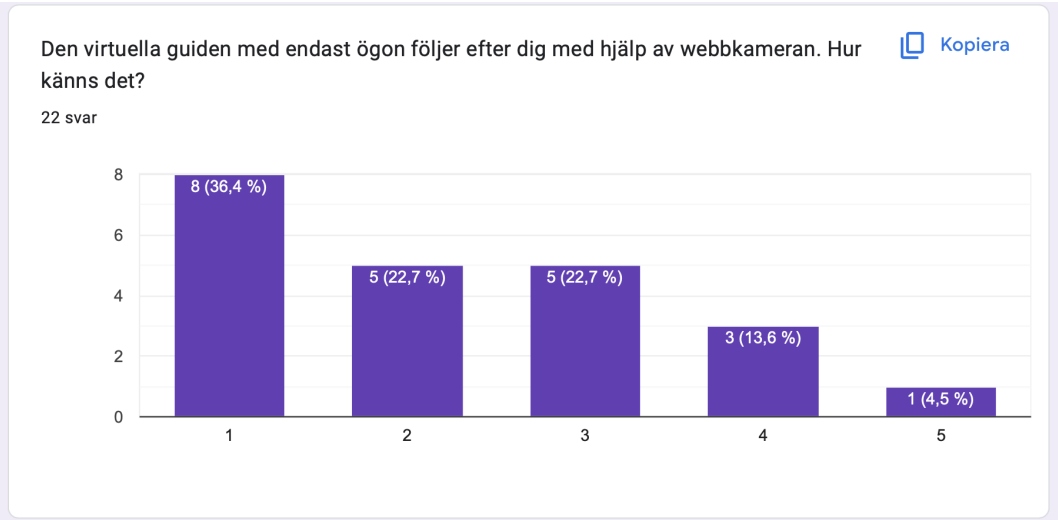
Figur 11: Fri text - del 1 - Varför föredrog användaren just denna grafiska representation?

Figur 12: Fri text - del 2 - Varför föredrog användaren just denna grafiska representation?

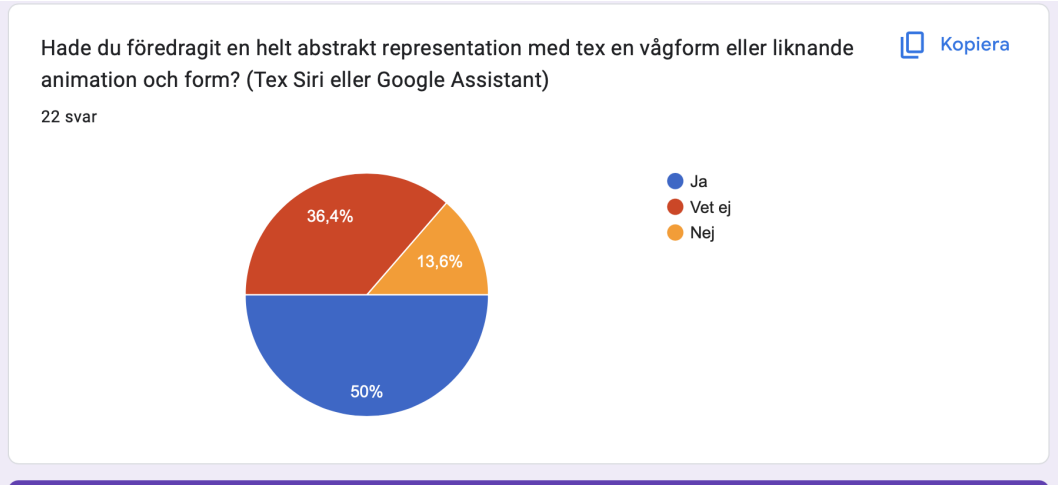
Figur 13: Fri text - del 3 - Varför föredrog användaren just denna grafiska representation?



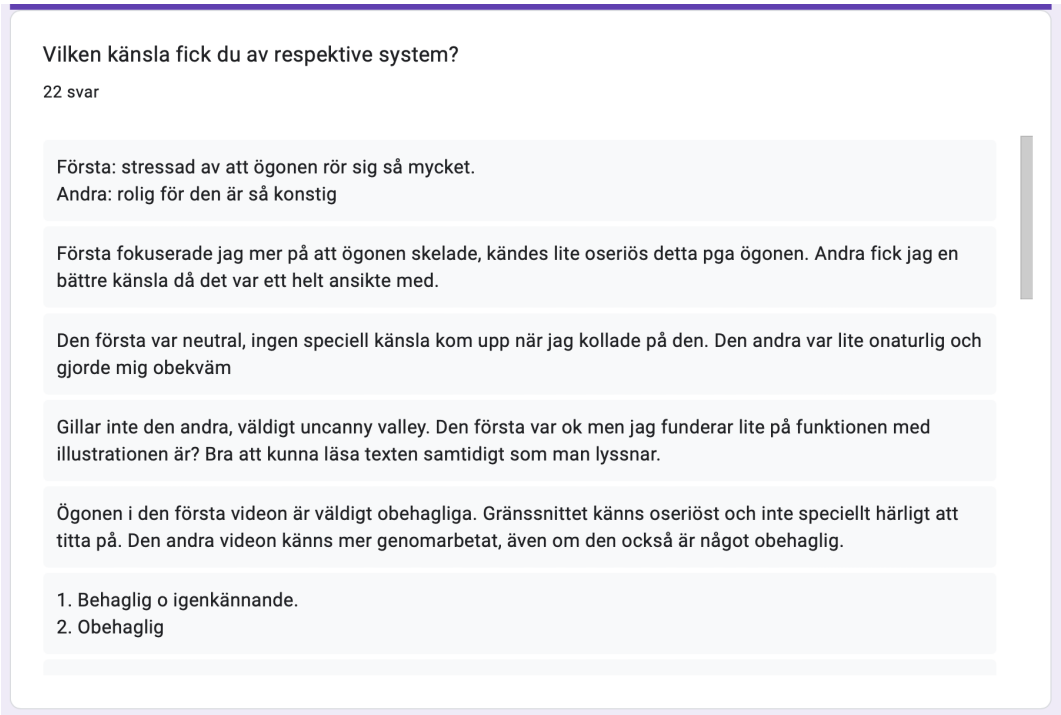
Figur 14: Är det viktigt för användaren att representera konversationen med text?



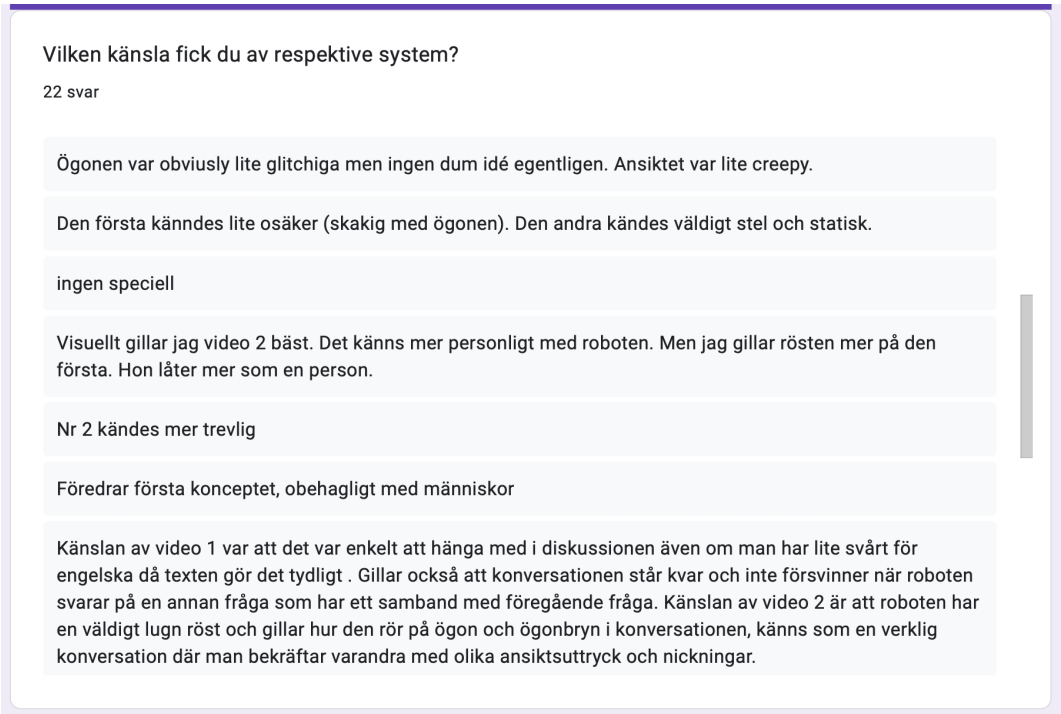
Figur 15: Att följa användaren med kamera, hur uppfattas det?



Figur 16: Alternativ grafisk representation.

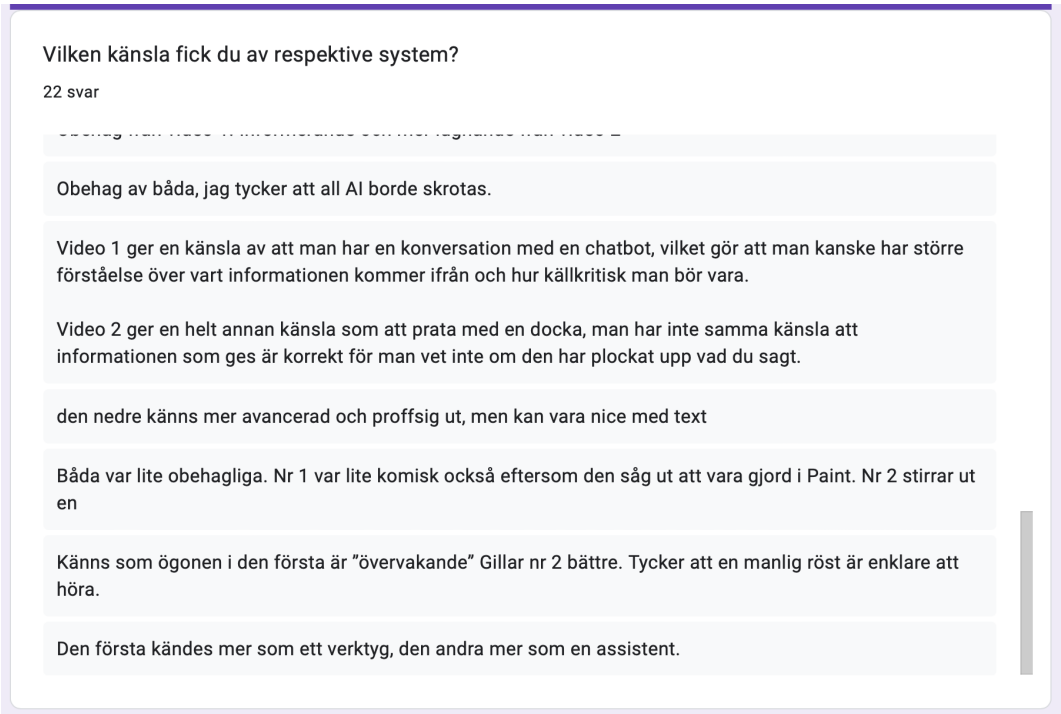


Figur 17: Fri text - del 1 - vilken känsla gav systemet användarna?



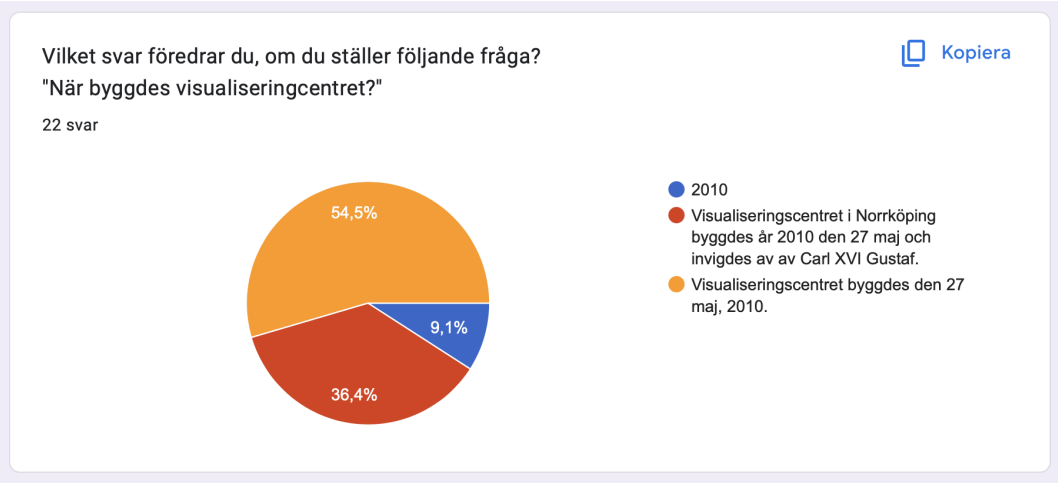
Figur 18: Fri text - del 2 - vilken känsla gav systemet användarna?

Figur 19: Fri text - del 3 - vilken känsla gav systemet användarna?

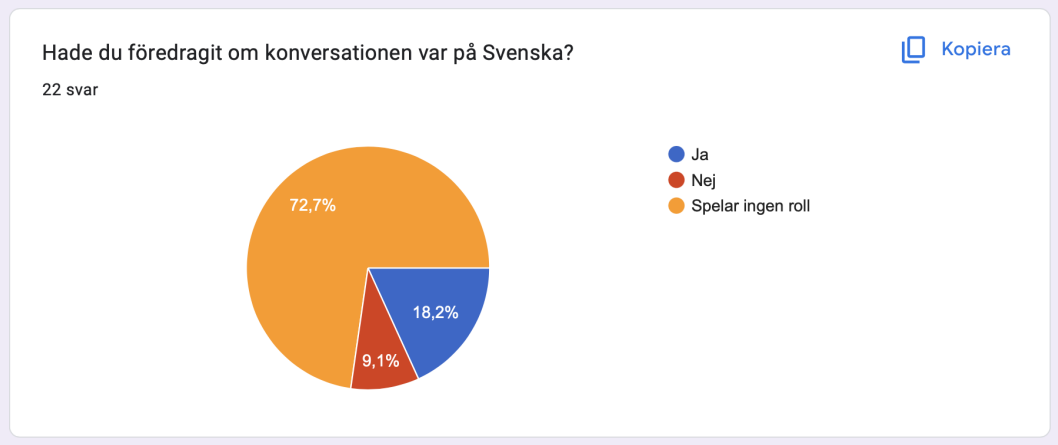


Figur 20: Fri text - del 4 - vilken känsla gav systemet användarna?

I Figur 22 presenteras resultatet angående språket. 18.2% föredrog om konversationen hade varit på svenska, 9.1% föredrog att behålla konversationen på engelska. 72.7% påstod att det inte spelade någon roll om konversationen var på engelska eller svenska. Majoriteten hade alltså ingen preferens över engelska eller svenska som språk för chattboten.



Figur 21: Formulering av svar från Chattbot.



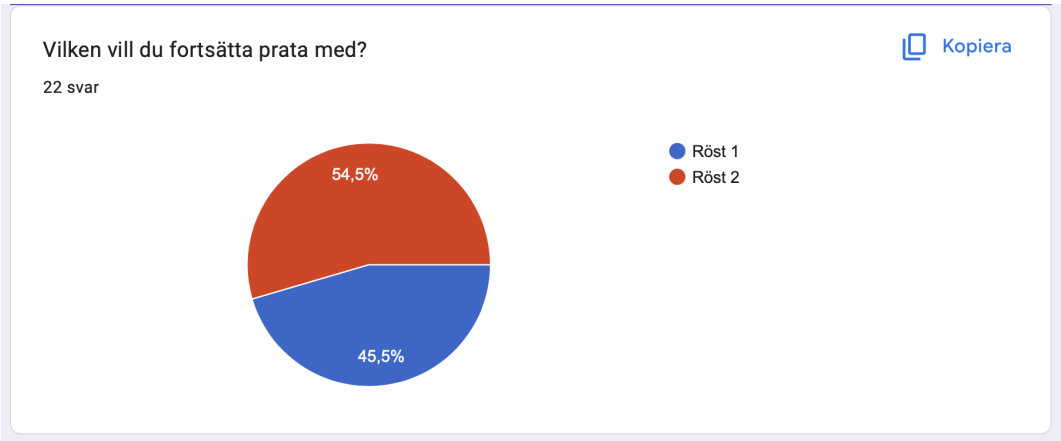
Figur 22: Vilket språk föredrar användarna?

4.4 Språk och röst

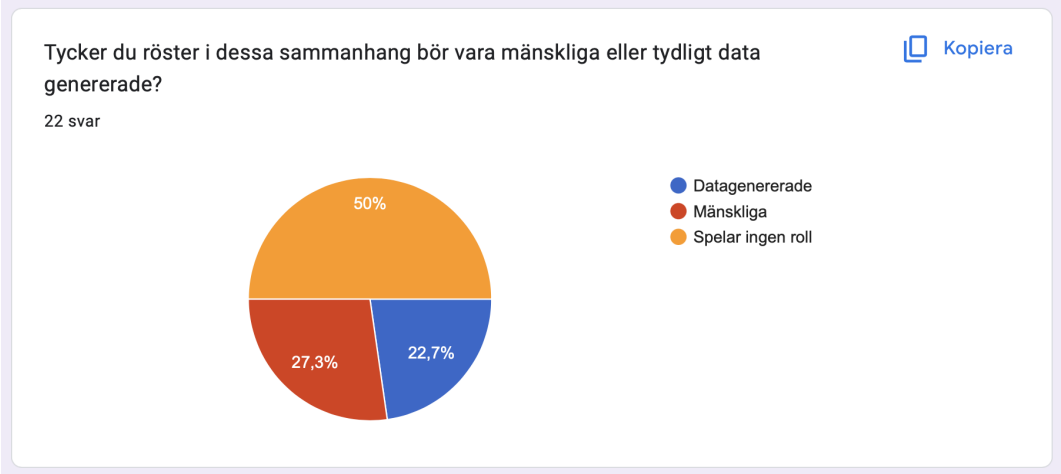
Figur 23 presenterar resultatet för vilken röst som föredrogs mellan PyQt6-gränssnittet och Furhat. 54.5% föredrog Furhats röst och 45.5% föredrog PyQt6. Figur 24 visar resultatet av huruvida rösten ska vara mänskligt eller robotaktig. Här har 50% svarat att det inte spelar roll, 27.3% tycker att rösten ska vara mänskliga och 22.3% tycker att den ska låta mer datagenererad.

4.4.1 Fördröjning

För att utvärdera fördröjningens påverkan på användaren testades sex olika situationer med olika lång fördröjningstid på varje konversation: 0, 1, 2, 3, 4 och 5 sekunder. Användaren fick inte informationen att det var fördröjningen som förändrades. Enligt figurerna 25 till 30 syns resultatet i detalj. En



Figur 23: Vilket system föredrar användarna?

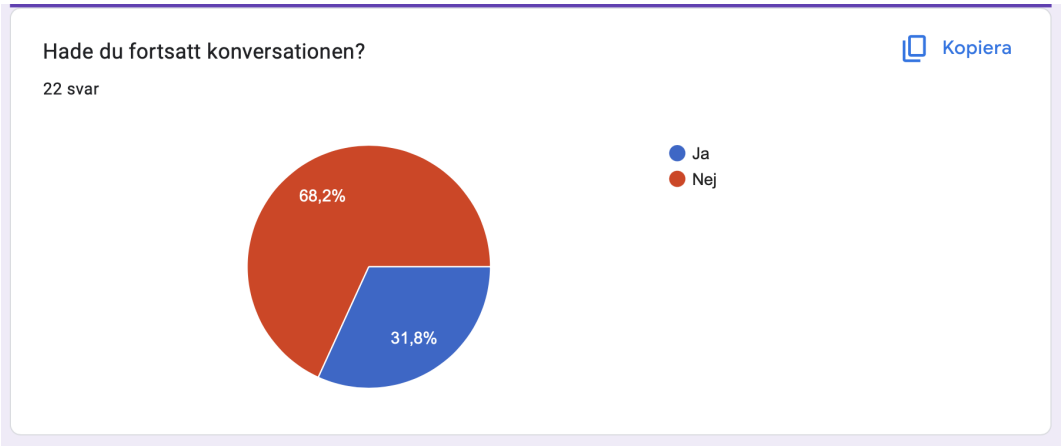


Figur 24: Vilken röst föredrar användarna?

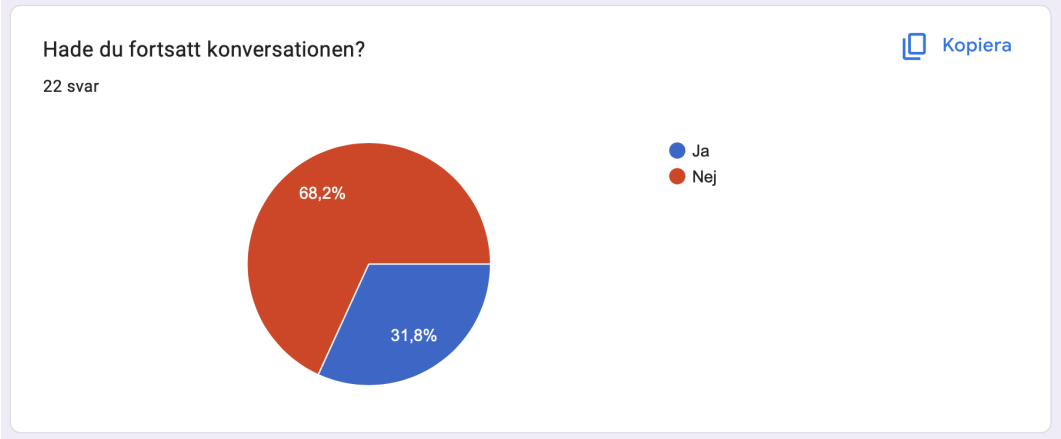
fördröjning på 3 sekunder eller mer ledde till att strax under 40% kunde tänka sig att fortsätta konversationen, medan 0 sekunder fördröjning gav ett resultat där 63.6% vill fortsätta konversationen. Ett sammanfattat resultat av detta syns i Figur 31.

I Figur 32 till 34 presenteras resultatet om användaren var medveten om att just fördröjningen förändrades. Av 22 deltagare kunde 10 individer korrekt dra slutsatsen att det var fördröjningen som förändrades, alltså 45%.

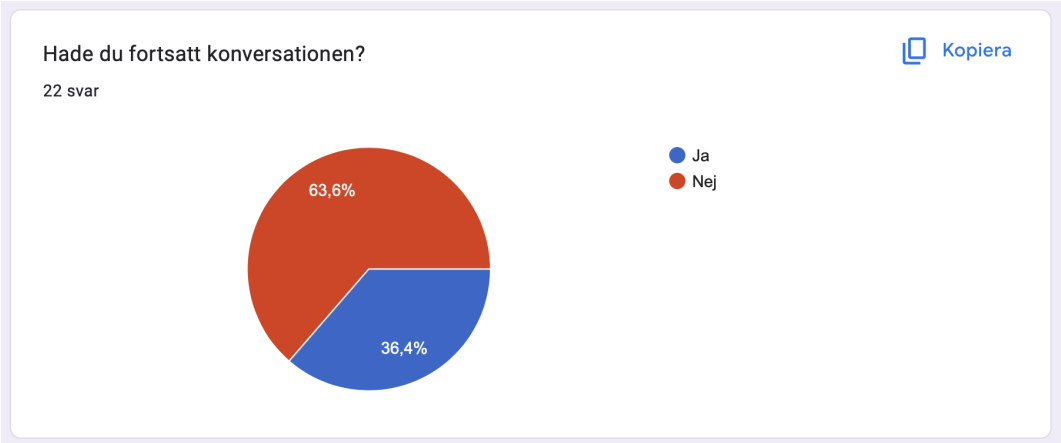
Figur 35 presenterar resultat för hur användarna värderar fördröjningsfaktorn, på en skala från 1 till 5, där 1 är ingen större skillnad och 5 är helt avgörande. 36.4% la ingen värdering i fördröjningen medan 13.6% tyckte detta var helt avgörande.



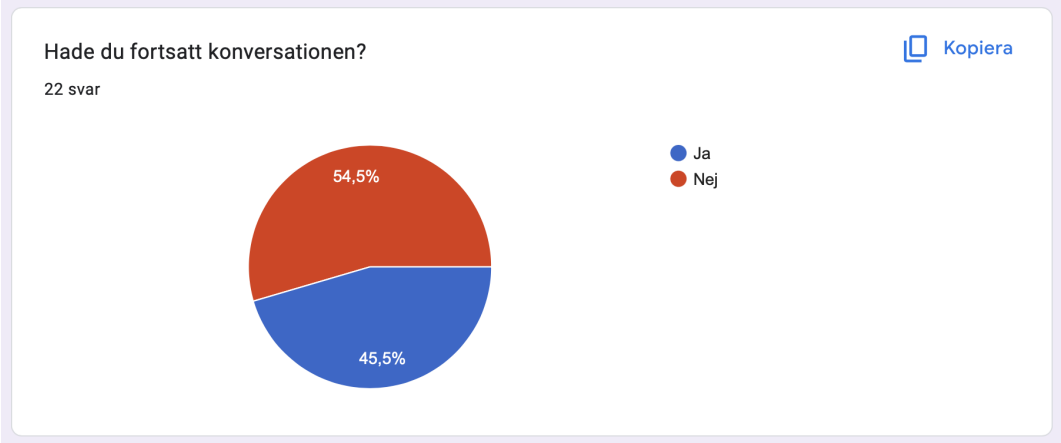
Figur 25: Konversation med 5 sekunder fördröjning.



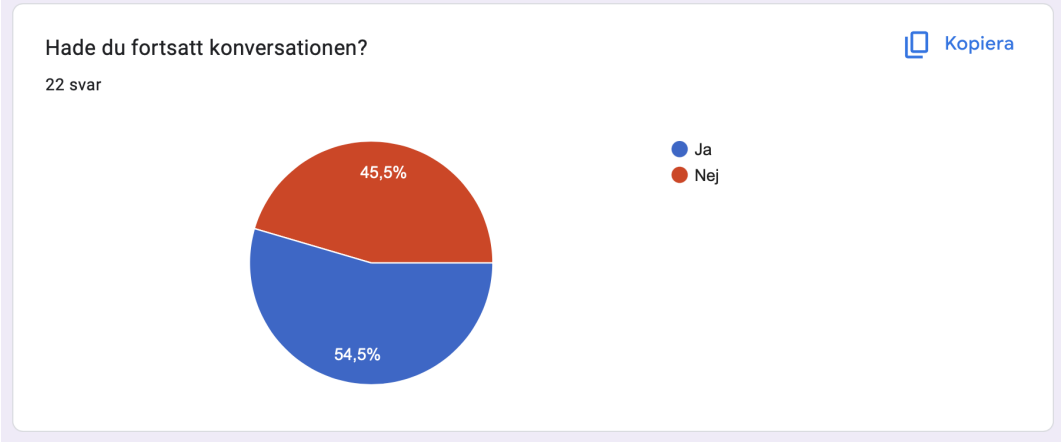
Figur 26: Konversation med 4 sekunder fördröjning.



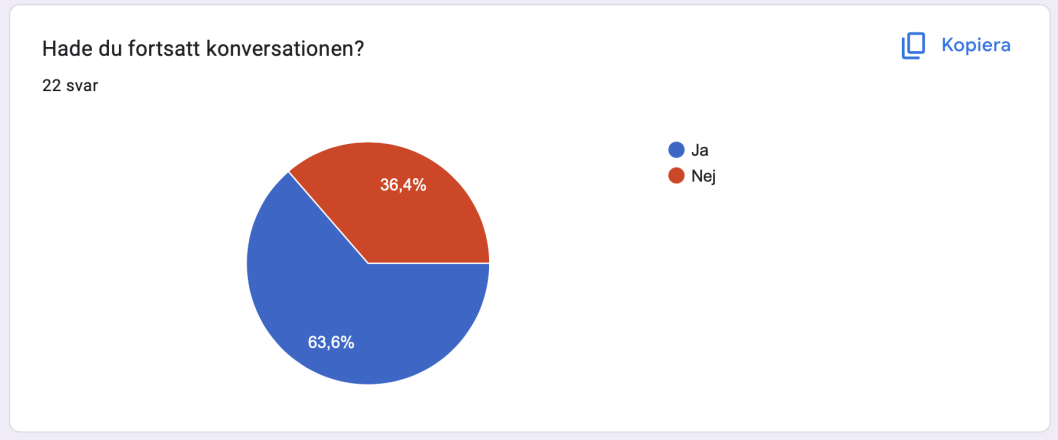
Figur 27: Konversation med 3 sekunder fördröjning.



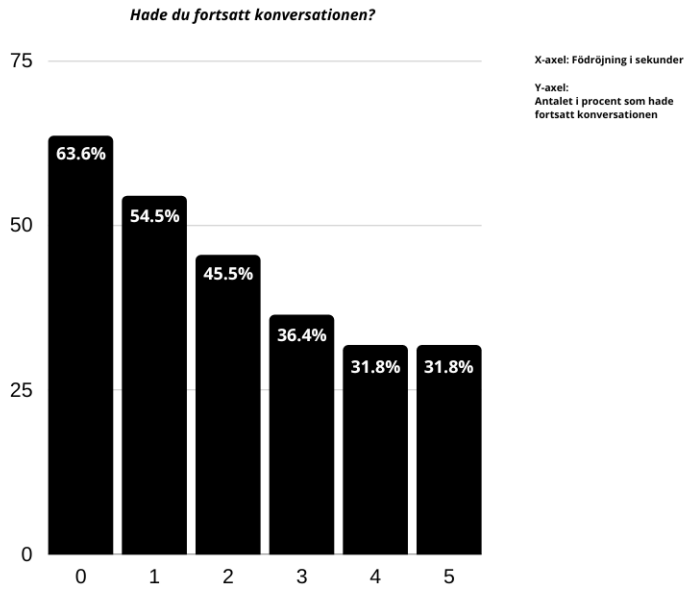
Figur 28: Konversation med 2 sekunder fördröjning.



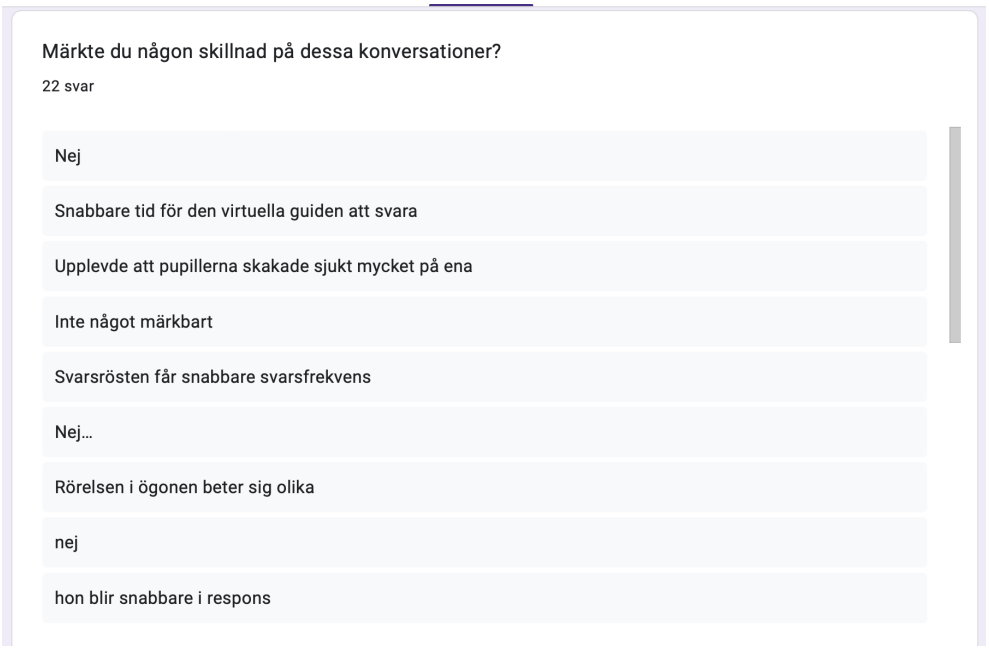
Figur 29: Konversation med 1 sekunder fördröjning.



Figur 30: Konversation med 0 sekunder fördröjning.

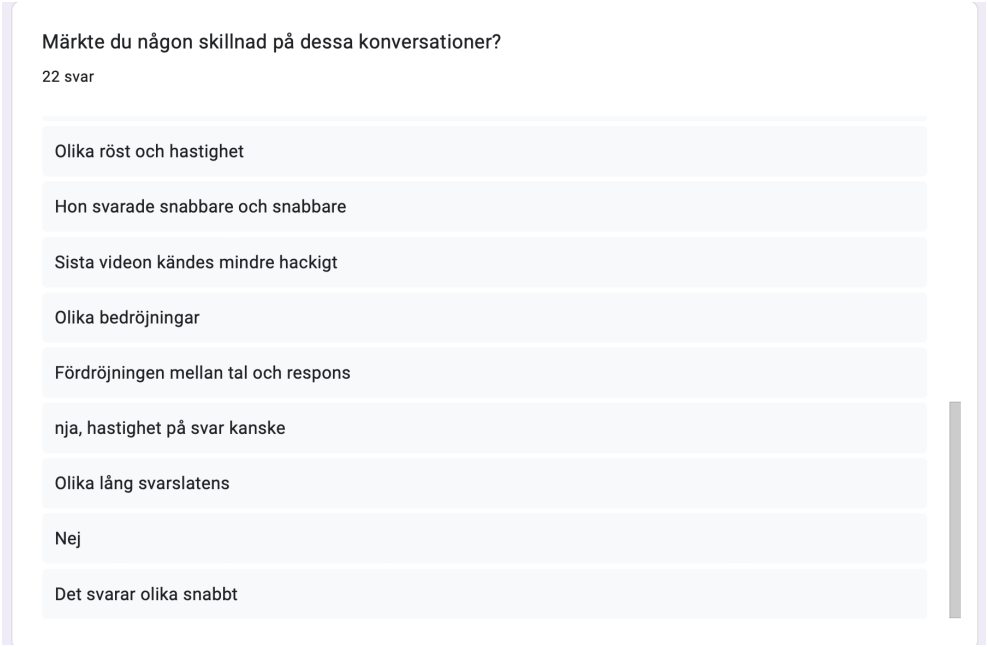


Figur 31: Fördröjning av svar : Användartest resultat sammanfattat.

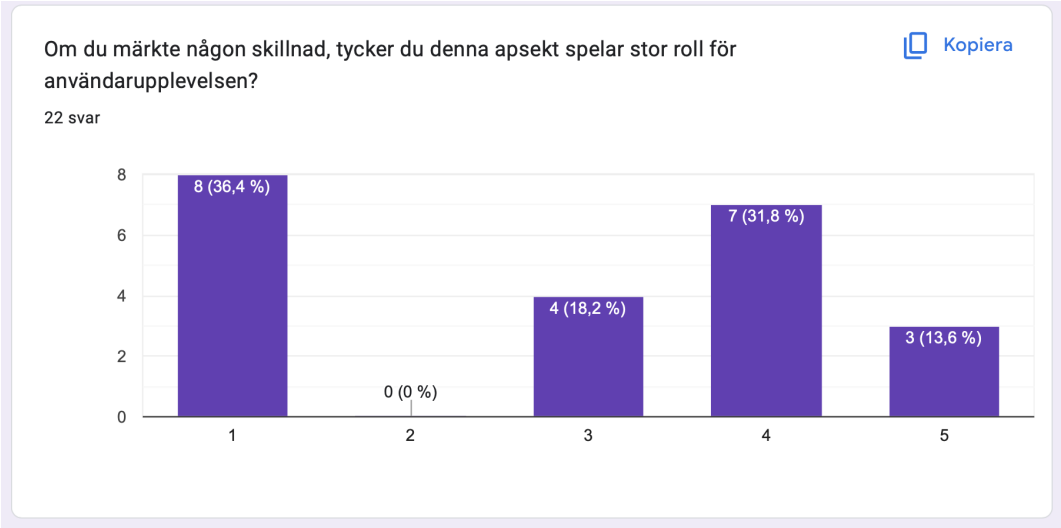


Figur 32: Svar del 1 : Fråga om användaren någon skillnad med födröjningsfaktorn.

Figur 33: Svar del 2 : Fråga om användaren någon skillnad med födröjningsfaktorn.



Figur 34: Svar del 3 : Fråga om användaren någon skillnad med födröjningsfaktorn.



Figur 35: Användarens värdering av födröjningsfaktorn.

Kapitel 5

Analys och diskussion

Det här kapitlet diskuterar resultaten och de tidigare studier som har presenterats ovan. Kapitlet går igenom de beslut som gjorts, vilka åtgärder som vidtagits för att uppnå de presenterade resultaten och vilka utmaningar som uppstod under processen. Vidare finns också en diskussion kring resultaten och deras betydelse för frågeställningarna. Slutligen presenteras även framtida forskningsområden och möjliga förbättringar av studiens metod och resultat.

5.1 System och komponenter

I detta avsnitt analyseras systemets olika delar som bidrar till den slutgiltiga modellen. Gruppen undersöker talmodeller och diskussionsmodeller som används för att bygga upp chattboten och tar även upp Blender, ett verktyg för att skapa och rendera grafiska representationer. Vidare diskuteras bildanalys och desktopprogram, samt hur Python och olika bibliotek används för att skapa systemet.

5.1.1 Talmodeller

En välfungerande talmodell är en grundpelare för att skapa en bra användarupplevelse med en virtuell guide. Utan en fungerande talmodell skulle ljudåterkopplingen vara begränsad, vilket skulle påverka helhetsupplevelsen negativt. Under utvecklingsprocessen upptäcktes det att vissa delar av arbetet var mer tidskrävande än förväntat. Trots detta lyckades utvecklingsteamet slutföra nödvändiga delar av talmodellerna, vilket resulterade i fungerande talmodeller som är avgörande för en bättre upplevelse med den virtuella guiden.

Tal-till-text Den färdiga tal-till-text-modellen uppfyller i stort sätt alla kundens krav, men det finns några utmaningar som inte löstes i bästa mån. Ett av dessa är att om användaren ställer en lång fråga kan inte hela frågan tas med, eftersom modellen har en fördefinierad tidsgräns att lyssna innan den börjar översätta. Även om det går att ändra tidsgränsen, kan detta leda till andra svarstider eftersom modellen behöver både lyssna mera och bearbeta mer data. Det finns dessutom en risk att andra störande ljud fångas upp under ljudinspelningen.

Även om modellen stödjer flera språk har det beslutats om att bara använda engelska eftersom modellen antingen lyssnar efter engelska eller försöker själv avgöra vilket språk användaren pratar. Med detta kunde modellen ibland missförstå användarens tal och översätta fel.

Att hitta en tal-till-text metod som fungerar bra med svenska är en utmaning, eftersom svenska är ett relativt litet språk och det krävs mycket träningsdata för att modellen ska förstå språket väl. Gruppen

använde sig av gratismodeller, eftersom det finns få modeller på svenska.

Text-till-tal Den färdiga text-till-tal-modellen ger ett snabbt och relativt bra resultat. I jämförelse med andra liknande text-till-tal-modeller presterar denna väldigt bra, vilket som kan vara av betydelse för användare som använder sig av denna typ av guide.

Kunden hade en stor önskan att kunna använda en personlig röst i guiden, men på grund av utmaningar med att hitta en modell som kunde stödja alla kundens krav samt tidsbegränsningar, valdes en lättare modell med en färdigimplementerad röst. Trots att denna modell inte kan erbjuda en personlig röst ger den fortfarande ett bra resultat.

I framtiden kommer text-till-tal modeller garanterat utvecklas, speciellt med tanke på efterfrågan och användningen av röststyrda enheter och virtuella guider. Det kan finnas en ökad efterfrågan på mer anpassningsbara och personifierade röster, och det är möjligt att framtida tekniker kommer att möjliggöra detta.

5.1.2 Diskussionsmodeller och chatbottar

En av de främsta fördelarna med att använda *Hugging Face* är dess stora samling av färdiga modeller inom språkteknologi och erbjudandet att importera dessa modeller med minimal kodning. Detta gör det möjligt att snabbt bygga ett frågebesvarande system och undersöka flera olika typer av modeller som det har gjorts i detta projekt. Dessutom erbjuder Hugging Face omfattande dokumentation och guider för att finjustera modellerna själv via *Pytorch*. Detta utöver att förtränade modeller redan sparar mycket tid har gjort arbetsprocessen mycket effektivt jämfört med att bygga chattboten från grunden.

En nackdel är att det finns många varianter på modeller som har undersökts i projektet att det inte finns tillräckligt med tid för att undersöka flera. Detta leder till en upprepande undersökning där det blir svårare att hitta den rätta modellen inom en rimlig tidsram.

En annan nackdel är att undersökningen av T5-modellen var ganska övergripande på grund av att projektet undersökte flera typer av modeller, vilket ledde till att tiden inte räckte till att göra en mer fördjupande analys av den valda modellen för chattboten.

5.1.3 Grafisk representation

Utifrån kundens önskan från projektets start skulle gruppen skapa en avatar genom att 3D skanning. Visionen var att skapa en verklig avatar, helst där en av gruppmedlemmarna skulle skannas för att sedan applicera en lip-sync modell där rösten skulle vara individens egen. Många redan etablerade interaktiva applikationer undviker att försöka efterlikna människor. Det finns en tydlig röd tråd att man försöker undvika detta, förmodligen för att det är för svårt och hamnar ofta i uncanny valley. Det vill säga, tillräckligt nära att individen uppfattas, men med nackdelen att användaren är medveten om att det är en robot och uppfattar istället systemet som obehagligt.

I jämförelse med de alternativa representationerna, kan användningen av mänskliga representationer ofta vara mindre flexibel och anpassningsbar i stor skala. De alternativa representationerna möjliggör mer designfrihet och flexibilitet och är lätta att anpassa för att matcha en mängd olika situationer och implementationer. Med anledningen i ovanstående resonemang valde gruppen att implementera en abstrakt form istället för att utforska möjligheterna med en mänsklig modell som representativ för PyQt6-systemet. Den slutgiltiga implementationen består av två ögon som på ett interaktivt sätt följer användaren. Det är en bra kompromiss som kombinerar många aspekter av liknande implementationer. Systemet undviker uncanny valley helt och hållet men lyckas ändå förmedla medkänsla och

systemet gör mer än bara prata. Med hjälp av bildanalysen får systemet också den interaktion med användaren som eftersöktes.

Den abstrakta figuren som tillhör den slutgiltiga representationen implementerades först i PyGame och därefter i PyQt6. Fördelarna med PyGame är att det avlastar utvecklarna i kodandet av illustrationen av den abstrakta grafiska representationen då verktyget innehåller draw funktioner. Därmed hamnar utseendet av representationen i fokus vilket var en betydelsefull önskan från kunden. PyGame kommer däremot inte till användning för den slutgiltiga grafiska representationen bygger på integrationen och kompatibilitet. Alltså för den färdiga Python-applikationen måste samtliga bibliotek fungera med varandra. PyGame bygger på SDL-bibloteket med dess egna rendreringsloop medans resterande delar av systemet är konstruerat i PyQt6 som använder Qt-biblioteket med dess rendreringsloop.

Det är möjligt att kombinera dessa bibliotek under ett och samma Python-system. Däremot kommer två stycken fönster att illustreras när systemet körs. Det är inte en optimal lösning för det här systemet då gruppen vill visa den abstrakta figuren och chattfunktionen under ett och samma fönster. Utöver att två fönster skulle visas skulle implementeringen i PyGame med resterande system i PyQt6 med största sannolikhet leda till komplexa problem som endast resulterar i konflikter i systemet. Konsekvenserna med PyGame blev att PyQt6 implementationen är den mest lämpade sättet att representera GUI:n.

PyQt6 har inte samma egenskaper som PyGame, där utvecklarna får hjälp att skapa ögonen som användes för grafiska representationen. Konsekvensen blir att utvecklarna måste skriva mer kod, vilket kräver extra förkunskap eller undersökning om hur PyQt6 har för funktioner och ska implementeras. Som nämndes tidigare i texten är resterande delar av systemet konstruerat i PyQt6 också. På så sätt undvek projektgruppen större konflikter när grafiska representationen bands samman med resterande delar av systemet.

5.1.4 Bildanalys

Det finns en viss felmarginal i det tränade datasetet och modellen därefter. Detta innebär att komponenten inte alltid känner igen ansiktet framför kameran. Koordinaterna är direkt kopplade till ögonens position och kommer därför också påverkas. Om ansiksigenkänningen anser att du flyttat dig, trots att ansiktet är stationärt, kommer ögonen att röra sig. Detta åtgärdades genom att använda ett ursprungspunkt i mitten av skärmen för grafiken att falla tillbaka på om bildanalysen sviker.

Eftersom komponenten söker igenom samtliga stillbilder som kameran, visar är det en relativt tung process som kräver mycket minne och processorkraft. Detta syns då också i den grafiska representationen med något hackig återkoppling.

5.1.5 Integration av system och sammankoppling av komponenter

Att utveckla ett gränssnitt och ett komplett desktopprogram var ingen självklarhet i projektplanen. Många inom gruppen menade på att det skulle vara ett enkelt program som endast visade den virtuella guiden. Efter mycket testning och diskussion visade det sig att implementera en chattfunktion var nödvändigt. Chattfunktionen visar konversationen i text och ger inte bara möjlighet till att förtydliga den givna konversationen, utan låter också användaren gå tillbaka och läsa igenom konversationen retroaktivt.

Ett gränssnitt med knappar, textfunktioner och den virtuella guiden kan också ge ett mer professionellt intryck. Applikationen upplevs mer fullständig än med endast en avatar som representerar hela systemet. I ett verkligt scenario behöver ofta användaren mer än bara en representation för att på en gång, förstå systemets syfte och hur interaktionen bör fungera.

Eftersom att gränssnittet är extremt enkelt och inte innefattar några avancerade funktioner i form av inmatning eller interaktion med själva gränssnittet togs ingen extra tid för användartester för just designen. Gruppen resonerade att systemet är godtyckligt enkelt att deras designförmåga och kunskap var godtycklig för att representera systemet på ett tillräckligt bra sätt, utan att konsultera utomstående användare.

Desktopprogrammet är ryggraden i systemet och tillåter inte bara användaren att få en grafisk representation av systemet utan sammanfogar också samtliga komponenter och tillåter dem att tala med varandra. Implementationen innebär också *threading* där ett flertal processer går parallellt. Detta innebär att programmet lyssnar, tittar och formulerar svar på en och samma gång, och behöver alltså inte vänta på att en process blivit färdig innan nästa påbörjas.

Anledningen till att just PyQt6 användes grundades i kompabilitetsproblem mellan Tkinter och den äldre varianten av Python, 3.8.16. Det uppstod ett flertal problem när dessa kombinerades och valet att använda PyQt6 gjordes. I samband med bytet upptäcktes det också att PyQt6-program kan designas med vanlig CSS kod, något Tkinter inte kan. Detta gjorde beslutet ännu mer självklart.

5.1.6 Python och bibliotek

För att kunna undersöka huruvida standarder och externa bibliotek påverkar arbetet och implementationen av systemet behöver dessa diskuteras. Av den anledningen kommer Python och ett flertal tredjehandsbibliotek att undersökas närmare. Den slutgiltiga Python-applikationen bygger nämligen på många tredjehandsbibliotek för att fungera som den ska. Dessa olika bibliotek stödjer olika varianter av Python. Av den anledningen körs applikationen i version 3.8 i Python, trots att version 3.11 är den senaste versionen, i skrivande stund. Detta är för att garantera att de bibliotek som används fungerar med varandra. Vissa bibliotek var inte optimerade för den nyaste versionen av Python. Projektgruppen fann att version 3.8 var en optimal kompromiss mellan de äldre biblioteken som inte längre stöds av den nyaste python-versionen samt de nyare biblioteken som inte längre stöds av de äldre python-versionerna.

Användningen av externa tredjepartsbibliotek är en vanlig praxis vid utveckling av mjukvara. Dessa bibliotek kan spara utvecklare tid och arbete genom att tillhandahålla funktioner och verktyg som är redan färdiga att använda. Detta kan leda till en ökad produktivitet och minskad tid för utvecklingsarbetet.

En annan fördel med att använda externa tredjepartsbibliotek är att de ofta har testats och validerats av andra utvecklare. Detta betyder att de har högre kvalitet och kan vara mer tillförlitliga än att utveckla egna lösningar från grunden. Att använda externa bibliotek kan också göra det enklare att underhålla och vidareutveckla projektet i framtiden då projektet bygger på mindre egenbyggd kod.

Trots de fördelar som användning av externa tredjepartsbibliotek kan ge finns det också vissa nackdelar som bör beaktas. En av de främsta nackdelarna är risken för beroenden. Om en tredjepartsbibliotek uppdateras eller förändras, kan det påverka hur den befintliga koden fungerar och orsaka problem som kan vara svåra att felsöka. Det kan även leda till en ökad säkerhetsrisk. Om utvecklarna inte undersöker bibliotekens säkerhet noggrant, kan det öppna upp för sårbarheter och utsätta applikationen för attacker. Detta är speciellt viktigt i ett projekt som detta då applikationen använder sig av både kamera och mikrofon.

Det är även värt att nämna att olika tredjepartsbibliotek kan ha olika licensieringskrav. Detta kan påverka hur koden får användas och distribueras, och kan leda till rättsliga problem om inte licensvillkoren följs noggrant.

5.1.7 Mänsklig representation

Resultatet i form av en mänsklig representation av den virtuella guiden mottogs väl enligt användartestet, enligt Figur 10. 59.1% av användarna föredrog den mänskliga representationen i jämförelse med den abstrakta representationen. Däremot uppstår det både positiva och negativa kommentarer i användartestet, gällande frågan "vilken känsla fick du av respektive system?", vilket kan ses i Figur 17. En del testare ansåg att den mänskliga represenationen var obehaglig och föll inom *uncanny valley*-phenomenet, medan andra testare kände att den var trevlig och att dess ansiktsuttryck gjorde den mer mänsklig. Något man kan komma fram till är att det finns starka åsikter åt båda hållen angående hur lämplig den är och att det finns utrymme för att förbättra upplevelsen för alla användare.

5.2 Diskussion av resultat

Det här kapitlet diskuterar de resultatet framtagit med användartesterna. Sektionen berör de svar som projektgruppen samlat in angående de två olika grafiska representationerna, chattboten, de olika rösterna systemet använder sig av samt vilken grad av fördröjning som är acceptabelt. Kapitlet kopplar resultatet till de tekniker som använts och diskuterar tekniska begränsningar i förhållande till frågeställningarna och optimala lösningar.

5.2.1 Jämförelse mellan mänsklig- och grafisk representation

Från användartesterna finns det ingen grafisk representation som verkar vara klart bättre än den andra. Utifrån de 22 användartesterna som gjordes ansåg 59.1% att Furhat-representationen är den bättre representationen över illustrationen med två ögon. Alltså kan gruppen konstatera att ingen av versionerna är varken sämre eller bättre än den andra, då nästan hälften av användarna tycker olika.

En notering från användartesterna var användarnas motivering till varför Furhat-representationen föredrogs över de två ögonen eller tvärtom. Bland de vanligaste motiveringarna till användarens val var att man störde sig över eller upplevde den andra versionen som obehaglig. Med andra ord röstade inte användarna på den versionen som de gillade mest utan med motiveringen att de föredrog den versionen som de störde sig minst på. Att den mänskliga representationen föredras med en liten marginal över ögonen indikerar förmodligen på att användarna kan uppleva konversationen mer som en vanlig konversation jämfört med ögonen då folk känner sig övervakade av ögonen.

5.2.2 Chattbot

Enligt användartesterna syns det tydligt att användarna föredrog ett svar som motsvarar en mänskligt interaktion, det vill säga ett svar där faktan ges som i en vanlig konversation. Det var färre som föredrog ett snabbt och avskalat svar eller ett svar med extra information. Resultatet är inte förvånande då svaret som föredrogs matchar det svar man skulle fått i en vanlig konversation med en annan människa. Det indikerar också att användarna tittar på konversationen som om det vore en vanlig interaktion, inte till exempel en Google-sökning där korta och koncisa svar ofta ges.

Det ska också noteras att det på en stor skala inte finns ett språk som föredras, då med avseende på svenska och engelska. Totalt 72.7% påstod att det inte spelar någon roll om systemet interagerar på svenska eller engelska. Detta resultat gynnar system av som detta då det finns betydligt fler hjälpmedel och tekniker på det engelska språket jämfört med det svenska. Det ska därför noteras att man bör utnyttja detta och hålla utvecklingen på engelska.

5.2.3 Språk och röst

Resultaten från de användartester som utfördes visade att majoriteten av användarna föredrog rösten hos Furhat framför den implementerad med PyQt6-gränssnittets röst. Det är dock viktigt att notera att det endast var 2 personer fler som föredrog Furhats röst framför den som var implementerad med PyQt6-gränssnittets röst, och om fler användartester hade genomförts kan det ha varit möjligt att få en tydligare bild av vilken röst som är mest populär.

Det är också värt att påpeka att båda rösterna som användes i testet var datagenererade, vilket kan påverka resultatet. Användarna hade kanske inget att jämföra med och därför valde de att det inte spelade någon roll vilken röst de föredrog. Om man exkluderar de användare som svarade att det inte spelade någon roll, visade resultaten att fler användare föredrog en mänsklig röst över en datagenererad röst.

För att få en mer tillförlitlig bild av vilken röst som är mest populär skulle fler tester behöva utföras. Det är också viktigt att ta hänsyn till att båda rösterna i testet var datagenererade, vilket kan påverka resultatet och gjort det svårt för användarna att avgöra vilken röst de föredrog.

5.2.4 Fördröjning

Att fördröjningen är en influensrik parameter är inget förvånande resultat. Enligt graferna i Figur 31 är det tydliga markanta skillnader i procent att användarna föredrar systemet mer och mer, desto mindre responstiden är, det vill säga fördröjningen. Dock syns det i samband med detta resultat att endast hälften av användarna lyckades identifiera vilken parametrar som ändrades. Slutsatsen blir då att fördröjningen är en viktig del av interaktiva system, även om uppfattningen kring den bildas subtilt och ibland undermedvetet. Det ska också noteras att parametern är helt avgörande för systemets uppfattning och i sin tur användarens vilja att interagera med systemet. En för långsam responstid, i detta fall över 3 sekunder, leder till att över 60% av användarna inte vill fortsätta interagera med systemet. Med detta resultat kan det konstateras att system av denna natur bör prioritera responstiden systemet innehar, och om det är möjligt prioritera bort andra faktorer eller funktioner om responstiden kan förbättras.

5.3 Etisk och samhällelig reflektion

För att använda sig av den virtuella guiden behöver den ta upp och analysera både användarens utseende och ljud. Denna analys aktiveras utan att systemet förvarnar användaren, eller låter denne godkänna det i syfte att minsta friktionen av att använda systemet och komma igång. Det är inte orimligt att användarna nyttjar systemet utan att tänka på att ljudet tas upp och analyseras, eller att de har en kamera som följer deras position.

Systemet har heller ingen funktion som motverkar att någon utomstående hackar sig in i systemet och upphäva funktionen som raderar inspelningen av användarens röst. Detta är en sårbarhet hos systemet. Om någon skulle hacka sig in i systemet skulle den personen i värsta fall få tillång till bild- och ljudupptagning av användaren. Detta skulle vara förödande om användaren delade med sig av personlig information. Däremot är det orimligt att användarna delar med sig av privat information till systemet då de sannolikt kommer hantera den virtuella guiden på samma sätt som en fysisk guide.

Vid vidareutveckling av virtuell guide uppstår ett flertal etiska och moraliska frågeställningar. Dessutom måste man ta hänsyn till hur mjukvaran ska undvika att svara med potentiellt kränkande eller felformulerade uttalanden, och vem som bär ansvaret för detta. Det är också viktigt att klargöra vem som bär ansvaret för detta om systemet skulle uttrycka en fras av denna natur. Detta är en både moralisk, etisk och juridisk fråga som måste tas hänsyn till i utvecklingen av produkten.

En viktig fråga som uppstår vid utvecklingen av virtuell guide handlar om hur tekniken påverkar individers integritet. Eftersom tekniken är baserad på bildanalys och artificiell intelligens kan den hämta information från privatpersoner och därmed utmana integriteten. Denna fråga är särskilt känslig när det gäller inspelning i offentliga forum och hur mikrofonen ska användas för att interagera med användaren.

Kapitel 6

Slutsatser

Projektet och systemet virtuell guide har undersökt och utrett interaktionen mellan dator och människa med avseende på utställningsmiljöer. I samband med utredningen har gruppen genomfört en grundlig process för utveckling med Python bibliotek, olika typer av artificiella intelligensmodeller och tekniker inom detta område. Syftet har således blivit uppdelat i två kategorier direkt kopplade till varandra. Projektet har utrett interaktionen mellan dator och människa i en utställningsmiljö. För att genomföra denna utredning har projektgruppen utvecklat en virtuell guide anpassad för dessa miljöer. Genom utvecklingen har gruppen forskat och implementerat ett flertal modeller inom den givna området och därmed kunnat undersöka och besvara de tillhörande frågeställningarna.

6.1 Frågeställningar

I följande sektion besvaras de frågeställningar som ställdes i kapitel 1.3.

Vilken grafisk representation är bäst lämpad för att representera ett system för interaktion mellan en mänsklig användare och dator i en utställningsmiljö? Hur påverkar den grafiska representationen användarupplevelsen och känslan systemet förmedlar?

Att representera ett system med syftet att försöka efterlikna en människa är en svår process. Det är en process som som ett flertal företag inom branschen och det givna området väljer att undvika. Ett tydligt mönster har visats, där implementationer av denna natur ofta använder alternativa metoder för att förmedla känslor i ett försök att personifiera systemen, utan att efterlikna personer. Genom användartester syns det att ett flertal personer tycker att en mänsklig representation är obehaglig och en representation med endast ögon uppfattas som "övervakande öch stirriga".

Utifrån genomförda tester, undersökningen och de studier och diskussioner gruppen har genomfört har slutsatsen dragits att systemet bäst representeras med en abstrakt representation. Detta återkommer igen till hur förtag väljer att representera sina system med likande figurer. Processen är inte bara svår att efterlikna en människa, utan de flesta företag i branschen undviker att konstruera sådana figurer då majoriteten av människor föredrar att figurerna visualiseras på andra sätt, till exempel genom en abstrakt Figur.

Tanken att representera ett system som detta med en människa är en process och ett förfarande som bör undvikas. Inte bara ur ett användarvänligt perspektiv där en del användare finner obehag men även ur ett tekniskt perspektiv, då implementationen av denna typ blir väldigt utmanade.

Hur kan färdiga modeller inom bildanalys, textanalys, talanalys och diskussionsmodeller användas för att skapa en interaktiv virtuell assistent, och hur påverkar skillnaden mellan lokalt processade modeller och externt baserade modeller implementeringen och utvecklingsmöjligheterna?

Det finns det ett flertal färdiga modeller, tekniker och system som möjliggör helhetsystem för en virtuell assistent med fokus på utställningsmiljöer och samtalsbaserade interaktioner. Processen att koppla ihop självständiga modeller för att bilda nya system och i sin tur applikationsområden är i dagsläget en fullt möjlig process med de utbud av kod, tekniker och mallar som finns.

En tydlig begränsning är istället hårdvaran som systemet utvecklas eller körs på. Många modeller och bibliotek använder den faktiska hårdvaran av klienten som systemet körs på. Detta som motsats till API- eller databaslösningar. Problematiken är att de flesta generativa AI-modellerna är mycket krävande och sällan optimisierade för att köras på en vardaglig laptop eller motsvarande hårdvara, detta gör det i sin tur svårt att implementera och utveckla dessa typer av system.

Om systemet i fråga ska distribueras kommersiellt beror det på hur mycket data som behandlas och vilken typ av hårdvara som finns tillgänglig. Slutsatsen blir därför att om systemet kräver mycket beräkningskraft och lagring kan det vara bättre att använda en API- eller databaslösning för att undvika att belasta den lokala hårdvaran. En API-lösning kan också vara mer skalbar och flexibel eftersom den nås via internetuppkoppling. Dock så har en lokal lösning fördelen att vara snabbare samt att det inte behövs en internetuppkoppling.

Hur kan standarder och externa bibliotek påverka eller hämma arbetet och implementationen av ett system starkt beroende på färdiga lösningar och tidigare implementerade funktioner och modeller? Hur påverkar en anpassad systemarkitektur och planering produktens utveckling ur detta perspektiv?

Att använda färdiga bibliotek, modeller och funktioner sparar tid och energi. Att implementera motsvarande modeller från grunden är ett skilt projekt. Det visade sig dock att versioner och kompatibilitet är en stor begränsning och kan bli väldigt tidskrävande. Det är en svår process att reda ut och kräver både kunskap och en tydlig målbild.

Projektet hade en tydlig bild kring systemarkitekturen och hur komponenterna skulle interagera, dock hjälper detta bara till en viss gräns. Även fast komponenterna inte är implementerade i samma filer kan kompabiltiet ställa till problem ändå. Problemen som uppstod berodde till mestadels på Python och dess versioner samt att vissa versioner funkade inte för alla utvecklare. Kompromissen som gruppen gjorde var att backa till en äldre version av programmeringsspråket, som i slutändan gjorde det möjligt att använda samtliga nödvändiga bibliotek.

Sammanfattningsvis kan gruppen konstatera att det rekommenderas att kompatibilitet mellan bibliotek, modeller och komponenter bör tas med som en faktor i projektplanen och tillhörande systemarkitektur. Genom att utvärdera och bestämma denna faktor redan i förstudien underlättar det för den kommande implementeringsprocessen och ramverket, då tidsplanen kan istället följas enligt plan.

6.2 Vidareutveckling

Undersöka hur komponenterna för tal-till-text och bildigenkänning kan utvidgas för att tillåta interaktioner med flera personer samtidigt. Detta skulle öppna upp applikationsområden avsevärt och därmed

bör detta anses vara ett av de viktigaste vidareutvecklingarna för systemet. En viktig aspekt inom detta område skulle då också bli att reducera brus och störningar från den omgivande miljön.

En möjlighet vore att kombinera chatmodeller av olika slag för att ge bästa lämpliga svar till användaren och inge en godare användarupplevelse. T5-modellen ger kortfattade svar utefter given kontext, vilket kan vara lämpligt för specifika frågor. Medan svar på mer generella frågor efterfrågar ett bredare och mer omfångsrikt svar och därmed skulle en modell som GPT-3.5 kunna användas. Utmaningen blir att hitta eller ta fram en funktion som bestämmer om en fråga är antingen av det specifika eller generella slaget.

Utveckla profiler för användare som skapas och underhålls av bildigenkänningen för att komma ihåg användare. Om systemet kommer ihåg individuella användare och deras tidigare konversationer finns möjligheten att anpassa framtida konversationer utefter individer. Något som den virtuella guiden även skulle kunna göra är att bemöta en återkommande gäst med frasen "Välkommen tillbaka", vilket förhoppningsvis inger en positiv återkoppling hos användare.

Litteraturförteckning

[1] Mike Seymour, Dan Lovallo, Kai Riemer, Alan R. Dennis, and Lingyao (Ivy) Yuan. Ai with a human face. https://hbr.org/2023/03/ai-with-a-human-face, 2023. Hämtad: 2023-04-27.

[2] Nancy Viola Wuenderlich and Stefanie Paluch. A nice and friendly chat with a bot: User perceptions of ai-based service agents. 2017.

[3] Furhat robotics. https://furhatrobotics.com/. Hämtad: 2023-05-08.

[4] Nvidia. Omniverse documentations. https://docs.omniverse.nvidia.com/. Hämtad: 2023-05-03.

[5] Nvidia. Audio2face overview. https://docs.omniverse.nvidia.com/app_audio2face/app_audio2face/overview.html#minimum-mesh-requirements-for-full-face-character-setup. Hämtad: 2023-05-03.

[6] Google Home. Google home. https://home.google.com/welcome/. Hämtad: 27 maj 2023.

[7] Apple Inc. Siri - apple. https://www.apple.com/siri/. Hämtad: 27 maj 2023.

[8] Amazon Developer. Amazon alexa developer. https://developer.amazon.com/en-US/alexa. Hämtad: 27 maj 2023.

[9] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33:12449–12460, 2020.

[10] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. *arXiv preprint arXiv:2212.04356*, 2022.

[11] Huggingface. Text2speech. https://huggingface.co/tasks/text-to-speech. Hämtad: 2023-05-09.

[12] Ryan McGrath. gtts. https://gtts.readthedocs.io/en/latest/. Hämtad: 2023-05-25.

[13] Resemble AI. Text2speech. https://www.resemble.ai/text-to-speech-converter/. Hämtad: 2023-05-09.

[14] Oshan Jarow. How the first chatbot predicted the dangers of AI more than 50 years ago. https://www.vox.com/future-perfect/23617185/ai-chatbots-eliza-chatgpt-bing-sydney-artificial-intelligence-history. Hämtad: 2023-05-03.

[15] L. Bordoni, F. Mele, and A. Sorgente. *Artificial Intelligence for Cultural Heritage*. Cambridge Scholars Publishing, 2016.

[16] Mihaj Duguleana, Victor-Alexandru Briciu, Ionout-Alexandru Duduman, and Octavian Mihai Machidon. A virtuel assistant for natural interactions in museums. *Sustainability*, 12(17), 2020.

[17] Timothy Bickmore, Laura Vardoulakis, and Daniel Schulman. Tinker: A relational agent museum guide. *Autonomous Agents and Multi-Agent Systems*, 27, 12 2013.

[18] Daniel Morena. IRIS+ part one: Designing + coding a museum AI. https://www.aam-us.org/2018/06/12/iris-part-one-designing-coding-a-museum-ai/. Hämtad: 2023-05-03.

[19] IBM. IBM Watson. https://www.ibm.com/watson. Hämtad: 2023-05-07.

[20] Standard, wavenet, neural2, and studio voices. https://cloud.google.com/text-to-speech/docs/wavenet, 2023. Hämtad: 2023-05-24.

[21] Hugging Face. https://huggingface.co/. Hämtad: 2023-05-03.

[22] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[23] Question Answering. https://huggingface.co/tasks/question-answering. Hämtad: 2023-05-03.

[24] Skanda Vivek. Extractive vs Generative Q&A - Which Is Better for Your Business? https://towardsdatascience.com/extractive-vs-generative-q-a-which-is-better-for-your-business5a8a1faab Hämtad: 2023-05-03.

[25] Text Generation. https://huggingface.co/tasks/text-generation. Hämtad: 2023-05-03.

[26] The National Library of Sweden / KBLab. Swedish BERT models. https://huggingface.co/KB/bert-base-swedish-cased. Hämtad: 2023-05-03.

[27] AI Sweden. GPT-SW3. https://www.ai.se/en/node/81535/gpt-sw3. Hämtad: 2023-05-03.

[28] Christian Di Maio and Giacomo Nunziati. T5 for generative question answering. https://huggingface.co/MaRiOrOsSi/t5-base-finetuned-question-answering?doi=true. Hämtad: 2023-05-03.

[29] Open AI. Models. https://platform.openai.com/docs/models. Hämtad: 2023-05-03.

[30] Face detection and recognition using opencv and python, 2023.

[31] Tejashree Dhawle, Urvashi Ukey, and Rakshandha Choudante. Face detection and recognition using openCV and python. *Int. Res. J. Eng. Technol*, 7(10), 2020.

[32] Pygame. Pygame. https://www.pygame.org/docs/, 2022. Hämtad: 2023-04-27.

[33] Martin Fitzpatrick. Pyqt vs. tkinter – which should you choose for your next GUI project? https://www.pythonguis.com/faq/pyqt-vs-tkinter/, 2019. Hämtad: 2023-04-23.

[34] History Computer. C++ vs python: A full comparison. https://history-computer.com/c-vs-python-2/, 2022.

[35] SCB. Att välja metod och intervjupersoner. https://www.scb.se/dokumentation/statistikguiden/undersokning-och-urval/att-valja-metod-och-intervjupersoner/. Hämtad: 2023-05-04.

[36] Git. https://github.com/, 2023. Hämtad: 2023-04-27.

[37] M Comstedt. Grunderna i scrum–roller, begrepp och aktiviteter. *URL: https://onbird.se/grunderna-i-scrum*, 2017.

[38] Grunderna i scrum – roller, begrepp och aktiviteter. https://onbird.se/grunderna-i-scrum/. Hämtad: 2023-05-02.

[39] Ken Schwaber & Jeff Sutherland. Scrumguiden. https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Swedish.pdf, 2020. Hämtad: 2023-01-31.

[40] R.A. Simmons. Software quality assurance (sqa) early in the acquisition process. In *IEEE Conference on Aerospace and Electronics*, pages 664–669 vol.2, 1990.

Bilaga A

Reflektion över systemutvecklingsprocessen

A.1 Projektmetodik

Följande kapitel beskriver projektmetodiken som använts i arbetet med rapporten. Metoden inkluderar källkritik, ramverk och utvecklingsmetodik, versionshantering och tidsplan. Genom att använda en strukturerad metodik har gruppen kunnat hantera projektet effektivt och säkerställt att arbetet utförts i enlighet med planen.

A.1.1 Ramverk och utvecklingsmetodik

Den underliggande utvecklingsmetodiken som gruppen använde sig av för att besvara frågeställningarna och effektivisera arbetet var den agila metoden *Scrum*. Systemet som kunden var ute efter är komplext och projektgruppen hade inte gjort något liknande förut. Därav behöver gruppen regelbundna granskningar med kunden för att se att resultatet liknar kundens vision. Scrum var tacksamt att utgå från då de korta sprintarna gjorde det lätt att byta riktning om kunden ändrade sig eller om projektgruppen inser att målet inte går att nå fullt ut. Gruppen planerade in *Daily Scrum*-möten som gick ut på att alla i projektarbetet talade om för varandra vad de gjort sedan sist och vad de kommer göra. Därmed är alla i gruppen medvetna om hur arbetet går och vad alla jobbar med.

Kunden var inblandad i början av varje sprint, vilket är viktigt då gruppen får en chans att informera om och visa vad de kommit fram till under den föregående sprinten, samt att kunna informera vad de tänker göra under nästkommande sprint. Därmed var kunden inblandad under samtliga sprintar och var med och påverkade i arbetsprocessen. Mest tid arbetade projektgruppen på planering och implementering då dessa kräver mer tid. När gruppen var noggran vid planerings- och implementeringsfasen krävde sprintgranskringen och sprintåterblicken mindre tid. Detta förutsatte att arbetet följer planeringen och inget oväntat händer.

En nackdel med Scrum var att det var svårt att anpassa arbetet till kundens behov då gruppen inte hade tillräcklig kunskap eller erfarenhet av det specifika problemområdet. Gruppen fann det svårt att fylla backloggen med stories då ingen i projektgruppen visste vilken väg gruppen skulle ta för att nå målet. Detta ledde till att de utdelade uppgifterna ibland var tillräckligt otydlig att inget arbete kunde ske. Det var inte ovanligt att det stod Skapa chattbot"eller Fixa desktopapplikation"i backloggen. Alla i gruppen var överens över att skapa en chattbot och att en desktopapplikation skulle användas, men ingen visste vilken väg som var den bästa vilket inte gjorde det möjligt att bryta ner uppgifterna i mindre delar. En lösning på detta hade varit att inkludera utvecklare som arbetat med likande projekt tidigare eller att minska arbetsmängden. En annan lösning hade varit att lägga mer tid på planering.

A.1.2 Versionshantering

Projektgruppen använde versionshanteraren Git, med hjälp av Github och Github Desktop, för att organisera och samarbeta kring projektet [36]. För att säkerställa en strukturerad och fungerande kodbas utnyttjade utvecklarna möjligheten att skapa grenar. Genom att skapa grenar kunde varje utvecklare experimentera och implementera nya funktioner utan att riskera att påverka andra delar av koden eller störa den befintliga funktionaliteten i projektets huvudgren.

Projektet bestod av flera separata lösningar som skulle integreras i ett gemensamt system. För att hantera detta skapade varje delsystem sin egen gren från huvudgrenen. Gruppen valde medvetet att inte slå samman grenarna förrän respektive delsystem var färdigt. Detta tillvägagångssätt säkerställde att huvudgrenen alltid var stabil och fungerande. Vid projektets slut kombinerades alla grenar för att skapa det totala systemet, vilket återigen skedde på projektets huvudgren.

Projektgruppen hade som mål att uppnå en mer kontrollerad struktur i projektplanen. Även om Git och Github användes flitigt, var användningen något ojämn. Pull-requests, som eftersträvades för att möjliggöra översikt och kontroll av koden, utnyttjades inte i den utsträckning som önskades. Detta ledde till att varje utvecklare blev specialist inom sitt eget område, vilket resulterade i att de andra utvecklarna hade svårt att granska och kontrollera varandras kod.

Att förgrena koden för huvudgrenen användes flitigt och fungerade väl för att ge utrymme åt utvecklarna att implementera och experimentera med sina egna funktioner utan att påverka resten av gruppen. Detta ledde dock till en ökande mängd grenar, ibland upp till åtta stycken samtidigt. Den ökade kodvolymen gjorde projektet svårt att navigera och gruppen tvingades vid flera tillfällen rensa upp både grenar och kod. Detta var delvis en följd av de kompatibilitetsproblem som uppstod, vilket resulterade i flera lösningar för samma komponenter. I dessa situationer hade gruppen svårt att avgöra vilka lösningar som skulle användas.

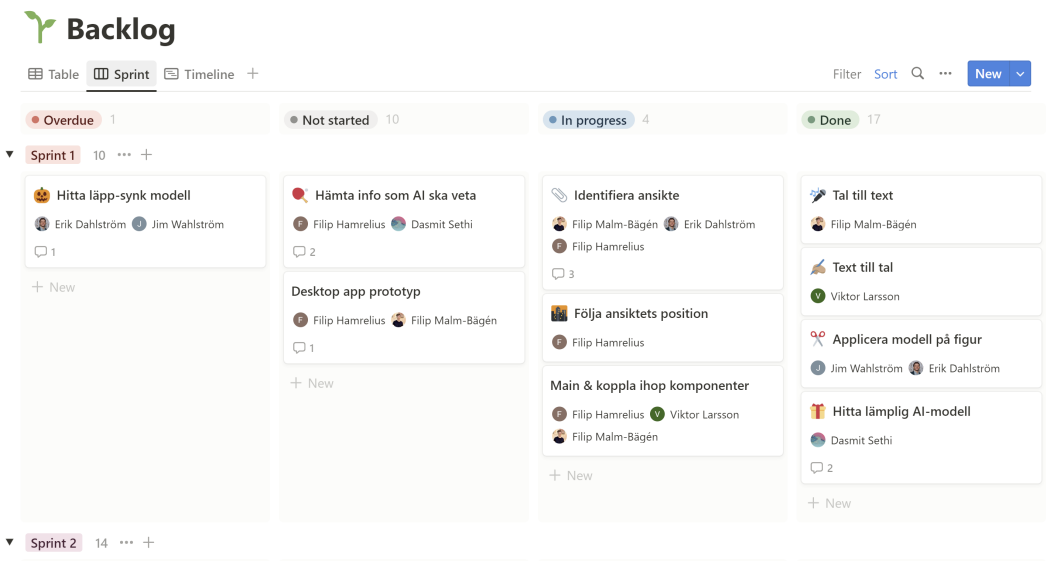
För att sammanfoga de olika komponenterna krävdes en djup förståelse för desktopprogrammets utveckling. Tyvärr hade de utvecklare som inte var involverade i utvecklingen av en specifik komponent svårt att integrera sina delar i systemet. Detta ledde till att pull-requests inte användes, och istället delades och implementerades koden manuellt. Det var en utmaning att säkerställa en smidig integration av komponenterna utan att ha en enhetlig process för att granska och godkänna ändringar.

A.1.3 Kvalitet och krav

Gruppen har utifrån dialog med kunden behandlat och prioriterat de uppdaterade kraven av projektet genom att använda webbapplikationen *Notion*. Notion kan beskrivas som ett allt-i-ett annteckningsverktyg för att öka produktiviteten inom till exempel organisera tankar och idéer för att hantera projektet på bästa möjliga sätt. Under varje kundemöte har annteckningar gjorts i Notion av Produktägaren för att samla kundens tankar, idéer och krav. Verktyget är en gemensam plattform som samtliga gruppmedlemmar kan ta del av under hela projektetsgång.

Vid kundmötets-slut förs dessa krav in i gruppens sprint-backlog sida som skapats i Notion. *Sprint-backloggen* ska ge en översikt och förtydliga produkt-backloggen för den givna tidsperioden. Likt *Kanban*-metodiken, formar gruppen sprint-backlogen utifrån den givna tidsplanen, Figur 37, som innehåller fyra stycken sprint-intervaller. Varje sprint-intervall innehåller fyra stycken kolumner, som säger vilka komponententer som ska göras under följande sprint, vem som är ansvarig för respektive komponent och vad komponenten har för status för den specifika sprinten. Detta visas i Figur 36[37].

Således används sprint-backloggen av samtliga utvecklare för att se vem som ska göra vad, samt att Scrum-mastern och Produktägaren kan prioritera och kontrollera att utvecklingsprocessen faller i fas med planeringen och kundens målbild. Målsättningen är att varje komponent för varje sprint ska vara



Figur 36: Notion, sprint-intervaller.

"in progress" innan halva sprinten har på gått och att samtliga komponenter är klara innan sprintens slut. Om en uppgift visar sig ta längre tid än vad som har planerats för den sprinten, då måste Scrum-mastern tillsammans med Produktägaren ta ett beslut om hur gruppen ska gå tillväga eller kontakta kunden och fråga om åtgärder.

Kvalitetssäkring kommer att göras vid två olika tillfällen. Tillfälle 1, sker efter varje genomförd sprint där nya komponenter ska vara färdiga. Alla färdiga komponenter ska därefter granskas av andra gruppmedlemmar. Utvecklarna granskar funktionaliteten, letar efter förbättringar och försöker upptäcka eventuella buggar som den ansvarige utvecklaren kan ha missat eller inte lyckats lösa.

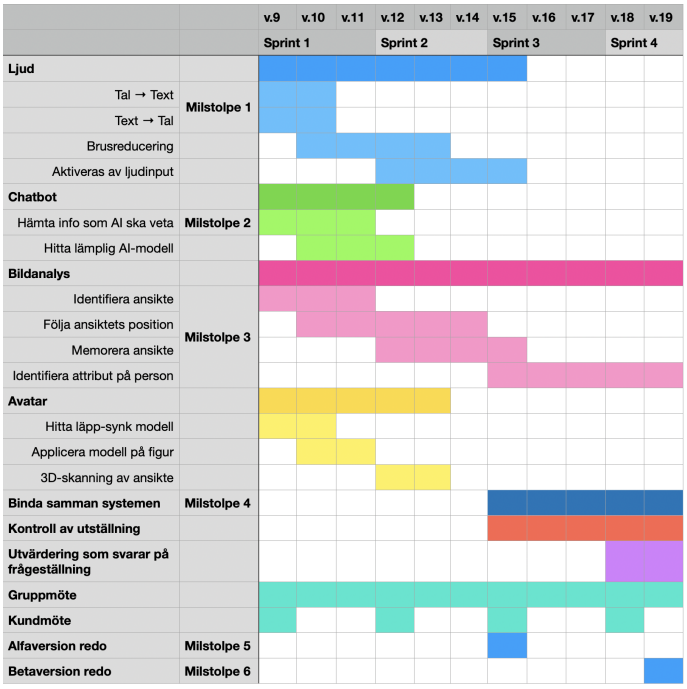
Tillfälle 2, kommer att vara när en merge ska göras med huvudbranchen för den utvecklarna som är klar med en huvudkomponent. Den ansvarige utvecklaren informera detta till Scrum-mastern innan en merge inleds. Vid en potentiell merge-konflikt måste den utvecklaren som är ansvarig vara beredd att lösa problemet. Efter lösta konflikter eller om inga konflikter uppstod vid en merge, ska utvecklarna ha som rutin att köra igenom alla funktioner av sin kod som en sista kvalitetssäkring. Av den orsaken att koden ska vara korrekt och felfri inför momentet då alla komponenter ska bindas samman.

A.1.4 Tidsplan

Tidsplanen, i Figur 37, framtogs enligt GANTT-modellen för att ge ett överskådligt perspektiv av projektets gång. Informationen som framgavs var relaterade till projektets milstolpar samt de tidsbegränsningar för när respektive milstolpe skulle uppnås. Milstolpar och tillhörande funktioner ses i det vertikala ledet till vänster, medan tidsperioden syns i det horisontella ledet i form av en-veckorsperioder för varje cell. I samband med att projektet utvecklades utefter metodiken Scrum visades även Sprintperioderna i planen.

Tidsplanen fungerade bra i början av projektet. Det var inget praktiskt märkbart som saknades när gruppen använde sig av den vid möten. Däremot var det mer en utmaning att gruppen hade underskattat utvecklingshastigheten av en del funktioner som tal-till-text, text-till-tal, 3D-modell för avataren och att identifiera ansikten. Tillhörande milstolpar uppnåddes väl innan planerat, vilket är generellt sätt en bra sak, men det innebar att tidsplanen inte hade en stor funktion under den andra halvan av projektets förlopp.

Underskattandet av utvecklingshastigheten för funktioner kommer troligtvis från att gruppen består av



Figur 37: GANTT-schema för utvecklingen av den virtuella guiden.

utvecklare vars kunskap inom området är minimal. För att kunna göra en mer informerad bedömning kunde gruppen ha vänt sig till handledaren för att få deras åsikt. Detta är en resurs man kan utnyttja i framtida projekt då man inte vill göra en bedömning där utfallet blir det motsatta, alltså att gruppen missar tidsgränsen för milstolpar istället.

A.1.5 Organisation

Organisationen består av sex personer som har tilldelats i tre olika roller. Alla sex kommer att fungera som utvecklare, men två av dem agerar också som Scrum-master och Produktägare. Dessa två personer har som huvuduppgift att säkerställa att projektet går enligt plan och att de kan erbjuda lösningar på eventuella problem som uppstår. Varje utvecklare har också tilldelats ett specifikt område att undersöka och bli expert inom ramen för utvecklingsprojetet. Kunden finns över organisationen och är i kontakt med Produktägaren.

A.1.6 Milstolpar och leverabler

- Milstolpe 1: Implementera tal-till-text- och text-till-tal-modellen.

  Detta är en av grundpelarna för projektet. Det är viktigt att gruppen får detta att fungera tidigt för att få en konceptvalidering. Milstolpen är nådd när gruppen har ett system som konverterar tal till text och tvärtom. Enligt Gantt-schemat som visas Figur 37, ska detta vara klart v.10.

- Milstolpe 2: Hitta information som AI-modellen ska hämta data från.

  När användaren ställer frågor om utställningen är det viktigt att den virtuella guiden ska vara till hjälp och kunna svara på alla sorters frågor. Denna information måste gruppen hitta. I denna milstolpe ingår även att hitta en AI-modell som ska analysera inmatningsdatan och returnera lämpligt svar till användaren. Detta kommer påbörjas i sprint 1, milstolpen ska vara avklarad senast v.11.

- Milstolpe 3: Implementera bildanalysen.

  Även detta är en av grundpelarna för ett lyckat projekt. Systemet måste identifiera människor som går förbi, samt följa dess position. Detta är viktigt för att avataren i sin tur ska kunna följa personen. Utöver det vore det bra om systemet kunde memorera ansikten och identifiera attribut hos personerna, till exempel som färg på tröjan eller om personen har glasögon eller inte. Milstolpe 3 kommer att delas in i delmål under rubriken Bildanalys, vilket illusteras i Figur 37. Ett delmål för samtliga sprint perioder och ska vara klart i slutet av varje sprint period.

- Milstolpe 4: Binda samman delsystemen.

  Ett viktigt mål är att dessa system ska kunna kommunicera med varandra. Detta ska leda till att systemet blir mer pålitligt och upplevas enhetligt av användarna. En avatar ska skapas för att underlätta detta och ge liv till systemet. Avataren skapas genom att binda samman bildanalysen, ljudintagningen och AI-modellen. Binda samman dessa delsystemet kommer att ske när samtliga delsystemen är helt färdiga. Utifrån Figur 37 ska detta påbörjas under starten av sprint 3, alltså v.15.

- Milstolpe 5: Färdigställd alfaversion av produkten.

  Detta är ett viktigt mål som innebär att projektgruppen har färdigställt en version där alla delmoment är sammankopplade. Systemet ska vara tillräckligt stabilt för att testa helheten inom utvecklingsgruppen, Figur 37 visar när milstolpe 5 ska vara klart.

- Milstolpe 6: Färdigställd betaversion av produkten.

  Följande milstolpe är nått när projektgruppen visar upp systemet för utomstående aktörer och utför användartester. När denna milstolpe är nådd, är projektet nära att kunna ställas ut på godtycklig utställning. Figur 37 visar när milstolpe 6 ska vara klart.

A.1.7 Mötesprinciper

Då den underliggande utvecklingsmetodiken är Scrum kommer gruppen använda sig av det ramverket under mötena. I det ramverket ingår Daily Scrum, dagliga planeringsmöten. Mötena är korta och får inte överstiga 15 minuter i längd för att hålla fokus och effektivitet. Varje deltagare svarar på tre enkla frågor: ”Vad har jag gjort sedan senaste mötet?”, ”Vad planerar jag att göra idag?” och ”Har jag några hinder?”. Scrum-Master kommer att leda mötet och visa sprint-backloggen för att göra det lättare att följa projektets framsteg. Dessa korta möten hjälper gruppen att snabbt identifiera hinder och hur de tillsammans kan hantera dessa, vilket gör att teamet kan planera tiden till nästa Daily Scrum på ett effektivt sätt. Daily Scrum är ett enkelt men effektivt sätt för teamet att hålla sig uppdaterade och arbeta tillsammans på ett strukturerat sätt under varje sprint, som i detta projekt varar i tre veckor [38].

Var tredje vecka, i början av varje sprint, kommer gruppen planera in ett möte med kunden. Mötet garanterar att projektet går i den riktning som kunden förväntade sig och ger projektgruppen en chans att komma med egna idéer. Det är lämpligt att ha möte i början av varje sprint för att kunna visa kunden det arbete som gjorts, samt presentera tankarna för kommande sprint.

Om inte kunden har något att säga till om kommer beslutstaganden baseras på majoriteten. Projektet kommer bygga på en platt organisation där allas röster respekteras. Detta är viktig att allas röster räknas i beslut för att alla ska känna sig inkluderade. Om inte, riskeras motivationen sjunka vilket skulle reducera effektiviteten.

A.1.8 Dokumentation

Som diskuterades tidigare i rapporten kommer all dokumentation att skötas på den gemensamma plattformen Notion. Det gemensamma verktyget kommer att användas för samtliga möten, sprint-genomgångar och förmedla status av samtliga komponenter i projektet. Följden av att sammanställa all dokumentation på en och samma plattform blir att missförstånd kommer att undvikas.

Dokumentation av kod kommer att ske direkt i koden för respektive komponent. Varje klass eller kodstycke ska ha en förklarande kommentar som beskriver vad funktionen gör. Att dokumentera i koden kommer underlätta processen för utvecklare som ska sätta sig in i koden eller bygga vidare på den. Kommentarerna kommer att formuleras kort och koncist, med innehåll av funktionsnamn som beskriver vad en funktion gör samt variabler som döps efter den information som de innehåller.

A.2 Filip Malm Bägén

I rollen som Scrum Master i projektet var min uppgift att se till att teamet följde det gemensamt uppsatta ramverket och höll en hög produktivitetsnivå. Det inkluderade att skapa tydliga mål och user stories, och att tilldela dessa till rätt personer med rätt deadline. Som utvecklare arbetade jag med att skapa en tal-till-text-modell, implementera en grundläggande ansiktsigenkänning och designa gränssnittet för desktop-applikationen.

I teorin skulle jag skapa och underhålla alla user stories i backloggen för att se till att jag själv och övriga utvecklare nått vårt gemensamma mål inom tidsramen. Planeringen bör ha varit planerad för att hantera de olika utmaningarna som uppstår i ett projekt. I litteraturen har Scrum visat sig vara ett framgångsrikt ramverk för att hantera komplexa projekt och upprätthålla en hög produktivitetsnivå. Enligt planeringen skulle även samtliga utvecklare kommentera sin kod och minst en utvecklare ska granska och godkänna all kod vi varje pull request. Detta hände aldrig på grund av de snabba förändringar som skedde samt tidsbristen. Eftersom att vi konstant utforskade ny mark var det inte orimligt att hela funktioner och filer raderades för att arbetet behövde gå i en ny riktning. Därmed var det inte lämpligt för utvecklarna att behöva vänta in varandras kod och kontrollera den i och med varje pull request.

Vi stötte på utmaningar under projektet, vilket har lett till förändringar i vår arbetsmetod. En av de största svårigheterna var att vi inte visste vilka steg som krävdes för att nå vårt mål eftersom ingen av oss hade arbetat med något liknande tidigare. Det blev därmed upp till varje utvecklare att själv testa en lösning i taget tills att någon lösning gör att vi kommer närmare målet. Det gjorde att Scrum blev mer av ett hinder än ett verktyg för oss. Den fundamentala bristen på tidigare erfarenheter inom området gjorde det irrelevant att skriva user stories och hålla i daily scrums. Hade vi varit erfarna inom området hade vi kunnat lägga mer tid på att få en mer fungerande produkt som bättre passar syftet.

A.3 Erik Dahlström

I rollen som Produktägare i projektgruppen var min uppgift att ha extra kontakt med kunden. För det här projektet hade inte jag ensamma kundmöten efter varje sprintåterblick, utan hela gruppen var med på så gått som samtliga möten. Däremot har jag som Produktägare behövt framföra ytterligare information till gruppen som kunden endast har skickat till mig i form av länkar till användbara tal-modeller samt sett till att kundens uppdaterade mål för projektet uppfylls. Som utvecklare skulle jag jobba med två huvudkomponenter i projektet, vilket var avataren och bildanalys. Över projektetsgång

övergick fokuset till att endast fokusera på avataren.

Utifrån teori till praktik har produktägar-rollen och utecklarrollerna inte varit helt optimala för det här projektet. Den faktiska arbetsprocessen har gjort att rollen inte har haft den påverkan som den vanligtvis har i en Scrum-modell. Gruppen har stött på många typer av utmaningar för samtliga huvudkomponenter något som har lett till förändringar i arbetsmetoden. Det största faktorn för svårigheterna har varit gruppens förkunskaper för att nå kundens samt våra egna mål, vilket har resulterat i att gruppen har inte kunnat strukturera en tillräckligt detaljerad backlog jämfört med den som har använts.

Gruppen hade som utgångsplan att samtliga utvecklare skulle vara involverad i två komponenter under projektet. Eftersom ingen hade de förskunskaperna som hade varit nyttiga att kunna, blev det mer en undersökningsprocess om vad som faktiskt kan fungera utifrån kraven för att sedan testa om det är möjligt att utveckla detta själva. Om vi hade varit mer erfarna hade mer tid av sprintarna gått till att utveckla produktens detaljer för att uppnå de ursprungliga målen.

Kommunikationen samt dokumentationen är något som fungerade bättre till en början av projektetsstart men som blev sämre mot sista veckorna. Webbapplikationen Notion användes för att dokumentera hur backloggen skulle struktureras, utifrån dess sprint-perioder. Samtliga huvudkomponenter delades in i mindre komponenter men på grund av gruppens oerfarenhet blev resultatet oftast att dessa mindre komponenter var fortfarande alldeles för stora för en eller två valda utvecklare per mindre uppgift. En följd av detta blev oftast att gruppmedlemmarna och framförallt Scrum-mastern visste inte hur det gick eller var utvecklaren var i komponentens-process. Detta kunde ha blivit mer effektivt om varje utvecklare kommunicerade och dokumenterade all den här informationen direkt i Notion istället för att för att nämna detta på näst kommande möte med gruppen.

A.4 Filip Hamrelius

I projektplanen hade jag rollen som utvecklare med ansvar för bildanalyskomponenten och desktopapplikationen. Men mina arbetsuppgifter blev mer flexibla än vad som förväntades, mestadels på grund av kompatibilitetsproblemen och förändringar i implementationen. Med anledning av detta tog jag på mig ytterligare ansvar för implementationsprocesserna. Desktopapplikationen blev grunden för sammanlänkningen av komponenterna, således implementerade jag även detta. Jag var också involverad i andra komponenter, bland annat text-till-tal- och tal-till-text-modeller. Detta berodde mer på mitt intresse till de funktionerna, men också för att få dem att fungera tillsammans med desktopapplikationen.

I teorin borde jag ha hållit mig till de stories och uppgifter som loggades i backloggen via Notion, precis enligt projektplanen och sin tur ramverket Scrum [39]. Uppgifterna borde ha implementerats och sedan loggats via en pull request på Github i inväntan på godkännande innan respektive push. I praktiken tyckte jag att denna process blev omständig, mestadels beroende på gruppens storlek och erfarenhet. Det blev problematiskt när varje utvecklare fick sitt eget område. I praktiken krävdes det först forskning inom området för att i sin tur genomföra själva implementationen. Detta var ofta en relativt lång process. Det blev därför besvärligt om jag skulle be en annan utvecklare att granska mitt arbete, eftersom det skulle kräva dubbelt arbete för att godkänna slutsatser och implementationer. Denna typ av praxis fungerar bra i teorin när man har utvecklare och ingenjörer med olika erfarenhetsnivåer, men också mer erfarenhet inom området, gärna där en av dem har i enskild uppgift att uppgift att granska och kritisera arbetet.

Utöver diskueratade avsteg kunde kommunikationen ha skett mer effektivt. Projektplanen menade på en flitig användning av Notion, något som gruppen la en del tid på under mötena men som jag sällan använde i praktiken. Iden var att föra en gedigen backlogg i förhållande till den aktuella sprinten. I backloggen skulle respektive utvecklare vända sig för att hämta en uppgift. Problemet var att ingen i

gruppen hade tillräckligt med kunskap eller erfarenhet inom respektive område för att kunna dela upp uppgifterna i små delar. Notion och backloggen blev därför väldigt övergripande och representerade snarare vad varje person jobbade på i helhet, inte i detaljnivå. För att undvika detta borde man efter sin forskning i området delat upp arbetet i små uppgifter och lagt in de i backloggen. Detta hade inte bara ökat effektivitet och preproducerbarheten men också kommunikationen mellan gruppmedlemmarna. I den genomförda processen blev det ofta att jag satt själv och implementerade en hel komponent utan att kommunicera med gruppen. Detta gjorde jag istället när funktionen var klar och adderad till Github.

A.5 Viktor Larsson

I projektet hade jag rollen som utvecklare och ansvarade för talmodeller, där jag skapade en text-till-tal modell och att programmet skulle starta diskussionen men virtuella guiden genom att säga “tina". Jag höll även på att experimentera med visuell feedback i form av ett pulserande objekt när guiden talade, men det bestämdes senare att detta inte skulle implementeras.

Att skapa en text-till-tal modell tog betydligt längre tid är förväntat då kund gärna ville ha en deepfake-röst. Detta var något jag kämpade med att lyckas implementera, med många olika modeller men fick det inte att fungera och till slut var jag tvungen att nöja mig med en enklare modell som gruppen var okej med.

Även om jag egentligen skulle ha haft mer ansvar för att koppla samman de olika delarna och fixa main filen, var det de andra två som också hade den uppgiften som gjorde detta. Detta berodde mest på att jag till en början var mer fokuserad på att få text-till-tal modellen att fungera medans de andra började implementera andra delarna. I detta kunde jag förbättrat kommunikationen med mina teammedlemmar och frågat om råd för att lösa detta tidigare.

Kommunikationen i gruppen var för det mesta bra, men det finns vissa delar som kunde varit bättre. Vi använde oss av Notion, men som mest uppdaterades under våra möten, vilket gjorde att man inte alltid visste hur långt alla hade kommit. Som det nämns i avsnitt A.1.8 var tanken att utvecklare skulle aktivt använda sig av backloggen för att dela information och möjliga problem som uppstått, men detta följdes inte av flera personer. Veckomöterna fungerade bra och kommunikationen under alla möten fungerade bra, men det kunde ha förbättrats genom att redan under mötet bestämma när nästa möte skulle vara. Även om vi hade veckomöten samma dag många av veckorna, bestämdes inte en exakt dag, vilket gjorde att man inte visste exakt hur veckan skulle se ut, vilket gjorde det lite svårare att planera veckan.

A.6 Dasmit Sethi

I rollen som utvecklare har min uppgift varit att hitta och undersöka förtränade AI-modeller för att skapa ett frågebesvarande system för den virtuella guiden. Som en nybörjare inom området började jag med att utforska plattformen *Hugging Face* och hittade många olika typer av modeller som i sin tur hade många olika varianter. Dessutom så ansökte jag om att få tillgång till modellen **GPT-SW3** via **AI Sweden** där mycket av tiden gick åt. Även om denna modell inte användes till beta versionen av systemet så fick jag lära mig mycket om området, såsom hur modeller som **GPT-SW3** fungerar och att köra dessa typer av modeller kräver stor mängd beräkningsminne som inte riktigt finns för typiska datorer idag. Att använda *Hugging Face* hade sina för- och nackdelar. Då det var väldigt hjälpsamt och effektivt att ha tillgång till så många typer av färdigtränade modeller som kunde importeras i *Python* med väldigt lite kodning, behövs det ändå kunskap och förståelse av AI inom språkteknologi för att faktiskt kunna anpassa en färdigtränad modell till en specifik uppgift. Även om *Hugging Face* har dokumentation och verktyg som *Trainer API* för att finjustera modeller till projektets behov krävs det tid för att förstå sig på verktygen innan man kan applicera det utöver hårdvaran som krävs av modellerna i sig.

A.7 Jim Wahlström

Under förloppet av projektet agerade jag som en utvecklare vars uppgift var att undersöka alternativ till avatar-komponenten och sedan utveckla en lösning med en av dem. Jag undersökte styrkor och svagheter med Furhat Robotics virtuella robot, Virtual Furhat, samt hur man på lämpligast sätt utvecklar mot det. När vi valde att arbeta vidare med Virtual Furhat så fick jag i uppgift att sätta upp en miljö som använder sig utav roboten tillsammans med funktioner skapade av mina kollegor. Det mesta av tiden gick ut på att förstå sig på de olika arbetssätten för roboten och sen läsa, tolka och bygga en lösning utifrån dokumentationen för verktyget.

Enligt projektplanen skulle det utföras tester för kvalitetssäkring vid tre olika tillfällen i projektet. Det första tillfället skulle ske vid sammanfogningar av utvecklingsgrenar. Det andra tillfället skulle ske vid slutet på varje sprint för att testa varandras utvecklade komponenter. Det tredje, och sista, tillfället skulle utföras efter halva projekttiden i form av användartester för en prototyp av virtuella guide-systemet. Kvalitetssäkring inom mjukvaruutveckling tillämpas för anledningar så som oväntade kostnader, ökad pålitlighet av programmet och försäkran om att utvecklingen följer standarder inom industrin. Projekt ska gärna tillämpa kvalitetssäkring vid ett tidigt stadie av projektet för att ha som störst effekt på mjukvarans kvalité [40].

Det enda tillfället vi kom att applicera var användartester i slutet av projektet. Däremot så användes mjukvaran flitigt av de som utvecklade mot det så stora fel har upptäckts och tillrättats, men det finns inga försäkringar om att det dyker upp så kallade *edge-cases*(en situation eller problem som sker under unika förhållanden) där programmet kan komma att missanvändas eller att det slutar fungera.

Bilaga B

Individuella bidrag

Filip Malm-Bägén

- Scrum Master. Planerade in och höll i möten.
- Implementering av tal till text.
- Grundläggande tracking och igenkännande av ansikte.
- Design av GUI samt sammankoppling av komponenter tillsammans med desktopapplikation.

Erik Dahlström

- Grundläggande förundersökning av Furhats olika modeller.
- Förundersökning och testat att konstruera 3D-animeringen.
- Konstruerat en 2D animering i Blender med lip-sync kopplat till en ljudfil.
- Konstruerat den grafiska representationen, två ögonen.
- Produktägare, extra kontakt med kunden.

Filip Hamrelius

- Bildanalyskomponent - Grundläggande tracking och igenkänning av ansikte.
- GUI, design och desktopapplikation.
- Sammankoppling av komponenterna tillsammans med desktopapplikation.
- Tal/Text-komponent: Testing och optimering.
- Text/Tal-komponent: Testing och optimering.

Viktor Larsson

- Implementering av text till tal.
- Skapa så program startar med aktiveringsfras.
- Test att skapa visuell återkoppling, tex som ett pulserande föremål.

- Sammankoppling av komponenter med desktopapplikationen.

Dasmit Sethi

- Implementering av chattbot.
- Undersökning och jämförelse av olika förtränade AI-modeller.
- Finjustering av förtränade AI-modeller.
- Mock-testning av chattbot.

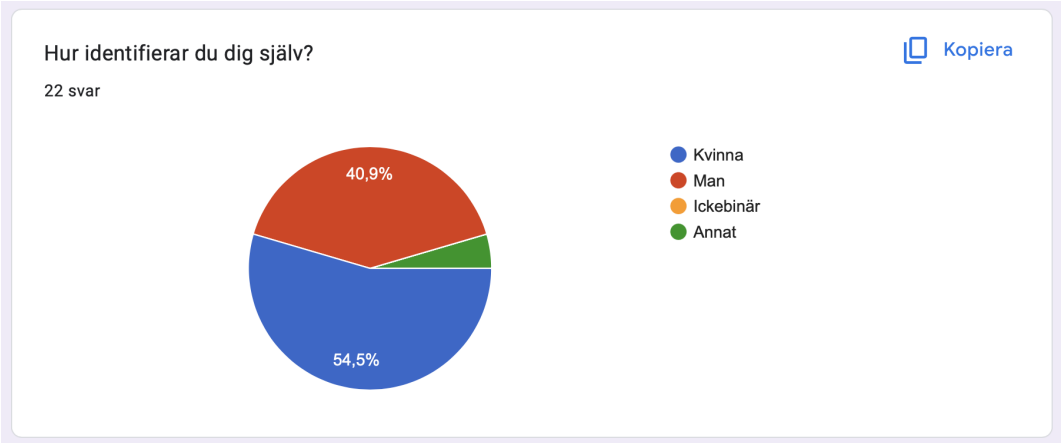
Jim Wahlström

- Undersökt Furhat robotens integrerade AI-modeller och funktioner.
- Undersökning kring de tre arbetssätten för utveckling av Furhat roboten.
- Implementerat den mänskliga representationen av den virtuella guiden via Python Remote API mot Furhat.
- Skapade en test version med OpenAI:s AI-modell gpt-3.5-turbo genom deras externa API.

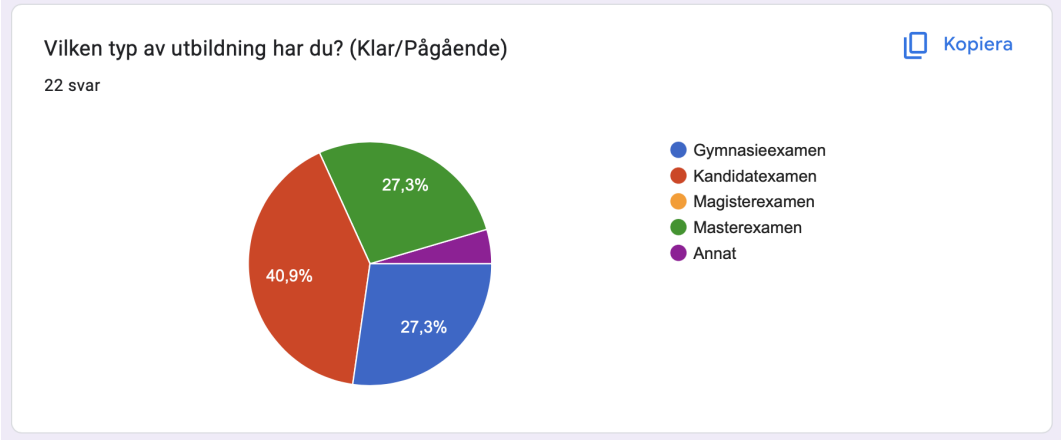
Bilaga C

Svar från användartester

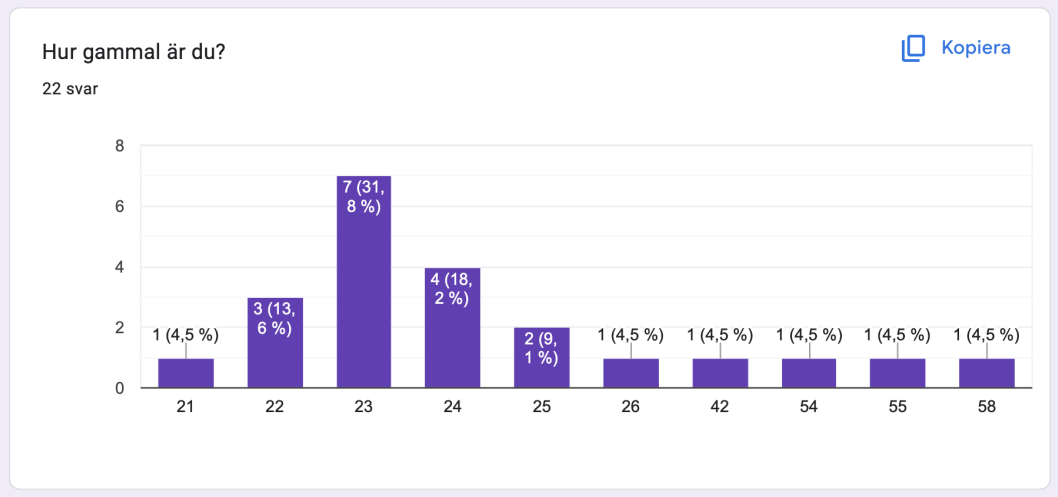
Bilaga C presenterar samtliga svar från de användartester som genomförts.



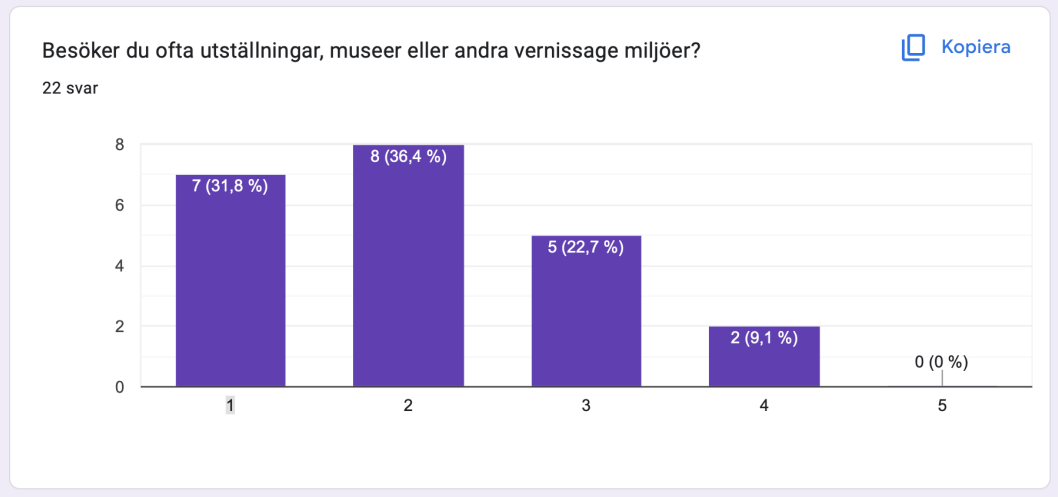
Figur 38: Könsfördelning på användartester.



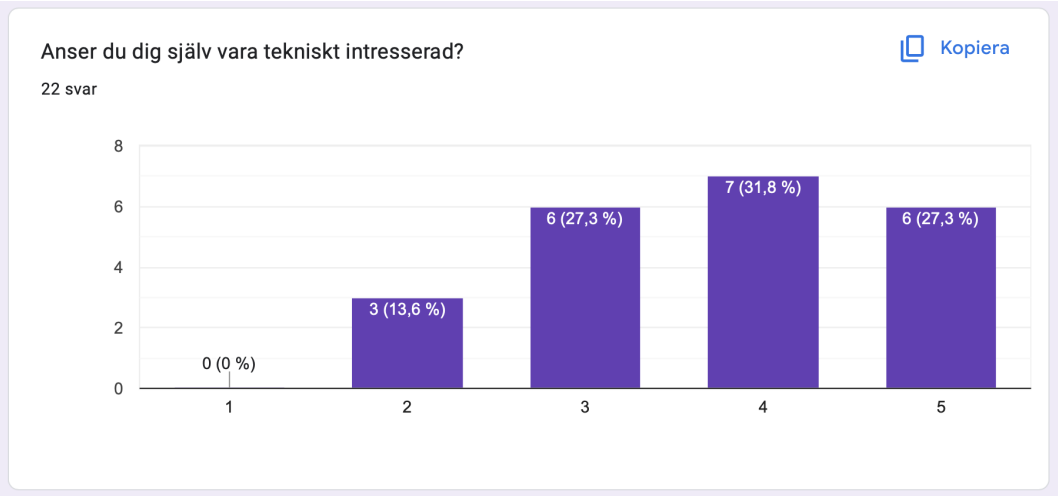
Figur 39: Utbildningsfördelning på användartester.



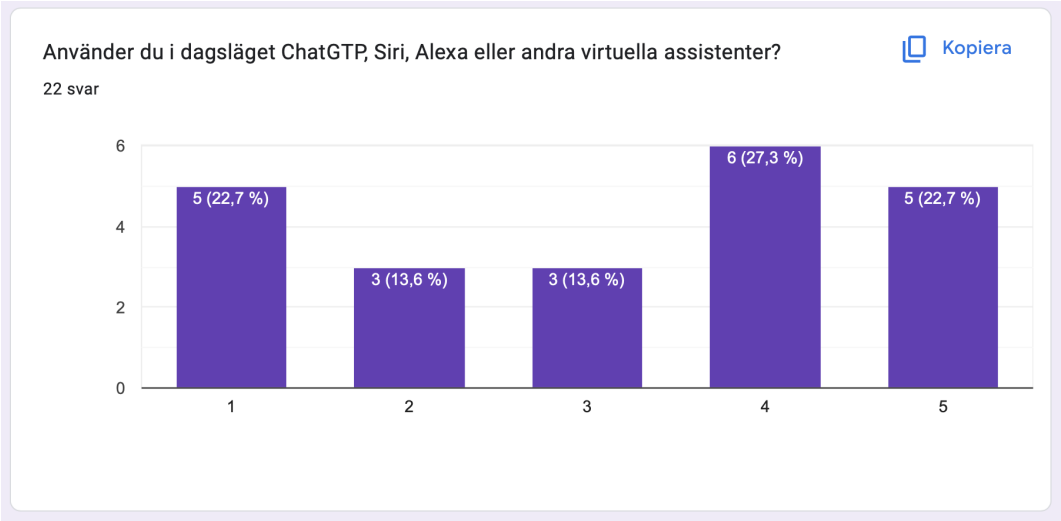
Figur 40: Åldersfördelning på användartester.



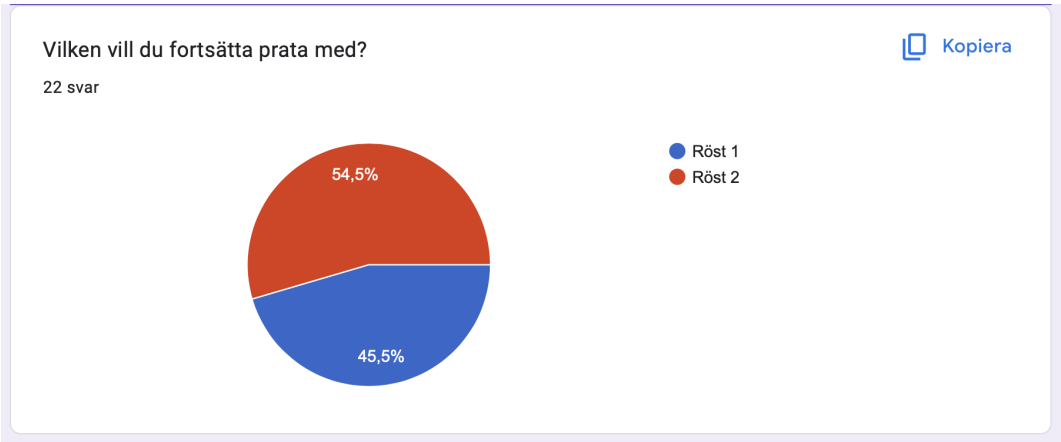
Figur 41: Intresse i utställningar hos användarna.



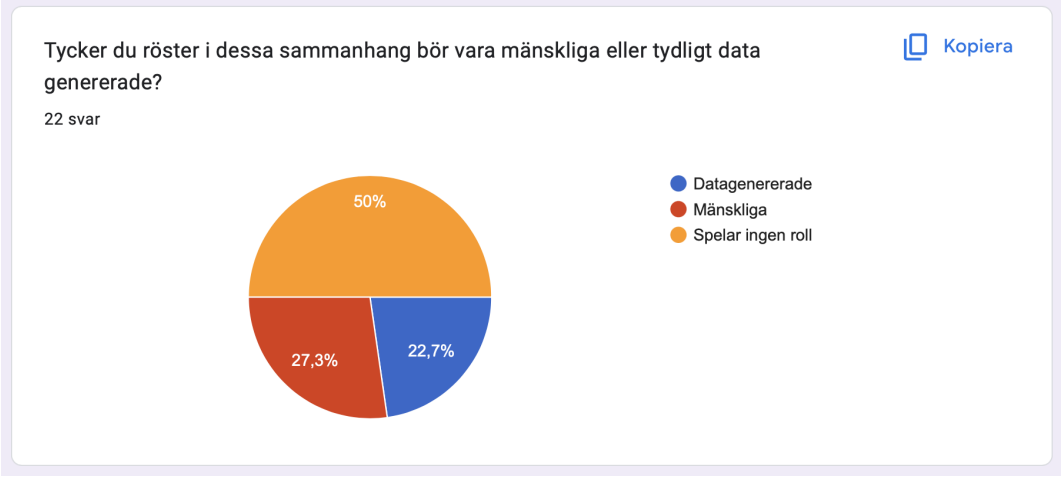
Figur 42: Tekniskt intresse hos användarna.



Figur 43: Användning av tekniska instrument i assistentform hos användarna.

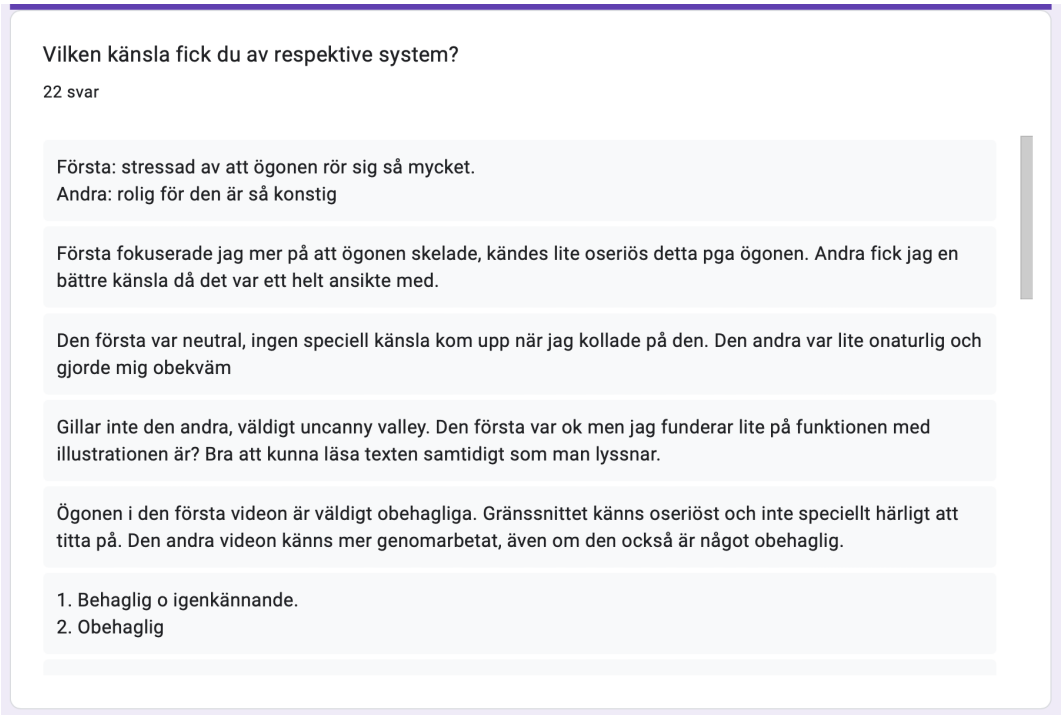


Figur 44: Vilket system föredrar användarna?

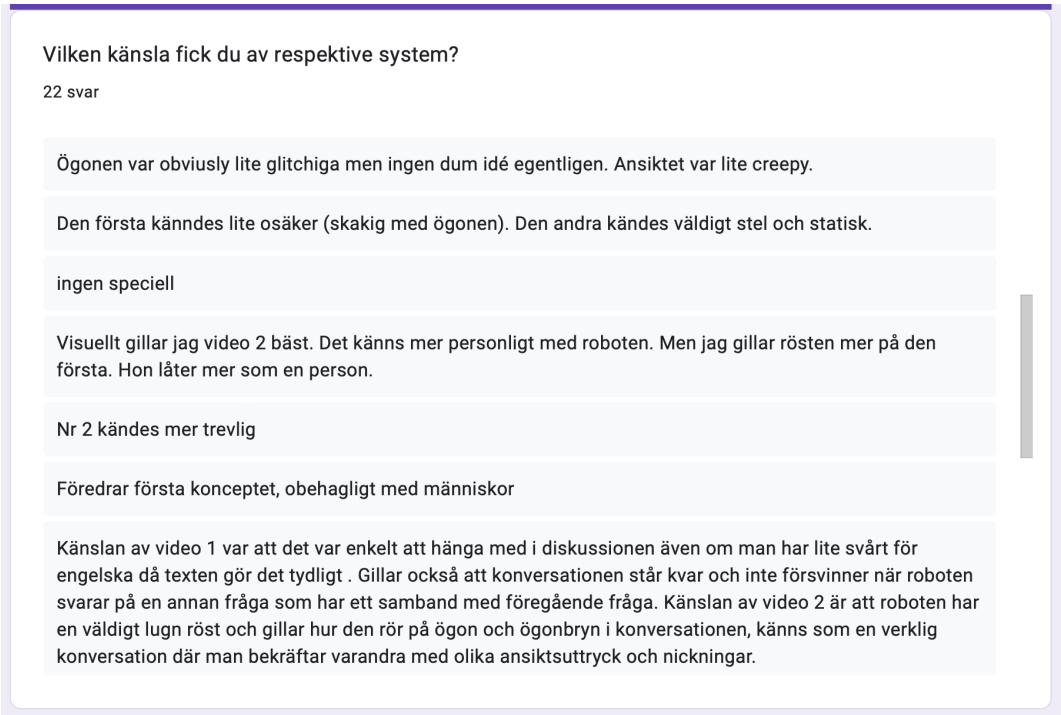


Figur 45: Vilken röst föredrar användarna?

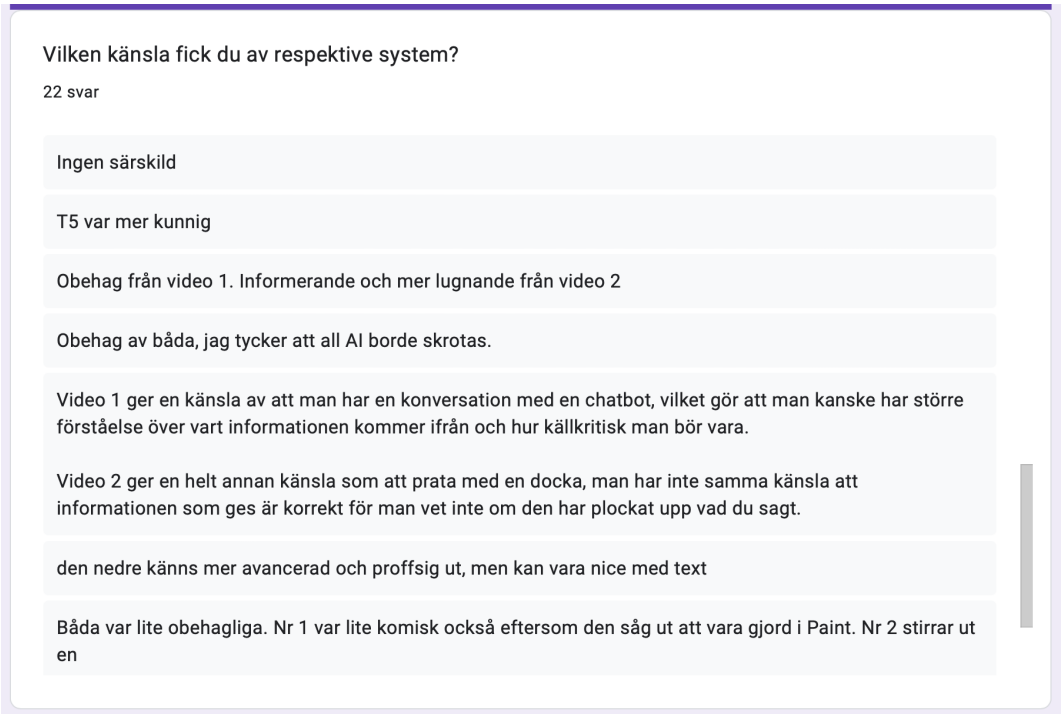
Figur 46: Vilket språk föredrar användarna?



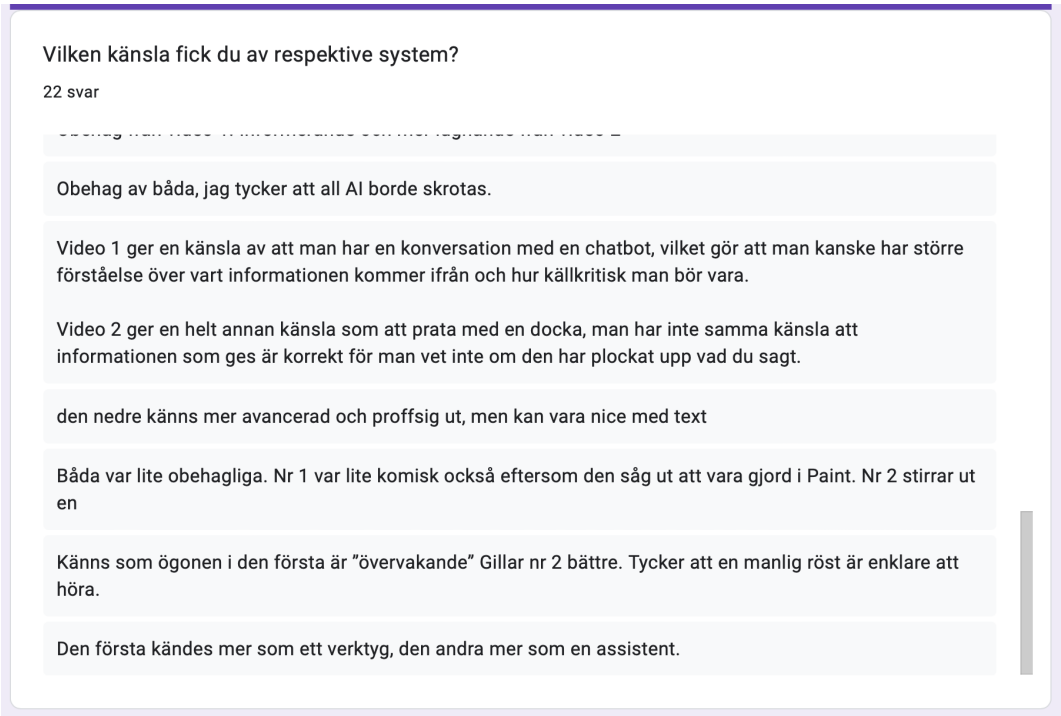
Figur 47: Fri text - del 1 - vilken känsla gav systemet användarna?



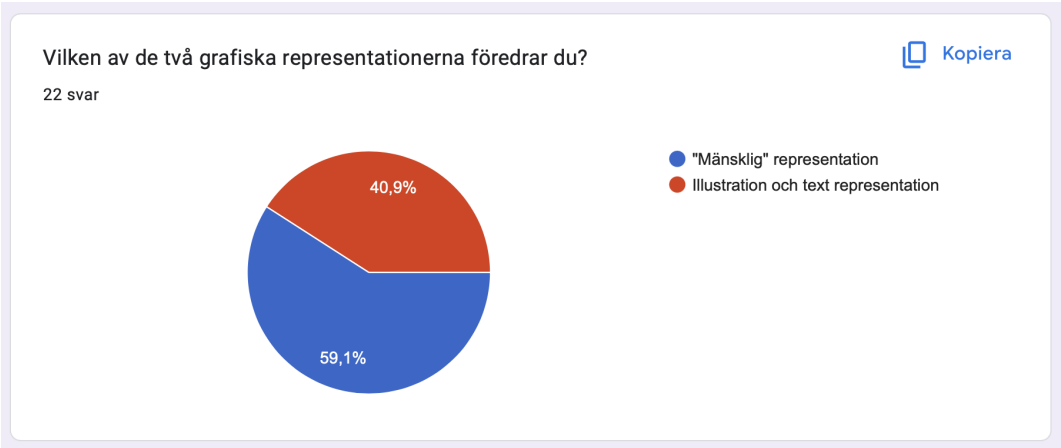
Figur 48: Fri text - del 2 - vilken känsla gav systemet användarna?



Figur 49: Fri text - del 3 - vilken känsla gav systemet användarna?



Figur 50: Fri text - del 4 - vilken känsla gav systemet användarna?

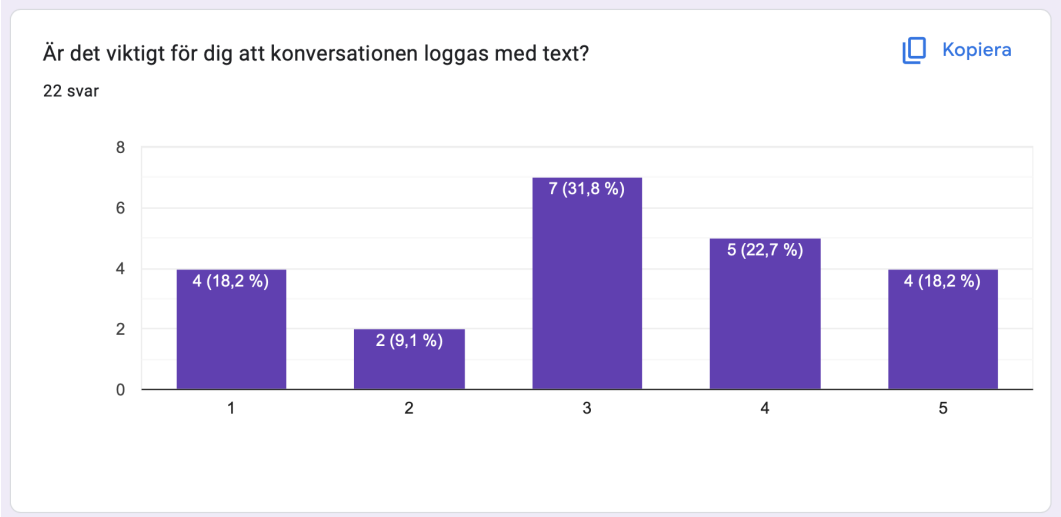


Figur 51: Vilken grafisk representation föredrar användarna?

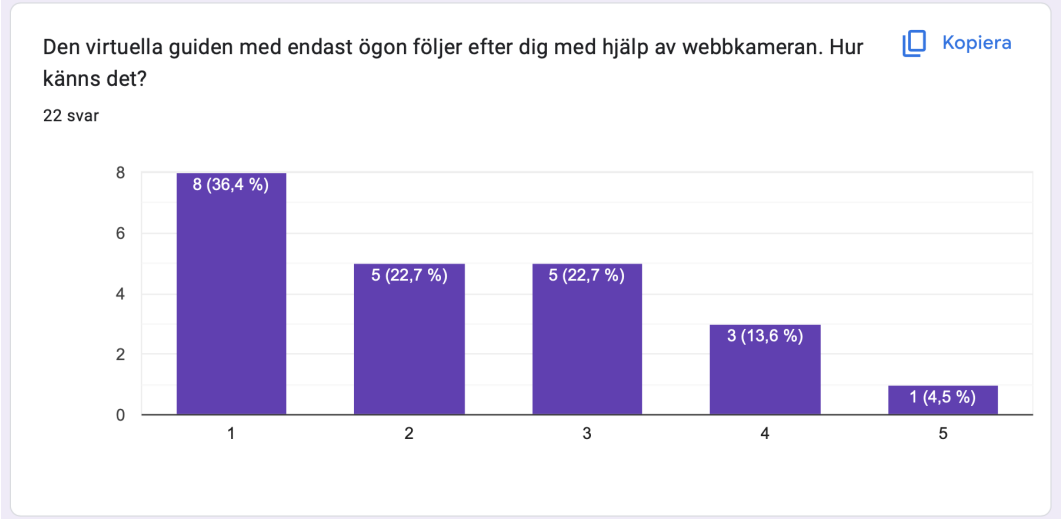
Figur 52: Fri text - del 1 - Varför föredrog användaren just denna grafiska representation?

Figur 53: Fri text - del 2 - Varför föredrog användaren just denna grafiska representation?

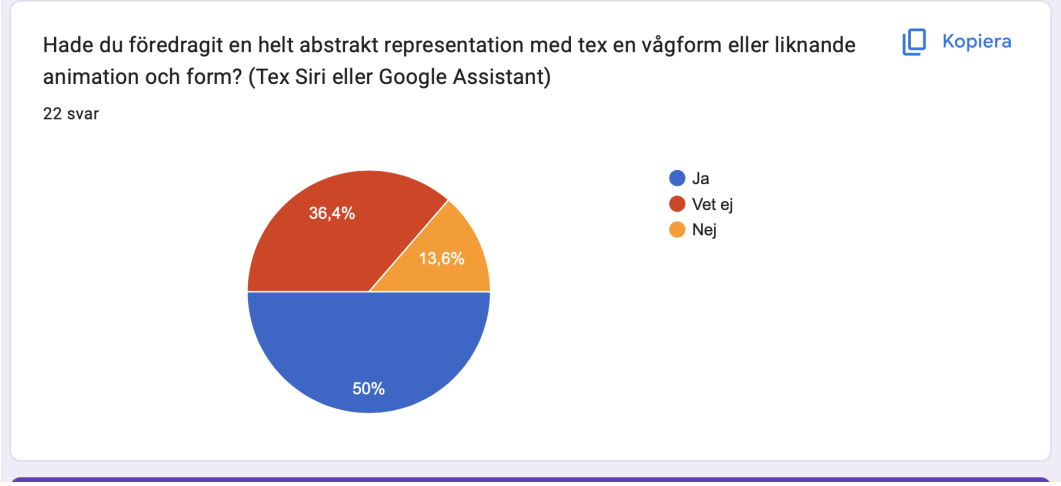
Figur 54: Fri text - del 3 - Varför föredrog användaren just denna grafiska representation?



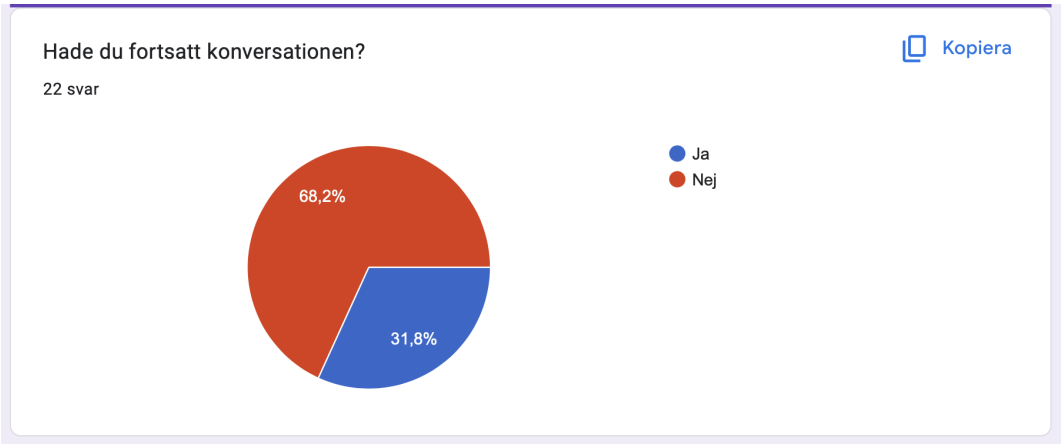
Figur 55: Är det viktigt för användaren att representera konversationen med text?



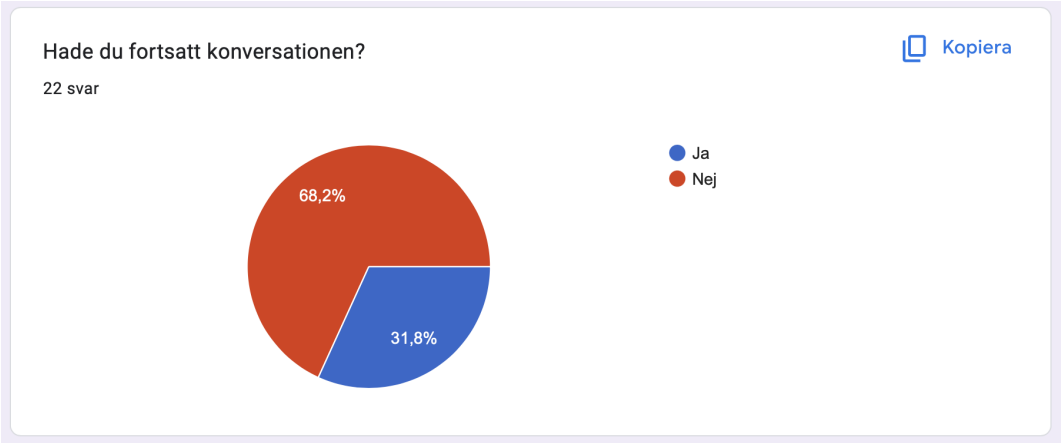
Figur 56: Att följa användaren med kamera, hur uppfattas det?



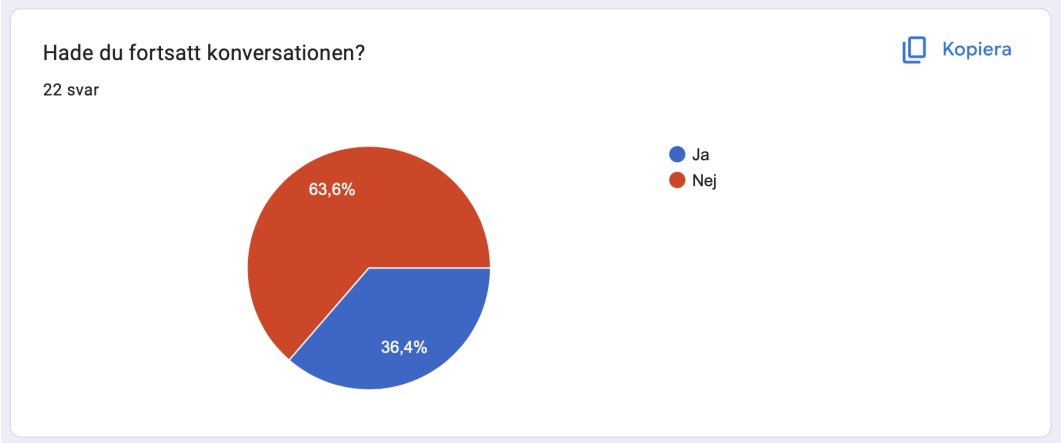
Figur 57: Alternativ grafisk representation.



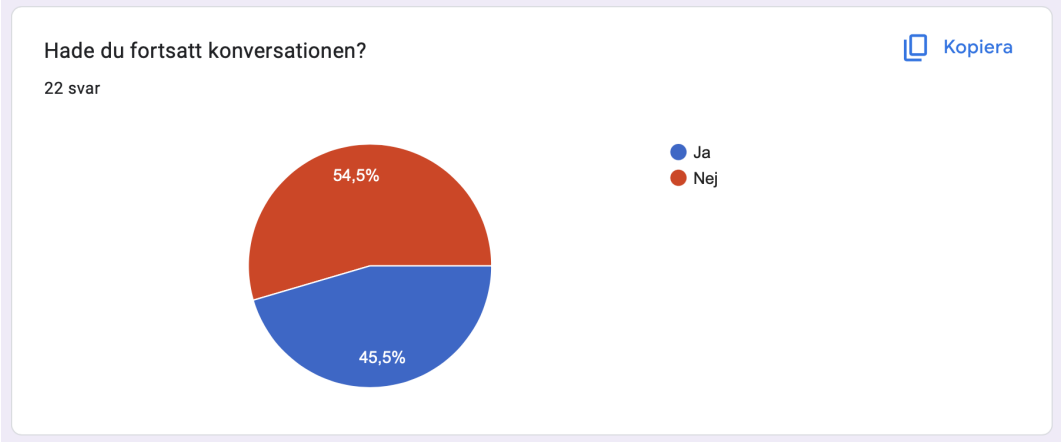
Figur 58: Konversation med 5 sekunder fördröjning.



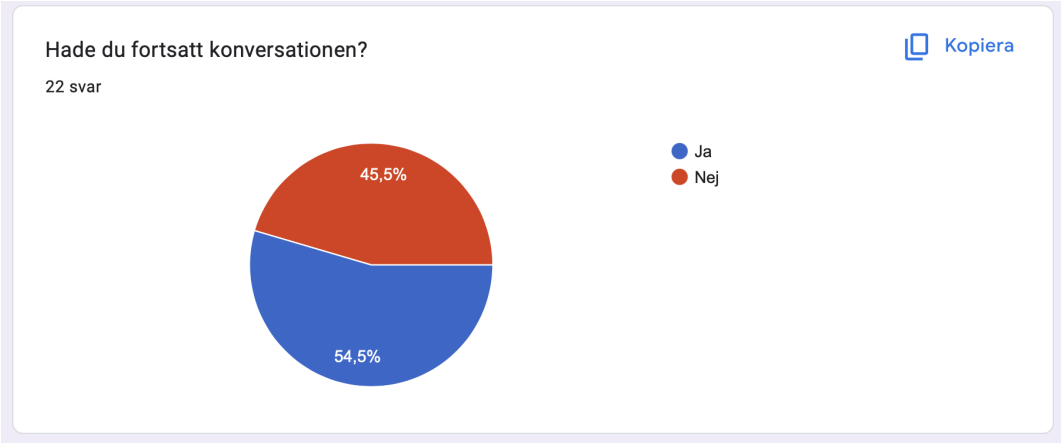
Figur 59: Konversation med 4 sekunder fördröjning.



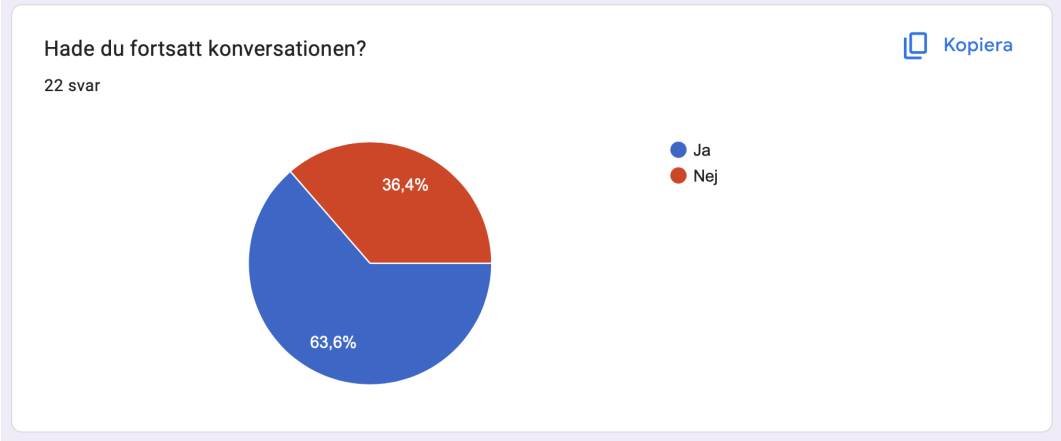
Figur 60: Konversation med 3 sekunder fördröjning.



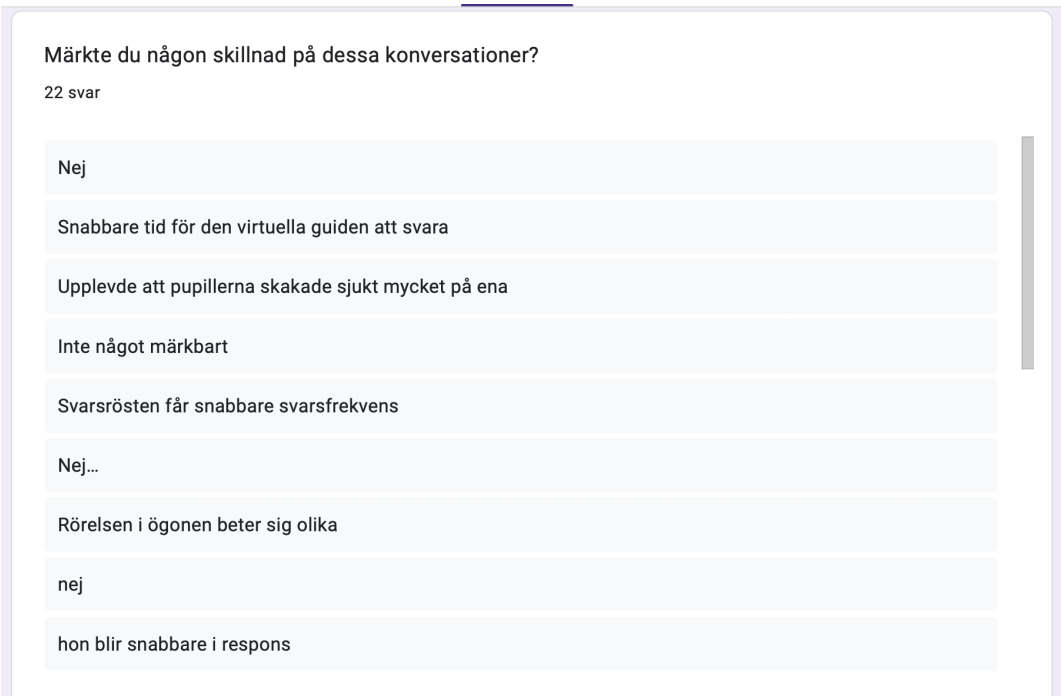
Figur 61: Konversation med 2 sekunder fördröjning.



Figur 62: Konversation med 1 sekunder fördröjning.

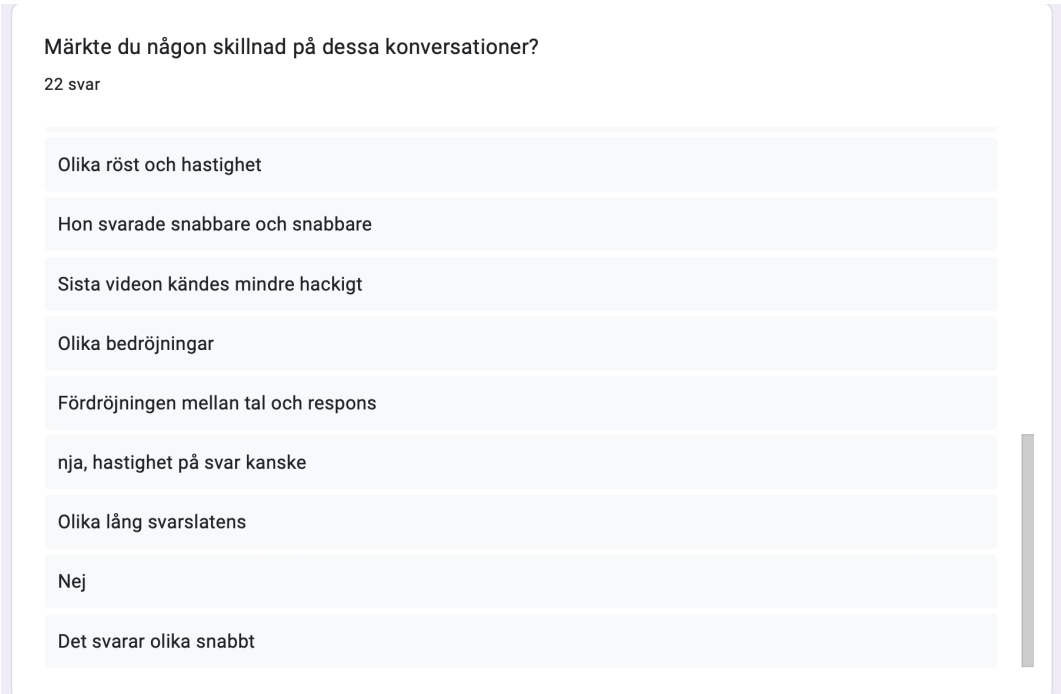


Figur 63: Konversation med 0 sekunder fördröjning.

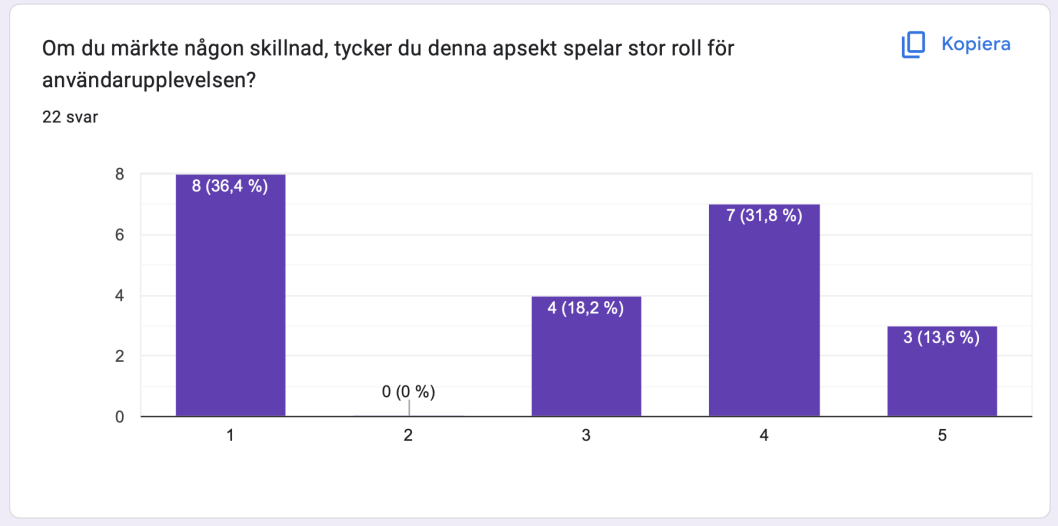


Figur 64: Svar del 1 : Fråga om användaren någon skillnad med födröjningsfaktorn.

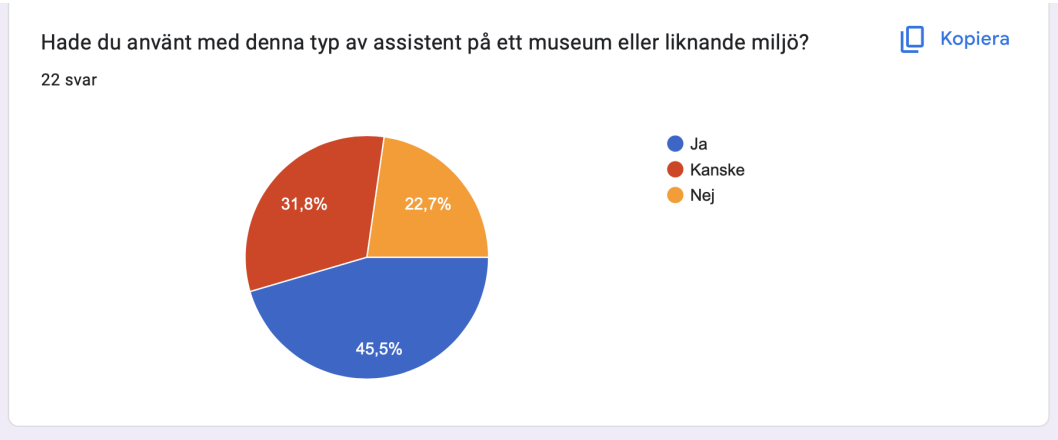
Figur 65: Svar del 2 : Fråga om användaren någon skillnad med födröjningsfaktorn.



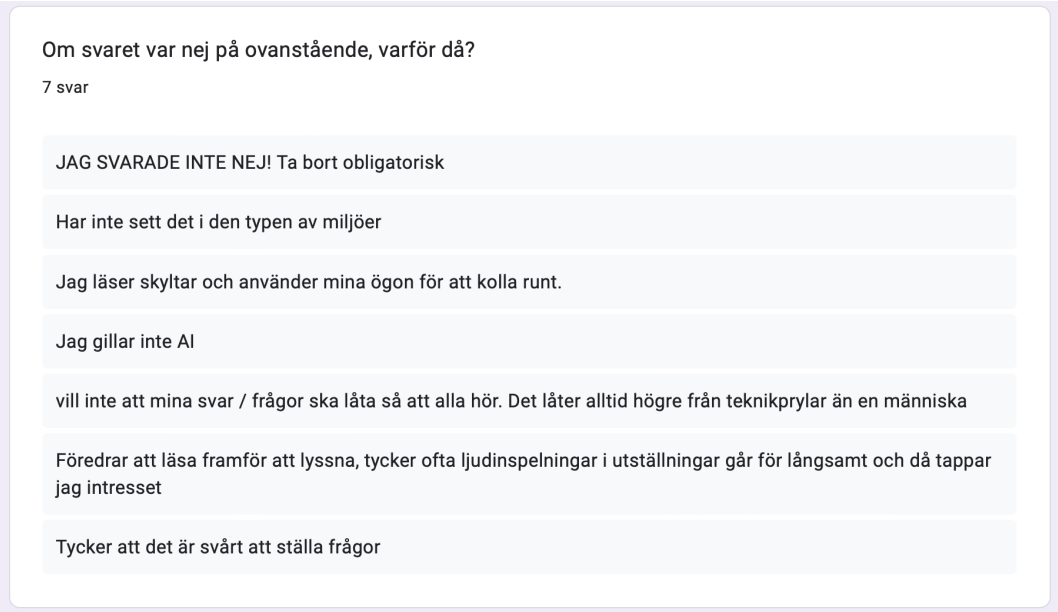
Figur 66: Svar del 3 : Fråga om användaren någon skillnad med födröjningsfaktorn.



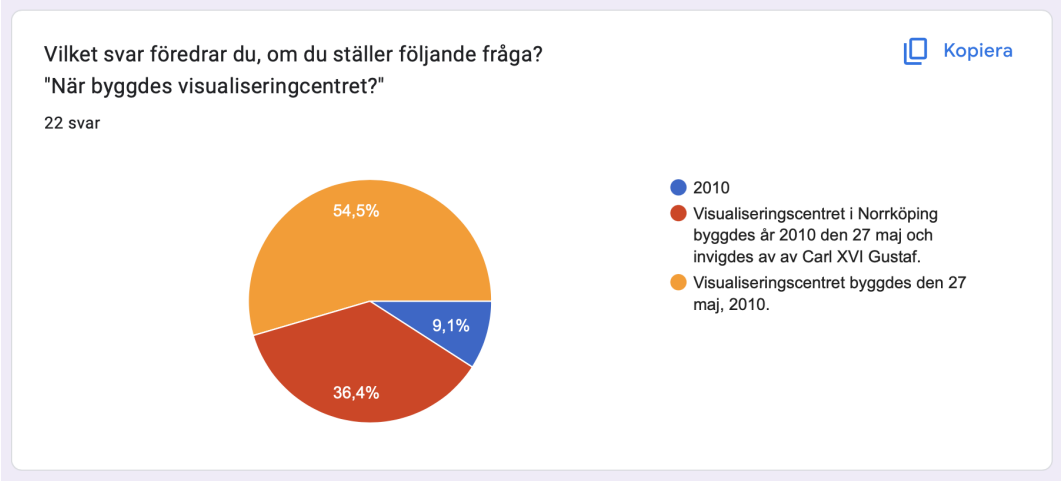
Figur 67: Användarens värdering av födröjningsfaktorn.



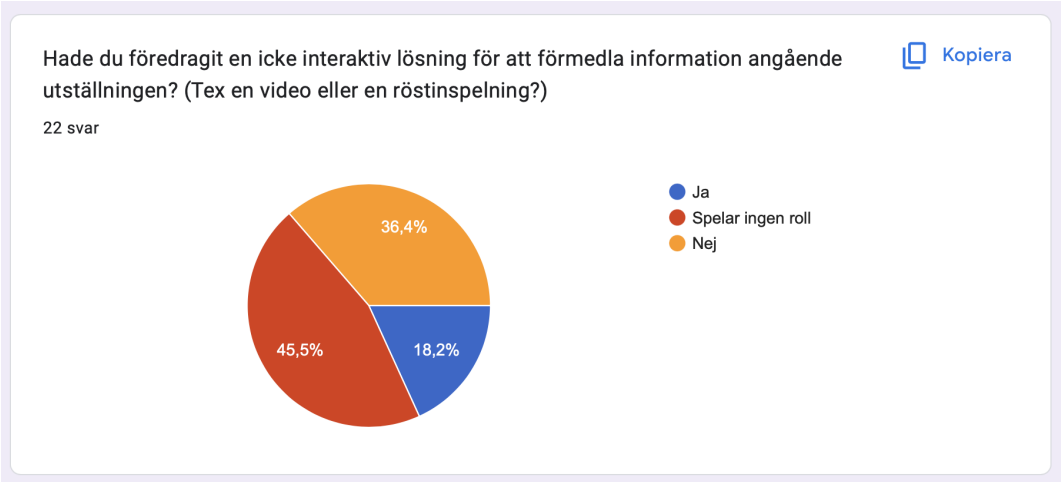
Figur 68: Intresset av denna typ av system.



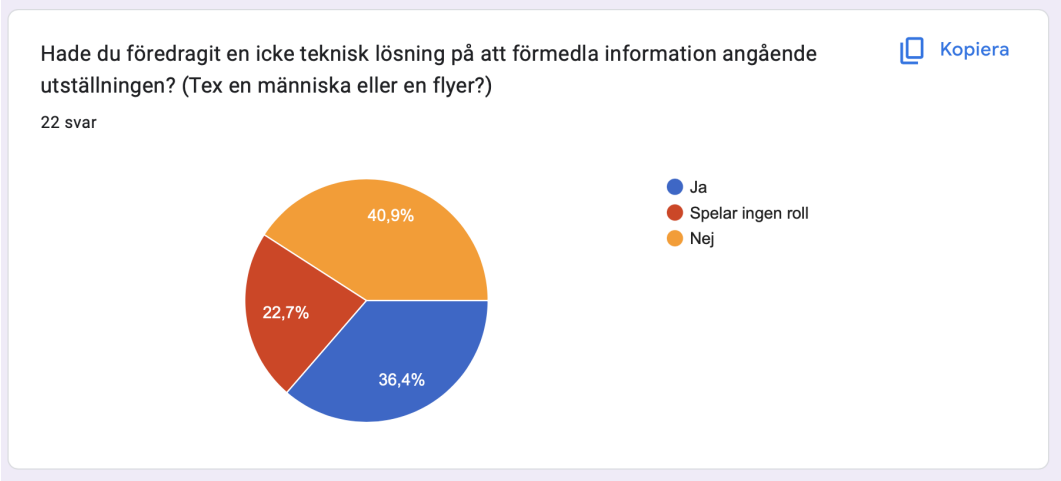
Figur 69: Fri text - Intresset av denna typ av system.



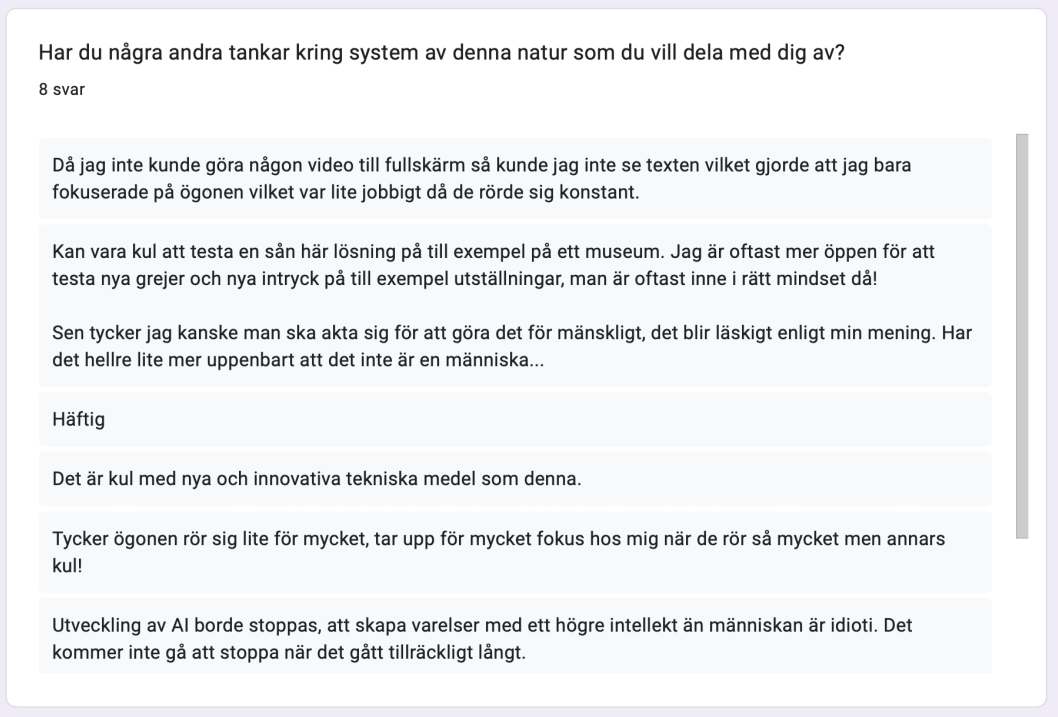
Figur 70: Formulering av svar från Chattbot.



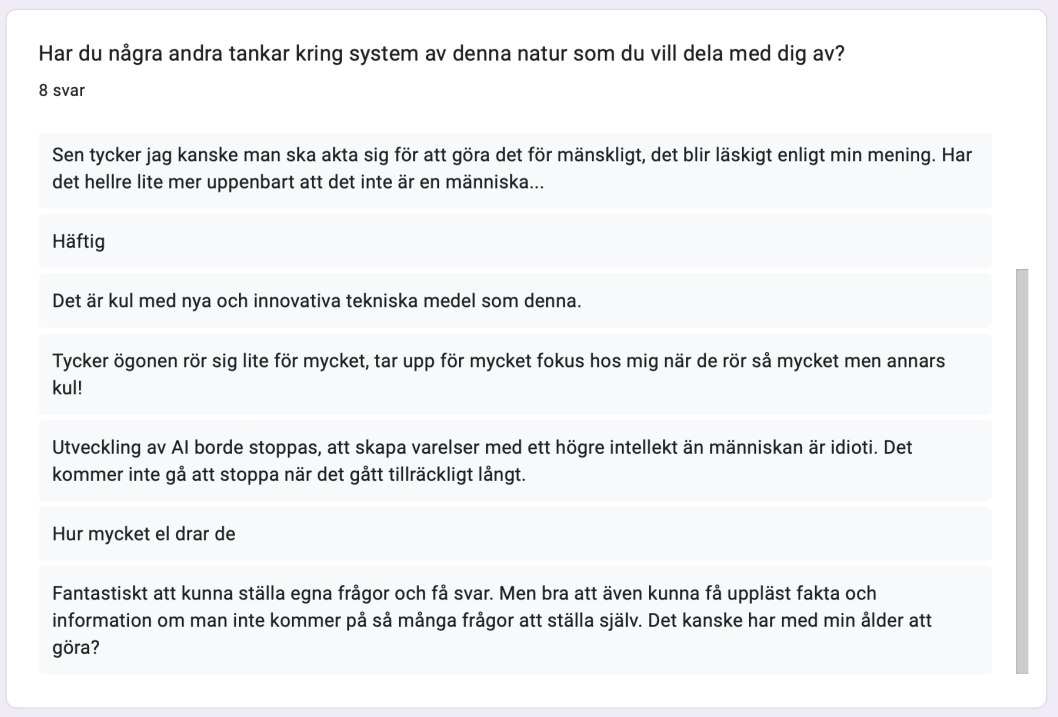
Figur 71: Alterntiva system.



Figur 72: Alterntiv informationsfördelning, icke teknisk.



Figur 73: Övriga tankar från användare - fri text - del 1.



Figur 74: Övriga tankar från användare - fri text - del 2.